**北京邮电大学**

**网络空间安全学院**



**《网络安全平台设计实践》课程总结报告**

**组 长 凌霄汉**

**组 员 王硕**

**组 员 谢庆贺**

**2019年 12月19日**

**目录**

[1. 说明 3](#_Toc21657)

[1.1报告说明 3](#_Toc1590)

[1.2分工介绍 3](#_Toc1590)

[2. 网站介绍 3](#_Toc32742)

[2.1功能概述 3](#_Toc1590)

[2.1.1登录 3](#_Toc9405)

[2.1.2注册 3](#_Toc14000)

[2.1.3重置密码 3](#_Toc9405)

[2.1.4主页 3](#_Toc14000)

[2.1.5题目界面 3](#_Toc9405)

[2.1.6公告栏 4](#_Toc14000)

[2.1.7排行榜 4](#_Toc9405)

[2.1.8战队界面 4](#_Toc14000)

[2.1.9 TOP10战队趋势图 4](#_Toc9405)

[2.1.10消息系统 4](#_Toc14000)

[2.1.11信息界面 4](#_Toc9405)

[2.1.12管理员界面 4](#_Toc14000)

[3. 漏洞扫描模块介绍 5](#_Toc28745)

[3.1 爬虫模块 5](#_Toc31920)

[3.2 sql注入漏洞扫描模块 6](#_Toc22643)

[3.3 xss漏洞扫描模块](#_Toc31920) 7

[3.4 弱口令破解模块 8](#_Toc22643)

[3.5 文件上传漏洞模块](#_Toc31920) 9

[3.6 输出模块 1](#_Toc22643)0

#### 说明

* 1. 报告说明

本报告为为简化报告，只说明精华内容，具体内容在打包的视频，ppt中，以及代码中。

模块目前还没有进行整合，因此会以三个压缩包的形式进行发送，在每一个压缩包下面已经跑通了，如果跑不通可能与环境的配置有关。项目在visual studio下面完成，需要自配虚拟环境以及安装包。

关于爬虫+弱口令+输出模块的源代码说明：

由于功能已完全但组合不完全，要想使用爬虫，请将641行的scanner\_scraper去掉注释，并加上646，689的注释。

如果要使用弱口令扫描+输出模块，则请注释641行并去掉646，689的注释。

另外由于使用绝对路径，故在运行时请自行修改路径。

##爬虫

scanner\_scraper()

##sqli

#sqli()

'''

* 1. 分工介绍

小组成员：凌霄汉、王硕、谢庆贺，凌霄汉是组长。

#### 网站介绍

* 1. 功能概述

网站地址：http://49.233.168.44/

总的来说，网站功能较为全面，不仅仅只是一个做题的平台。首先是基本的登录、注册、重置密码。登录成功后，导航栏分别代表了主页、题目界面、公告栏、排行榜、战队界面、top10战队趋势图、站内消息、个人信息、战队信息、登出。在题目界面里面实现了Markdown提交writeup功能，在个人信息界面可以进入管理员界面，管理员可以对公告、消息、题目进行管理。下面详细说明各个功能的细节和实现逻辑：（具体的演示请详见录屏）

2.1.1登录

只有当用户正确输入了用户名、密码、验证码才可以进入网站主页。验证码可以防止弱密钥碰撞攻击。

2.1.2 注册

用户名、学号、邮箱所有用户中均不能有重复，注册采用邮箱认证的方式，只有用户在邮箱认证成功才可以激活账号。注册界面实现了基本的检测，比如用户名是否被注册、邮箱是否合法、密码强度等等。

2.1.3 重置密码

同样采用邮箱验证的方法，为每一个用户设置一个布尔属性resetable，默认为false，代表用户当前是否可以修改密码，当用户邮箱认证成功设置成true，修改完密码再修改成false. 认证成功自动跳转到修改密码界面，修改成功即可跳转到登录界面

2.1.4主页

主页有三个直通车，为最新公告、我的进展、最新消息。方便用户实时查看网站内部的变化。

2.1.5 题目界面

进入题目界面，所有题目显示在一个界面中，点击分类导航，可以筛选每一类型的题目。每一类题目显示用户完成情况的进度条，当用户成功提交某一题的flag后，会在题目框上显示一个对勾，代表已经完成该题。

另外除了样例题目外，我们每个人还在学习的基础上部署了两道题。

点击某一题的按钮，进入题目详情界面，题目的属性有：题目名称、描述、链接、flag等。完成题目后，界面显示：您已经完成了这道题您可以选择提交文件/点击右上角编写writeup

点击writeup，进入一个Markdown界面，一人一题只能有一个writeup，当再次编写，会显示“修改成功”，而不是“发布成功”。在writeup列表可以看到自己和别人的writeup，方便用户之间互相学习。

2.1.5 公告栏

管理员可以发布公告，发布后的公告在公告栏以列表形式列在公告栏界面，点击公告的标题，可以进入每一条公告的详情界面。

2.1.6 排行榜

个人排行

战队排行

分类排行

2.1.7 战队界面

2.1.7.1 战队列表

战队按照战队积分列表显示，点击战队名称，可以转到战队信息界面。

2.1.7.2 战队管理

（1）创建战队

（2）申请加入战队

申请后必须队长同意才可以加入，一次只能申请一个战队

1. 退出战队

队员退出，战队信息、排行等动态变化即可

当队长退出时候，队长身份自动交给战队成员中贡献度最大的用户，如果战队中只有队长，则战队解散

1. 解散战队

2.1.8趋势图

一个多折线图，显示top10战队的分数变化。可以拖动滚动条查看各个时间段的战队分值变化

2.1.9消息系统

* + 1. 消息有多种类型：

1. 系统消息：当用户注册成功，创建、退出、解散战队等时系统自动发送消息
2. 战队申请：申请战队、同意申请战队需要通过消息系统来实现，并不是直接加入
3. 管理员消息：管理员可以指定用户发送消息进行一些通知啥的
4. 用户消息：用户之间可以发送站内信
   * 1. 消息管理由管理员管理

2.1.10 信息界面

2.1.10.1个人信息界面

基本信息、题目完成的饼图、提交情况、每一提排行（一二三血）

2.1.10.2 战队信息界面

除了采用个人信息界面的形式展示战队信息，还有战队管理的界面，队长可以选择退出、解散战队、移除某用户等；普通队员可以选择退出战队等

2.1.11 管理员界面

我们在数据库为某些用户赋予管理员的身份，在个人信息界面可以看到进入管理员界面、其他用户没有。三个基本模块：公告管理、消息管理、题目管理

2.1.11.1公告管理

公告的发布、编辑、删除

2.1.11.2 消息管理

管理员可以选择为某些用户发送通知，也可以查看用户之间的消息情况。当然也包括对消息的编辑、删除等

2.1.11.3 题目管理

解决了的后台人员在数据库中发布题目的麻烦，采用图形化的界面发布题目，可以设置题目的题目、描述、类型、难度、分值、链接、flag等，也可以对题目进行编辑、隐藏等操作

#### 漏洞扫描模块介绍

* 1. 爬虫模块

3.1.1 大致描述：

爬虫模块使用遍历搜索进行网页url的爬取。具体逻辑是，将网站的url进行输入，将其作为root url进行操作。

第一步是初始化，为了保证尽可能爬取该网站下较多的url，对url实现了切割操作，将url中协议+域名+端口以外的内容进行了排除形成了一个新的url后再次存入要查找的url队列中，这样一般情况下就有两个url，可以增大查找页面的范围。之后，使用request请求，得到返回后观察是否url可以访问，如果不能则从查找队列中取出，并且不作为有效url存放在字典中。如果当前的url全部无效，则退出逻辑。

第二步是查找，对于每一个在队列queue中待查找的url，都对其进行请求，然后在得到的html界面中寻找超链接标识，并将值获取。

第三步是判断有效性，每一次查找都会获得一串url的列表，这之后对当前得到的url列表进行request判断，如果成功则他可以作为url放入可用url的字典中，并且将加入到下一次查找queue中。

之后重复迭代二三步，直到全部查找队列为空。

3.1.2 处理细节：

会对于一些特殊链接进行排除，比如”javascript:”或者”mailto”，因为这样会造成未知请求错误，并且没有必要进行request判断，因为它是最耗时间的。

另外，对于.html的页面，只将它作为待查找对象放入queue中，但是不会将他作为有效url放入字典中，因为从html中可以获得有效链接，但是html本身对我们并没有意义。

除此之外，爬虫在3.1.1中的第三步里采用了多线程以集中提高速度（多次request是拖慢速度的元凶）。这里尽量将多线程的范围缩小，以避免线程之间进行互相干扰的操作。这里的操作是线程之间无干扰的。

另外request判断会进行读取和请求的时间限制，如果时间过长则认为无效链接。如果重定向次数过多，也会认为是无效链接。主要是考虑效率以及爬虫安全性的问题。

3.1.3 信息伪造

本爬虫对proxy和user-agent进行了伪造和随机化处理，这里写的是每次一遍，实际上可以改成5次一变。这主要是防止以同样的请求头请求过多后被服务器识别，这样做的话会增加爬虫的隐蔽性。 user-agent直接随机从文件中读取即可；proxy则使用连接池，从专门提供proxy的网站中获取代理ip，并将信息存储到数据库中，此时再从数据库中获取。通过本地127.0.0.1：8080即可获取返回的可用ip。

3.1.4 数据处理

对于得到的有效url数据我们有两种处理方法：1. 将url存入到文件中，便于其他模块取用。 2. 将该缓存的字典dict直接在内存中被其他模块调用。

这里我们只写了方法2，然后由于网站的url都太多，并没有真正完整的输出过，故此处只说明原理。主要说明方法2：方法2是调用之前的有效url字典，字典以{url:response}格式实现，也就是说对于之前判断过有效的url，之后的模块不需要再次请求一遍url，只需要直接从response中获得信息即可，这样可以大幅减少时间，但缺点是必须将流程综合化，不利于灵活调用，因此还存在第一种方式。

3.2 SQL注入漏洞模块

3.2.1 基本实现功能与原理

实现了五种漏洞检测：基于报错的注入，时间盲注，布尔盲注，万能密码，宽字节注入，核心思想就是**利用后端代码在与数据库交互时对用户的输入检查不仔细导致出现泄露数据库版本，种类等信息，数据库中用户信息甚至数据库管理员的账户密码。**在写检测脚本的时候主要利用在本地搭建的SQLabs进行各种类SQL注入的测试。比较全面且方便。脚本可以实现get和post两种注入检测，但实际中一般都是post通过表单来进行注入，get的检测不太常见。

3.2.2 基于报错注入

我们可以故意构造带有特殊符号（例如：’,’’等）的数据库语句利用数据库查询时语法不正确的报错信息来泄露数据库的版本信息，从而进行进一步的注入，还可以通过不同特殊字符输入的返回结果对比来判断后端是整数检测还是字符检测，从而进行进一步的注入。**基于报错的注入最主要的作用就是获取服务器数据库的版本信息以及后端注入类型（整数型还是字符型）和语法模式，为进一步注入收集信息。**

3.2.3 时间盲注

当出现即使我们输入使数据库语法错误的特殊构造语句时，网页也不会报错的情况，（界面返回值只有一种,true 无论输入任何值 返回情况都会按正常的来处理），这时候就可以使用时间盲注来判断我们的特殊构造到底是不是数据库的语法，主要利用sleep()函数来使服务器沉睡来判断我们的闭合符号是否使用正确，例如：’ or sleep(5)#，若数据库使用’来闭合，则会沉睡5秒钟才会相应用户的请求，所以我们检测其相应时间是否大于5秒来判断是否存在输入，实现的方法为在向网站发送请求时设置timeout为sleep()中的值少一，当其在这个时间范围内未响应的话就说明大概率存在时间盲注。**存在的问题：当网络条件不好时，没办法判断是网络太卡导致服务器没有响应还是服务器真的在沉睡，当然，在本地测试当然不存在这个问题。**

3.2.4 布尔盲注

布尔很明显分为Ture和Fales，也就是说网站只会根据用户的注入信息返回Ture跟Fales，也就没有了之前的报错信息。所以我们就只能用and/or/or not等配合注释符号等不断注入看返回结果来不断推测是否存在注入点和和数据库的信息，例如：1’or 1=1 --+，其实脚本的作用真的很有限，只有人工不断测试，才能获得数据库的有效信息，一个注入脚本就可判断出数据库名字和表名字的才有意义，这种基础的注入其实没啥意义，

3.2.5 万能密码

主要原理为利用数据库的注释语句和闭合方式来使数据库在搜寻是否存在用户输入的用户名和密码时，将密码注释掉且用户名恒等成立，从而使登录成为百分百的情况，例如：username=’ or 1=1 --+&password=任意 当闭合方式为 ’ 时，即可成功登录。这种输入方式的危害就不用说了吧，直接绕过检查直接登陆，可以对网站进行违法操作却不留下痕迹，如果是管理员的话则危害更大。

3.2.6 宽字节注入

有攻击就有防御，网站的开发人员在知道常见的sql注入测试闭合符号时就会对用户的输入进行过滤，比如当我们输入’时，后端代码有函数会将’转义为\’，从而让数据库认为其是一个’字符，而不是语法上的闭合字符，从而防止了部分注入检测攻击。但是如果后端是gkd编码方式的话（一种存在汉字的编码，两个字节编码为一个汉字），我们就可以利用宽字节注入使\’的\和我们输入的一个字符在一起转义为一个汉字，从而使’再次暴露出来，造成前面所说的那些基础类型的注入漏洞。常见的测试为：%df，%5c等。

3.2.7 心得和遇到的麻烦

大部分代码都是自己写的，但是也的确上网找了很多资料，东家凑一点，西家拼一点，布尔和基于报错的注入是资料较多的，其他三个注入是在了解了原理之后自己写的，然后用SQLabs进行对应类型的逐个测试，能扫出来就说明成功。

遇到的一些麻烦：因为是用脚本扫描，所以我们没法看到真的网页，所以我们常用方法为先输入一个正常值，将其返回的网页记录下来，然后再输入我们特殊构造的值，将两者返回的网页进行对比，若两者不同，则认为后者存在错误即存在注入点。但是遇到一些特殊情况，比如网页会把你的用户名和密码打印到屏幕上，这样就导致了每一种输入就有一种返回值，所以没办法判断哪一个返回网页是错误时返回的。还有就是万能密码，当登录成功时，我们通常需要检测返回的网页带有透露成功信息的语句时才会判断为登录成功，例如返回了success，flag等，但是假如返回了一个图片，我们则没了办法判断。。。想实现自动化真的太难了。。

3.3 xss漏洞扫描模块

3.3.1 基本实现功能

实现了检测经过后端的xss，在扫描网页后输出：xss漏洞类型、payload有效要求，响应中需要去除的内容、以及可供参考的攻击方法

3.3.2 扫描逻辑

首先是发送payload，然后查找payload的输出点，接着再看这个输出点对应上边正则表达式中的哪个位置。找到对应的位置之后再看如果在这个位置要利用成功需要哪些字符不被转义，然后在响应中查找payload，确定payload中‘不可被转义的特殊字符’前是否被加上了反斜杠，如果有则认为被转义，并继续往下找，直到每个不可被转义的字符至少在响应中未被转义地出现了一次，才会认为存在漏洞。

如果可以找到，就说明这个参数的值是会被输出到响应中的（先不管有没有过滤），下面的逻辑就要检测这个参数是否可以利用。首先需要确定response-payload的输出点在哪个位置（script标签内？注释内？引号内？等等），然后再根据输出点来确定在这个位置利用成功需要哪些字符不被过滤。

3.3.3实现细节

3.3.3.1防止后端使用waf/ids时对payload进行拦截

构造payload的时候，两个随机字符串中间添加上<>"';字符的随机排序。但是后端往往使用waf/ids时对payload进行拦截，所以使用一个for循环，针对larger\_char\_pool和smaller\_char\_pool进行单独的测试（这两个list分别是：(''', '"', '>', '<', ';') 和('<', '>')。故使用了一个smaller\_payload来避免过于明显的payload[即有较少的特殊字符]）。

3.3.3.2 payload之前加一个单引号'

当使用larger\_char\_pool(''', '"', '>', '<', ';')时候，在payload之前加一个单引号的原因如下：

1) .试图触发xss

2) .故意构造一个错误的sql语句用于报错，试图在报错信息中寻找触发点

构造完payload之后直接发送，然后获取返回内容。这里就要去掉之前的单引号'，至此已经完成了它的使命，从响应中去掉，防止干扰。

3.3.4 总结

总的来说，经过后端型xss，这里先是通过正则判断输出点在dom树中的位置，然后确定如果想要触发xss需要哪些字符不被过滤，接着再根据响应包中的payload中的特殊字符前是否有反斜杠来确定其是否被过滤从而确定是否满足上面的条件，满足则存在漏洞。这样的检测确实存在很多不足。对于某些过滤某些标签字符的网页也发现扫描结果错误现象。

针对dom xss，一个貌似可行的检测方法是通过正则表达式来对一些有可能造成危害的函数进行匹配，并且匹配其参数是否可控。限于时间原因没有实现。

对于存在cookie或者需要登录才能进入的网站不能扫描。也尝试着带着cookie进入网页，但是结果并没有想象的那样。

3.4 弱口令破解模块

3.4.1 爆破基本原理

该爆破模块使用字典爆破，就是找到对应页面所含有的表格，利用requst.post方法传入对应文本框所需要的数据。但在此基础上，还针对验证码的识别实现了人工智能识别验证码的方法。

3.4.2 实现细节

3.4.2.1 发现真正的表单

考虑到有可能会出现页面中多张表单的情况，于是不能仅仅只扫描带”form”关键字的条目。这里采用的方法是，寻找输入框带有type=password的，即输入的消息以”\*”显示的输入框，再找到它对应的父元素表单，再从该表单中获取所有信息。因为即便有多张表，但需要输入密码的寥寥无几，一般来说可以认为是真正的登陆表。

3.4.2.2 对于表单的信息的处理

除了基本的定位用户名，密码的输入框以外。还需要定义这样几项：1. csrftoken的隐藏输入框，有些网站为了防止csrf引入了这个，由于显示在前端，不能此时造成困难，然而还是需要将其作为数据提交到表单中，不然是无法登录的。 2. captcha，即验证码的输入框，这里也需要定位，如果网站的登录需要验证码的话（这里仅指图片验证码）。

3.4.2.3 信息伪造

同爬虫

3.4.2.3 cookie的的处理

要使用request.session自动维护cookie或者手动维护cookie,这样可以欺骗服务器。有的服务器会在登录时set一遍cookie，获取图片验证码时再set一遍cookie，最后点击登录后再会重定向一遍进行一遍setcookie，尤其是最后一步如果不正确就会出现问题。本模块对这一个功能进行了模拟，然而并没有生效，并没有找到具体原因，但是出现的现象是：对于浏览器提交表单成功后，有一个302重定向并setcookie，但是对于爬虫程序来说，即便做了完善的伪造，还是无法得到与浏览器同样的结果。

3.4.2.4 对验证码的处理

这里主要是对我们自己的网站进行的针对性实验。实际上，由于时间不够，并不能训练出对任何网站都适用的图来，因此只对本网站的captcha进行实现，另外，由于时间原因没有对验证码进行剪裁，因此验证码的大小也必须是130\*50才可以，不然会无法识别。

逻辑主要是，如果判断有验证码的话，那么会对验证码进行请求，并下载到本地的文件夹中。为了判断识别的成功，这里做了一个判断，即可以选择智能判断或人工判断，加入人工判断的选项主要是为了保证程序运行的流畅性，毕竟下载图片这一步是任何网站的验证码都适用的（图片验证码）。当我们选择智能判断时，可以打开对应的文件夹观察是否人工只能输出的字符串与图片里的文字一致。

人工智能在google的colab上进行训练，对单个字符的识别率达到0.9999，目前没有出现过失败。然而人工智能的训练，任重而道远，对于带有较多划线的验证码，人工智能的识别率较低，需要在训练时增加切割技术等进行多次训练才可能达到效果。

3.5 文件上传模块

3.5.1 基本原理与核心思想

文件上传漏洞是一种危害性大且很常见的漏洞，核心原理为上传一个可执行的网页文件（例如php，apx等），当服务器执行了这个文件时，会种下病毒或者泄露重要信息等，从而破坏服务器。但是现在大部分的网站都会对上传文件的类型进行过滤（基本对于后缀进行过滤）。所以我们的扫描模块也分为了两大部分：文件后缀的白名单检测与上传成功的文件是否可执行的检测。写脚本时上网找了很多资料，大概有一半是自己写的吧，然后本地就用自己搭建的upload-labs进行逐个测试，调Bug，测试，调Bug。。。。

3.5.2 后缀检测模块，

经上网搜集资料，整理了基本市面上能看到的文件后缀类型，有400多种（为了缩短测试时间，还整理了一个基本的类型，共172个后缀）。通常一个mime会对应多个文件后缀类型（例如：image/jpeg jpeg jpg jpe），然后用线程池以多线程方式异步生成临时文件，与后缀连接逐个测试，然后将返回的页面与正确或者错误的正则关键字进行匹配，来判断哪些文件后缀是允许上传的。

3.5.3 是否可执行检测模块

写了php，gif，jpg，jsp，asp，aspx，imagetragick，.htaccess八个template（已经经过特殊构造好的，可执行的本源漏洞文件，例如gif文件实际上是将php文件的头文件改为了G98强制转变为gif,实际上是可执行的php文件），然后将这八个template分别与上面检测的后缀白名单中的后缀进行拼接，还会进行大量的后缀混淆（例如：["php1","php2","php3","php4","php5","phtml","pht","Php","PhP","pHp","pHp1","pHP2","pHtMl","PHp5"]），零字节截断，双后缀拼接等，然后测试是否可以上传成功，然后再访问上传文件对应的路由，将返回的页面与提前配置好的如果执行了会出现的正则表达式进行匹配，匹配成功则说明可以成功执行，并将其提取出来，结束搜索。

3.5.4 心得与遇到的问题

其实我个人觉得文件上传是一个受限制很大的漏洞，因为其需要知道的信息和需要人工参与的成分太多，例如：我们需要知道上传失败时的报错信息或者成功时的提示信息（没有提示时我们则无法测试扫描），我们需要知道上传文件的存储路由（因为没有存储路由的话我们就没办法访问，也就没办法判断是否执行成功，即使可以执行我们也不法获取信息），但是一个正常的网页都不会告诉用户自己的文件上传路径，所以这本身就是一个困难，因为在本地用Upload-Labs测试所以可以知道对应的上传路径。此外，即使我们知道了文件上传的路径，但是如果服务器端对上传的文件名进行了随机化，我们就无法访问到对应的路由，仍然无法用脚本来判断是否可执行。所以离真正实现自动化还有很远距离。

3.6 输出模块

输出模块将以弱密钥破解模块为范本进行输出，由于时间原因并不能全部相连并输出。

3.6.1 html输出模块

这里html由jinja2引擎实现，可以将后端的数据以render\_template的形式输出到html文本中，并依照jinja2语法做一些python的操作，非常的方便。会在程序结束时开启一个本地的127.0.0.1:5000端口进行页面输出（仅在调式模式下进行过观察）

3.6.2 word输出模块

对弱密钥破解模块的一些信息进行格式化后输出到word中，在一个文件夹中生成word文本。