

**本科生毕业设计[论文]**

**基于模糊测试的Web漏洞挖掘技术研究与实现**

院 系 网络空间安全学院

专业班级 信息安全1805

姓 名 肖昂

学 号 U201812212

指导教师 陈凯

2022年 5月 7日

**学位论文原创性声明**

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包括任何其他个人或集体已经发表或撰写的成果作品。本人完全意识到本声明的法律后果由本人承担。

作者签名： 年 月 日

**学位论文版权使用授权书**

本学位论文作者完全了解学校有关保障、使用学位论文的规定，同意学校保留并向有关学位论文管理部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权省级优秀学士论文评选机构将本学位论文的全部或部分内容编入有关数据进行检索，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。

本学位论文属于 1、保密囗，在 年解密后适用本授权书

2、不保密囗 。

（请在以上相应方框内打“√”）

作者签名： 年 月 日

导师签名： 年 月 日

**摘 要**

随着互联网的繁荣和Web技术的普及，许多互联网企业和公共机构通过Web应用向用户提供服务。相比较传统的C/S架构，Web应用程序的分布性强，可移植性更好，然而用户在购物、娱乐和办理业务时海量的个人信息将通过Web应用程序传输和处理，而且Web应用程序的开发水平参差不齐，由此而引发的信息安全问题也越来越值得重视，因此对Web程序进行漏洞挖掘和评测有助于提高Web应用程序的安全性。

在Web漏洞挖掘技术研究与实现中，首先介绍常见了Web漏洞挖掘技术，然后介绍模糊测试技术的发展历程和基本原理，并讨论将模糊测试技术引入Web漏洞挖掘的可行性。其次，考虑到模糊测试的局限性，引入遗传算法来改进模糊测试中数据完全随机生成导致的测试效率较低的问题。最后，在技术实践阶段，基于python语言和相关依赖模块实现了结合遗传算法和模糊测试的Web漏洞挖掘系统，生成被混淆的数据爆破字典以绕过网站应用级入侵防御系统防护，并对具有常见漏洞的Web网站进行漏洞挖掘，以减少其受到攻击的可能性。

基于上述方案实现了Web漏洞挖掘系统，并在DVWA(Damn Vulnerable Web Application)测试平台上进行了有效的漏洞挖掘测试，测试结果证明相较于没有利用遗传算法和模糊测试所构造的Web漏洞扫描器，在SQL注入、文件上传漏洞和跨站点脚本攻击的探测上有更好的覆盖率，利用遗传算法改善后在数据包有效率、数据包发送速度和面对WAF时具有更好的表现。

**关键词：**Web漏洞挖掘技术；模糊测试；遗传算法；网站应用级入侵防御系统

**Abstract**

With the prosperity of the Internet and the popularization of Web technology, many Internet companies and public institutions provide services to users through Web applications. Compared with the traditional C/S architecture, web applications are more distributed and have better portability. However, when users shop, entertain and handle business, massive amounts of private information will be transmitted and processed through web applications. The development level of Web applications is uneven, and the information security issues caused by this are more and more worthy of attention. Therefore, the vulnerability mining and evaluation of Web programs can help improve the security of Web applications.

In the research and implementation of Web vulnerability mining technology, the common Web vulnerability mining technology is firstly introduced, then the development process and basic principles of fuzzing testing technology are introduced, and the feasibility of introducing fuzzing testing technology into Web vulnerability mining is discussed. Secondly, considering the limitations of fuzzing testing, a genetic algorithm is introduced to improve the problem of low testing efficiency caused by completely random data generation in fuzzing testing. Finally, in the technical practice stage, a web vulnerability mining system combining genetic algorithm and fuzzing is implemented based on python language and related dependency modules, and an obfuscated data blasting dictionary is generated to bypass website application-level intrusion prevention system protection. Vulnerable Web sites conduct vulnerability mining to reduce their possibility of being attacked.

Based on the above scheme, a Web vulnerability mining system is implemented, and an effective vulnerability mining test is carried out on the DVWA (Damn Vulnerable Web Application) test platform. The test results show that compared with the Web vulnerability scanner constructed without genetic algorithm and fuzzing, It has better coverage in the detection of SQL injection, file upload vulnerabilities, and cross-site scripting attacks, and has better performance in data packet efficiency, data packet transmission speed and WAF after using genetic algorithm improvement..

**Key Words：**Web vulnerability mining technology; fuzzing test; genetic algorithm; Web Application Firewall

**目 录**

[**摘 要** I](#_Toc103650892)

[**Abstract** II](#_Toc103650893)

[**1 绪论** 1](#_Toc103650894)

[1.1 研究背景与意义 1](#_Toc103650895)

[1.2 国内外研究现状 2](#_Toc103650896)

[1.2.1 漏洞挖掘技术 2](#_Toc103650897)

[1.2.2 模糊测试 3](#_Toc103650898)

[1.2.3 遗传算法 4](#_Toc103650899)

[1.3 论文主要研究内容 4](#_Toc103650900)

[1.4 论文组织结构 5](#_Toc103650901)

[**2 相关理论和技术** 6](#_Toc103650902)

[2.1 漏洞挖掘技术 6](#_Toc103650903)

[2.2 模糊测试技术 7](#_Toc103650904)

[2.3 遗传算法 8](#_Toc103650905)

[2.3.1 基因编码方案 9](#_Toc103650906)

[2.3.2 适应度函数 9](#_Toc103650907)

[2.3.3 遗传算子 10](#_Toc103650908)

[2.3.4 迭代和演化 11](#_Toc103650909)

[2.4 本章小结 11](#_Toc103650910)

[**3 Web漏洞挖掘系统理论和需求分析** 12](#_Toc103650911)

[3.1 Web漏洞分析 12](#_Toc103650912)

[3.2 网站应用级入侵防御系统 13](#_Toc103650913)

[3.3 本章小结 14](#_Toc103650914)

[**4 Web漏洞挖掘系统设计和实现** 15](#_Toc103650915)

[4.1 系统架构 15](#_Toc103650916)

[4.2 请求和响应报文分析 15](#_Toc103650917)

[4.3 模糊测试字典 18](#_Toc103650918)

[4.5 本章小结 20](#_Toc103650919)

[**5 实验及分析结果** 21](#_Toc103650920)

[5.1 实验环境 21](#_Toc103650921)

[5.2 漏洞挖掘系统测试 21](#_Toc103650922)

[5.3 漏洞挖掘系统的优点 24](#_Toc103650923)

[5.4 漏洞挖掘系统的不足 26](#_Toc103650924)

[5.5 本章小结 26](#_Toc103650925)

[**6 总结和展望** 27](#_Toc103650926)

[6.1 论文总结 27](#_Toc103650927)

[6.2 研究展望 27](#_Toc103650928)

[**致 谢** 29](#_Toc103650929)

[**参考文献** 30](#_Toc103650930)

**1 绪论**

**1.1 研究背景与意义**

自从计算机网络技术开始在社会各方面广泛被应用，如何为网络上的数据提供安全保障一直是非常重要的问题。Web2.0技术兴盛的当下，用户会通过Web应用与网络平台进行大量的数据交互，而这些网络平台上会存储海量的涉及隐私的个人数据，如何保护敏感数据的机密性、完整性和可用性一直是网络服务提供商所考虑的问题。

根据国际电信联盟(International Telecommunication Union, ITU)的数据，全球互联网用户数量从2019年的41亿增长到2021年的49亿[1]。人们通过网络访问在线视频网站、在社交平台分享信息、享受线上教育和进行远程工作等等。在互联网已成为全球化的重要组成部分的同时，由中国国家信息安全漏洞共享平台(China National Vulnerability Database, CNVD) [2]二零二一年发布的统计数据可知(如图1-1所示)，网络应用漏洞占比为22.3%，仅次于应用程序漏洞，网络安全问题越来值得重视。



图1-1 2021年CNVD漏洞影响类型分布

为了保护众多互联网用户的信息安全和保障Web内容提供平台的正常运作，以达到营造安全可靠的互联网环境。本课题的目的是针对测试用例有限、测试方法多样、测试目标盲目等问题，通过动态生成测试用例对指定网站进行渗透测试和漏洞挖掘。通过使用并优化现有的Web网站模糊测试方法，对指定的Web网站进行动态测试和漏洞挖掘。

**1.2 国内外研究现状**

**1.2.1 漏洞挖掘技术**

漏洞一般是指程序或协议存在逻辑漏洞或者其安全策略设计不周，使黑客能够未经授权非法访问系统，获取或破坏私有的系统数据，甚至直接对系统造成永久性损害。如今有海量的个人隐私数据在互联网上存储和传播，其中数据的价值甚至远高于互联网结构本身，例如社交平台被攻破导致的隐私数据的泄露会给用户和平台都造成严重的损失。Web应用安全已经成为安全研究的热门内容，开放式Web应用程序安全项目[3](Open Web Application Security Project, OWASP)是一个由开源社区主导的软件项目平台，除了检测软件外，该平台还分享了很多Web安全相关的知识。除了这种大型的开源平台外，还有许多研究团队针对Web安全中的各个方面发表了自己的技术成果。

针对Web搭建中常用的PHP语言，Jiazhen Zhao等人利用静态污点分析搭建了一个PHP Webshell分析框架[4]。要想进行污点分析，先将PHP源码转化带解析器的抽象语法树(abstract syntax tree, AST)，然后生成控制流图(control flow graph, CFG)。污点分析就在控制流图上进行，可以分为两个步骤。首先是识别污染源，包括用户的直接输入和一些敏感函数，然后是传播污染节点，在确定污染源之后，通过遍历控制流图确定所有和污点源相关的变量并进行标记。被SonarSource收购后的RIPS(Research and Innovation to Promote Security)是对Web中的PHP代码进行静态分析方面的一个成熟的软件，它建模了952个内置函数来记录PHP的特定漏洞并利用反向分析以提高计算性能[5]。除了前面所谈及到的白盒测试，漏洞挖掘领域还有许多黑盒测试方法。Widia等人在黑盒测试中结合边界值分析和划分等价类[6]，其中边界值分析用于测试具有范围值的数据类型，而等价类划分的作用是的定义数据标准以划分所有可能的数据。Byun等人结合深度神经网络结构定义了一个用于测试机器学习系统的黑盒覆盖标准[7]，通过建立黑盒覆盖标准的系统已评估其有效性。黑盒测试也包含传统模糊测试技术，该项技术会在下文中提及。

**1.2.2 模糊测试**

经典模糊测试是一种向应用程序提供随机输入的简单技术。随机测试经过了几十年的发展，最初的经典模糊测试具有以下的特性。首先，输入是随机的ASCII字符流，而非使用任何程序行为、应用程序类型或系统模型，即是黑盒测试。在命令行模糊测试研究中，随机输入是非结构化的。对于更复杂的系统进行测试时则可以采用随机的结构化的输入。其次，程序的可靠性标准被简单的定义为程序是否崩溃或异常，如果由则记录下发生异常的输入数据。在这个测试中，应用程序不必以预设的方式响应输入或处理崩溃，它可以不处理异常而直接退出。基于上述特征，经典的模糊测试可以跨应用程序、操作系统进行，并实现高度自动化。由于这些不规则的输入，模糊测试更容易发现以前未知的漏洞，这也是模糊测试发挥重要作用的关键原因之一。

如今模糊测试已逐渐演变为一种综合技术，协同其他技术已提高测试覆盖率和效率。考虑到模糊测试器对一般程序的测试代码覆盖率较低，因为不受约束的随机输入可能没有沿着许多不同的路径执行，相比之下符号执行[8]通过系统地同时探索许多可能的执行路径而不必需要具体的输入。解释器跟随程序，假设程序输入值是符号而非正常程序执行中的实际输入，从程序中的那些表达式符号和变量推断逻辑表达式，并从这些符号输出表达式作为每个条件分支的可能结果。符号执行提供了更好的路径覆盖率。BS Pak等人展示了通过结合符号执行和模糊测试来实现一种混合的模糊测试技术[9]。该技术首先使用符号执行来发现测试程序中的边界节点。测试人员可在指定资源下收集尽可能多的前沿节点后约束，然后使用预处理的随机输入对程序进行模糊测试，其实验表明，结合符号执行和模糊测试的混合模糊测试生成的输入能够比以前的方法获得额外的广度和深度，在时间和空间上都是更有效率的。Böhme等人则提出了定向灰盒模糊测试[10]，目的是有效的到达一组给定的目标程序位置，开发了基于模拟退火(simulated annealing, SA)分配不同的能量给种子，对于接近目标程序位置的种子给予更多的能量，其AFLGo[11]的实验表明结合了AFL的定向灰盒模糊测试在测试效率上高于同类的其他测试工具。

**1.2.3 遗传算法**

遗传算法(Genetic Algorithm, GA)由JH Holland等人提出，其灵感来自生物进化过程[12]，属于元启发算法(Metaheuristic)的一种。元启发算法除了遗传算法外还有Kirkpatrick等人的模拟退火[13]和Glover的分散搜索[14]，在离散空间上寻求最优解。对于随机生成输入数据时过大的解空间，单纯应用穷举搜索难以解决问题，可以借助遗传算法优化搜索效率。随着技术发展，遗传算法被用来解决很多组合优化问题，Stepanov等人利用遗传算法建立了数学模型识别安全漏洞已评估信息系统机构安全的信息安全级别[15]。Sun等人将遗传算法和卷积神经网络(convolutional neural network, CNN)结合起来[16]，针对卷积神经网络架构需要手动设计而专业知识不足的用户较难为自己感兴趣的图像分类任务提出较好的卷积神经网络架构的问题，提出了一种使用遗传算法的自动卷积神经网络架构设计方法，可以有效的解决图像分类任务，其中最大的优点便是可以自动化进行。Mirjalili等人则对遗传算法中选择、交叉合变异进行了改进，利用遗传算法在图像处理领域从完全随机的图像中重建二值图像[17]。遗传算法和模糊测试技术结合可以改善构造数据完全随机化导致测试效率低的问题。

**1.3 论文主要研究内容**

本论文基于Web应用漏洞挖掘中的模糊测试技术，结合遗传算法，设计并实现了一个自动化进行Web漏洞挖掘的工具，其主要内容包括

（1）研究了当下漏洞挖掘技术、模糊测试技术和遗传算法在相关领域的技术论文和实践工具，介绍了白盒测试、黑盒测试和灰盒测试的发展历程，分析模糊测试结合符号执行或者卷积神经网络技术进行的改进，讲述了遗传算法在处理随机化过程中的优点。

（2）分析了Web漏洞挖掘系统的基本需要和功能，对系统的模糊测试模块、字典生成模块和遗传算法模块等进行了理论分析，阐述了各个模块以何种方式被组织起来，以及优化模块的方式和可能。

（3）调用pycurl库设计了一个可以并行的HTTP报文发送和接受处理模块，在构造向量输入模块可以高效地进行模糊测试；利用pyparsing进行作为过滤器处理构造和筛选数据包；基于python语言完成遗传算法模块，用于优化模糊字典的生成。

（4）测试了不同的Web防护类型下注入漏洞的情况，对于没有WAF防护的Web网站来，分析了利用模糊测试字典爆破的有效性以及使用并发快速处理的高效性；即使是一些WAF防护的网站也可以利用遗传算法生成的模糊测试字典混淆语句绕过WAF防护并进行漏洞挖掘。

**1.4 论文组织结构**

本文分为六个章节，每个章节的内容如下：

第1章是绪论，说明了Web漏洞挖掘技术的具体背景和研究目的，接着介绍了漏洞挖掘技术、模糊测试技术和遗传算法的研究现状。

第2章讲述了白盒测试、黑盒测试和灰盒测试的当先研究现状；分析了模糊测试技术和其他技术结合的可能，对遗传算法中各个模块进行了介绍，为之后构造漏洞挖掘系统铺垫。

第3章阐述了介绍Web漏洞挖掘的系统需求对SQL注入、跨站脚本攻击等漏洞的检查和利用，以及WAF模块在Web中如何部署以及绕过，为实践中绕过WAF进行注入漏洞挖掘提供理论指导。

第4章对系统的总体架构和实现细节进行了讲述，从信息收集、输入向量构造、报文分析和模糊字典生成多个步骤介绍系统细节、可能遇到的问题和解决方案。

第5章是实验与分析，介绍了实验情况和分析结果，给出了系统在自动化运行上高效和绕过WAF系统的方案，也指出在面对可能需要利用缓冲区溢出或者其他复杂漏洞是绕过WAF较难的情况。

第6章是研究总结与展望，对本文所做的理论研究和系统实验进行总结分析，并讨论未来可能的方案和完善的系统的思路。

**2 相关理论和技术**

本章将论述系统实现所涉及到的技术理论，主要包括Web漏洞挖掘技术、模糊测试技术和遗传算法原理，并讨论这些技术与Web漏洞挖掘相结合的方法。

**2.1 漏洞挖掘技术**

测试软件的逻辑错误和安全漏洞一直是软件开发周期的一部分，然而从设计和开发阶段一直到生产部署和维护，许多不同的人编写了很多代码，造成了错误无法避免的情况。为了维护软件安全，减少软件漏洞，对软件进行漏洞挖掘是十分必要的。Web漏洞测试方法大致可以分为白盒测试和黑盒测试。现在也有结合白盒测试和黑盒测试的灰盒测试，只需了解程序内的部分结构，例如程序内部数据结构或算法。

白盒测试需要保证程序中重要路径的可靠，覆盖程序内部所有关键逻辑分支，并且检查代码是否符合编码规范。白盒测试的基本程序要求测试人员对被测试的源代码有深入的了解，才能知道要创建什么样的测试用例，以便对每条可见路径进行测试。白盒测试需要分析源代码以创建测试用例。其具体步骤是，首先要了解程序的功能模块、详细的设计文档，这是测试的准备阶段，用于了解代码细节。然后构建测试计划，构建测试用例并执行，确保这些测试用例可以充分测试程序并记录测试用例对应的结果。最后输出依据测试计划生成的报告即可。白盒测试的优势是可以了解源代码的副作用并进行彻底的测试，并优化代码，对于何时停止测试也提供了清晰的规则。然而缺点是白盒测试是十分复杂的，测试人员必须对程序有一定了解，需要测试团队中有优秀的程序员。而且由于测试方案和软件紧密耦合，当需要更新软件结构时必须做额外的工作以再次匹配。而且虽然白盒标准具有有效地发现实现缺陷的优势，但当完全依赖白盒测试时，会带来只见树木不见森林的风险。例如程序所缺失的功能可能会在未被检测到的情况下免于检测。而黑盒测试提供了一种互补的自上而下的视角，这是克服这些缺点所必需的。在实践中，两者往往同时使用并且都是必要的。

黑盒测试是一种不需要任何有关被测软件内部信息的测试方法。在测量测试充分性时，黑盒技术依赖于更高抽象级别的工件，测试人员只需知道软件应该做什么，但无需了解程序的实现细节，对某一个特定输入会返回一个特定的、不变的输出，但是不知道软件内部的实现细节。根据规范、需求和设计参数构建好测试样例输入，然后观察程序的异常和崩溃信息。白盒和黑盒以及结合两者所用的灰盒测试，在现在技术发展下，其中的界限轮廓已经逐渐模糊了。白盒最初意味着利用源代码，黑盒意味着使用需求，然而现在测试是需要依赖抽象层级不同的设计结构的，对于自顶向下的软件结构来说，测试时需要的模型可能需要细致到逻辑谓词，也可以是抽象的输入输出。划清白盒黑盒等术语的界限或许不再重要，在实际测试中需要依据需求和程序环境来决定。

**2.2 模糊测试技术**

模糊测试技术(fuzz testing)是一种自动生成精心设计的输入并将其发送导应用程序，同时监视该程序的行为以确定是否在特定输入下会导致未定义或恶意行为的技术。该技术最开始由Barton Miller等人在一篇探究UNIX程序可靠性的论文[18]中实践，该论文为测试程序而构建了一个模糊测试的随机字符生成器还有相关的交互程序和自动化脚本，并对导致的崩溃进行了识别和分类。随着技术的发展，模糊测试器可以发现从可执行二进制文件到网络协议等各个方面的漏洞[19]。

模糊测试技术的步骤如下。首先需要选择模糊测试的目标应用程序，考察程序中的文件或其所调用的库。可以针对某个被多次调用的库进行检测，因为这样的库如果出现安全漏洞就会影响到多个应用程序。模糊测试的输入变量往往是程序中用户的输入数据，例如HTTP请求中的消息头，也可以是文件内容、路径和环境变量等。

生成测试数据可以划分成以下多个方法。随机生成数据，通过纯粹随机所生成的数据有较低的测试效率，而且需要花费很多时间去检索生成的大量崩溃数据，其中可能很多数据都是无效的。但是该方法的优点便是简单粗暴有效。除了纯粹随机生成，还可以手动或自动根据协议生成被构造好的数据。手动生成的测试数据即需要测试者对被测程序具有一定的了解，知道用什么方法更容易触发漏洞。对于Web漏洞挖掘来说，SQL注入等漏洞的Fuzzing数据构造就可以由测试者手动构造。自动化地根据协议生成输入数据也需要预先研究了解输入数据格式和规划，并根据此创造一个输入模板，将模糊测试生成的数据填入对于模板中，相对于纯随机生成来说，这种方法的效率更高。除了上述生成方法，还可以使用已有数据变异的方法生成测试数据。针对正常的程序输入内容，对其中的数据样本的字节或输入向量的各个模块进行变异，在不断变异中同时向目标应用程序中发送，这种方法的优点是易于持续自动化测试，但是由于对字节或模块变异的随机性，生成的无效数据比较多，效率偏低。构造好测试数据后，即可向目标程序发送数据包并监视程序异常或崩溃，在这个过程往往可以实现自动化执行。经过一段时间的自动化执行后，即可分析处理数据结果，收集那些使程序出现异常的数据包并进行进一步的分析。

Barton Miller在1988年首次提出模糊测试[20]概念时即是对UNIX程序的本地模糊测试，通过命令行输入随机的ASCII字符流。此外，还有针对文件格式的模糊测试器，因为许多程序都把文件作为输入或输出，而文件格式的模糊测试器可以通过生成不规则的文件格式，然后运行处理这些文件的应用程序，查看应用程序是否会出现异常和崩溃。除了本地模糊测试，还有针对Web应用程序的远程模糊测试。通过互联网可以访问网页和发送电子邮件等，这其中也存在漏洞可能被恶意利用，因此对网络协议进行模糊测试是十分有必要的。面向Web应用程序的模糊测试器会对HTTP报文进行构造并分析Web应用的响应来检查是否存在漏洞。有些报文是构造是比较简单的，没有认证和校验，但是还有些复杂的报文需要cookie等额外信息甚至经过了加密。

总的来说，模糊测试技术可以分为两种，基于变异测试和基于生成测试，前者是改变数据样本的字节构造输入向量，后者是对程序输入格式进行构造，自行生成符号输入格式的数据。虽然模糊测试可以发现未知漏洞并高度自动化，但是无法挖掘到导致程序崩溃或返回异常的漏洞和病毒。本文所探讨的模糊测试技术即是针对程序输入向量进行构造的，面向Web应用程序使用HTTP报文格式进行发包测试，并且依据网站返回码来判断是否触发漏洞。

**2.3 遗传算法**

遗传算法是利用计算机仿真运算结合计算[数学](https://zh.wikipedia.org/wiki/%E8%AE%A1%E7%AE%97%E6%95%B0%E5%AD%A6)用于获得[最优化](https://zh.wikipedia.org/wiki/%E6%9C%80%E4%BD%B3%E5%8C%96)的搜索[算法](https://zh.wikipedia.org/wiki/%E7%AE%97%E6%B3%95)，十分适合利用随机数据优化。遗传算法对实际问题的参数空间进行搜索，对于模糊测试来说，参数空间即是被构造的输入向量。遗传算法是借鉴生命科学中的[遗传](https://zh.wikipedia.org/wiki/%E9%81%97%E4%BC%A0)、[突变](https://zh.wikipedia.org/wiki/%E7%AA%81%E5%8F%98)、[杂交](https://zh.wikipedia.org/wiki/%E6%9D%82%E4%BA%A4)以及[自然选择](https://zh.wikipedia.org/wiki/%E8%87%AA%E7%84%B6%E9%80%89%E6%8B%A9)，在多次迭代后可以提高种群中个体的适应度。在多次迭代过程中保留种群中的优秀个体。遗传算法的实现方式为计算机模拟进化过程，本课题根据其算法流程图(如图2-1所示)介绍算法实现细节。

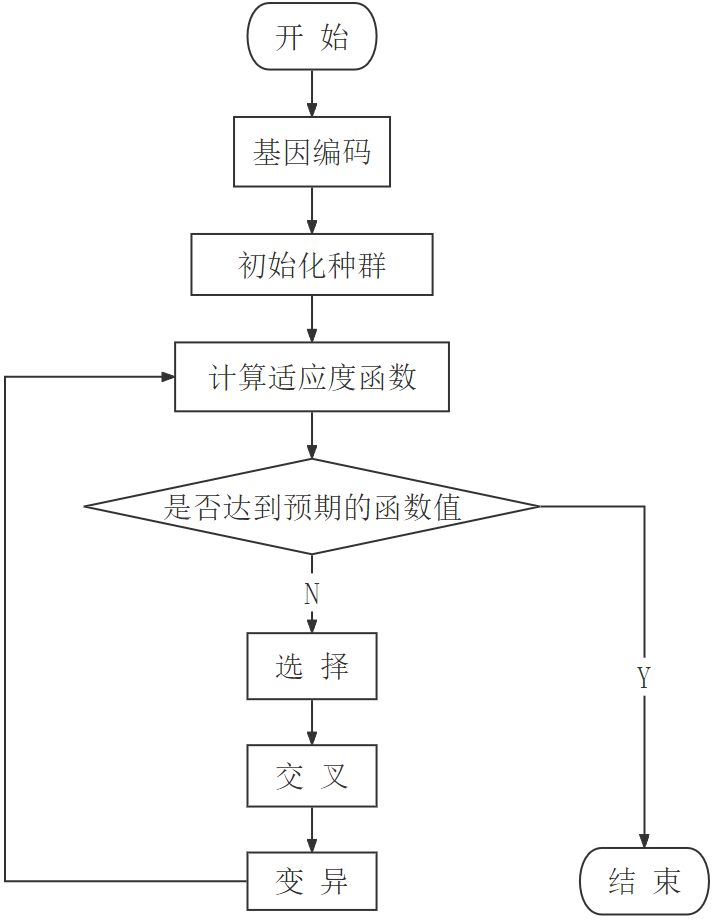


图2-1 遗传算法流程图

**2.3.1 基因编码方案**

遗传算法需要将所处理的数据转换为基因表示，将数据串在测试中的表现型转换为。在基因编码中可以使用二进制编码、格雷码编码或者浮点数编码。使用二进制编码即使用简单的01比特串来表达数据。这样处理的好处是数据编码简单，然而二进制编码在变异时的局部搜索能力不强，仅仅变异一个比特位可能会导致表现型发生巨大的变化。格雷码可以解决二进制编码的这个问题，格雷码所表示的两个相邻整数只有一位不同，这样进行比特位变异时，表现型变化较小，在探索解空间时有较好的局部搜索能力。浮点数编码是数据中的每个基因对应成一个实数，输入数据个数决定基因个数。相较于前两种编码，浮点数编码的精度更高，搜索空间更大但是算法设计更复杂。适合解决输入向量维度较高和对结果精度较高的问题。

**2.3.2 适应度函数**

适应度函数的目的是评判个体的优劣程度以决定其遗传概率大小。定义适应度函数是十分重要的，不当的适应度函数可能会导致遗传算法迭代后趋于局部最优解。适应度函数应具有如下特征，首先适应度函数应便于计算，可以直接反应个体的优劣，在数学上适应度函数应该是单调连续且非负。对于模糊测试过程，每一个测试样例发现漏洞的时间和个数等，在远程模糊测试中，则可以看服务器的响应时间和某种特定返回码的数量。

**2.3.3 遗传算子**

遗传算子是模拟自然选择中多代种群的选择、交叉和变异过程，为了在迭代演化中趋于最优解，需要根据数据模型和方案要求设计合适的遗传算子。

选择算子的作用是尽量将适应度高的个体保留到下一代。目前流行的选择算子有：轮盘赌选择(Proportionate Roulette Wheel Selection)[21]的实现方法是，个体被选则的概率等于个体的适应度值除以群体适应度总值。除了最基本的轮盘赌选择算子，还有应用更广泛的锦标赛选择算法(Tournament Selection)[22]，算法细节是再整个种群中抽取n个个体，选取其中最优的个体并保留完成筛选，在实践中往往取n=2。锦标赛选择算法的优点在于选择策略简单，复杂度低。指数排名选择(Linear Ranking Selection)[23]和指数排名选择策略(Exponential Ranking Selection)[24]是为了解决适应度函数为0的个体不会被选择的问题，根据适应度函数值对所有个体进行排序，这两种算法的区别在于排序后个体的被选则概率是线性分布还是指数分布。

交叉算子的作用是在群体适应度提升的同时保障种群的多样性。目前使用的交叉算法有单点交叉、两点交叉和均匀交叉。单点杂交的过程是，在染色体上选择一个交换点，并将两个个体按照交换点的位置交换前后基因，多点杂交则是确定多个交换点，具体实现过程于单点杂交算法类似。均匀杂交算是是为了解决避免染色体中的部分基因被过早的收敛而舍弃的问题[25]。本实验根据模糊测试输入向量选择一般选择两点交叉，即对筛选后的个体两两配对，交换两个位置的基因。

变异算子是为了加强算法的全局搜索能力，变异算子有：基本位变异(Simple Mutation)比如在二进制编码和格雷码中，进行比特位翻转变异即是基本位变异算法。均匀变异(Uniform Mutation)是用均匀分布的随机数直接替换，而边界变异(Boundary Mutation)等选取边界基因值附近的随机数去替代原有值。在算法中，设计者还需要要决定变异率，如果变异概率过大，会使得算法退化成随机搜索过程如果变异概率较小会导致算法全局搜索能力下降，可以采用在算法前期取较大的变异概率，在算法趋近收敛的后期逐渐降低。

**2.3.4 迭代和演化**

每次迭代并计算适应度函数进行进化，理论上来说通过多代演化可以得到一些比较优秀的测试用例，再针对这些测试用例对可能的漏洞进行深挖。针对Web漏洞挖掘技术的实践来说，遗传算法的作用是优化其输入向量的值，也即被构造HTTP报文中的值。这些值可以存储在字典，而遗传算法的迭代会生成触发漏洞更多、效率更高的爆破字典。

**2.4 本章小结**

本章从漏洞挖掘技术、模糊测试技术和遗传算法的技术发展细节和实践中的算法选择过程进行了详细阐述，下一章将会针对具体漏洞类型、安全防御手段和绕过安全防御的方法进行展开。

**3 Web漏洞挖掘系统理论和需求分析**

结合上章对相关理论技术的详细描述，本章将将从系统需求和功能出发讲解Web漏洞挖掘系统理论，其中会针对SQL注入、跨站脚本攻击等和网站应用级入侵防御系统(Web Application Firewall, WAF)以及模糊测试字典生成的问题进行描述。

**3.1 Web漏洞分析**

本课题研究的目的是自动化挖掘Web漏洞。主流的Web漏洞有：SQL注入、跨站脚本攻击、跨站请求伪造等。SQL注入攻击主要是在未设置严格安全检查的SQL查询语句的情况下，直接将带有来自用户输入的SQL查语句在数据库运行。一个成功的SQL注入漏洞挖掘可以获得很有有用的信息，比如从存储大量用户隐私的数据库中进行读取修改甚至删除，管理或变更数据库权限或留下后门，具体情况取决于漏洞的危险等级。

对于SQL命令“SELECT ID, FIRSTNAME, LASTNAME FROM PLAYERS”，如果输入“FIRSTNAME: KAMI'SATO, LASTNAME: AYAKA”。则查询字符串变为“SELECT ID, FIRSTNAME, LASTNAME FROM PLAYERS WHERE FIRSTNAME = ' KAMI'SATO' AND LASTNAME =AYAKA”，数据库尝试运行会报错“Incorrect syntax near SATO' as the database tried to execute KAMI.”

上述 SQL 语句可以用Java重写为带安全检查的SQL语句(如图3-1所示)。

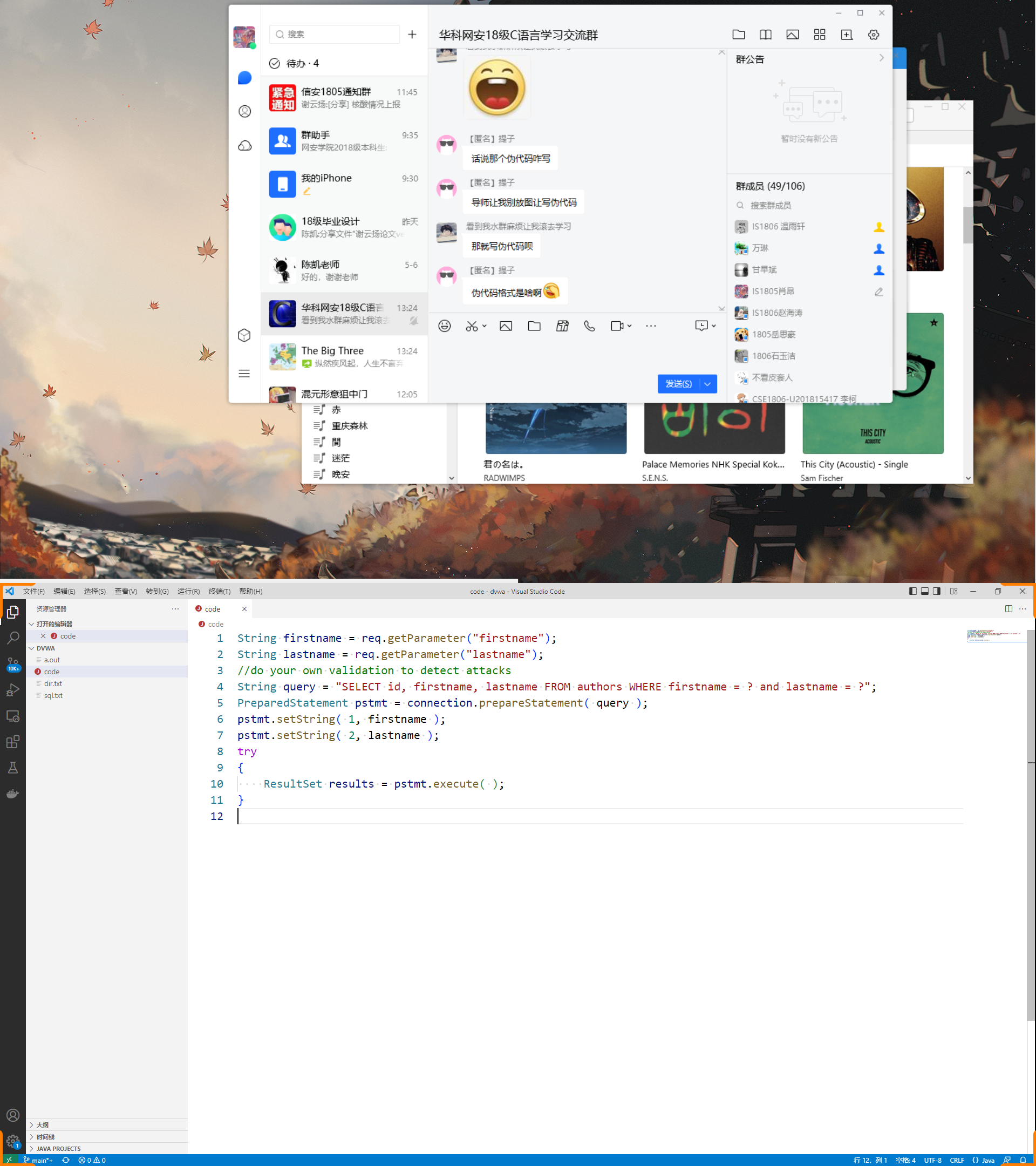


图3-1 有安全检查的SQL语句

[SQL盲注与普通SQL注入](https://owasp.org/www-community/attacks/SQL_Injection)的区别是从网页响应攻击者的方式，在SQL盲注中，当数据库不向网页输出数据，攻击者只能多次尝试获得数据库的不同响应来判度是否存在漏洞。相比于直白的数据库报错。缺乏数据库响应使得利用 SQL 注入漏洞变得更加困难。但这并非是对SQL语句进行了安全检查，因此安全漏洞依旧存在，而且其爆破方法与SQL注入是一样的。只有采用了有效的SQL安全检查语句则很难被找到安全漏洞。

跨站点脚本攻击(Cross Site Scripting, XSS)可以分为三种：存储型XSS、非持久型XSS和基于文档对象模型(Document Object Model, DOM)的XSS。

存储型XSS攻击和非持久型XSS都是将恶意代码注入到Web页面中，例如在论坛评论区向服务注入恶意代码后，受害者通过浏览器加载含有恶意信息的网页时，恶意代码会被运行。非持久型XSS有所不同的是不能在页面直接加载时运行，需要欺骗用户区点击才可以触发。但这两者攻击方法在实际上没有区别。

除了上述两种XSS攻击方法，[Amit Klein在2005年](http://www.webappsec.org/projects/articles/071105.shtml)发现了即[基于DOM的XSS](https://owasp.org/www-community/attacks/DOM_Based_XSS)[26]。与其他XSS漏洞不同在于，基于DOM的XSS漏洞并非因为在HTML页面中不安全地使用客户端输入，而是不完全由服务器提供的页面控制的DOM被在客户端代码中不安全的引用和调用。

还有很多常见Web安全漏洞，但是本文不能完全研究，故仅简单提及。远程文件包，黑客使用这种类型的攻击将文件远程注入 Web 应用程序服务器。这可能导致在应用程序中执行恶意脚本或代码，以及数据盗窃或操纵。跨站点请求伪造(Cross Site Request Forgery, CSRF)，一种可能导致未经请求的资金转移、更改密码或数据盗窃的攻击。它是由恶意 Web 应用程序使用户的浏览器在用户登录的站点中执行不需要的操作时引起的。

**3.2 网站应用级入侵防御系统**

考虑到Web网站往往具有防护，在实际挖掘中需要绕过这些安全防护才可能获取有价值的信息。网站应用级入侵防御系统(Web Application Firewall, WAF)将一组规则应用于HTTP会话，分析双向流量并过滤掉不安全的输入包信息，例如SQL注入和跨站脚本攻击。按照OWASP的观点来看，WAF是不依赖于应用程序本身的 Web 应用程序级别的安全解决方案[27]。

对于布置了WAF系统页面，想要继续注入SQL则需要其他的技术操作。例如SQL 注释允许攻击者绕过很多过滤和WAF的一些规则，如图3-2所示。



图3-2 利用注释绕过WAF

有些WAF在过滤SQL关键字时可能只过滤了大写或小写关键字，对于图3-3所示正则表达式，可以采用更改大小写的方式绕过。

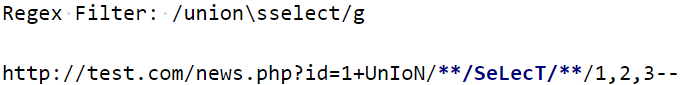


图3-3 利用大小写绕过正则表达式过滤器

有些WAF使用preg\_replace函数删除所有SQL关键字。但攻击者可以通过图3-4中的简单的写法绕过WAF对UNION和SELECT的检查。



图3-4 利用大小写绕过正则表达式过滤器

有的WAF会将SQL关键字会被替换为其他字符，例如Apache的mod\_rewrite函数，此时无法用注释"/\*\*/"再绕过，但可以用"%0b"替换"/\*\*/"，如图3-5所示。

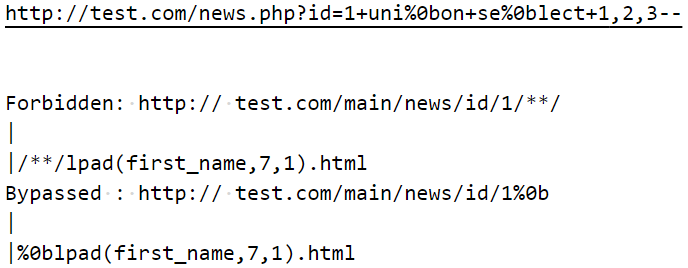


图3-5 利用"%0b"绕过检查

除了上述绕过方法，还可以采用通过利用缓冲区溢出(Buffer Over Flow)防火墙崩溃，但文主要是探究绕过WAF，故不探讨使WAF崩溃的相关技术了。

**3.3 本章小结**

本章根据挖掘漏洞在实际中需要考虑的需求和理论分析，介绍了SQL漏洞和XSS漏洞以及WAF防护机制，为下一章系统设计和实验提供理论依据。

**4 Web漏洞挖掘系统设计和实现**

本章将详细结构模块算法和代码，以及实现逻辑和过程。在构造向量输入模块利用pycurl进行并行发包处理，并接收所需返回码。pyparsing可以帮助简化正则表达式的语法处理，并用遗传算法迭代后的字典多次生成知道达到较高的发包效率和有效性。

**4.1 系统流程**

本漏洞系统逻辑流程图如图4-1所示，该系统大致可以分为两个部分，一个是进行HTTP发包测试的模糊测试模块，另一个则是遗传算法优化模块。

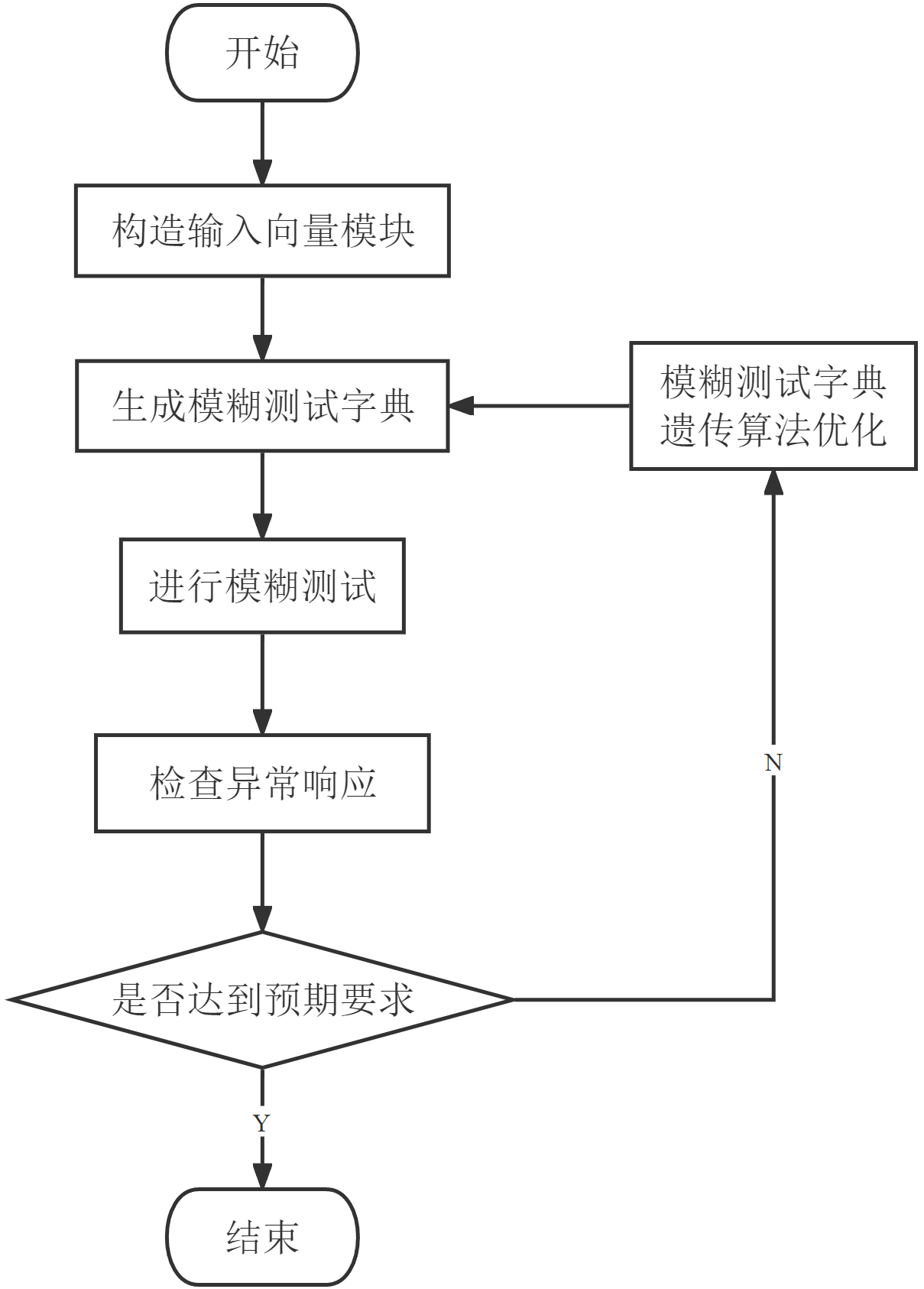


图4-1 Web漏洞挖掘系统流程图

**4.2 请求和响应报文分析模块**

HTTP发包需要事先对网站架构和输入格式进行详细了解，这需要对Web渗透有一定知识并花费实际进行调研，然后对写脚本对Web进行发包，请求行包含设置请求类型和资源路径等信息，请求体则包含需要发送的数据、定义语言还有连接状态。还要分析HTTP的相应包，其内容有状态行，包含协议版本、状态码、连接状态和服务器类型等其他信息，还有响应正文，既是浏览器对请求包所请求内容数据和相关信息。如图4-2所示，给出了HTTP请求和响应报文的示例。

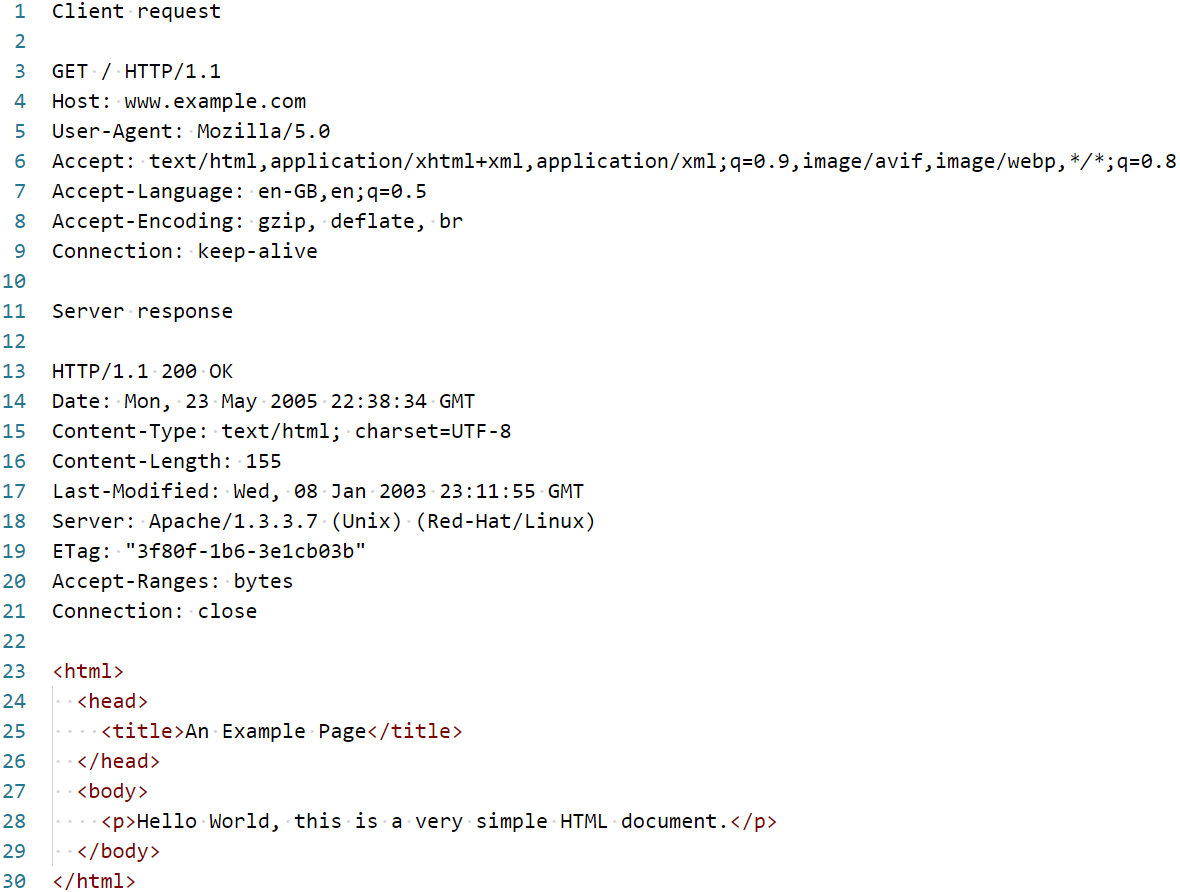


图4-2 HTTP请求和响应报文示例

为了执行HTTP请求，需要调用pycurl库，pycurl可以实现并发HTTP连接并且支持HTTP表单上传、代理、cookie和用户密码等身份验证，如图4-3所示。

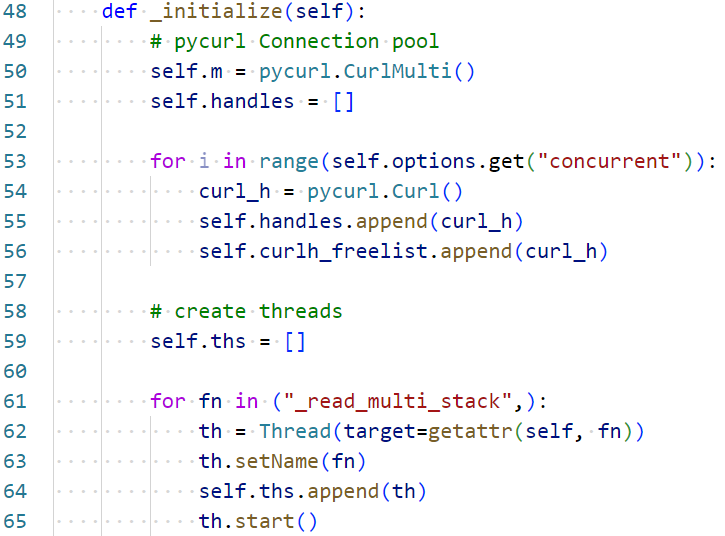


图4-3并行发送请求报文

发送报文后需要查看响应报文头信息以查看十分有崩溃或异常，异常是取决于漏洞类型的，对于路径探索模糊测试和SQL注入模糊测试，200 OK的返回码往往意味着探索到新目录或者存在可利用的SQL漏洞，如图4-4所示，通过pycurl来获取响应头信息。

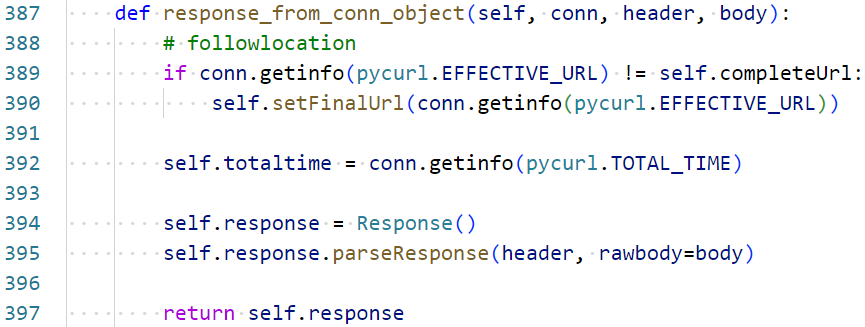


图4-4 获取响应报文信息

利用pyparsing模块对语法进行解析更为简单，而且可以筛选大量模糊测试返回码结果，方便找出值得分析的返回码。调用pyparsing模块创建过滤器语法如图4-5所示。

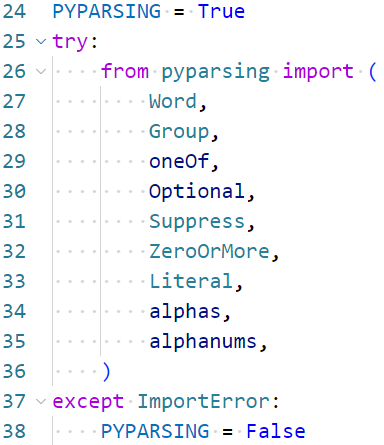


图4-5 调用pyparsing模块

在获取到完整的一轮模糊测试字典发包完成后，即可查看预期返回码的个数和所耗费的时间，有时候会期望40X或30X的返回码，但是在SQL注入时会期望得到200 OK返回码，因此具体返回码需要根据想挖掘的漏洞类型来决定。

**4.3 模糊测试字典**

在完成一轮测试后，本应该进行遗传算法改进模糊测试字典，但是这种改进是取决于是否有WAF防护，对于那些缺乏WAF防护的网站来说，在经过对网站仔细调研和信息收集后往往可以找到SQL漏洞，但是对于有WAF防护的网站来说需要生成随机数据混淆WAF。混淆WAF有两种办法，一种是在输入语句中夹杂一些特殊转义的十六进制数，另一种是触发缓冲区漏洞导致WAF系统宕机而无法判读，遗传算法迭代框架如图4-6所示，模糊测试字典实例如图4-7所示。



图4-6 遗传算法迭代框架

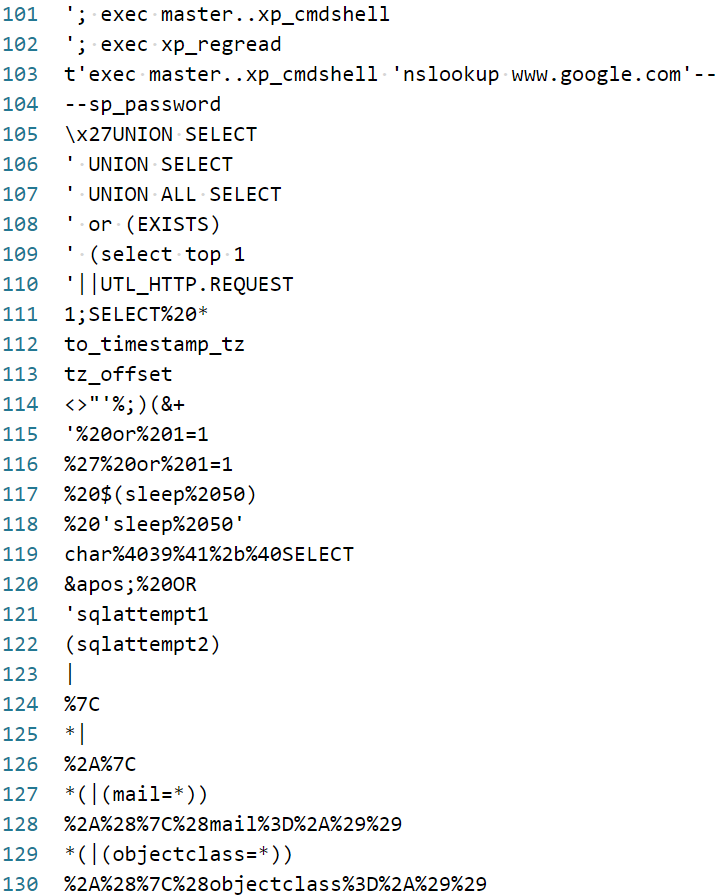


图4-6 模糊测试字典

在生成新的模糊测试字典后即可迭代使用新字典并多次分析返回结果，可以人工查看每次得出的模糊测试结果再决定是否继续迭代，抑或设置阈值进行自动化检查。

**4.5 本章小结**

本章详细介绍了系统主要模块的组成逻辑和细节，利用迭代演化生成优秀的测试样例，借助pycurl模块和pyparsing模块HTTP包接收和发送以及过滤器语法规则。

**5 实验及分析结果**

本章将通过实践阐述从搜集信息开始，到生成模糊测试字典并测试和分析返回数据中所困难遇到的细节和问题，展现系统的优点和不足之处。

**5.1 实验环境**

运行环境：Microsoft Windows 10 家庭版 10.0.19043；Intel64 Family 6 Model 158 Stepping 10 GenuineIntel ~2208 Mhz

相关依赖：chardet 4.0.0；pycurl 7.43.0.5；pyparsing 3.0.7；six 1.16.0

**5.2 漏洞挖掘系统测试**

系统界面如图5-1所示，本系统采用命令行输入，读取需要挖掘的url和fuzz参数。

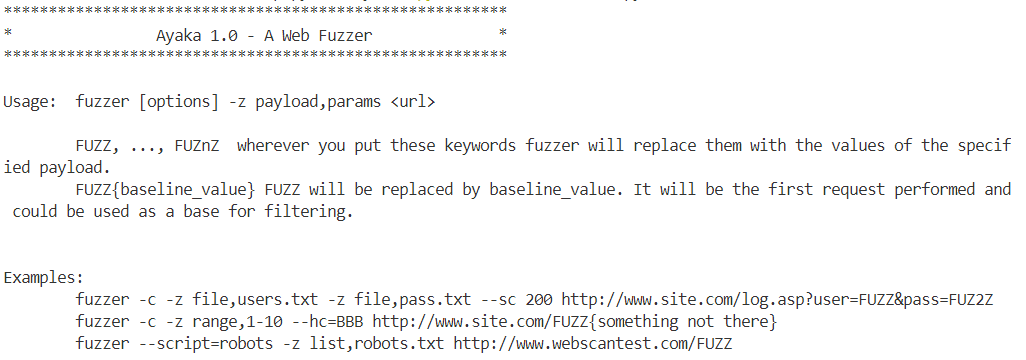


图5-1 系统初始界面

为了得到所需参数，首先需要对Web网站进行信息收集以了解其输入格式。使用Burp Suite进行发包接包分析。如图5-2所示，可知PHPSESSID=6hfnipfii3gh5 g93vu259k2ii7这个值将作为cookie参数输入到本系统中。

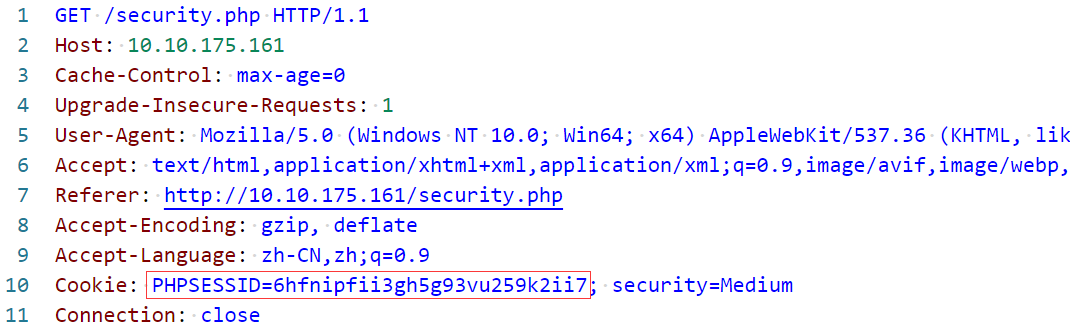


图5-2 利用Burp Suite收集网站信息

构造好的代码如图5-3所示，FUZZ处即将被替换成已经训练过的字典值。

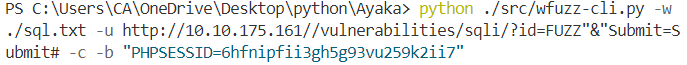


图5-3 根据信息构造输入

等待一会后即可看到返回码，返回行数，单词数和字节数以及对应的注入内容，如图5-4所示，该图中所有的返回码都是302，表示资源存在但是被临时调换了位置。但是与此同时的所有其他数据都是0，便可以知道此时302等价于404 Not found。

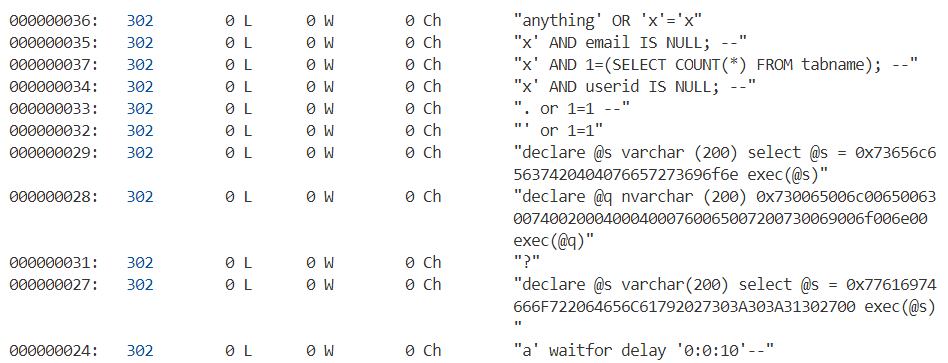


图5-4 无效的注入

仔细分析数据，如图5-5所示，有两个注入数据返回码是200 OK，而且其他数据也是有意义的值，可以猜测这两个数据可以触发注入漏洞，此时可以带入网页尝试，如图5-6所示，由此可知这种格式的注入都可能是可行的。

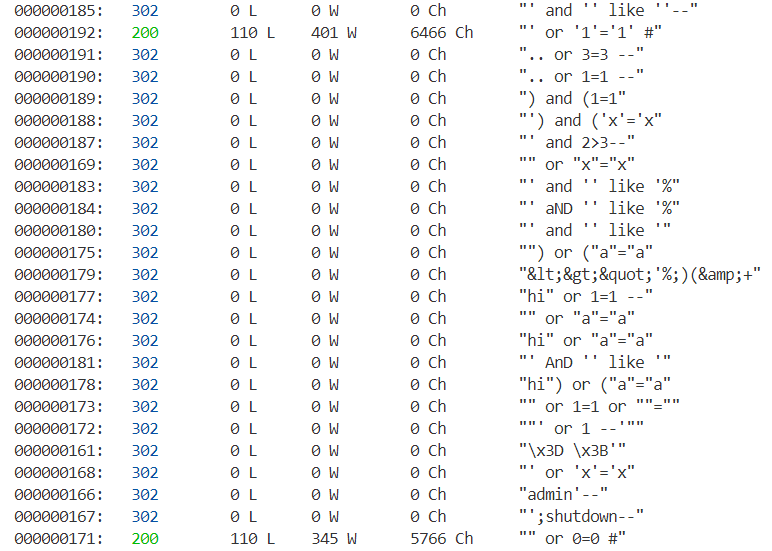


图5-5 200 OK的返回码

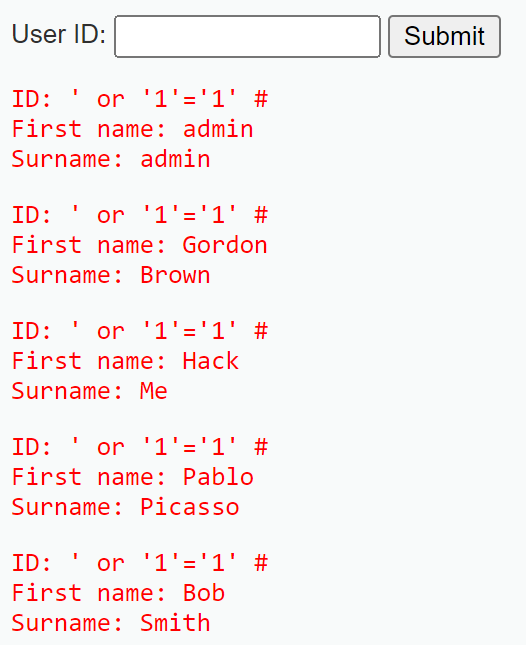


图5-6 注入实例

然而并非所有200 OK的回复码都是有效的，比如图5-7所示，虽然返回码是200，其他信息也都有意义，但是无法利用这个格式进行注入，受限于Web模糊测试分析返回码的局限性，分辨是否是注入格式需要手动操作。

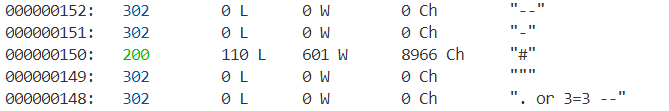


图5-7 无效的200 OK返回码

利用遗传算法和模糊测试生成的被混淆的注入语句，目的是绕过WAF的检查如，图5-8所示。对于某些WAF，仅仅对语句进行了简单而非全面的检查，这会导致一般语句无法注入但是经过混淆之后的语句变可轻松绕过WAF，如图5-9所示。



图5-8 生成被混淆的注入语句



图5-9被混淆的注入语句依旧有效

**5.3 漏洞挖掘系统的优点**

表5-1阐述了本漏洞挖掘系统在的优点并说明了各个系统功能上的差异，其中系统A即是用多次迭代后用遗传算法改善模糊测试字典的本漏洞挖掘系统，系统B是去掉了遗传算法，直接进行模糊测试的漏洞挖掘系统，系统C是没有利用pycurl等库的模糊测试发包系统。

表5-1 漏洞挖掘系统效果对比

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试目标 | 是否可以自动化运行 | 并行发包效率 | 是否可以绕过WAF | 有效数据包率 |
| 系统A | 是 | 30.2/sec | 可以 | 100% |
| 系统B | 是 | 28.5/sec | 不行 | 3.125% |
| 系统C | 否 | N/A | 不行 | N/A |

如图5-10所示，仅用模糊测试技术构造的漏洞挖掘系统的数据有效率仅为3.125%，大部分数据无法成功注入，每秒钟发包数为28.5，而如图5-11所示，利用遗传算法迭代改进后的数据包有更高的都可以通过过滤器检查，且发包效率为30.2每秒，更好的模糊测试字典提供了更快的并行处理速度。

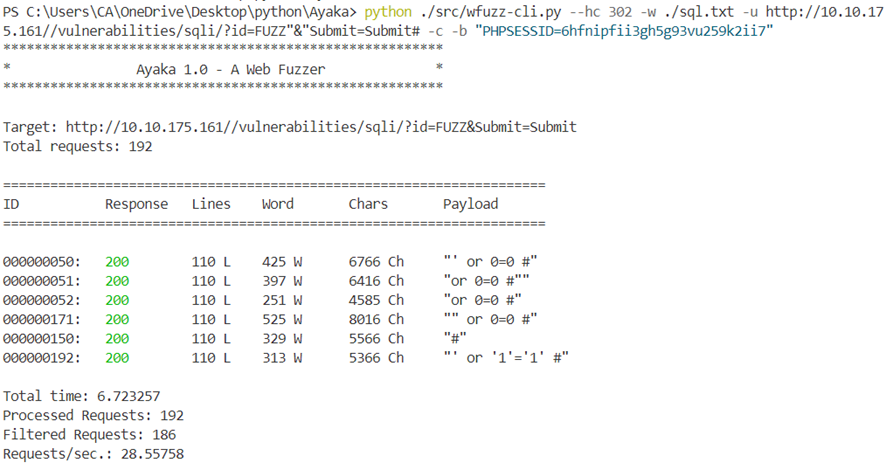


图5-10 不使用遗传算法

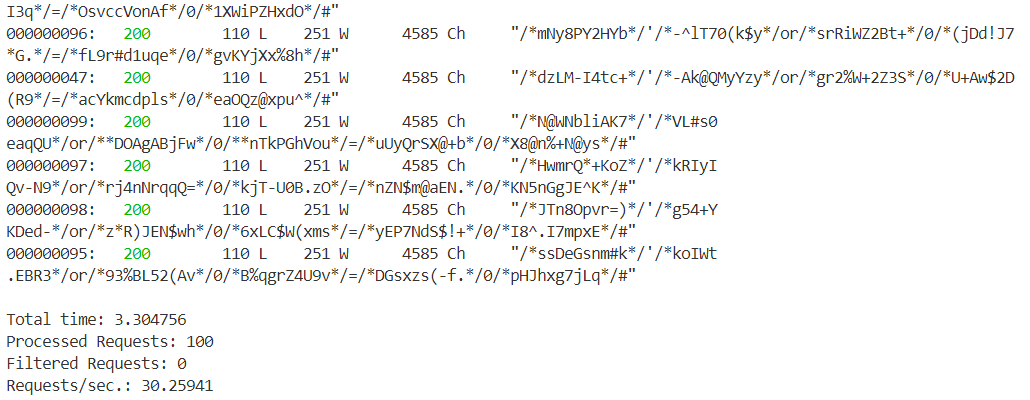


图5-11 使用遗传算法多次迭代

本系统的优点主要在于，通过模糊测试的工具，可以有较高的自动化进行并行的模糊测试发包，并且记录数据的状态返回码、返回字节数和行数等信息。而且对于用WAF系统进行防御的部分网址可以照样渗透，而一般的SQL注入会被WAF拦截。对于那些不理解网络协议而出于好奇的初学者，也可以在找到比较好的模糊测试字典后使用。还有就是将遗传算法框架于模糊测试模块相结合，对于那些想要了解遗传算法的使用者来说，可以帮助其掌握算法的要领，且这两个模块耦合度较低，方便其日后的优化和发展。

**5.4 漏洞挖掘系统的不足**

本系统的缺点是在复杂的商业网址中寻找注入漏洞有一定难度，单靠本系统构造的模糊测试和遗传算法模型难以成功，而且其WAF的防护是十分完善的，可能无法逃过正则表达式的规则过滤和相关安全检查，因此更多意义上是作为工具了解网络渗透、模糊测试和遗传算法。对于遗传算法在绕过WAF模块上的有化也存在能够直接绕过的WAF往往无需优化即可，而安全严密的WAF在优化混淆后也无法通过防火墙，想要绕过这类检查可能需要结合缓冲区溢出或其他漏洞，虽然模糊测试在缓冲区溢出等方面也有作用，但是本文的研究尚未设计，且缓冲区溢出漏洞可能需要对网站系统有更多的了解。

**5.5 本章小结**

本章通过实践讲述了从网站信息搜集开始，到利用遗传算法生成模糊测试字典并测试，以及分析返回数据包和避免了一般情况下可能被WAF检查的问题，对比了不同系统之间的差异，体现本系统在WAF绕过和挖掘效率上的优势，最后也阐述了系统的不足之处。

**6 总结和展望**

**6.1 研究总结**

本文主要研究使用模糊测试技术在互联网上搜索漏洞，其中使用遗传算法进行模糊测试，以减少测试用例的盲目性。首先，介绍了研究的背景和意义，以及国内外的研究现状，并讨论了与系统实施相关的技术理论。主要涵盖漏洞分析技术、模糊测试技术和遗传算法，并讨论了将这些技术与Web漏洞分析相结合的细节。然后，基于漏洞分析技术、模糊测试技术和遗传算法的发展细节，以及在实践中的算法选择过程，对遗传编码方案、适应度函数和遗传算子的选择进行了理论分析。在系统理论和需求分析中讲解了一个Web漏洞分析系统的原理和设计方案，包括SQL注入、跨站脚本攻击等和网站应用级入侵防御系统，以及生成模糊测试字典创建的方案。接下来基于SQL漏洞、XSS漏洞和WAF安全机制分析如何挖掘Web漏洞，为系统设计和实验提供理论基础。在系统设计和实现章节详细介绍了结构模块的算法和代码，以及逻辑和实现过程。向量输入模块设计使用pycurl并行执行多线程处理，分析需要的返回码。利用pyparsing可以帮助简化正则表达式的语法处理，并使用遗传算法对字典进行多次迭代，直到达到预期的阈值。在实验测试阶段表明本系统在自动化运行、并行发包效率的高效性和在绕过WAF系统提过数据包过滤率的有效性。

**6.2 研究展望**

现代模糊测试技术与Web漏洞挖掘已经紧密结合，Google的OSS-Fuzz库[28]已经为很多模糊测试者提供丰富的库接口，FFUF[29]和V-Fuzz[30]等一系列模糊测试工具也在开发中，可以预见的是在模糊测试自动化程度越来越高的未来，自动化的收集信息、进行测试和数据分析将会慢慢替代部分枯燥的人工测试。而发展了很多年的遗传算法在面对优化随机解空间搜索问题时会一直都有不错的表现，遗传算法对模糊测试这类随机过程的优化会有更好的解答。本文在遗传算法、模糊测试和Web漏洞挖掘这个三个方向上做了一些尝试，并尝试将自动化的系统引入以帮助那些想在Web漏洞挖掘方向尝试的入门者。本文的缺点是利用遗传算法优化后的模糊测试字典绕过复杂的WAF和成熟的商业网站是比较困难的事情，简单的遗传算法模型和迭代次数很难应对周密的WAF系统。希望将来有学者可以就更细粒度的角度依据遗传算法或机器学习等知识优化包含用模糊测试进行Web漏洞挖掘等各类随机数据测试中取得理论上的突破。

**致 谢**

2018年的我第一次踏入这片校园的时候，将如何想象四年后的此时此刻的呢？我已经记不起来了。但是现在的我的确记得，大一的时候和队友第一次参加建模比赛的凌晨，也兴奋地加入LabVIEW社团接触单片机相关的知识，在IOS技术协会准备WWDC，在经济学院报告厅听浙大郑强教授的“科学精神与实践”讲座，报告厅里座无虚席，很多同学都是坐在过道上或站在窗户旁；大二的时候每天都骑车很远去西十二楼上课，在科技楼做完物理实验后和同学在夜色中走回韵苑，在百景园的24h自习中心尚未建成的时候，经常去集贸的麦当劳点份12元套餐然后自习一下午。那时没有疫情，可以到处去旅游，我没有忘记在解放碑和春熙路的故事，在玉龙雪山上和洱海湖畔的时光。疫情突然来临后，在家里和同学们线上开会，在Ubuntu16.04 LTS上搞大创和信息安全比赛，在宿舍里远程参加NUS的夏令营然后和同学们通宵完成论文，现在终于完成了毕业论文……回顾过往，在华中科技大学四年奋斗的岁月的确在我人生中留下了浓墨重彩的一笔。

感谢我的指导老师陈凯老师。老师在研究方向和技术理论上给了我很多帮助，根据我的研究进展情况给了我很多有用的建议。在论文选题，技术路线的选择和论文的撰写上给了我很多宝贵的意见，让我在茫茫的知识海洋中找到努力的方向，在此向陈凯老师致以最诚挚的感谢。

感谢李升辉老师、高鹏毅老师，认真督促我的毕设进度，每周开会根据我们的报告和展示提出了宝贵的建议，热情帮助我解决了在研究中遇到的技术问题，老师们认真的教学态度和对科研的求索精神让我敬佩。

感谢单成顶学长和赵睿同学，他们在论文写作和实验测试中遇到问题给了我很多帮助，他们对研究的理解绕我少走了很多弯路，也感谢我的朋友贾韵锋和胡浩然在毕设期间给我的帮助和鼓励。

感谢我的家人对我无限的支持，没有他们我无法在求学路上更进一步，这四年时光里他们给了我太多的鼓舞的理解，他们是我继续坚持奋斗的最大动力。

肖昂

2022年5月于华中科技大学

**参考文献**

[1] Measuring digital development Facts and figures 2021. https://www.un.org/sustainabledevelopment/blog/2021/12/as-internet-user-numbers-swell-due-to-pandemic-un-forum-discusses-measures-to-improve-safety-of-cyberspace/

[2] China National Vulnerability Database. <http://www.cnvd.org.cn/>.

[3] OWASP Foundation | Open Source Foundation for Application Security. https://owasp.org/

[4] Zhao J, Lu Y, Wang X, et al. WTA: a static taint analysis framework for PHP webshell[J]. Applied Sciences, 2021, 11(16): 7763.

[5] "RIPS - Wikipedia", En.wikipedia.org, 2021. <https://en.wikipedia.org/wiki/RIPS>.

[6] Widia I D M, Rosalin S, Asriningtias S R, et al. Black Box Testing Menggunakan Boundary Value Analysis dan Equivalence Partitioning pada Aplikasi Pengadaan Bahan Baku Batik dengan Pendekatan Use Case[J]. JIMP (Jurnal Informatika Merdeka Pasuruan), 2022, 6(1).

[7] Byun T, Rayadurgam S, Heimdahl M P E. Black-Box Testing of Deep Neural Networks[C]//2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE). IEEE, 2021: 309-320.

[8] Baldoni R, Coppa E, D’elia D C, et al. A survey of symbolic execution techniques[J]. ACM Computing Surveys (CSUR), 2018, 51(3): 1-39.

[9] Pak B S. Hybrid fuzz testing: Discovering software bugs via fuzzing and symbolic execution[J]. School of Computer Science Carnegie Mellon University, 2012.

[10] Böhme M, Pham V T, Nguyen M D, et al. Directed greybox fuzzing[C]//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017: 2329-2344.

[11] Directed Greybox Fuzzing with AFL. <https://github.com/aflgo/aflgo>

[12] Michalewicz, Zbigniew, and Marc Schoenauer. "Evolutionary algorithms for constrained parameter optimization problems." *Evolutionary computation* 4.1 (1996): 1-32.

[13] Kirkpatrick S, Gelatt Jr C D, Vecchi M P. Optimization by simulated annealing[J]. science, 1983, 220(4598): 671-680.

[14] Glover F. Heuristics for integer programming using surrogate constraints[J]. Decision sciences, 1977, 8(1): 156-166.

[15] Stepanov L V, Koltsov A S, Parinov A V, et al. Mathematical modeling method based on genetic algorithm and its applications[C]//Journal of Physics: Conference Series. IOP Publishing, 2019, 1203(1): 012082.

[16] Sun Y, Xue B, Zhang M, et al. Automatically designing CNN architectures using the genetic algorithm for image classification[J]. IEEE transactions on cybernetics, 2020, 50(9): 3840-3854.

[17] Mirjalili S, Song Dong J, Sadiq A S, et al. Genetic algorithm: Theory, literature review, and application in image reconstruction[J]. Nature-inspired optimizers, 2020: 69-85.

[18] Miller B P, Fredriksen L, So B. An empirical study of the reliability of UNIX utilities[J]. Communications of the ACM, 1990, 33(12): 32-44.

[19] P. Amini M. Sutton, A. Greene. Fuzzing: Brute Force Vulnerability Discovery. Addison-Wesley, Boston, MA, 2007.

[20] Barton P. Miller (September 1988). ["Fall 1988 CS736 Project List"](http://pages.cs.wisc.edu/~bart/fuzz/CS736-Projects-f1988.pdf) (PDF). Computer Sciences Department, University of Wisconsin-Madison. Retrieved 2020-12-30.

[21] Mousavirad S J, Moghadam M H, Saadatmand M, et al. RWS-L-SHADE: An Effective L-SHADE Algorithm Incorporation Roulette Wheel Selection Strategy for Numerical Optimisation[C]//International Conference on the Applications of Evolutionary Computation (Part of EvoStar). Springer, Cham, 2022: 255-268.

[22] Bian C, Qian C. Running Time Analysis of the Non-dominated Sorting Genetic Algorithm II (NSGA-II) using Binary or Stochastic Tournament Selection[J]. arXiv preprint arXiv:2203.11550, 2022.

[23] Shukla A, Pandey H M, Mehrotra D. Comparative review of selection techniques in genetic algorithm[C]//2015 international conference on futuristic trends on computational analysis and knowledge management (ABLAZE). IEEE, 2015: 515-519.

[24] Zhang C, Oh S K, Fu Z. Hierarchical polynomial-based fuzzy neural networks driven with the aid of hybrid network architecture and ranking-based neuron selection strategies[J]. Applied Soft Computing, 2021, 113: 107865.

[25] Syswerda G. Uniform crossover in genetic algorithms[C]//ICGA. 1989, 3: 2-9.

[26] “DOM Based Cross Site Scripting or XSS of the Third Kind” (WASC writeup), Amit Klein, July 2005 http://www.webappsec.org/projects/articles/071105.shtml

[27] Dermann M, Dziadzka M, Hemkemeier B, et al. Best practices: Use of web application firewalls[J]. The Open Web Security Application Project, OWASP Papers Program, 2008.

[28] Ding Z Y, Le Goues C. An Empirical Study of OSS-Fuzz Bugs[C]//2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR). IEEE, 2021: 131-142.

[29] Mattsson M. A comparison of FFUF and Wfuzz for fuzz testing web applications[J]. 2021.

[30] Li Y, Ji S, Lv C, et al. V-fuzz: Vulnerability-oriented evolutionary fuzzing[J]. arXiv preprint arXiv:1901.01142, 2019.