



中北大学
NORTH UNIVERSITY OF CHINA

毕业设计说明书

基于 Python 的 SQL 注入漏洞扫描器 的设计与实现

学生姓名：_____ 学号：_____

学 院：_____ 软件学院

专 业：_____ 软件工程

指导教师：_____

20** 年 06 月

基于 Python 的 SQL 注入漏洞扫描器的设计与实现

摘要

由于 Web 应用越来越普及，Web 应用中存在 SQL 注入漏洞而带来安全隐患的事情并不少见，为了检测 web 应用是否存在 SQL 注入漏洞，本文设计并开发了基于 Python 的 SQL 注入扫描器，用于检测 web 应用的 SQL 注入，SQL 注入扫描器向目标网站发送携带注入 BUG 的 HTTP 注入，通过分析返回的结果(如错误信息、响应时间等)检测漏洞。

支持 GET、POST 请求，支持布尔盲注、时间盲注等多种检测模式，支持多线程加快检测速度，支持命令行式简单界面设计，支持按需，按定义自选参数，按命令行式导出扫描结果。扫描器结果检测良好，对常见的 SQL 注入型漏洞可以较好的检测，误报率较低，成功检测后可以导出漏洞报告，还会提供相应修复漏洞的方案，用于渗透测试或安全审计。

关键词：漏洞扫描，自动化检测，SQL 注入

Design and Implementation of a Python-Based SQL Injection Vulnerability Scanner

Abstract

In order to detect whether there are SQL injection vulnerabilities in web applications, this paper designs and develops a Python-based SQL injection scanner to detect SQL injection in web applications, and the SQL injection scanner sends HTTP injections with injection bugs to the target website, and detects the vulnerabilities by analyzing the returned results (such as error messages, response time, etc.).

It supports GET and POST requests, supports Boolean blinds, time blinds and other detection modes, supports multi-threading to speed up detection, supports command-line simple interface design, supports on-demand, self-selected parameters according to definition, and exports scan results according to command-line. The scanner results are well detected, common SQL injection vulnerabilities can be well detected, the false positive rate is low, the vulnerability report can be exported after successful detection, and the corresponding vulnerability repair plan will be provided for penetration testing or security audit.

Keywords: vulnerability scanning, automated detection, SQL injection

目 录

1 引言	1
1.1 选题的背景与意义	2
1.1.1 选题的背景	2
1.1.2 选题的意义	2
1.2 国内外研究的现状	3
1.2.1 国内研究的现状	3
1.2.2 国外研究的现状	3
1.3 本研究的作用和特点	4
1.3.1 本研究的作用	4
1.3.2 本研究的创新	5
1.4 小结	6
2 相关技术研究	7
2.1 SQL 注入原理与分类	7
2.1.1 SQL 注入原理	7
2.1.2 SQL 注入分类	7
2.2 现有检测技术对比分析	10
2.2.1 静态分析	10
2.2.2 动态分析	11
2.2.3 混合分析	12
2.3 Python 在安全领域的应用优势	13
2.4 Web 应用架构与常见 SQL 注入点	15
2.4.1 不同 Web 应用架构中的数据交互流程	15
2.4.2 SQL 注入漏洞在不同架构中的产生根源	15
2.4.3 常见的 SQL 注入漏洞点场景	16
3 系统需求分析	17
3.1 功能需求	17

3.1.1	自动化扫描功能	17
3.1.2	漏洞检测功能	18
3.1.3	漏洞报告生成功能	19
3.1.4	用户界面功能	19
3.2	性能需求	20
3.2.1	扫描速度	20
3.2.2	资源占用	21
3.2.3	检测准确性	21
3.2.4	系统兼容性	21
3.3	安全需求	22
3.3.1	扫描器自身安全性	22
3.3.2	对目标系统的安全性影响	23
3.3.3	漏洞报告的安全性	23
3.3.4	合规性要求	24
4	系统设计	25
4.1	大体设计	25
4.2	扫描模式	25
4.3	核心功能	26
4.4	高级功能	27
4.5	支持库	27
5	系统实现	28
5.1	Python 环境搭建	28
5.2	核心模块实现	28
5.3	关键技术难点的解决方案	28
5.4	用户界面实现	29
5.4.1	Web 界面设计概述	29
5.4.2	WEB 页面预览	30

6	系统测试.....	31
6.1	测试环境搭建.....	31
6.2	功能测试用例设计.....	31
6.3	功能演示.....	32
6.3.1	SQL 注入检测.....	32
6.3.2	SQL 登陆表单测试.....	33
6.3.3	防御绕过策略.....	35
6.3.4	系统配置.....	36
6.3.5	基于漏洞类型检测.....	37
6.3.6	生成对应的漏洞报告.....	40
7	结论与展望.....	41
7.1	测试结论.....	41
7.2	改进建议.....	41
7.3	未来展望.....	42
	参 考 文 献.....	44
	致 谢.....	46

1 引言

现如今我们生活在数字化时代，无论是在生活上、工作上、娱乐上，我们都离不开互联网。Web 应用系统的兴起与发展更是让我们的种种隐私信息、商业信息、财务信息等等都存储在了 Web 应用系统的数据库中，而数据库作为信息系统的重要组成部分，其安全问题，更是影响信息系统，直接影响信息的安全问题。网络安全也面临着各种各样的考验与漏洞，其中最常见也是最具有威胁的便是 SQL 注入。

SQL 注入攻击一般是指攻击者利用 Web 应用程序输入字段的方法，在输入字段中插入一条 SQL 语句，绕过应用程序的输入字段检测，与数据库直接建立连接，完成恶意操作的一类攻击方法，通过上述手段，攻击者就能够直接窃取数据库中的内容，盗取数据库中的账号、密码、信用卡信息甚至篡改、删除数据库中的信息，造成数据库丢失、数据库系统崩溃等。这会给企业带来极大的损失，也会使企业名声尽失，客户信息尽失。

由于 SQL 注入的危害性巨大，所以必须及时发现并修补此类漏洞，但是传统的人工查找效率低、工作量大、容易出现疏漏，已经无法满足网络安全的快速发展，因此迫切需要一种准确高效的 SQL 注入漏洞扫描器。

Python 是一种简单且拥有丰富库函数的编程语言，适合于编写安全检测相关工具。使用 python 编写 SQL 注入漏洞扫描器，就是模拟注入者发送各种可能恶意的 SQL 代码给 Web 应用，根据系统的响应来判断是否存在漏洞。自动扫描可以快速地完成检查工作，节省人力，并且更加精确、全面。

还有，扫描的 SQL 注入工具是 python 的，可定制可扩展，开发人员随意扩展扫描功能、增加新的规则检测、加强算法、甚至加入其他安全工具。灵活与 python 开源强大社区使其工具本身开发与后期维护更简单。开发人员可借鉴开源社区先进经验，缩短开发周期。

对于公司企业来说，对于测试人员来说，正确、快速的 SQL 注入扫描器的诞生，是抵御网络攻击的利器，对于今后基于 python 的扫描器将会在防治复杂的 SQL 注入攻击中起着不可忽视的作用，在未来的科技发展中，成为保护网络的利器，保护我们的网络世界。

1.1 选题的背景与意义

1.1.1 选题的背景

当今社会已处于网络化时代，互联网已深入社会生活的各个领域，Web 应用系统随处可见，从社交网站到购物网站，从网络办公到金融网站等等，Web 应用系统背后都有一个庞大的数据库，存放着客户的信息、机密、财务等等，数据库是数据的心脏，是整个信息系统的数据存储、管理机构，它的安全与否，关乎着整个系统的运行安全和数据的完整保密性。

然而网络安全形势严峻，黑客、恶意攻击者不断利用漏洞获取不正当利益或进行破坏，SQL 注入攻击就是常用的高危害性破坏攻击。SQL 注入攻击，是指 SQL 注入对 web 应用程序处理用户输入过程中存在的安全漏洞进行利用，将恶意构造的 SQL 语句输入到输入检测和输入验证之间跳过应用程序本身直接与数据库直接进行交互，利用 SQL 注入漏洞获取存储于数据库中的用户账号、密码、信用卡号等信息进行身份窃取、金融诈骗等犯罪行为；篡改、删除数据库中数据，导致数据丢失、系统瘫痪等，给企业和用户带来巨大的经济和名誉损失。

近些年来，SQL 注入导致的安全事件屡有发生。大型的公司企业单位等都曾经出现过 SQL 注入问题，如大型的电商网站平台信息泄露，金融机构的交易信息被篡改等，都引起了不少的社会关注度，同时也说明 Web 应用程序加强安全防护势在必行。

1.1.2 选题的意义

当今社会已处于网络化时代，互联网已深入社会生活的各个领域，Web 应用系统随处可见，从社交网站到购物网站，从网络办公到金融网站等等，Web 应用系统背后都有一个庞大的数据库，存放着客户的信息、机密、财务等等，数据库是数据的心脏，是整个信息系统的数据存储、管理机构，它的安全与否，关乎着整个系统的运行安全和数据的完整保密性。

然而网络安全形势严峻，黑客、恶意攻击者不断利用漏洞获取不正当利益或进行破坏，SQL 注入攻击就是常用的高危害性破坏攻击。SQL 注入攻击，是指 SQL 注入对 web 应用程序处理用户输入过程中存在的安全漏洞进行利用，将恶意构造的 SQL 语句输入到输入检测和输入验证之间跳过应用程序本身直接与数据库直接进行交互，利用 SQL

注入漏洞获取存储于数据库中的用户账号、密码、信用卡号等信息进行身份窃取、金融诈骗等犯罪行为；篡改、删除数据库中数据，导致数据丢失、系统瘫痪等，给企业和用户带来巨大的经济和名誉损失。

近些年来，SQL 注入导致的安全事件屡有发生。大型的公司企业单位等都曾经出现过 SQL 注入问题，如大型的电商网站平台信息泄露，金融机构的交易信息被篡改等，都引起了不少的社会关注度，同时也说明 Web 应用程序加强安全防护势在必行。

1.2 国内外研究的现状

1.2.1 国内研究的现状

（1）开源工具活跃：国内开源社区涌现大量 python 编写的 SQL injection 漏洞扫描器扫描工具，十分活跃。SQLiv 为 python 开源工具，采用多线程扫描大量 URL，扫描方式丰富多样，多域扫描、定向扫描、反向扫描等，还可输出 json 结果。SQLiFinder 是针对 GET 注入的 SQL 漏洞，扫描效率高，基于智能爬虫、Pathwayback Machine URL 和巧妙设计的 SQL injection 的 payload，依赖简单^[1]。

（2）高校和科研机构深入研究：许多高校和科研机构针对 SQL 注入漏洞的检测技术开展了深入研究。一些研究聚焦于如何提高扫描器的检测准确性，通过改进算法和模型，结合机器学习等技术，让扫描器能够更精准地识别各种类型的 SQL 注入漏洞，包括盲注、时间延迟注入等较为隐蔽的漏洞类型。同时，也有研究致力于优化扫描器的性能，使其能够在大规模网络环境下快速完成扫描任务，满足企业和机构对 Web 应用安全检测的高效需求。

（3）企业应用与实践：随着网络安全形势日益严峻，国内企业也逐渐意识到网络安全的重要性，越来越多的企业开始利用 python 编写的 SQL 注入漏洞扫描器对 Web 程序进行安全测试。甚至一些互联网公司也开始推出了自己的扫描器并针对自己的业务进行二次开发，用于解决更为复杂的应用场景问题，并参与到了开源扫描器的维护建设中，共同推进开源扫描器的开发与应用。

1.2.2 国外研究的现状

（1）技术领先与创新：以 python 为开发语言编写的 SQL 注入漏洞扫描器，国外先进，不断将新的技术方法创新应用在扫描器中，提升扫描器扫描水平，有的扫描器利用

模糊化测试技术，自动生成复杂化的测试用例，对 Web 应用进行扫描，应用人工智能、机器学习的神经技术，使扫描器具备自学习、自适应自主学习能力，自动对不同应用、不同注入漏洞进行自适应调整检测策略检测，提升检测的准确率和效率。

(2) 商业化工具成熟：除此之外，国外同样存在许多商业的 SQL 注入漏洞扫描器，其扫描器功能强大，拥有良好的用户体验，不仅提供了图形化的界面，操作简单方便，让用户操作和使用更为简便直观，并且结果分析简单、清晰，同时支持多语言、多环境，可以完美兼容企业现有的安全管理平台，进行无缝接入，实现漏洞的自发现、自报告、自修复。其配套的技术团队非常专业，漏洞库和扫描检测算法在不断的更新，从而保证扫描器可以对新 SQL 注入漏洞进行检测^[2]。

1.3 本研究的作用和特点

1.3.1 本研究的作用

(1) 全面覆盖漏洞类型：根据本文对 LIGHTPayloadS, STANDARDPayloadS, DEEPPayloadS, ADVANCEDPayloadS, loginFORMPayloadS 等 SQL 注入的有效载荷分析、SQL 注入中布尔盲注、登录绕过、联合查询、时间盲注、文件读取、堆叠查询等有效载荷分析对系统有效漏洞全覆盖。

(2) 检测隐蔽漏洞：启动二阶注入检测，潜在隐蔽性二阶注入检查将注入标记和 URL 对应的字典存储在 CANARY_TOKENS 中。二阶注入比较隐蔽，攻击者将攻击代码存入系统，当某个动作触发后，执行恶意代码。这个功能可以检测这种潜在的攻击。

(3) 评估 WAF 防护能力：WAF 防火墙能力检测功能，通过 WAF_BYPASSYSTPayloadS、CRM_WAF_BYPASSYSTPayloadS 等 WAF 绕过载荷扫描测试 WAF 是否能够抵御 SQL 注入，帮助系统管理员定位目标 WAF 的漏洞。

(4) 合理并发控制：为保障扫描速度及可靠性设置合理大小的 (THREADpool_size=5) 线程池并发数，避免并发过大导致的保护，也避免给目标系统过多负担，并确保大扫描时扫描速度足够快及不易被发现与封解封禁。

(5) 请求参数优化：分别设置请求超时 (REQUEST Timeout) 参数为 30，请求间隔 (REQUEST Delay) 参数为 1.5，启用智能延迟 (SMART_DELAYed True)，请求超时可以防止长时间排队，请求间隔可以防止快速连续请求对速率的要求，智能延迟可以

自适应调整请求间隔时间进行扫描，MAX_RESETS (maximal REpeats) 防止了由于网络波动等原因导致的请求失败都可以再次请求的情况下的扫描的正确性。

(6) 安全防护方面：如果存在 SQL 注入漏洞，开发人员、系统管理员可根据扫描结果进行系统漏洞修补，避免系统被用来攻击系统敏感信息、恶意破坏系统而导致系统安全与完整性缺失。定期进行漏洞扫描有助于开发人员、系统管理员增强对漏洞的重视程度，使其在系统开发维护的过程中，更加注意系统的安全问题。

1.3.2 本研究的创新

为了提高扫描器的检测成功率，我们引入了大比例的绕过技术，在国防部类目 DefenseBypassStrategy 下有很多绕过，如：注释、随机大小写绕过、URL 编码、分键码绕过、空格、十六进制、空位、超长 UTF-8 码、多重码、追加换行符、Unicode 规范化、HTML 转义码等绕过技术，这些绕过可以帮助我们绕过目标系统的可能存在的检测，提升检测成功率。

(1) 针对性的 WAF 绕过：我们不仅提供通用 WAF 绕过载(WAF_BYPASSYSTLOADS)，还提供定制不同系统 WAF_WAF_BYPASSYSTLOADS(CRM_WAF_BYPASSYSTLOADS)，定制让扫描器对不同系统使用不同的 WAF 值，定制让扫描器适应更广阔的环境和更有效的扫描。

(2) 智能延迟策略：扫描器智能延迟策略主要是为了避免造成目标系统因为过多的请求而出现过度疲劳、被系统识别的问题。智能延迟扫描策略可以根据目标系统的响应情况、当前的扫描进度情况，对扫描过程进行智能的调整。扫描进度可调节、扫描效率高，扫描器智能延迟策略可以有效地防止目标系统被扫描、被侦查。

(3) 用户代理轮换：扫描机还存在用户代理轮转：定义多个用户代理 (users AGOINT)。扫描机随机请求其中一个用户代理，给它发送一个 request 对应请求，模拟不同浏览器请求或者不同设备请求。这样可以避开被用户代理发现被扫描机扫描，增加隐蔽性。

(4) 多层次有效载荷集：应对不同的复杂 SQL 注入，提供基础有效载荷 (LIGHT PayloadS)、高阶复杂有效载荷 (ADVANCEDPayloadS)、常见 SQL 注入、不常见 SQL 注入等不同复杂程度的多类型有效载荷集，供用户灵活选择，从而提高扫描器的扩展

性及完备性。

(5) 登录表单专用负载：最后针对登录表单中的 SQL 注入点进行设计，我们设计了专用的 `LOAD_FORMPayloadS`，此载荷集包含了用户名和密码两个不同字段、不同注入、不同组合的测试的优化载荷，对于表单中的漏洞有较好的发现效果。

1.4 小结

数字时代 web 应用程序的使用，由于 web 应用程序的数据库存储着许多敏感的应用数据库，数据库里存储着很多敏感信息，SQL 注入攻击非常普遍，严重威胁信息系统安全，python 编写 SQL 注入漏洞扫描器，对计算机信息系统安全有重要理论和实际意义。

国内相关开源工具正在蓬勃发展，高校研究机构在探索，企业在实践和摸索，国外已经成熟商用，开源社区合作热闹非凡。

本项研究也发挥了重要作用，大量有效负载覆盖各种类型的漏洞、检测深埋漏洞，测试 WAF 的保级，通过并发合理化请求、参数修改、设置最大重试机制等策略来提升扫描的鲁棒性对扫描器进行安全保护，树立安全意识。创新之处，首次将绕过集成在一个研究中，设计 WAF 绕过负载通过智能延期策略，轮换 User-Agent 构建多层有效负载集，通过构建定制登录表单。创新点提升扫描器的有效性、适用性，鲁棒性、隐蔽性、灵活性。

2 相关技术研究

2.1 SQL 注入原理与分类

2.1.1 SQL 注入原理

(1) Web 应用与数据库交互基础

在通常的 web 应用程序的工作流程中，用户通过浏览器输入数据，提交表单，web 服务器端接收到数据后将进行数据处理，应用程序根据业务逻辑构建出 SQL 语句发送到数据库服务器，数据库端执行 SQL 语句并将处理结果返回给 web 服务器，web 服务器将处理结果返回给用户。举例一个简单的用户登陆功能，应用程序将用户输入的用户名和密码构建出如下 SQL 语句：`SELECT*FROM users WHERE username=<输入用户名'AND password=A 输入密码'`该语句会在数据库中查找是否有和用户输入一致的用户名和密码，如果有则用户登陆成功，否则失败。

(2) SQL 注入产生的根源

在通常的 web 应用程序的工作流程中，用户通过浏览器输入数据，提交表单，web 服务器端接收到数据后将进行数据处理，应用程序根据业务逻辑构建出 SQL 语句发送到数据库服务器，数据库端执行 SQL 语句并将处理结果返回给 web 服务器，web 服务器将处理结果返回给用户。举例一个简单的用户登陆功能，应用程序将用户输入的用户名和密码构建出如下 SQL 语句：`SELECT*FROM users WHERE username=<输入用户名'AND password=A 输入密码'`该语句会在数据库中查找是否有和用户输入一致的用户名和密码，如果有则用户登陆成功，否则失败。

这种攻击原理就是由于对用户输入的不当安全处置，导致恶意用户修改 SQL，从而引发数据泄漏、修改或其他不良后果。

2.1.2 SQL 注入分类

SQL 注入漏洞产生于应用程序对用户输入未做严格过滤和处理时，通过构造恶意输入内容，改变原有 SQL 语句语义，进行未授权操作。不同的 SQL 注入利用数据库查询特性，通过不一样的途径实现敏感数据泄露、数据改动或其它恶意行为。

(1) 基于注入方式分类

联合查询注入(Union-based Injection): 对于联合查询注入, 因为使用了 UNION 运算符, 因此将查询结果与原有查询结果联合获取数据库中其他表中的结果。若应用程序对用户的输入不进行检测, 攻击者可以使用如“UNION SELECT column1column2FROM other_table 获得与原有查询结果不相关的结果, 攻击者可能获取用户姓名、密码等。

报错注入(in errorbased injection): 报错注入是指利用数据库执行报错 SQL 语句后返回的信息。不同数据库返回错误信息不同, 攻击者可以通过引发错误返回不同的错误信息, 从而观察错误信息来推测出数据库结构。比如 MySQL 数据库执行类型不匹配错误返回数据库版本、返回表名、列名信息, 攻击者可以通过构造查询让数据库返回错误信息, 从而获取数据库的可能信息。

盲注 (Blind Injection): 盲注是指应用程序没有返回任何实际的查询结果, 攻击者通过观察页面其他线索 (如返回、正常等等) 来判断是否成功注入, 根据所得到线索间接推断数据库信息。在布尔盲注中, 攻击者通过构造真或假的条件 (1AND1=1) 来判断是否存在注入, 在时间盲注中, 攻击者通过构造数据库的延时 (SLEEP 函数) 来判断是否成功进行了查询。攻击者通过以上方法推断数据库结构或者敏感信息。

Width-byte injection: Width-byte injection:Width-byte injection 是指由于多字节字符集中的多字节字符不能通过输入校验转义而引起的 SQL 注入, 例如 addslashes 函数转义, 在某些情况下对部分字符不能转义, 而有些字符的组合 addslashes 不能转义。攻击者可以构造%dfOR1=1--等绕过输入有效性安全过滤器进行 SQL 注入。

堆叠查询注入 (Stacked QueryInjection Injection): 堆叠注入即一次数据库的 SQL 语句查询可以执行多个 SQL 语句, 攻击者就会利用这个注入执行多个语句。攻击者可以在原始的数据库查询之后执行一些危险的语句, 如: Drop table、Insert into 等。例如: Attacker:drop table users;--删除数据库当中的 users 表格。

内联标注注入 (Inline Comment 注入): 内联标注注入是数据库对标注处理属性进行的简单输入过滤如 MySQL①50000 是一个固定模式的注释其内容会导致在 MySQL 的某些版本中执行的 SQL 语句不被过滤清理掉黑客通过在被注入的语句后加入一个注释语句绕过程序应用过滤器执行恶意 SQL 语句 1①50000UNION SELECTUser PasswordFromAdmin_table*/窃取敏感信息。

这些 SQL 注入都是基于数据库特性的，描述了攻击者在各种方法中绕过应用程序安全并访问、修改和删除数据库，而无需任何许可以外的行为。避免 SQL 注入的方法是使用参数化查询，输入检查和过滤器等。开发人员应该防止应用程序被 SQL 注入，如：安全参数化查询、安全检查和过滤器等。

（2）基于攻击目标分类

数据窃取型 SQL 注入：数据窃取型 SQL 注入顾名思义就是通过注入 SQL 语句从而窃取敏感数据。利用注入语句、联合查询语句或其他注入语句来获取用户信息、商业秘密和机密信息等，例如获取用户信息，包括姓名、身份证号码、银行卡号等，或者获取一些商业机密或内部信息等。该类 SQL 不需要更改数据库的数据，只需要从表中获取信息，例如数据库中存储了用户的个人信息的表中，需要获得数据库内所有用户的账号和密码信息，以便利用 UNION SELECT 注入语句来获得数据库内所有用户的账号和密码，以便于以后的盗用身份及其他犯罪行为。

数据篡改型 SQL 注入：数据篡改型 SQL 注入即通过注入 SQL 语句来修改数据库中数据，改变软件的正常工作。攻击者注入 SQL 语句，通过 Update，DELETE，INSERT 等 SQL 语句对数据库中的数据进行更改、删除和插入，可能影响整个业务流程。比如网店购物流程中，攻击者通过 SQL 注入的方式将订单金额字段通过 SQL 注入修改为自己获取不正当利益，或者攻击者删除竞争对手的订单数据，破坏正常的竞争，给企业造成经济损失。

权限提升 SQL 注入：权限提升 SQL 注入是指不同权限的用户通过 SQL 注入语句获得较高权限，从而可以操控整个系统。这种 SQL 入侵存在于不同权限的系统用户中，通过 SQL 注入语句对用户权限字段进行修改，从而将普通用户的权限提升为系统管理员权限，例如注入更新语句将自己的账号权限改为管理员权限，就可以完全控制这一系统，使用 SQL 注入语句让攻击者执行不可能完成的操作，比如删除用户数据，读取敏感数据，修改系统配置，等等。

这些 SQL 注入的攻击模型显示了攻击者通过这些注入来盗取数据、篡改数据，提升权限等方式对数据库及应用系统进行极大的攻击。通过检查输入、参数化查询、最小权限、数据加密等可以防止这些注入攻击。

2.2 现有检测技术对比分析

2.2.1 静态分析

(1) 原理

词法分析：将源代码或二进制代码看作字符流，并依编程语言的词法分析规则分析出一大单词（key、标识、常量等）。SQL 注入检测的词法分析就是找出 SQL 关键字或者 SQL 特殊字符等可能跟 SQL 注入相关的单词，比如找出关键字的 UNION、SELECT、OR 关键字的位置或频率等。

语法分析：语法分析是在词法分析后将程序语言按照语法规则，将一个个单词组织成语法树的过程，其中最核心的就是判断该段代码是否存在 SQL 语句结构，SQL 语句是否正确等，例如组织 SQL 语句的方法，如何将参数传递等。SQL 注入检查：语法分析有助于检查是否含有 SQL 注入问题，如将用户未经过滤直接拼接到了 SQL 查询中。

语义分析：语义分析就是语义分析，分析代码运行中的变量类型、变量的作用域、变量的流等。通过分析代码中数据流，静态分析用户输入是否没有做正确校验就直接拼接到 SQL 语句中去了，如用户输入没做正确转义或参数化，直接拼接到 SQL 语句中的 SQL 注入等等。

(2) 优点

全面性：静态扫描可以对代码实现全覆盖扫描，扫描整个程序的应用程序路径，包括正常情况下程序运行不会经过的代码、程序运行过程中异常的代码，让静态扫描可以找到正常情况下不容易触发的 SQL 注入点。

早期检测：静态分析早期静态分析通常是在研发过程中，如编码期间，代码检查期间，或者程序编译期间。漏洞的发现成本很低，不需要对软件系统进行大的改动或更新，不需要对最终用户进行更改。

不用运行的环境：静态分析不需要程序运行的环境（数据库、服务器等），它只依赖于源代码或者二进制代码，可以在任何一个地方（平台或者环境）进行运行，所以静态分析可以用于任何类型的应用程序，包括 web 应用程序、桌面应用程序甚至移动应用程序。

（3）缺点

误报率较高：静态分析是基于代码结构进行分析，而非根据运行状态，故容易对正常代码结构产生误报，例如：存在静态 SQL 构造方法，或真运行时输入经过严苛过滤，而静态分析无法识别，存在误报。

不能处理复杂逻辑：现代程序复杂的逻辑与语句动态结构 SQL 语句静态分析很难分析处理，而且，SQL 语句是函数调用或者复杂语句构成结构语句构造出来的，静态难以分析追踪数据流转，数据处理过程，存在 SQL 注入。

依赖代码质量：静态分析工具依赖于源代码的质量。不规范代码、混乱代码、无注释代码，可能直接导致静态分析工具无法理解代码，代码质量差，可能造成静态分析的准确率和完整率下降，导致部分 SQL 注入代码无法被找到或报错率更高。

2.2.2 动态分析

（1）原理

网络流量监视：动态监视网络接口上的数据包以获取应用程序和数据库之间的通信信息，主要包括监视 HTTP 请求或其他网络协议数据包，检查其中的请求参数和 URL 等，通过监视其中是否含有 SQL 关键字、字符或含有特殊值的参数，来检查是否含有 SQL 注入，例如，检查是否含有不规范的、含有 SQL 构造的恶意输入数据包发往数据库中。

数据库交互分析：动态分析还可以通过数据库代理或者数据库拦截，对应用程序到数据库的交互进行分析，通过给数据库发送的 SQL 语句及数据库的响应，发现 SQL 注入，通过这种方法可以发现是否存在未经授权的访问，是否存在 SQL 语句异常，SQL 语句是否是业务逻辑错误。

运行时环境检查：运行时环境查看（运行日志、运行服务器信息等），是否存在 SQL 注入的异常情况，数据库是否存在连接异常、查询执行时间异常，异常查询信息等，可能是攻击者利用 SQL 注入进行入侵的位置。

（2）优点

准确性高：动态分析是在程序的实际运行环境检测，能够看到真实的用户输入和系统响应，通过对数据库交互以及错误反馈的结果分析，能够有效判定出真正 SQL 注入

攻击，减少误报，检测结果可靠性高。

能检测动态语句：动态语句适用于动态检测动态运行时 SQL 语句，需要运行时上下文信息的语句。动态检查应用在运行时动态生成 SQL 语句，对静态运行时上下文信息无法检测的 SQL 语句注入漏洞。尤其适用于静态分析无法检测的应用在运行时动态生成的 SQL 语句。针对用户输入或配置文件动态生成 SQL 语句的场景。

不用查看源码：动态分析关注软件的输入输出以及运行时行为，不用查看软件的源代码，因而可以用于动态分析不能查看源码的第三方软件、黑盒软件或者其它复杂的应用，通过监控外部行为(网络流量、数据库操作等)就可以发现 SQL 注入漏洞。

(3) 缺点

覆盖面有限：由于动态分析时只能捕获到测试过程中实际发生的漏洞，而针对特定的用户角色、业务过程或输入的组合而构造的注入 SQL 可能并未在测试过程中发生，即漏测了某些边缘情况或罕见情况。由于某些边缘情况或罕见情况可能并未

难以检测复杂攻击有些复杂的 SQL 注入式攻击可能需要多请求串联、数据库特性、特殊上下文信息等条件才能完成，而动态分析往往是基于单请求、短时行为，容易造成没有捕获到攻击全貌导致无法发现其内在逻辑的问题。

需要运行环境：对需要运行的应用系统环境，进行建设和维护，比如：服务器、数据库、网络等，对检测成本和难度提升。同时对系统性能产生影响，为保证准确，需在近似生产环境检测环境下测试，对环境建设和维护的要求^[3]。

2.2.3 混合分析

(1) 原理

静态分析阶段：静态分析阶段首先采用静态分析对应用中的源码、二进制进行扫描，采用词法分析、语法分析、语义分析等技术，分析出有 SQL 注入风险点，可能为未做校验的用户输入码，可能为用于动态构造 SQL 语句的语句位置，可能为可能泄露信息的函数调用位置，分析器标记出危险点作为动态分析的观察点。

动态分析阶段：动态分析程序的实时执行过程静态分析确定的可能性风险点，动态分析是监视网络请求、数据库交互、执行情况等实时分析可能性风险点位的 SQL 注入漏洞，利用动态分析对静态分析确定的用户输入点位等进行监视，实时分析是否存在 S

QL 注入点、是否出现非预期的数据库操作、数据泄露。

结果整合与分析：将静态和动态分析的结果进行整合，在静态分析中有风险点标记为静态分析，在动态分析中有 SQL 注入漏洞或异常的可以认定为有漏洞，否则，修改静态分析规则或增加动态测试用例，确保测试结果的准确性。

（2）优点

静态分析阶段：静态分析阶段首先采用静态分析对应用中的源码、二进制进行扫描，采用词法分析、语法分析、语义分析等技术，分析出有 SQL 注入风险点，可能为未做校验的用户输入码，可能为用于动态构造 SQL 语句的语句位置，可能为可能泄露信息的函数调用位置，分析器标记出危险点作为动态分析的观察点。

动态分析阶段：动态分析程序的实时执行过程静态分析确定的可能性风险点，动态分析是监视网络请求、数据库交互、执行情况等实时分析可能性风险点位的 SQL 注入漏洞，利用动态分析对静态分析确定的用户输入点位等进行监视，实时分析是否存在 SQL 注入点、是否出现非预期的数据库操作、数据泄露。

结果整合与分析：将静态和动态分析的结果进行整合，在静态分析中有风险点标记为静态分析，在动态分析中有 SQL 注入漏洞或异常的可以认定为有漏洞，否则，修改静态分析规则或增加动态测试用例，确保测试结果的准确性。

（3）缺点

复杂程度高：由于混合分析采用静分析、动分析和整合分析相结合的方式，对分析工具配合要求高、对分析结果要求高。检测难度高、对人员要求高。

性能开销大：混合分析静态分析与动态分析同时进行，会占用较多的系统资源和时槽，比如静态分析同时要分析大量的代码，同时占用时间和内存。动态分析同时要动态执行应用程序，对系统性能产生损耗，比如增加应用响应时间、资源使用率等，对一些性能要求较高的软件系统，混合分析会导致较大的性能开销。

2.3 Python 在安全领域的应用优势

在信息化时代，网络安全显得尤为重要，Python 语言在安全领域的使用越来越广泛，Python 在安全领域中的运用，特别是在扫描器方面的实现有着不可替代的作用。通过上面在工具中的运用可以总结 Python 在工具中的使用有：工具简单，高效、工具拓展性

强等特点。

(1) 简洁高效的语法: Python 的语法简洁直观, 使得开发人员能够用较少的代码实现复杂的功能。特别是在开发网络请求模块时, Python 的 requests 库允许以极简的代码发送 HTTP 请求并获取响应, 从而减少了开发时的错误机会, 并加速了开发进程。相较于其他编程语言, 开发者可以把精力更多集中在核心逻辑和漏洞检测算法上, 而不是在复杂的语法结构上浪费时间。

(2) 丰富的库和模块: Python 拥有丰富的标准库和第三方库, 极大地提高了漏洞扫描器开发的便利性。例如, paramiko 库可以实现对远程服务器的 SSH 连接, scapy 则能够操作网络数据包, 帮助识别网络层面的漏洞。这些库提供了现成的工具, 使得开发人员可以快速集成所需功能, 构建一个全面而专业的漏洞扫描器。

(3) 跨平台兼容性: Python 天生具有跨平台特性, 使得基于 Python 开发的漏洞扫描器能够在多个操作系统上运行, 如 Windows、Linux、macOS 等。这使得同一套扫描器代码可以适用于不同的系统环境, 减少了平台适配的开发工作量, 提高了扫描器的适用性和维护效率。

(4) 强大的数据分析能力: 漏洞扫描过程中会生成大量的数据, Python 提供了如 pandas 和 numpy 等强大的数据分析工具。借助这些库, 开发者可以高效地处理和分析扫描数据, 生成可视化报告, 帮助安全人员更直观地了解目标系统的安全状况并及时识别潜在的风险点。

(5) 易于集成与扩展: Python 很容易与其他技术和工具进行集成。例如, 可以将漏洞扫描器与现有的安全管理系统或日志分析工具结合, 增强系统的综合安全性。还有, 随着新漏洞的不断出现, Python 也支持快速添加新的检测模块或算法, 从而保证扫描器能够不断跟进最新的安全威胁。

(6) 良好的代码维护性: Python 代码遵循严格的缩进规范, 确保了代码结构清晰。对于长期开发和维护的漏洞扫描器, 就算是不同的开发人员接手, 也能快速理解代码逻辑并进行修改。这不仅减少了维护时间和成本, 也确保了工具在长期使用中能够持续稳定运行, 并适应新的安全需求。

(7) 庞大的社区支持: Python 拥有一个庞大且活跃的开发社区。在遇到技术难

题时，开发者可以轻松通过社区获得帮助，查阅代码示例，获取解决方案。无论是针对新型漏洞的检测方法，还是优化扫描器的建议，社区的资源都极大地推动了漏洞扫描器的开发和进步。

（8）在信息化时代，网络安全显得尤为重要，Python 语言在安全领域的使用越来越广泛，Python 在安全领域中的运用，特别是在扫描器方面的实现有着不可替代的作用。通过上面在工具中的运用可以总结 Python 在工具中的使用有：工具简单，高效、工具拓展性强等特点^[4]。

2.4 Web 应用架构与常见 SQL 注入点

2.4.1 不同 Web 应用架构中的数据交互流程

（1）传统三层架构：传统的 web 应用三层架构大概为表现层、业务逻辑层、数据访问层，浏览器发出请求，到业务逻辑层，再到数据访问层，与数据库交互。如用户登陆，用户输入用户名密码，业务逻辑验证用户名密码，如果输入正确则调用数据访问层，读取数据库内符合要求的用户。python 的 web 框架常用 Django，业务逻辑常用视图函数，数据访问使用 orm（Objectrelations Mapping，对象关系映射）。

（2）微服务架构：微服务架构是指将一个应用构建为多个微服务，每个微服务就是一个功能，每个微服务之间通过轻量级的通讯方式进行通讯，例如 RESTful API。比如一个电商系统中有商品服务，订单服务、用户想要查询商品信息，那么商品服务就会去查询用户商品信息并且可能调用其他的微服务。在微服务架构中，Python 可以使用 FastAPI 等快速构建 API 接口，进行数据交互^[5]。

2.4.2 SQL 注入漏洞在不同架构中的产生根源

（1）传统三层架构中的 SQL 注入风险：在传统三层架构中，如果业务逻辑层缺乏对用户输入进行验证，数据访问层通过拼接 SQL 语句后与数据库进行交互，则存在 SQL 注入漏洞。例如在用户的注册业务中，未对用户的输入进行验证，将用户的注册信息中的用户名、密码传入数据访问层，以拼接 SQL 语句的方式与数据库交互，攻击者可以通过用户名、密码进行恶意注入来执行恶意 SQL 语句。

（2）微服务架构中的 SQL 注入隐患：传统三层架构下业务逻辑层未充分验证用户传入输入值下，在数据访问层采用拼接 SQL 语句方式对数据库访问，容易导致 SQL

注入漏洞，用户在注册时，业务逻辑层未对用户传入值进行充分验证，将用户传入密码和用户名传递至数据访问层，数据访问层采用拼接 SQL 变语句访问对数据库进行访问，攻击者通过用户名+口令绕过验证或恶意输入 SQL。

2.4.3 常见的 SQL 注入漏洞点场景

(1) 登录与注册模块：传统三层架构下业务逻辑层未充分验证用户传入输入值下，在数据访问层采用拼接 SQL 语句方式对数据库访问，容易导致 SQL 注入漏洞，用户在注册时，业务逻辑层未对用户传入值进行充分验证，将用户传入密码和用户名传递至数据访问层，数据访问层采用拼接 SQL 变语句访问对数据库进行访问，攻击者通过用户名+口令绕过验证或恶意输入 SQL。

(2) 搜索功能模块：Web 应用提供的搜索引擎容易存在 SQL 虚点，Web 应用将用户输入的搜索语句直接拼接在 SQL 查询语句上，攻击者将特殊构造的字符串拼接在非法构造的 SQL 语句上，实现信息获取。例：应用中提供的新闻网站搜索引擎，由于拼接不规范，攻击者将非法构造的字符串拼接在检索结果上，返回全部的新闻信息。

(3) URL 参数传递：Web 应用使用 URL 传递参数时，若对传入的参数未进行有效过滤，就可能导致 SQL 注入攻击。例如，若应用通过 URL 参数传递用户 ID，且未对该 ID 参数进行安全检查，攻击者可以修改 URL，注入恶意 SQL 语句，导致查询返回所有用户的信息，甚至执行其他危险操作。

评论模块：评论功能的存在导致 SQL 注入存在可能，由于用户在提交评论时若不采用拼接的写法插入评论，攻击者则可以通过向评论中注入恶意的 SQL 语句执行对 SQL 的攻击，如删除其中存储的数据，使之失效或者崩溃。

了解这些常见的 SQL 注入漏洞情景可以帮助开发人员构建更好的防御措施来构建 Web 应用程序。通过编写 Python 语言开发 Web 应用程序可以利用 Django 等框架提供的安全措施对 Web 应用程序的输入进行严格的审查和筛选，降低 SQL 注入的风险，构建更加安全的 Web 应用程序。

3 系统需求分析

3.1 功能需求

3.1.1 自动化扫描功能

(1) 输入方式多样性: 扫描器支持多种输入 URL 的方式, 可以在图形界面中通过文本框输入 URL, 也可以通过上传文本文件的方式将含有多个 URL 的文本文件输入, 扫描文本文件支持一般的 txt 文件。系统提供对输入的 URL 进行合法性校验的功能, 提示用户输入的 URL 不符合 URL 的格式(缺少协议头、域名不合法)等。

(2) URL 解析深度: 扫描器支持多种输入 URL 的方式, 可以在图形界面中通过文本框输入 URL, 也可以通过上传文本文件的方式将含有多个 URL 的文本文件输入, 扫描文本文件支持一般的 txt 文件。系统提供对输入的 URL 进行合法性校验的功能, 提示用户输入的 URL 不符合 URL 的格式(缺少协议头、域名不合法)等。

(3) 请求构建灵活性: 扫描器应当支持构造灵活的 HTTP 请求, 对合理范围内的 web 应用进行合理扫描, 针对 GET 支持对 GET 请求中的查询参数进行操作, 对合理的请求添加或修改 GET 请求, 针对 POST 表单编码为 application/x. www-form-urlencoded 以及多路(form-map)的 POST 数据进行正确模拟表单提交, 支持用户自定义 refer、Cookie 等, 模拟真实场景下用户的真实要求。

(4) 响应处理全面性: 扫描器应当支持构造灵活的 HTTP 请求, 对合理范围内的 web 应用进行合理扫描, 针对 GET 支持对 GET 请求中的查询参数进行操作, 对合理的请求添加或修改 GET 请求, 针对 POST 表单编码为 application/x. www-form-urlencoded 以及多路(form-map)的 POST 数据进行正确模拟表单提交, 支持用户自定义 refer、Cookie 等, 模拟真实场景下用户的真实要求。

(5) 并发控制策略: 扫描器要控制并发请求的数量, 不宜过度增加扫描效率而给目标 Web 应用带来额外的工作量, 使用人员可以根据目标系统的承载能力来设置并发数量, 系统会根据系统 URL 数量、性能合理控制并发请求数量, 避免并发请求数量过大导致服务器宕机、触发安全防御程序, 扫描器需具备检测并发请求状态, 对无效请求(过期、连接失败)自动放弃并重新发起请求。

(6) 任务优先级管理：扫描器支持对于扫描批量任务进行优先级管理，根据用户的需求对于不同的 URL 和任务按照其重要程度和紧急程度分配不同的优先级，按照任务优先级执行扫描任务，确保重要目标优先执行扫描检查。

3.1.2 漏洞检测功能

(1) 特征匹配算法优化：基于特征匹配的检测算法，是检测 SQL 注入的第一步，扫描器构建 SQL 注入常用特征库，用于检测常用注入关键字、注入语句、注入方式。扫描器对请求参数，进行对比，如果匹配特征，则报检测结果。将特征匹配算法，优化为模糊匹配、正则表达式匹配，来提升变形、混淆注入语句的检测率。

(2) 语义分析技术应用：语义分析技术可以对请求和应答进行语义解释，通过分析 SQL 语句中的语句结构，数据类型、操作语义等，判定 SQL 操作是否存在异常，如扫描器检查中不合理的条件判定、不合理的类型转换等。语义分析技术基于上下文的语义语境，可以提高检测的准确性，避免生成误报^[6]。

(3) 动态测试策略设计：动态测试是 SQLLintner 不可或缺的环节，扫描器需要设计合理的动态测试方案，依据动态测试中的不同场景，设计不同的动态测试用例用于不同数据库中，例如针对布尔的盲注，扫描器设计布尔条件语句，观察不同响应结果。针对时间盲注，扫描器控制 SQL 执行时间，观察不同响应时间的差异，测试用例应优化，减少多余的测试^[7]。

(4) 数据库特定检测规则：每个数据库的 SQL 语句都不相同，例如：MySQL、Oracle、SQL Server、PostgreSQL 等不同的数据库扫描器选择的检测原理都不相同，如：使用 UNION 联合查询注入对 MySQL 数据库；使用特殊语法 SQL 语句（selective）SQL 语句对 Oracle 数据库执行 SQL 语句。扫描器需要正确识别不同并检测到。

(5) 数据库指纹识别：扫描器扫描需要有数据库指纹识别功能，通过 web 返回的错误信息、web 响应信息等识别出目标的应用运行在什么数据库中，选择不同的规则、设置不同的测试用例，提高检测的准确性^[8]。

(6) 隐藏字段和 Cookie 检测：除了常见的 URL 和表单输入框外，还需要扫描器对一些隐藏字段和 Cookie 值进行扫描，隐藏字段中可能包含了敏感字段、控制业务流程字段，攻击者利用隐藏字段注入来绕过安全检查拦截，扫描器对字段进行测试，看

该字段是否存在注入标志或用于数据库操作，扫描器自动抓取隐藏字段和 Cookie 值。

(7) HTTP 头自定义字段检测：随着 Web 应用的日益增多，某些 Web 应用会使用自定义的 HTTP 头字段封装其它信息，扫描器会通过构造带有注入语句的自定义 HTTP 头字段发送给 Web 应用，通过检测 Web 应答来检查自定义 HTTP 头是否存在 SQL 注入^[9]。

3.1.3 漏洞报告生成功能

(1) 精确的 URL 和参数定位：扫描器定位到 SQL 注入漏洞后，应该将漏洞发生的 URL 和参数准确地报告出来，如果是同一个 url 的多个参数存在注入，那么应该报告具体哪个参数存在注入问题，同时将这个 url 的完整上下文 context，包括但不限于 request method、header 等等报告出来，便于开发人员进行复现修复。

(2) 页面元素关联：页面元素相关：当漏洞与页面中元素相关时（如表单、链接等），扫描器需关联页面元素的信息，如元素 ID、元素名称、元素位置等，扫描器根据 HTML 页面的结构，定位到扫描出问题的页面元素，可以为开发人员提供更直观易理解的漏洞定位信息。

(3) 详细的类型分类和解释：扫描器检查的 SQL injection 类型分类与说明：除以上常见的注入类型：布尔盲注，时间盲注，联合查询注入，报错注入等等常见类型之外，还要包括每种类型的描述，入侵手段，利用技巧，例如：布尔盲注描述：构造的布尔盲条件语句通过返回的结果来获取数据库信息。

3.1.4 用户界面功能

(1) 直观的布局设计：用户界面设计的目的是要让用户方便操作。用户的主要操作按键和输入框的位置一定要在屏幕显眼的位置，例如“开始扫描”“查看报告”等按键以及目标 URL 输入框、设置框等输入框的位置，这些位置要放在显眼方便寻找的位置，分好类组，如“扫描设置”和“报告查看”为一类，层次分明，操作流畅。

(2) 操作提示和引导：对用户使用进行提示、引导，如输入 URL 地址时提供 URL 的格式提示；扫描参数的设置提供参数说明、解释；对一些复杂的操作，提供操作说明、视频，以方便用户对扫描器的理解和操作。

(3) 多维度进度信息：扫描过程中，用户界面应该随时显示多维度进度信息，扫

描信息除了显示扫描已完成 URL 的数量和已检进度百分比外，还需要展示扫描速度(如:扫描请求数 / 秒)、剩余时间等信息。将扫描进度用图表或者数字的形式直观地展现给用户，让用户随时掌握扫描进度。

(4) 异常情况提示：扫描过程中如果出现异常（扫描请求超时、连接失败、返回目标失败等）的提示，界面应及时提示，并给出处理建议，如提示扫描用户检查网络连接情况或修改扫描参数，尽快解决异常问题，使扫描顺利完成。

(5) 详细报告查看：每一个漏洞，均可根据表中的按钮，查看漏洞的具体汇报。具体汇报中包括漏洞的具体描述、攻击样例，和漏洞的修补方法等，以文本+代码样例的形式展示，方便用户理解漏洞，修补漏洞。

(6) 详细日报查看：扫描器应详细记录每次扫描的日志，包括扫描 URL、扫描开始时间、扫描结束时间等，明确扫描任务。扫描报告记录所有发现的 SQL 注入漏洞的详细内容，包括漏洞 URL、漏洞注入类型（盲注、注入错误等）、漏洞危害程度等信息，如检测到时间盲注漏洞，记录时间漏洞的属性信息，漏洞 URL。扫描器应记录发送的请求和响应，记录请求 URL、请求方法（GET、POST 等）、请求参数、响应状态码、响应内容等。对扫描中发生的网络中断、请求超时等错误，记录出错类型、错误时间、请求和响应等定位问题并解决问题。

3.2 性能需求

3.2.1 扫描速度

(1) 并发处理能力：扫描器需要具备并发处理的能力，能同时扫描多个目标 URL 或同一目标 URL 下的不同的参数，在多核 CPU 环境下，扫描器需要合理的分配 CPU 资源，将多线程或多进程并发扫描。以扫描 100 个目标 URL 为例，在普通配置 4 核 8G 服务器上扫描器扫描时间应该保证在 10 分钟以内完成，并发数要依据服务器的能力和目标系统的负载能力而设定，建议默认不小于 10 个并发请求。

(2) 请求响应时间优化：扫描器需要具备并发处理的能力，能同时扫描多个目标 URL 或同一目标 URL 下的不同的参数，在多核 CPU 环境下，扫描器需要合理的分配 CPU 资源，将多线程或多进程并发扫描。以扫描 100 个目标 URL 为例，在普通配置 4 核 8G 服务器上扫描器扫描时间应该保证在 10 分钟以内完成，并发数要依据服务器的能力

和目标系统的负载能力而设定，建议默认不小于 10 个并发请求^[10,11]。

3.2.2 资源占用

(1) 内存占用：扫描器程序在运行过程中不能产生内存泄漏或者占用内存的情况，一个包含 100 个目标的扫描任务，不能占用过多的内存资源，内存容量控制到小于 200 MB，而对于包含更多个目标 URL 的扫描任务，例如包含大量测试用例，需要占用不超过 500MB 的内存容量，确保扫描器程序在不同的环境中正常运行。

(2) CPU 占用：扫描器程序在运行过程中不能产生内存泄漏或者占用内存的情况，一个包含 100 个目标的扫描任务，不能占用过多的内存资源，内存容量控制到小于 200 MB，而对于包含更多个目标 URL 的扫描任务，例如包含大量测试用例，需要占用不超过 500MB 的内存容量，确保扫描器程序在不同的环境中正常运行。

(3) 磁盘 I/O 占用：对于扫描器需要将日志、测试用例、扫描结果写入磁盘的情况，减少对磁盘 I/O 的占用，降低对磁盘性能影响。日志文件要异步写入，避免高频率占用主线程，合理限制日志文件大小和存储频率，避免出现写满磁盘导致扫描速度慢的情况发生。

3.2.3 检测准确性

(1) 误报率控制：扫描器应通过改造检测算法及规则，将误报率控制在一个较低的范围内，对上万次安全可靠的 web 应用扫描，误报率确保在 5% 以下，更为复杂的 web 应用不能超过 10%。真正注入点经过数次验证检测打点才是 bug。

(2) 漏报率控制：漏报率是扫描器的重要指标，扫描器应能将每一种已知的 SQL 注入漏洞扫描出来，且应有较低的漏报率，对于已知的种类的 SQL 注入漏洞应保持较低的漏报率(小于 3%)；对于新的注入技术、注入技术变种，应在较短的时间内更新扫描器中的检测规则、算法及较低的漏报率。

(3) 检测精度：扫描器将漏洞类型、漏洞位置及漏洞危害级别判别正确，检测结果 SQL 注入漏洞类型正确率正确率达 SQL 注入漏洞检测 95% 及以上，检测结果 SQL 注入漏洞定位正确率正确率达 SQL 注入漏洞检测 98% 及以上，危害级别按照漏洞危害程度、利用难易程度正确分类，危害级别正确率达 SQL 注入漏洞检测 90% 及以上。

3.2.4 系统兼容性

(1) 操作系统兼容性：扫描机的工作环境是当前主流操作系统，如 Windows10、Windows Server2019、Ubuntu20.04、CentOS8、macOS BigSur 等，扫描机在各操作系统的使用应不具有性能差异，不应因操作系统的不同而造成差异产生。

(2) 数据库兼容性：扫描器对常用的多种数据库类型注入 SQL 漏检漏洞测试检测应具备兼容性，如 Oracle 数据库、MySQL 数据库、SQL Server 数据库、PostgreSQL 数据库等，扫描器应对数据库版本不同、不同语法变更检测规则和测试用例准确、快速、高效、有效、不被改变。

(3) Web 服务器兼容性：扫描器应兼容常见的 Web 服务器，如：Apache、Nginx、IIS 等，能正确响应不同类型、不同版本的 Web 服务器，且扫描器扫描不同类型、不同版本的 Web 服务器不出现误报和扫描失败的情况。

3.3 安全需求

3.3.1 扫描器自身安全性

(1) 输入验证：扫描器应兼容常见的 Web 服务器，如：Apache、Nginx、IIS 等，能正确响应不同类型、不同版本的 Web 服务器，且扫描器扫描不同类型、不同版本的 Web 服务器不出现误报和扫描失败的情况。

(2) 安全编码实践：扫描器应兼容常见的 Web 服务器，如：Apache、Nginx、IIS 等，能正确响应不同类型、不同版本的 Web 服务器，且扫描器扫描不同类型、不同版本的 Web 服务器不出现误报和扫描失败的情况。

(3) 代码审计：扫描器应兼容常见的 Web 服务器，如：Apache、Nginx、IIS 等，能正确响应不同类型、不同版本的 Web 服务器，且扫描器扫描不同类型、不同版本的 Web 服务器不出现误报和扫描失败的情况^[12]。

(4) 用户认证和授权：用户需要通过认证和授权才能访问扫描器提供的用户界面或 API 接口。只有认证后的用户才可以访问扫描器，并且用户需要依据角色和权限才能使用，例如只有管理员才能配置和管理操作系统，而只有普通用户才能执行扫描任务并查看扫描结果。

(5) IP 访问控制：控制非法访问通过限制可以访问扫描器的 IP 地址域的范围，使扫描器只能被可信任的网络或者 IP 访问，增强安全性。

(6) 数据加密：扫描器将扫描、配置等敏感信息进行加密存储及传输。扫描器使用密码学中的数据加密算法(AES)等对存储数据的内容进行加密、通过信息传输协议(HTTPS 协议)等的支持进行传输，确保数据在传输过程中不会被窃取、篡改。

(7) 数据备份和恢复：及时对扫描器数据进行备份防止数据丢失，做好数据恢复的准备，当数据丢失、损坏时，及时进行数据恢复，将数据备份至安全位置，如远程机离线设备或其他服务器。

3.3.2 对目标系统的安全性影响

(1) 请求频率控制：使用扫描器控制请求频率，防止 DoS 拒绝服务攻击控制目标。可以根据需要控制的目标系统的性能设定每个请求的间隔。对性能较差的系统可以增加系统请求间隔，对性能较好的系统可以修改系统的请求频率。

(2) 并发请求限制：并发的请求数量不能太多，不能并发太多请求到目标系统。并发请求限制，应该根据目标系统处理能力的大小来设置。例如小型的 web 应用可以并发 5-10 个并发请求，企业级应用可并发多些，但也不宜过多并发。并发请求限制，一定要根据目标系统处理能力。

(3) 测试用例安全：测试用例扫描器应当选择合适的测试用例，测试用例不能破坏系统、不能引入新的漏洞。测试用例的设计应当遵循不破坏系统、不引入新漏洞的原则，不使用极端测试用例，不出现系统崩溃的情况。

(4) 回滚机制：扫描器在进行测试扫描对扫描系统进行改动时（插入测试用数据），应该有回滚机制，扫描后扫描系统应该能够恢复到原始状态，避免扫描行为对扫描系统的干扰。

3.3.3 漏洞报告的安全性

(1) 敏感信息处理：脱密漏洞报告中的敏感信息（数据库用户名、数据库密码、服务器 IP 地址等），敏感信息只含有必要的信息，数据库密码显示*、IP 地址显示等。

(2) 报告完整性：漏洞报告内容包括漏洞描述、漏洞检查、危害程度、修复意见等内容，报告内容完整正确，目标系统管理员可以针对目标系统进行修复等意见。

(3) 加密传输：在传输漏洞报告的过程中应采用加密的方式进行传输，例如邮件加密方式传输或 SFTP 传输等，保证在传输的过程中，不会导致漏洞报告外漏。

（4）访问控制：对存储漏洞报告服务器或存储介质进行访问控制，对报告人授予访问权限，防止对存储的漏洞报告非法访问和泄露。

3.3.4 合规性要求

（1）扫描器的研发使用遵守国家/地区的法律法规（例如：《网络安全法》），扫描器遵守合法扫描，不进行非法网络扫描、攻击。

（2）对采集和使用的目标系统相关信息，遵循隐私保护法规，保护用户的个人信息和隐私，避免用户敏感信息被泄露和滥用的发生。

（3）参考相关的行业安全标准和最佳实践，如参考 OWASP 安全组织的相关指南。借鉴 OWASP 相关的防止 SQL 注入指南，优化扫描器的检测规则和算法，提升扫描器的检测效率和扫描的安全性。

4 系统设计

SQL 注入扫描系统的整体系统使用 flask 库,通过界面接收用户输入的 url 及扫描等级,根据不同的扫描等级进行相应的扫描,系统由多个模块和子模块组成,通过不同模块之间的相互配合来完成 SQL 注入的扫描。

4.1 大体设计

(1) 用户与系统交互界面,是个 web 页面基础 html、css、js 相结合,通过一个 flask 库的渲染模式引擎如金像(金像 2)渲染的界面。用户通过界面输入扫描 URL,扫描级别 Light, Standard, Deep。功能:接收用户输入的 URL 以及扫描等级信息,并发送到 Flask 主控制器。

(2) 基于 flask 库模式定义路由,路由定义了不同类型的用户请求:路由(/), 扫描(/scan), 路径(/debug.inf)等。定义路由时,在代码中用 app.route 对路由进行装饰定义。功能:(1)接受用户界面传来的 url 和扫描等级。(2)根据扫描级别选择合适的扫描模式(轻量扫描、标准扫描或深度扫描)。(3)协调各个模块的工作,如调用扫描模块进行扫描,并将扫描结果返回给用户界面。

(3) 根据用户选择的扫描级别,决定使用哪种扫描模式。可以通过条件判断语句(如 if-elif-else)来实现。功能:根据用户选择的扫描级别,调用相应的扫描模块进行扫描^[13]。

4.2 扫描模式

- 轻量扫描:使用基本 Payload 库(BASICPayloadS),快速 URL 扫描和多级爬虫扫描。使用多线程或异步扫描。基本 Payload 库,快速 URL 扫描和单级爬虫扫描,使用线程扫描。功能:(1)对 URL 做简单扫描(获取 basicSQL 注入 Pool 测试)。(2)单层采集,获取部分链接,扩大扫描范围。

- 标准扫描:利用 Payloads 扩展库(STANDARD_PAYLOADS),配合 WAF 绕过库(WAF_BYPASS_PAYLOADS)进行表单、URL 参数、HTTP 头部的扫描,并进行两层深度爬取。功能:(1)扩展 Payload 库进行扫描。(2)WAF 过滤机制,尝试绕过目标网站 Web 应用防火墙。(3)表单、URL 字符串和 HTTP 头部扫描 SQL 注入漏洞。(4)两层的链

接爬取扫描。

- **深度扫描：**对全量 Payloads（包括 DEEPPayloadS、ADVANCEPayloadS 等）进行二阶检测、时间盲注分析、特殊 CRM 检测以及三层数爬取操作。功能：（1）使用全量的 Payload 库进行最全面的扫描。执行时间盲注检测，通过时间戳生成与时间戳追踪，检查是否存在时间盲注漏洞。（2）执行二阶注入测试，通过标注生成与标注追踪，检查是否存在二阶 SQL 注入漏洞。针对不同的 CRM（灵当 CRM、糖糖 CRM，以下简称糖糖 CRM），使用对应的 library 底部（LINGDANG_CRMPayloadS、CRMPayloadS）来检测。（3）三层爬取，尽可能的爬取目标网站的所有内容^[14,15]。

4.3 核心功能

- **智能延迟：**通过设定延迟时间的最小值（MIN_REQUEST_DELAY）和最大值（MAXREQUEST_DELAY）、动态调整延迟时间（DELAY time dependenton Scanning），通过设定随机数发生器可以智能的延迟。功能：控制请求发出的间隔时间，避免给目标主机造成过大影响，避免引起目标主机的防御。

- **用户代理轮换：**定义一个用户代理列表（USER_AGENTS），通过随机选择列表中的用户代理来实现轮换。功能：随机选择不同的用户代理，模拟不同的浏览器或设备，增加扫描的隐蔽性。

- **请求控制：**使用 request 发送 http 请求，配置超时时间（REQUEST_TIMEOUT）和最大尝试次数（MAX_RETRIES）。功能：负责发送请求到目标服务器，并管理请求的并发和超时。

- **响应分析引擎：**分析返回的相关信息，抽取对应服务器返回的信息，例如返回状态、返回信息等。可以使用正则表达式(re 模块)匹配相应模式。功能：对目标服务器响应的数据，进行响应解析，提取相关数据，用于漏洞匹配。

- **漏洞匹配：**定义一系列 SQL 错误模式（SQL_ERROR_PATTERNS），将响应分析结果与这些模式进行匹配。功能：将响应分析结果与错误模式库进行匹配，判断是否存在 SQL 注入漏洞。

- **日志记录：**使用 python 日志库记录日志，指定日志级别(日志等级 logging)，日志格式。功能：保存扫描过程的详细情况，如请求信息、响应信息、漏洞检测结果，便

于后期调试分析^[16]。

4.4 高级功能

- 二阶注入检测：通过生成唯一的标记（CANARY_TOKENS），将标记注入到请求中，然后在后续的请求中检查响应中是否包含该标记。功能：检测是否存在二阶 SQL 注入漏洞，即注入的恶意代码在后续的请求中被执行。

- 时间盲注检测：运用带有时间延迟功能的 DELAYayer（诸如睡眠、等待、延时等操作），借助查看延时情况来判断是否存在时间盲注漏洞。功能：判断是否存在时间盲注漏洞，即通过限制数据库的回答时间来获取信息。

- CRM 专用检测：对不同的 CRM 系统进行定义，比如灵当 CRM、SugarCRM，针对不同 CRM 系统分别定义专门的有效载荷库（LINGDANG_CRM_PAYLOADS、CRM_PAYLOADS）并使用这些有效载荷进行检测。功能：对具体的 CRM 系统进行 SQL 注入漏洞检测，让漏洞检测更具针对性和有效性^[17]。

4.5 支持库

- Payload 管理器将不同的等级 PayloadS(LIGHTPayloadS,STANDARDPayloadS,DEEPayloadS 等)分类管理，根据扫描等级，选择不同的 PayloadS。功能：控制各个级别的 POP 库，提供不同种类的 SQL 注入测试语句。

- 错误模式库定义一系列 SQL 错误模式（SQL_ERROR_PATTERNS），用于漏洞匹配。功能：存储常见的 SQL 错误模式，为漏洞匹配提供参考。

- CRM 特征库针对不同的 CRM 系统，定义专门的特征信息，如特定的错误模式、数据库表名等。功能：存储不同 CRM 系统的特征信息，为 CRM 专用检测提供支持。

5 系统实现

5.1 Python 环境搭建

- (1) Python 版本: 3.8+
- (2) 依赖管理: 使用 requirements.txt 精确控制版本

5.2 核心模块实现

核心检测模块负责检测目标 URL 或表单是否存在 SQL 注入漏洞,主要通过以下步骤实现:

(1) 定义有效载荷: 系统定义了多种类型的 SQL 注入有效载荷,包括轻量级、标准、深度、高级、登录表单专用、CRM 系统专用等。这些载荷覆盖了不同的 SQL 注入技术,如报错注入、布尔盲注、时间盲注、UNION 注入、堆叠注入等。

(2) 识别注入类型: 通过 identify_injection_type 函数,根据响应文本、原始文本、有效载荷和响应时间,识别 SQL 注入的类型。例如,检测报错注入通过匹配特定的错误模式,检测 UNION 注入通过检查响应文本长度的变化等。

(3) 检查漏洞: is_vulnerable 函数用于检查响应是否表明存在 SQL 注入漏洞。它调用 identify_injection_type 函数识别注入类型,并根据响应文本与原始文本的差异判断是否存在漏洞。

(4) 测试 URL 参数: test_url_parameter 函数用于测试单个 URL 参数。它的有效载荷插入到 URL 参数中,发送请求并检查响应是否存在漏洞。对于 DVWA 特殊情况,还会处理用户令牌。

(5) 测试表单字段: test_form_field 函数用于测试单个表单字段。它的有效载荷插入到表单数据中,根据表单方法(GET 或 POST)发送请求,并检查响应是否存在漏洞。同时,会检查重定向情况,判断是否成功绕过认证^[18]。

5.3 关键技术难点的解决方案

● 误报与漏报控制难点: 如何准确判断目标是否存在 SQL 注入漏洞,避免误报和漏报。不同的 Web 应用和数据库系统对 SQL 注入的响应各不相同,这增加了判断的难度。(1) 多维度检测:通过分析响应文本内容、长度、响应时间等多个维度来判断是

否存在漏洞。例如，识别特定的数据库错误信息、检查响应长度的变化、监测响应时间是否因注入时间延迟语句而显著增加。（2）多种注入类型检测：针对不同的 SQL 注入类型（如报错注入、布尔盲注、时间盲注、UNION 注入、堆叠注入等）设计专门的检测逻辑，提高检测准确率。（3）智能比较：将注入有效载荷后的响应与原始响应进行智能比较，判断差异是否由 SQL 注入引起。

- 反爬虫与防护机制绕过难点：目标网站可能有反爬虫机制或 WAF（Web 应用防火墙），会阻止扫描请求或误判为攻击行为。（1）智能延迟策略：在请求之间引入随机延迟，避免频繁请求触发速率限制。延迟时间在最小和最大延迟之间随机选择，模拟人类用户的浏览行为。（2）用户代理池：使用多个不同的 User-Agent 发送请求，降低被识别为爬虫的风险。（3）请求头多样化：在请求中添加各种合法的 HTTP 头信息，使请求更像正常的浏览器请求。（4）降低并发度：控制线程池的大小，减少并发请求数量，避免对目标系统造成过大压力。

- 表单与会话管理难点：处理复杂表单和维护会话状态，特别是需要 CSRF 令牌或用户认证的情况。（1）表单解析与处理：自动解析 HTML 表单，提取表单字段、提交 URL 和方法等信息，支持对各种类型的表单字段进行测试。（2）CSRF 令牌处理：识别并提取表单中的 CSRF 令牌，在测试过程中正确使用这些令牌，确保请求的合法性。（3）会话管理：使用会话对象（如 requests.Session）维护会话状态，处理 Cookie 和登录会话，支持对需要认证的页面进行漏洞扫描。

- 性能与效率平衡难点：在保证扫描准确性的前提下，提高扫描效率，减少扫描时间。（1）并行扫描：使用线程池并行执行多个扫描任务，提高扫描效率。（2）智能任务取消：一旦发现某个参数存在漏洞，立即取消该参数的其他测试任务，避免不必要的请求。（3）重试机制：对于失败的请求进行有限次数的重试，确保扫描的完整性。（4）结果缓存：缓存已测试的结果，避免重复测试相同的参数和有效载荷^[19,20]。

5.4 用户界面实现

5.4.1 Web 界面设计概述

整个用户界面主要由几个大的部分构成，分别是扫描设置区域、扫描过程中的加载提示区域、扫描结果展示区域以及调试信息区域。页面的布局使用了 HTML 的 <div> 标

签进行模块化划分，每个部分都有明确的功能和展示内容。

5.4.2 WEB 页面预览

当我们成功运行程序，访问指定的 url，我们就会来到 SQL 注入漏洞扫描器的工具页面，如图 5.1，5.2。



图 5.1 WEB 页面预览图 1

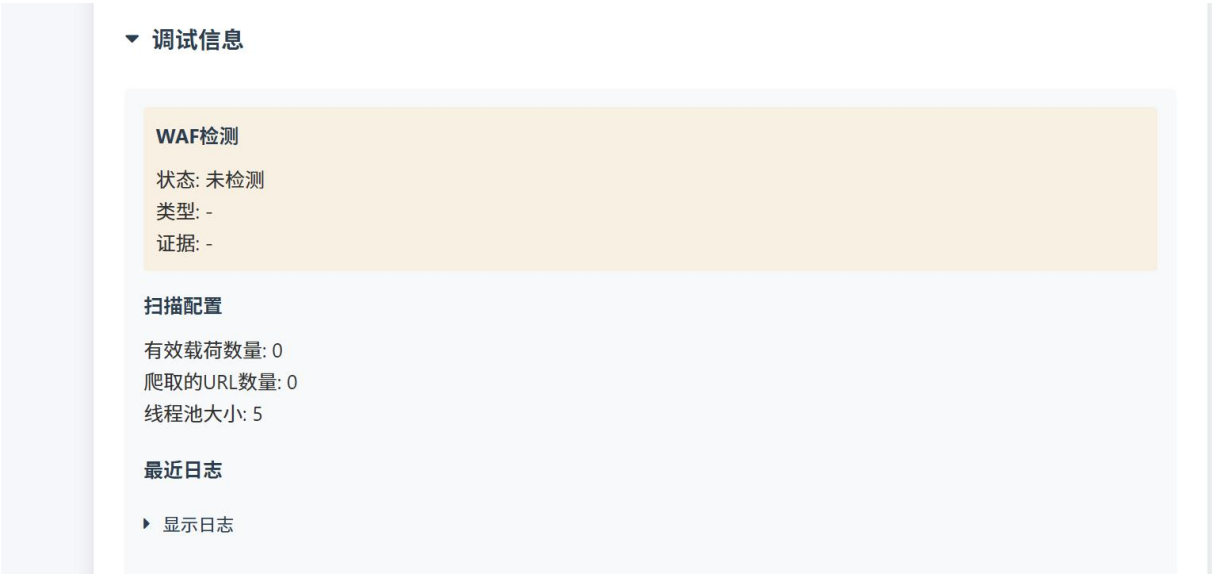


图 5.2 WEB 页面预览图 2

6 系统测试

6.1 测试环境搭建

所需软件有 Python 3.7+, pip (Python 包管理器), Docker (可选, 用于容器化测试, 启用前用 pip 安装相关版本的依赖。在当前目录的终端使用命令 `pip install -r requirements.txt` 安装依赖, 使用 `python app.py` 启动扫描器。

6.2 功能测试用例设计

本次功能测试我们是采用的 sqlmap 靶场的指定关卡, 来模拟各种 SQL 注入漏洞的环境, 去测试该工具在各种情况下功能的完整性, 如表 6.1。

表 6.1 功能测试用例表

测试模块	测试场景	环境选择 (统一使用 SQLmap 靶场)
SQL 注入检测	轻量级有效载荷	第 65 关
	标准有效载荷	第 65 关
	深度有效载荷	第 65 关
登录表单测试	用户名注入	第 11 关
	密码注入	第 12 关
	特殊组合注入	第 20 关
防御绕过策略	注释绕过	第 23 关
	随机大小写	第 28 关
	双写绕过	第 28 关
	URL 编码绕过	第 21 关
系统配置	请求超时设置	http://example.com:81
	URL 格式错误	123456798
基础漏洞类型检测	时间盲注	第 9 关
	宽字节注入	第 36 关
	参数污染	第 29 关
	二次注入	第 24 关

6.3 功能演示

6.3.1 SQL 注入检测

采用 SQLi 靶场的 65 关对轻量查询，标准查询，深度查询做了功能的和对比，轻量查询快但只是最基本的查询，标准查询一般推荐有够用的有效载荷，和相对较快的时间，深度查询的有效载荷最多，与之相对应的是时间是最慢的。如图 6.2，6.3，6.4 所示。

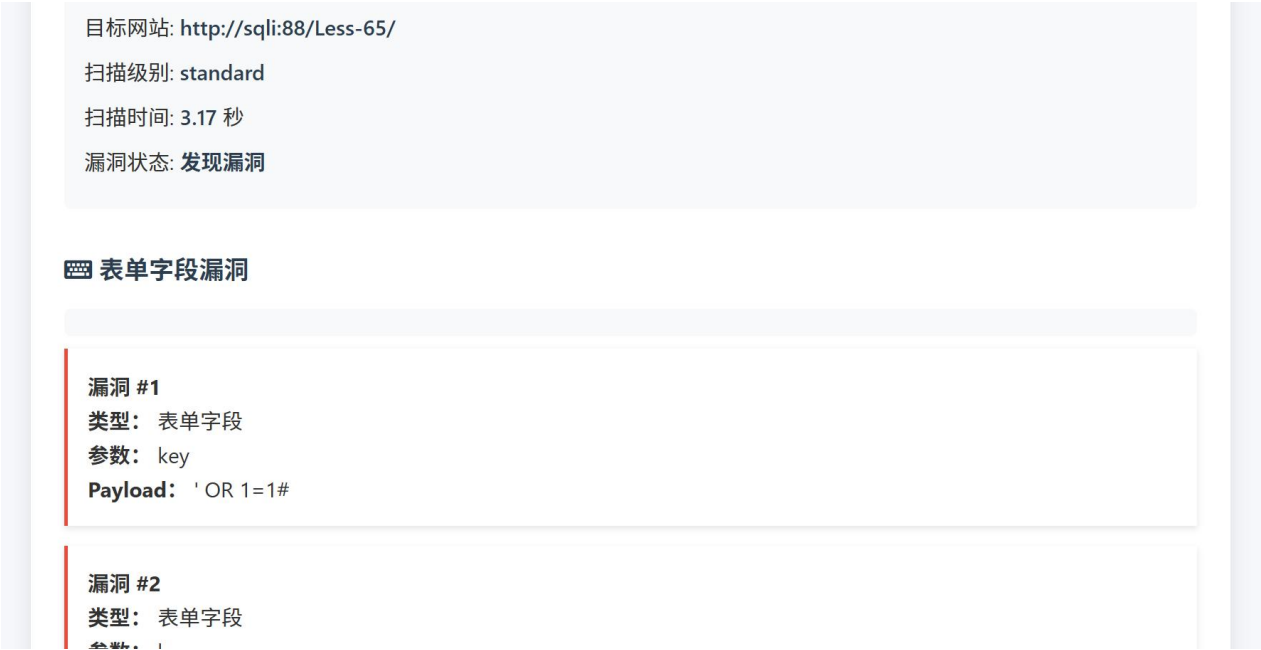


图 6.2 标准查询

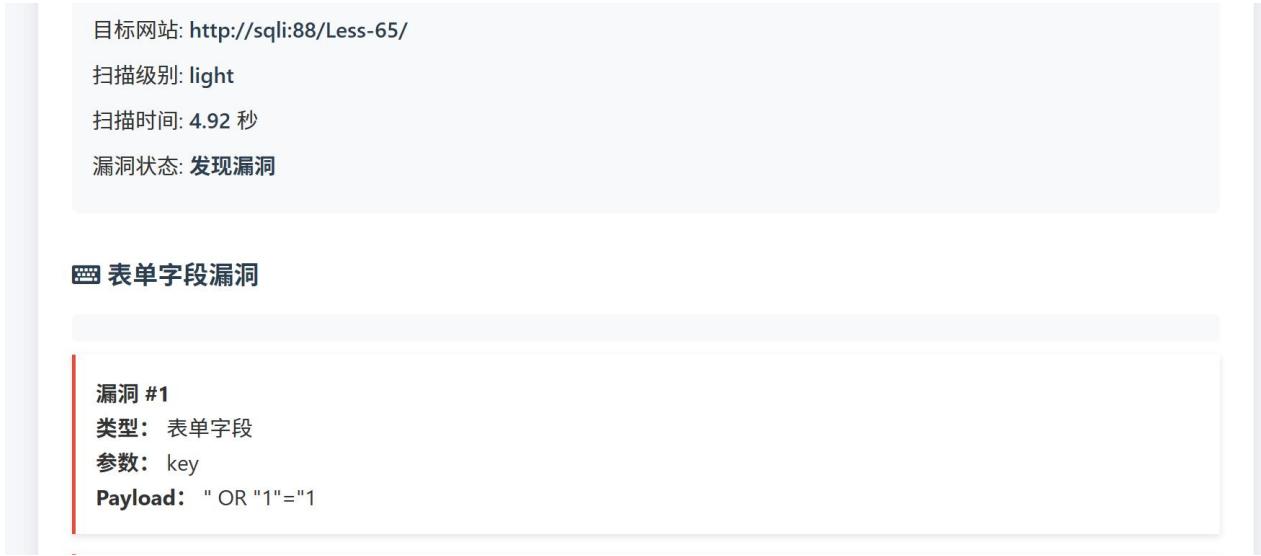


图 6.3 轻度查询

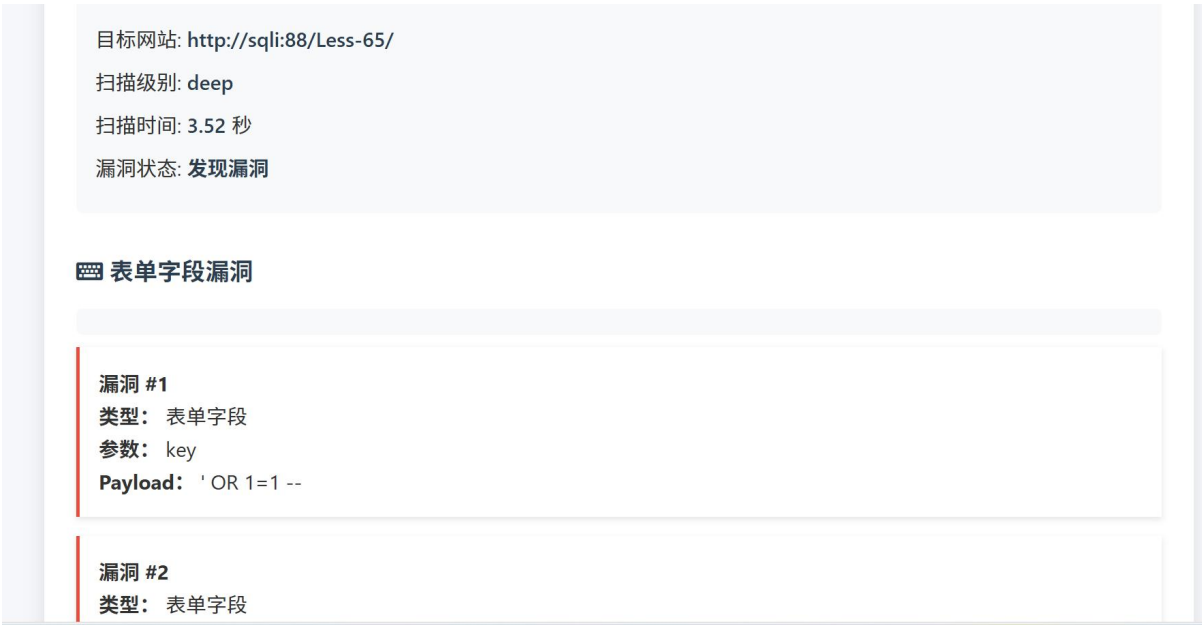


图 6.4 深度查询

6.3.2 SQL 登陆表单测试

采用 SQLi 靶场的 11, 12, 20 关对用户名注入, 密码注入, 特殊组合注入 (cooike) 做了功能的和对比, 主要检测该系统对登陆表单的 SQL 注入检测机制上是否完善, 能否应对多种情况的多种绕过。如图 6.5, 6.6, 6.7 所示。

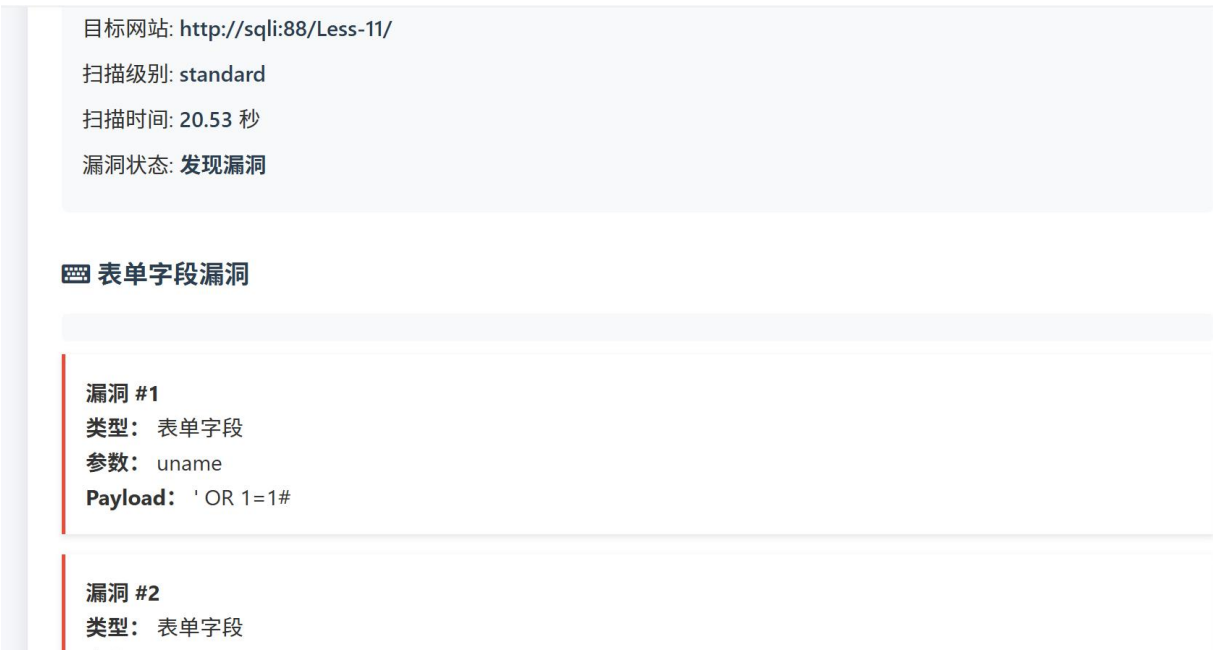


图 6.5 用户名注入

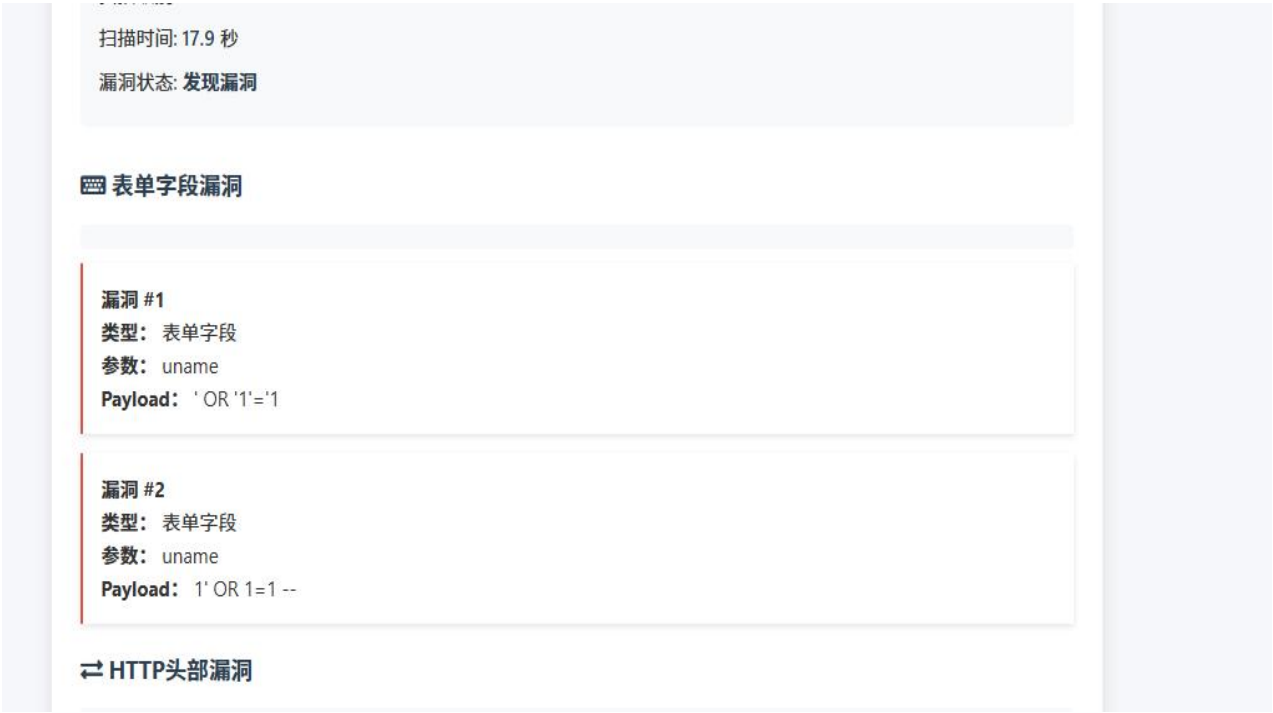


图 6.6 密码注入



图 6.7 特殊组合注入

6.3.3 防御绕过策略

Waf 的绕过方式有好多总经常见的有大小写，双写，url 编码，注释绕过等等，在这里我就演示几个最常见和最经典的，用检测系统的功能是否全面。如图 6.8，6.9，6.10 所示。

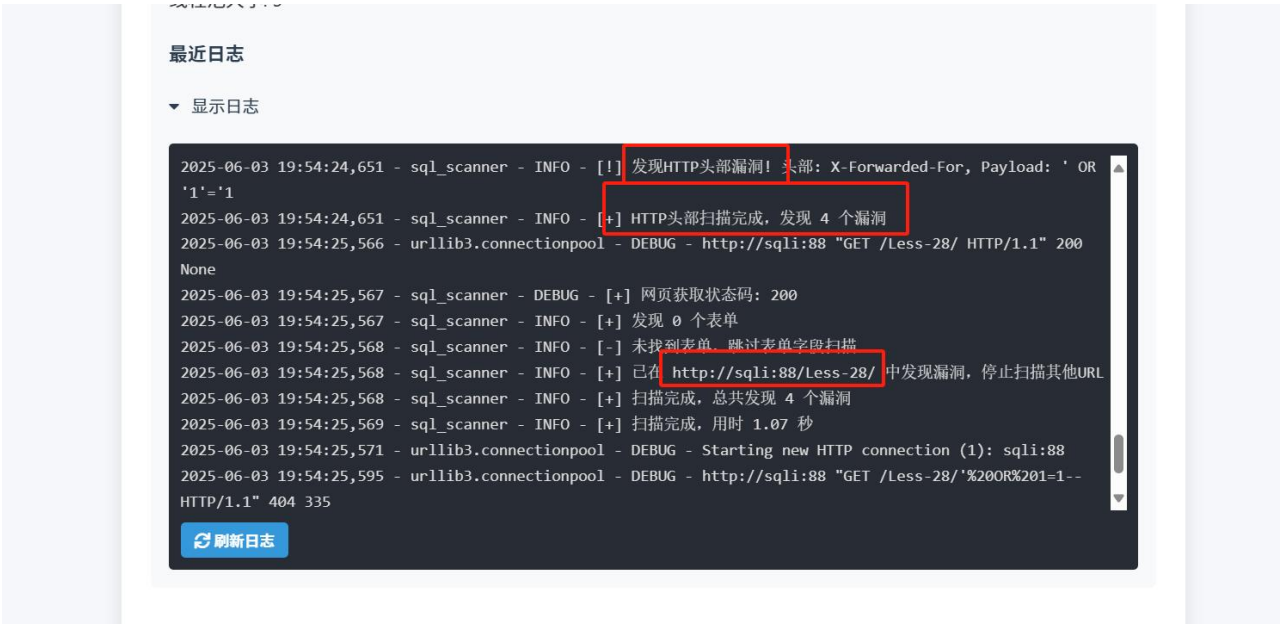


图 6.8 大小写双写绕过

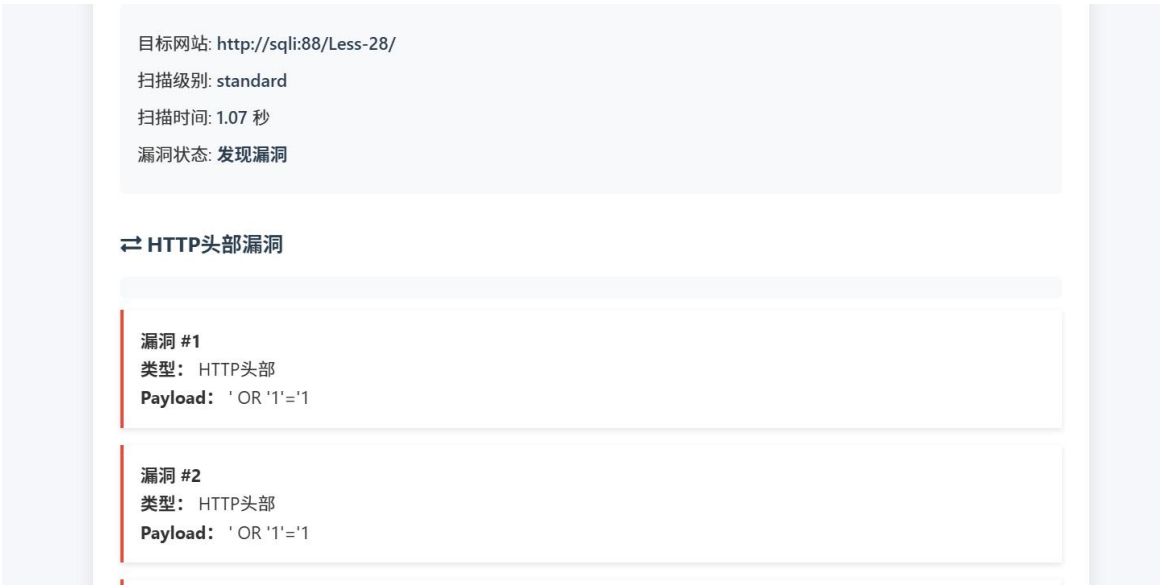


图 6.9 url 绕过

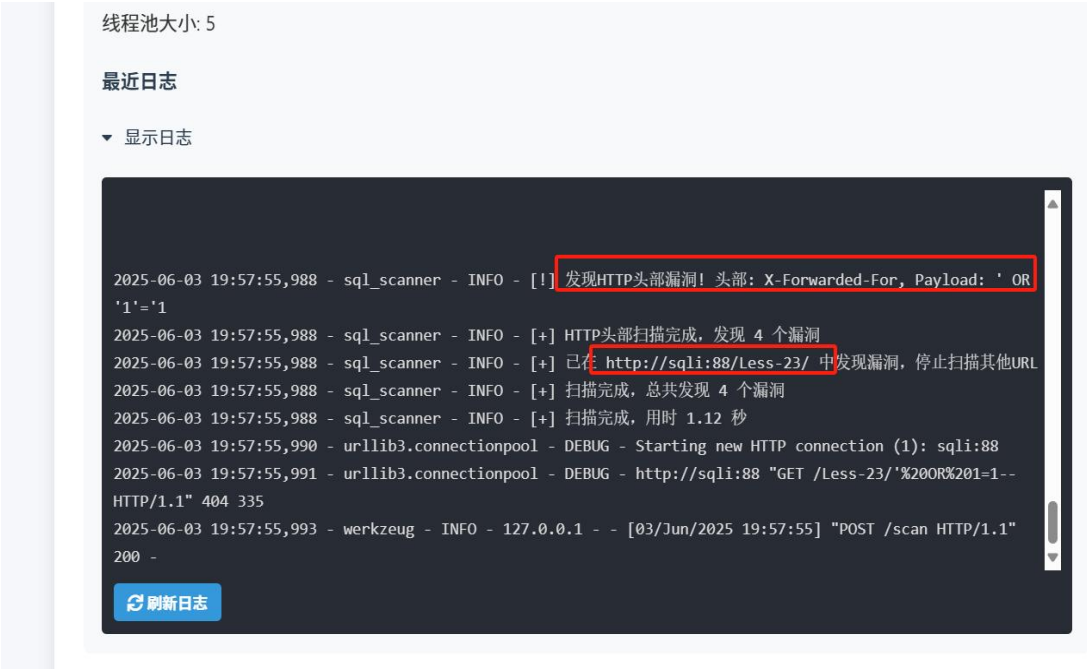


图 6.10 注释绕过

6.3.4 系统配置

本测试主要展示系统在面对一个超时链接和根本不存在的链接时，可不可以正常回显问题如图 6.11，6.12 所示

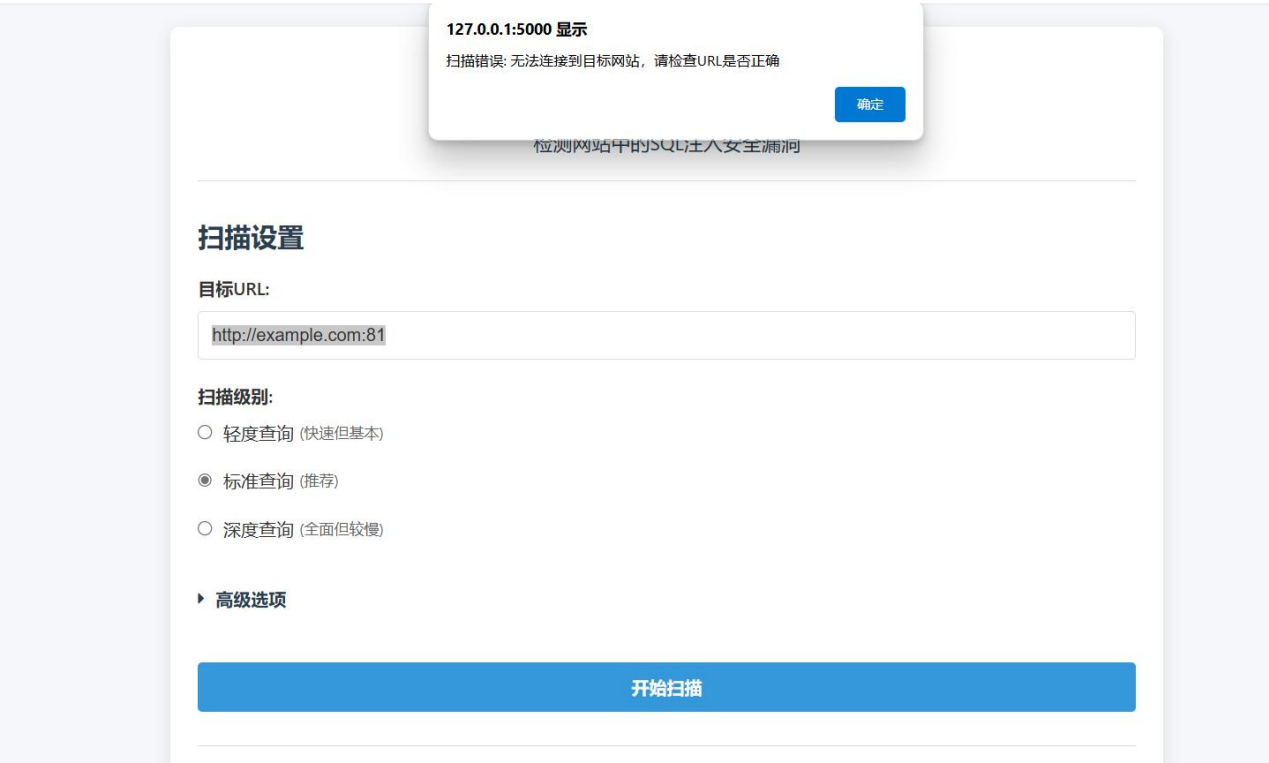


图 6.11 超时链接回显



图 6.12 错误链接回显

6.3.5 基于漏洞类型检测

SQL 注入的类型有很多种，盲注，二次注入，宽字节，参数污染是一些中等偏难的注入类型，测试系统是否有扫描出隐蔽 SQL 注入类型的能力，如图 6.13，6.14，6.15，6.16，6.17 所示。经测试发现可以成功检测漏洞，在二次注入打开时也可以成功检测二次注入漏洞。



图 6.13 时间盲注测试

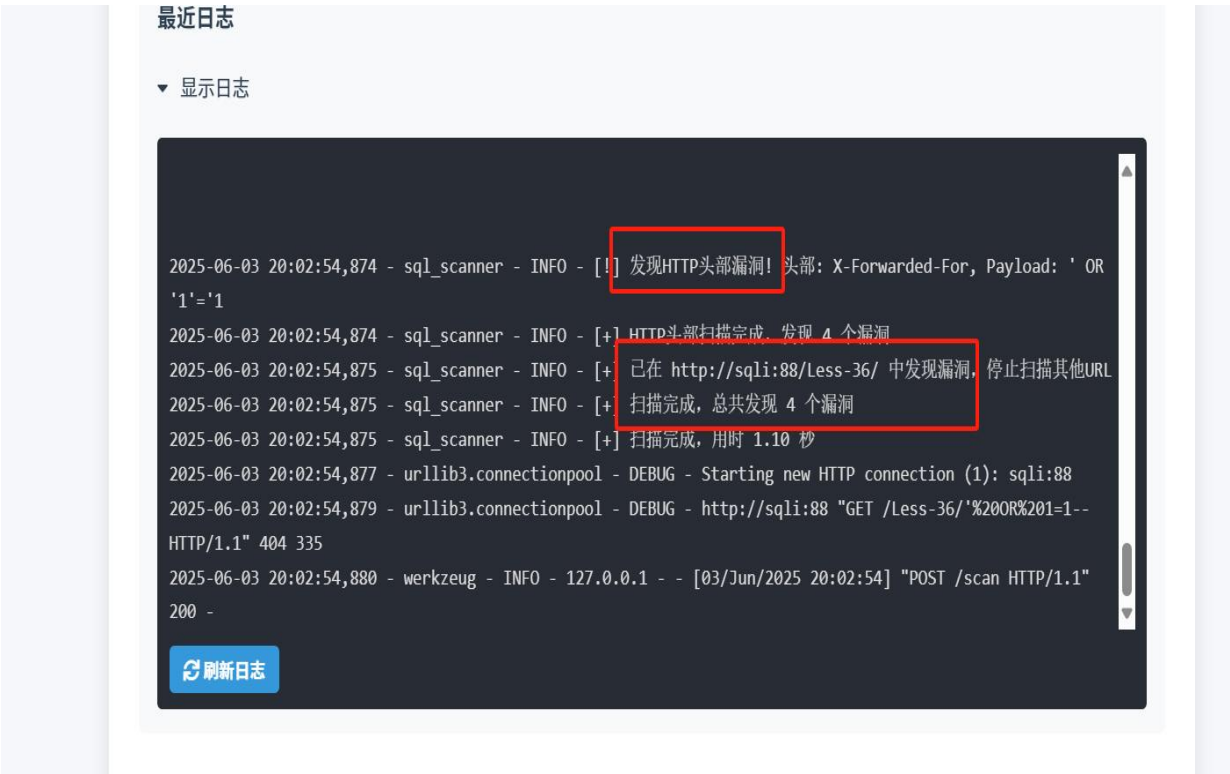


图 6.14 宽字节注入测试



图 6.15 参数污染测试

扫描结果

目标网站: http://sqli:88/Less-24
扫描级别: 标准查询
扫描时间: 34.48 秒
漏洞状态: 安全



未发现SQL注入漏洞

这并不意味着网站完全安全，可能存在其他类型的安全问题。

图 6.16 二次注入测试（未开启二次注入时）

生成报告

目标网站: http://sqli:88/Less-24/

扫描级别: standard

扫描时间: 19.21 秒

漏洞状态: 发现漏洞

表单字段漏洞

漏洞 #1

类型: 表单字段

参数: login_user

Payload: ' OR 1=1 --

漏洞 #2

类型: 表单字段

参数: login_user

Payload: ' OR 1=1#

图 6.17 二次注入测试（开启二次注入时）

6.3.6 生成对应的漏洞报告

在成功检测出 SQL 注入漏洞后可以手动直接生成这次检测的漏洞报告详情，便于查看，如图 6.18，6.19，所示。经测试发现可以成功导出漏洞报告。



图 6.18 导出报告

SQL注入漏洞扫描报告



图 6.19 漏洞报告

7 结论与展望

7.1 测试结论

(1) 功能完整性:

- 系统实现了全面的 SQL 注入检测功能，包括 URL 参数、表单字段、HTTP 头部的扫描
- 支持多种扫描级别（轻量、标准、深度）和自定义扫描
- 具备二阶注入检测和 WAF 绕过能力

(2) 技术特点:

- 采用多线程并发扫描，提高了检测效率
- 内置丰富的 SQL 注入 payload 库和错误模式识别规则
- 实现了智能延迟策略和用户代理轮换，有效规避防护机制
- 特别针对 CRM 系统(如灵当 CRM)设计了专用检测逻辑

(3) 检测能力:

- 能够准确识别多种 SQL 注入漏洞，包括布尔型、时间盲注、联合查询等
- 对常见的 WAF 防护机制有良好的绕过能力
- 能够检测二阶 SQL 注入和可疑 JavaScript 文件引用

(4) 系统稳定性:

- 完善的错误处理和重试机制
- 详细的日志记录便于问题排查
- 资源控制合理(线程池大小、请求超时等)

(5) 安全性:

- 扫描行为可控，不会对目标系统造成破坏
- 请求速率控制避免对目标系统造成过大负载

7.2 改进建议

(1) 功能增强:

- 增加对 NoSQL 注入的检测能力

- 添加 XSS 和命令注入等其他 Web 漏洞的扫描功能
- 实现自动化的漏洞验证功能，减少误报

(2) 性能优化：

- 引入更智能的爬虫策略，提高链接发现效率
- 优化 payload 执行顺序，优先测试高成功率的 payload
- 添加分布式扫描支持，提升大规模站点扫描效率

(3) 用户体验：

- 开发更友好的 Web 界面，提供可视化扫描结果
- 添加扫描报告生成功能(HTML/PDF 格式)
- 实现扫描进度实时展示

(4) 技术改进：

- 引入机器学习算法优化漏洞识别准确率
- 增强对新型 WAF 的绕过能力
- 添加 API 接口支持，便于集成到其他安全系统

(5) 维护性：

- 建立 payload 和错误模式的自动更新机制
- 添加插件架构，支持功能模块化扩展
- 完善单元测试和集成测试覆盖^[21]

7.3 未来展望

(1) 智能化发展：

- 结合 AI 技术实现更精准的漏洞识别
- 开发自适应扫描策略，根据目标响应动态调整

(2) 集成化方向：

- 与漏洞管理系统、SIEM 系统集成
- 支持作为持续集成/持续部署(CI/CD)流程的一部分

(3) 商业化潜力：

- 可作为企业级 Web 应用安全扫描解决方案

- 发展 SaaS 模式，提供在线扫描服务

(4) 社区生态：

- 开源核心引擎，吸引社区贡献
- 建立 payload 和规则共享平台

(5) 合规支持：

- 添加对 OWASP Top 10 等安全标准的支持
- 生成符合 PCI DSS、等保等合规要求的报告

参 考 文 献

- [1] 秦泽坤, 聂闻. 基于 RASP 和机器学习的 SQL 注入检测方法[J]. 信息技术, 2025, 3: 69-75, 85.
- [2] 郑志慧, 邹雨琛. Python 在网络安全漏洞扫描中的实现与优化[J]. 信息记录材料, 2024, 7: 89-91.
- [3] 李舟军, 余杰. 主动 Web 漏洞扫描中的场景技术研究[J]. 计算机工程与科学, 2022, 3: 31-34, 65.
- [4] 叶梦雄. 基于 Web 的 SQL 注入漏洞扫描系统的设计研究[J]. 电子设计工程, 2022, 16: 20-23.
- [5] 李培峰, 钱峰, 周伟. 计算机通信网络安全与防护策略研究[J]. 信息与电脑, 2025, 1: 110-11.
- [6] 吴迪, 郑宇, 王强. 基于 Python 的网络安全工具开发实践[J]. 计算机工程与设计, 2021, 42(11): 3154-3160.
- [7] Jiang, X., Lora, M., and Chattopadhyay, S. (2020). An experimental analysis of security vulnerabilities in industrial IoT devices. *ACM Transactions on Internet Technology (TOIT)*, 20(2), 1 - 24.
- [8] 牛咏梅. 面向 Web 应用的漏洞扫描器的设计与实现[J]. 南阳理工学院学报, 2022, 6: 66-69.
- [9] 孟清, 路贺俊. 基于 Python 的自动代理 Web 漏洞扫描器的设计与实现[J]. 科技视界, 2022, 17: 41-45.
- [10] Yu, M., Zhuge, J., Cao, M., Shi, Z., and Jiang, L. (2020). A survey of security vulnerability analysis, discovery, detection, and mitigation on IoT devices. *Future Internet*, 12(2), 27 - 36.
- [11] 吴文绛. Python 在 Web 渗透测试中的应用[J]. 信息与电脑, 2023, 24: 227-229.
- [12] 王可, 康晓凤. 基于机器学习的日志分析系统的设计与实现[J]. 软件工程, 2022, 5: 56-59.

- [13] 徐贵江, 黄媛媛. 基于 Python 的 Web 漏洞扫描器[J]. 软件工程, 2023, 4: 19-21, 11.
- [14] 赵琳, 钱峰, 周伟. 数据库指纹识别技术在 Web 漏洞扫描中的应用[J]. 数据库与信息管理, 2022, 4(02): 33-42.
- [15] 吴迪, 郑宇, 王强. 跨平台安全扫描器的设计与实现[J]. 计算机工程与科学, 2021, 43(12): 2175-2182.
- [16] Ranaweera, P., Jurcut, A., and Liyanage, M. (2021). MEC-enabled 5G use cases: a survey on security vulnerabilities and countermeasures. *ACM Computing Surveys (CSUR)*, 54(9), 1 - 37.
- [17] Ponta, S. E., Plate, H., and Sabetta, A. (2020). Detection, assessment and mitigation of vulnerabilities in open source dependencies. *Empirical Software Engineering*, 25(5), 3175 - 3215.
- [18] 朱振南, 金京犬. 基于 SQL 注入漏洞的攻击技术研究[J]. 电脑知识与技术, 2024, 1: 98-100, 103, 105.
- [19] 粟圣森. SQL 注入漏洞与安全防范技术探讨[J]. 轻工科技, 2025, 2: 100-103.
- [20] Yaacoub, J. P. A., Noura, H. N., Salman, O., and Chehab, A. (2022). Robotics cyber security: Vulnerabilities, attacks, countermeasures, and recommendations. *International Journal of Information Security*, 21(1), 115 - 158.
- [21] 钱丽. 基于 SQL 注入漏洞的攻击技术研究[J]. 网络安全技术与应用, 2023, 7: 55-59.

致 谢

我想感谢我的指导教师曹峙和魏月娟老师，在整个毕业设计过程中给予我很多的鼓励和关怀，从毕业论文的选题，研究、系统设计、论文写作以及论文的选题中，您为学之术、为术之术、为经验之术犹如一灯火在我身上燃烧，在我研究过程中遇到困难和瓶颈时能够在我身上看到问题的原因，新的思路和方法，让我从困境中跳出来，不单是教会我怎样做论文，更重要的是教会我怎样做学问，怎样独立思考问题解决问题，您的一言一行将会使我终身受益，我永远铭记于心。

也感谢畅洋同学，在这过程中遇到技术问题一起研究，遇到问题一起讨论，在我弄不懂细节技术的时候，你都是不顾一切跟我交流，给我新的方向新的办法，你的帮助你的陪伴使我能够有胆量有底气继续往下进行，让我在毕业设计上能走得更有底气走得更有力量，我们共同度过了许多难关，也让我对知识能有更深刻的理解，也让我和畅洋同学的友谊更加深刻。

感谢这充满挑战而又充实的毕业设计时光，感谢帮助过我的所有人，毕业设计作品不仅是个人的收获，更是一群人的辛劳，本人将带着感恩的心继续努力学习、工作和生活。