

**软件测试课程设计报告**

( 2020 -- 2021 年度第 二 学期)

名 称： Python+Flask留言板系统

院 系： 控制与计算机工程学院

班 级： 软件1801

学 号： 120181080414、120181080419、1201801080412

学生姓名： 王嘉承、王舰航、姜逸龙

成 绩：

日期： 年 月 日

**一、课程设计(大作业)的题目和要求**

基于Python+Flask，开发一个web系统；包含常见的软件漏洞范例页面，并给出正确的消除缺以后的程序；

具体要求和内容： 开发的系统包含以下漏洞的功能页面，以及正确的页面或者系统（可以开发出2个一样的系统，其中一个存在以下的问题和缺陷；另外一个不存在）；

1. 存在跨站脚本（XSS）漏洞的系统:跨站脚本攻击发生在客户端，可被用于进行窃取隐私、钓鱼欺骗、偷取密码、传播恶意代码等攻击行为。
2. 存在跨站请求伪造（CSRF）漏洞的系统：恶意网站通过脚本向当前用户浏览器打开的其他页面的URL发起恶意请求，由于同一个浏览器进程下Cookie可见性，导致用户身份被盗用，完成恶意网站脚本中指定的操作。该漏洞会导致用户信息泄露，被盗用后进行恶意操作。
3. HTTP Header注入漏洞系统：WEB程序代码中把用户提交的参数未做过滤就直接输出到HTTP响应头里，导致攻击者可以利用该漏洞来注入到HTTP响应头中实现攻击。
4. 目标遍历漏洞系统：目标遍历是由于WEB服务器或WEB应用程序对用户输入文件名称的安全性验证不足而导致的一种安全漏洞，使得攻击者通过HTTP请求和利用一些特殊字符就可以绕过服务器的安全限制，访问任意受限文件，甚至执行系统命令。
5. SQL注入漏洞：SQL注入攻击，被广泛用于非法获取网站控制权，是发生在应用程序的数据层上的安全漏洞。
6. 文件下载漏洞：WEB应用程序在处理文件下载时，接收用户指定的路径和文件名进行下载，攻击者利用此漏洞来下载服务器的其他文件甚至任意文件。
7. 文件上传漏洞文件：文件上传的WEB程序未对文件类型和格式做合法性验证，导致攻击者可以上传WEB Shell或者非期望格式的文件。
8. 日志不全或无操作日志的系统；
9. 渗透测试安全漏洞系统：

A 身份验证漏洞；

B 越权访问数据（垂直越权，平行越权）；

C 敏感数据泄露或者传输显示时可见

1. **课程设计（实验）正文**
2. **XSS漏洞**

1.1 产生原因

跨站脚本攻击是指恶意攻击者往Web页面里插入恶意Script代码，当用户浏览该页之时，嵌入其中Web里面的Script代码会被执行，从而达到恶意攻击用户的目的。

1.2 系统中xss漏洞的体现

在一个类似留言板的功能中，让用户在显示框输入一个内容：<button>123</button>，然后就把这个内容当做显示框内容显示。但是，该网页把用户发送过来的数据，未经处理直接写入返回的html流，最终这条语句被执行，生成了一个按钮。这就是XSS漏洞所在。

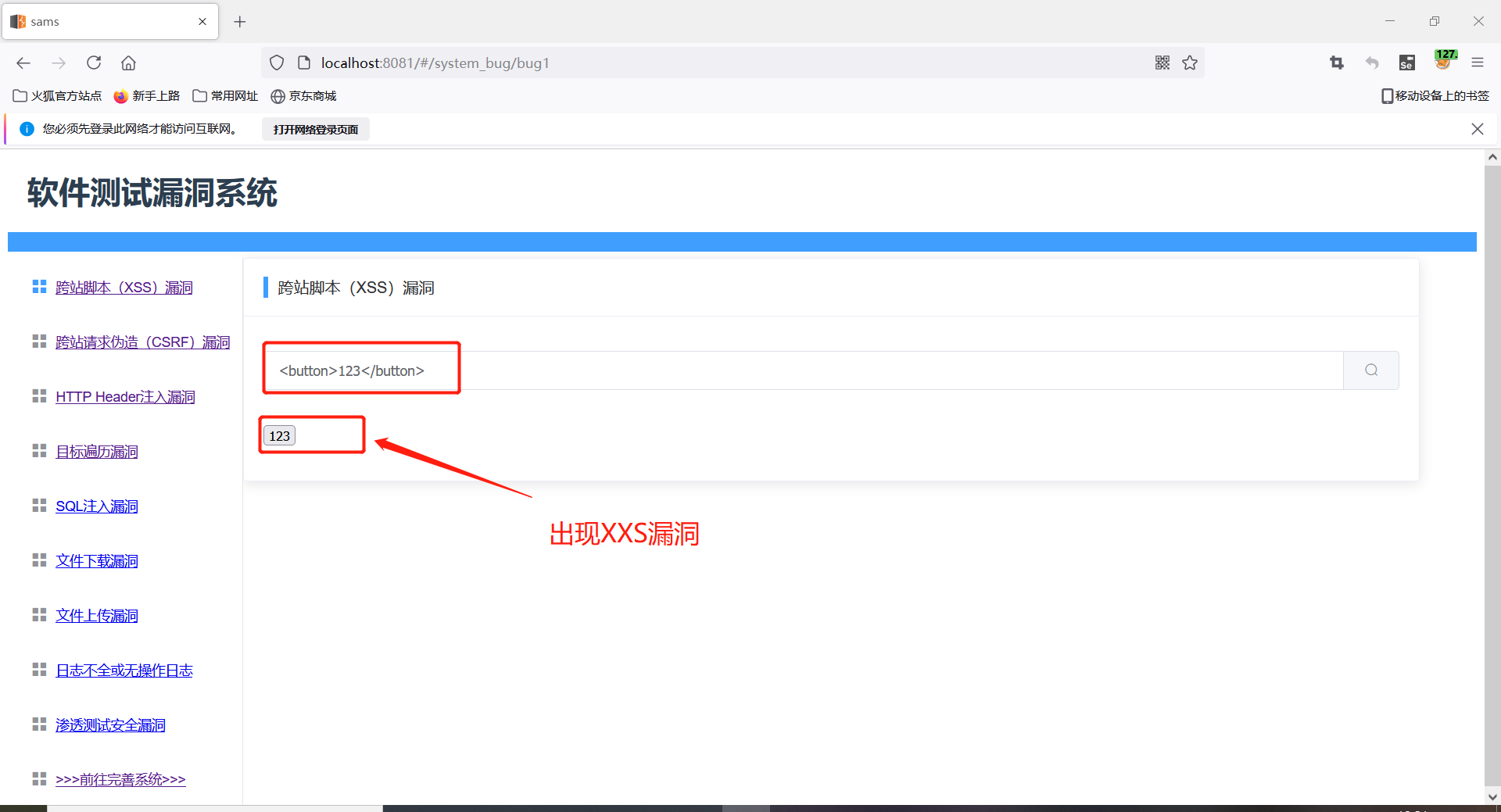
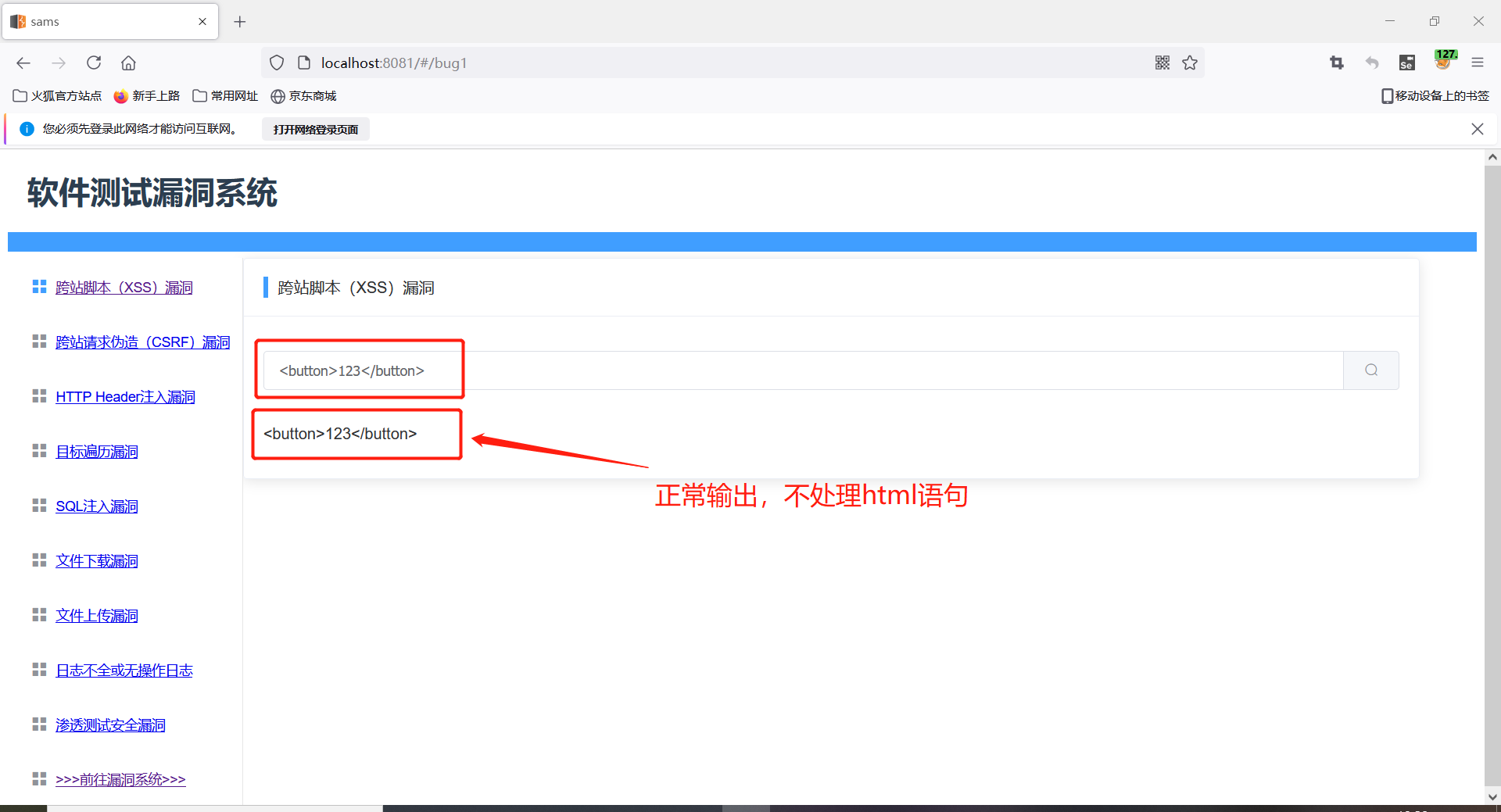


图1.XXS漏洞的体现

1.3 防范和修复

XSS防御的总体思路是：对输入进行过滤，对输出进行编码。

也就是对提交的所有内容进行过滤，过滤掉会导致脚本执行的相关内容；然后对动态输出到页面的内容进行html编码，使脚本无法在浏览器中执行。虽然对输入过滤可以被绕过，但是也还是会拦截很大一部分的XSS攻击。

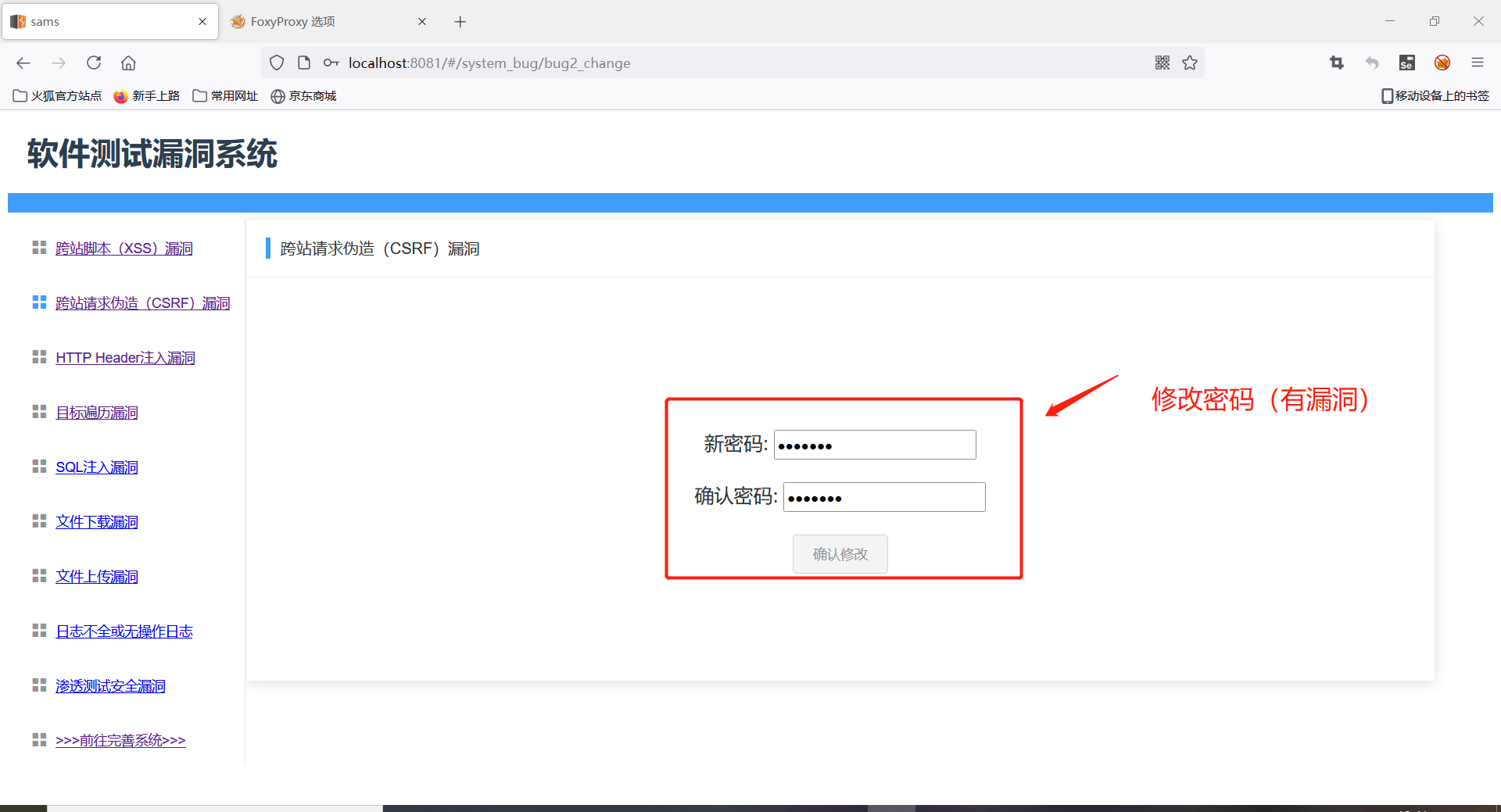


1. **CSRF漏洞**

2.1 产生原因

一个用户通过浏览器成功登录一个网站，登陆成功后，服务器会返回一个该用户的唯一标识放入浏览器Cookie中，以此作为用户之后操作的唯一凭证。假设此时该用户在此网站中请求一个表单类的网页，这时候用户又打开了另外的一个网站，而这个网站是一个病毒网站，它直接窃取了唯一身份凭证，通过唯一身份凭证，病毒网站直接进行用户所做的表单提交，而服务器是通过这个凭证来匹配用户信息的，服务器这时候无法识别病毒网站所做的操作，误以为是用户操作，此时可能造成用户严重损失。

2.2 系统中csrf漏洞的体现

​用户成功登录后，进行改密操作，由于系统没有进行任何防护，通过burp suite拦截用户的改密请求，抓取到用户的账号和密码，生成csrf攻击代码页面，若用户在登陆状态点击了此网页的按钮，密码被成功修改。

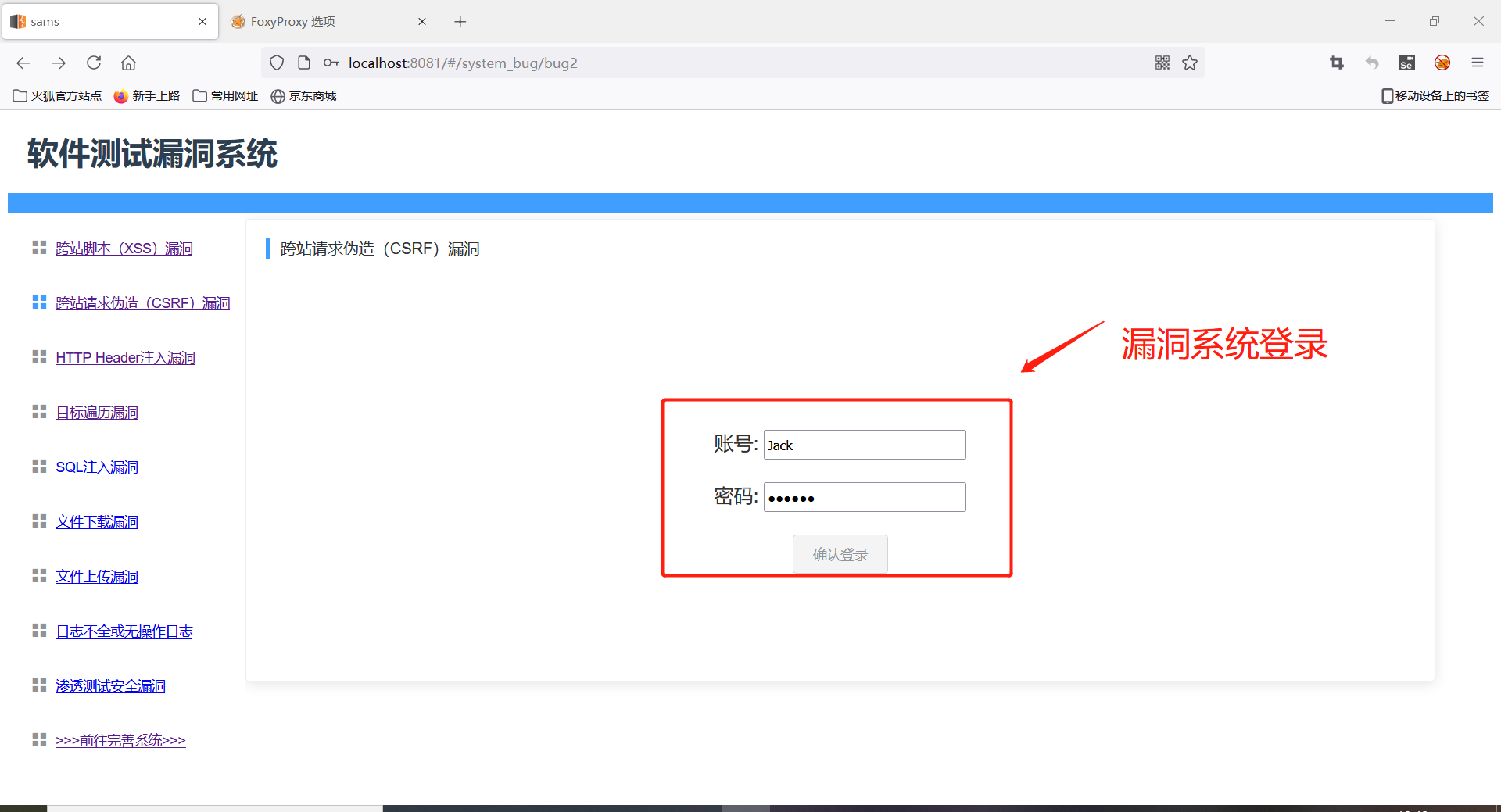
图3.出现CSRF漏洞

图4.漏洞系统登录

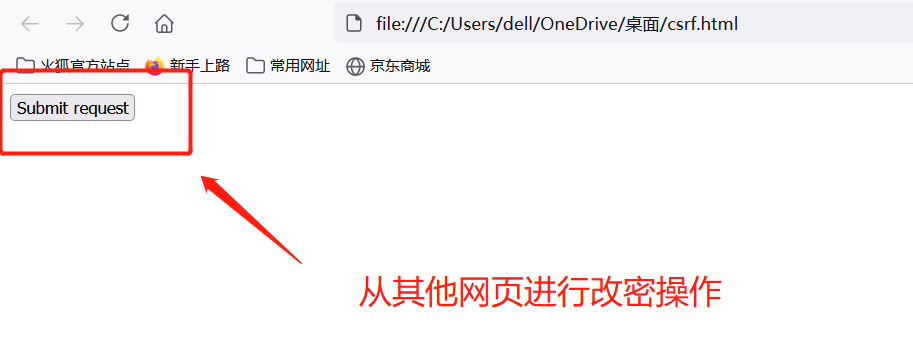
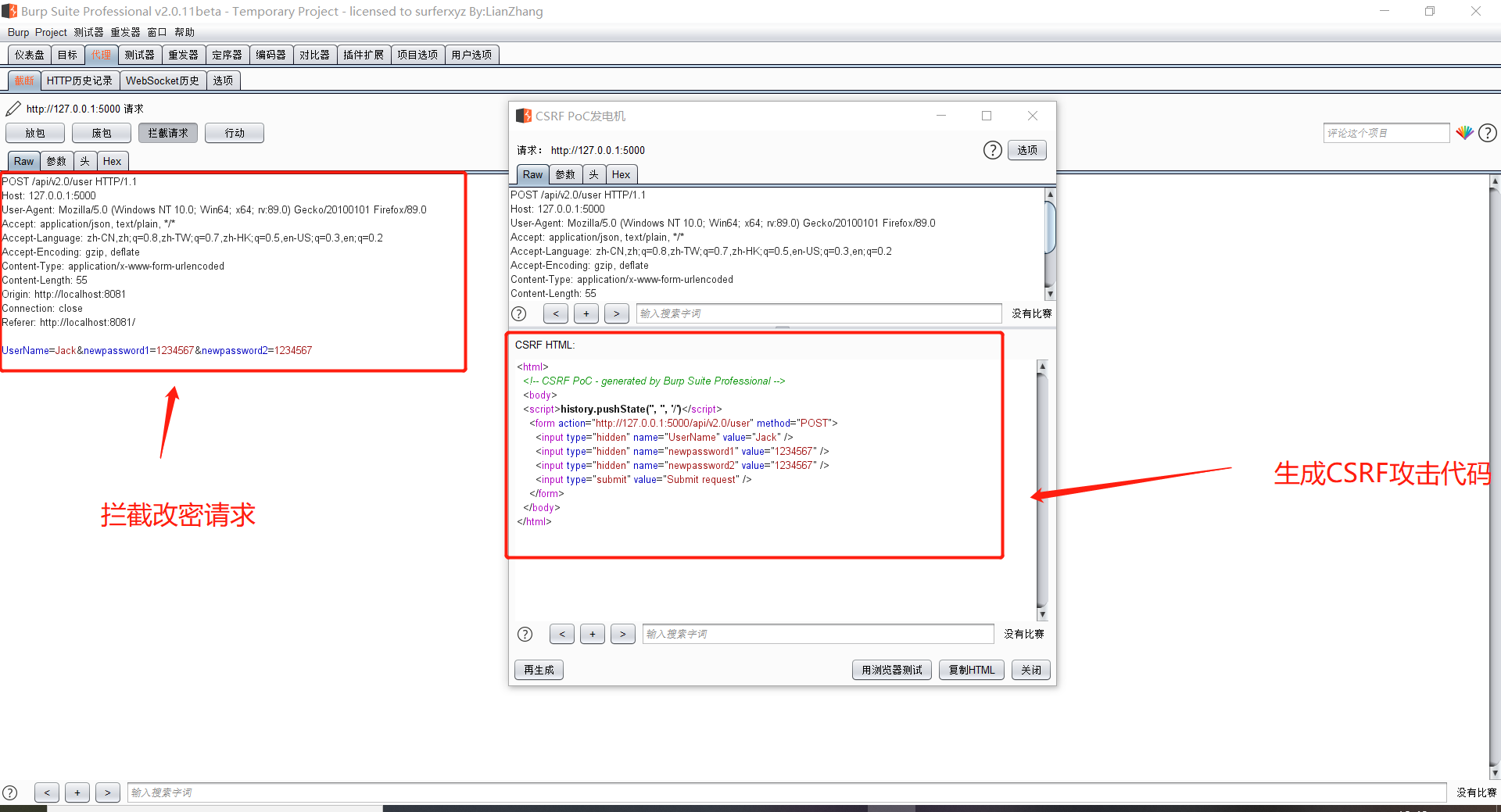
图5.利用burpsuit改密请求

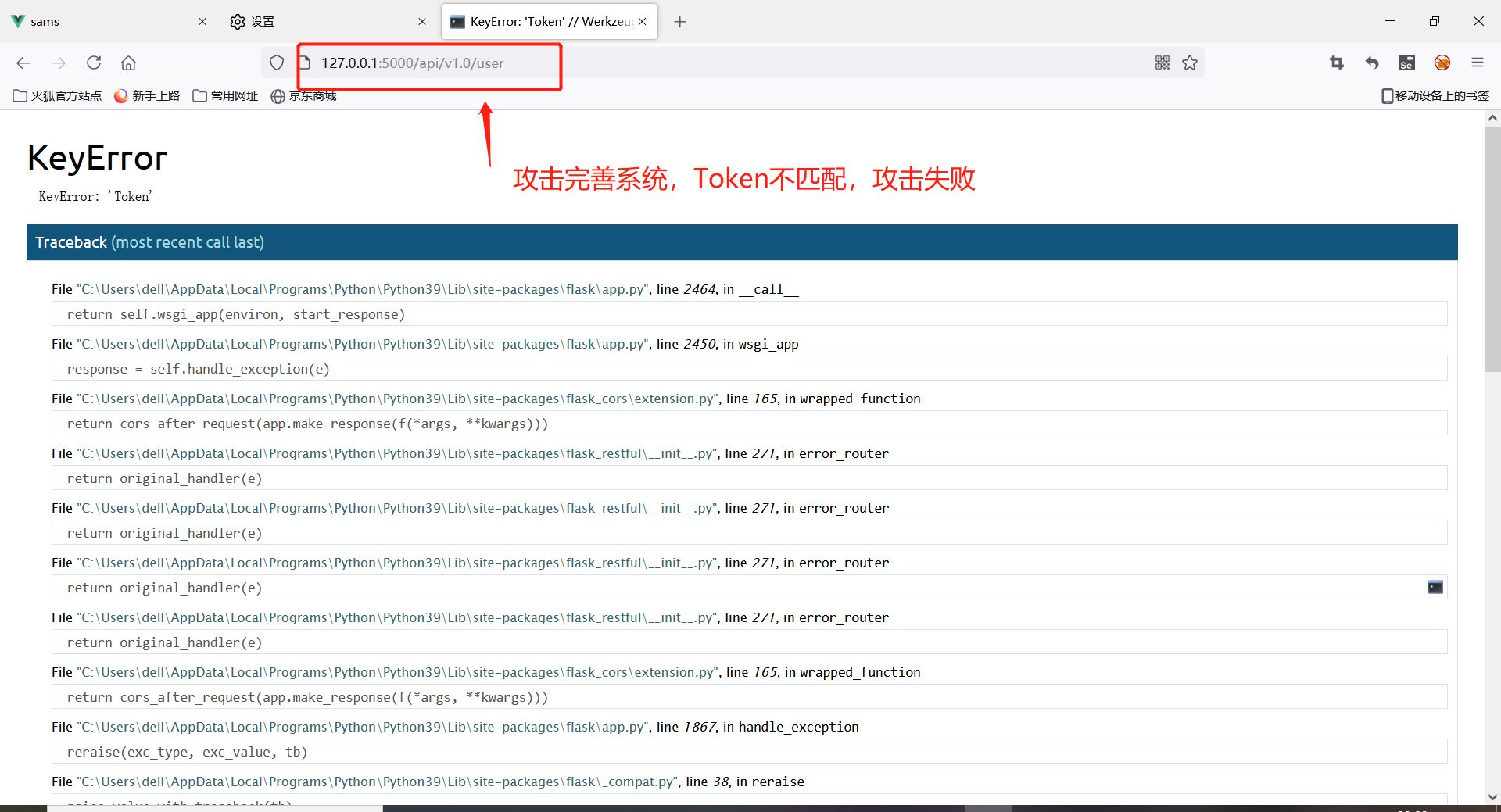
图6.从其他网页进行改密操作

2.3 防范和修复

（1） 用户登录成功后，获得后端分配的一个随机token，病毒网站无法获取到token，后端进行token验证，不成功则拒绝请求

（2） 二次验证，通过让用户输入原始密码，攻击者在不知道原始密码的情况下，无法进行CSRF攻击

图7.改密成功

图

1. **HTTP Header注入漏洞**

3.1介绍及产生原因

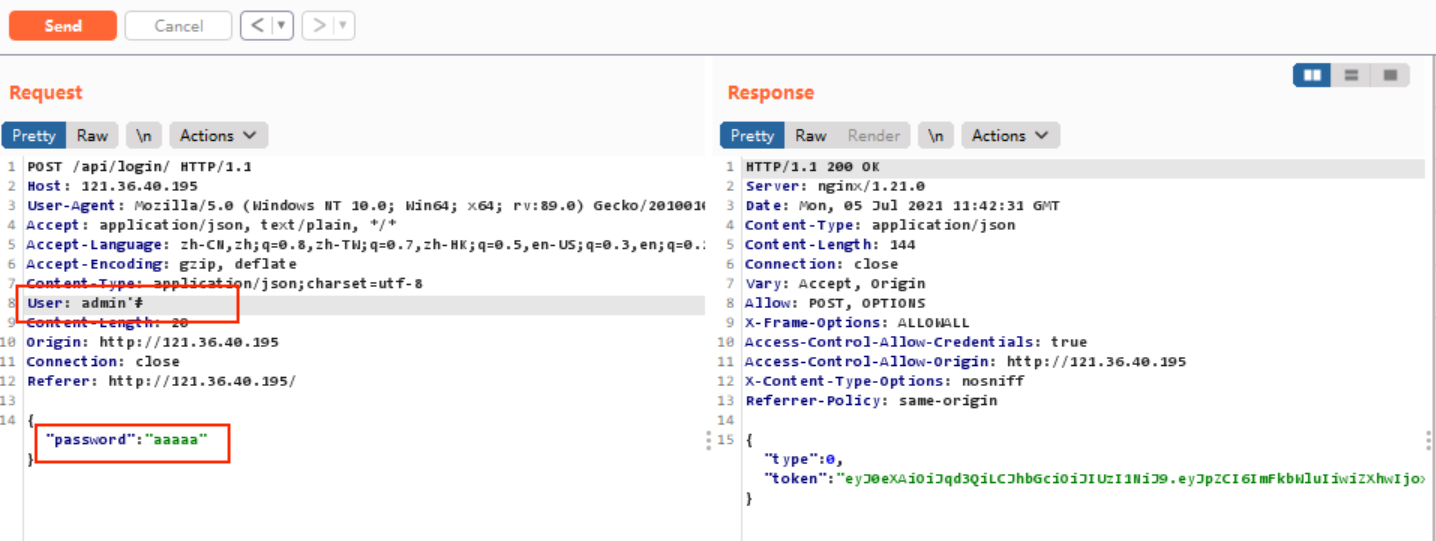
有些时候，后台开发人员为了验证客户端头信息（比如常用的cookie验证），或者通过http header头信息获取客户端的一些信息，比如useragent、accept字段等等，会对客户端的http header信息进行获取并使用SQL进行处理，如果此时没有足够的安全考虑则可能会导致基于http header的SQL注入漏洞。

3.2如何攻击

首先，用bp工具对网站登录页面进行抓包，然后对user agent数据（这里不一定是user agent，你怀疑哪一个你就改哪一个）进行测试，看是否存在sql注入，也就是输入单引号或者其他违规语法，看后台返回的信息中是否存在syntax error（语法错误），确认存在sql注入以后，就可以渗透了。

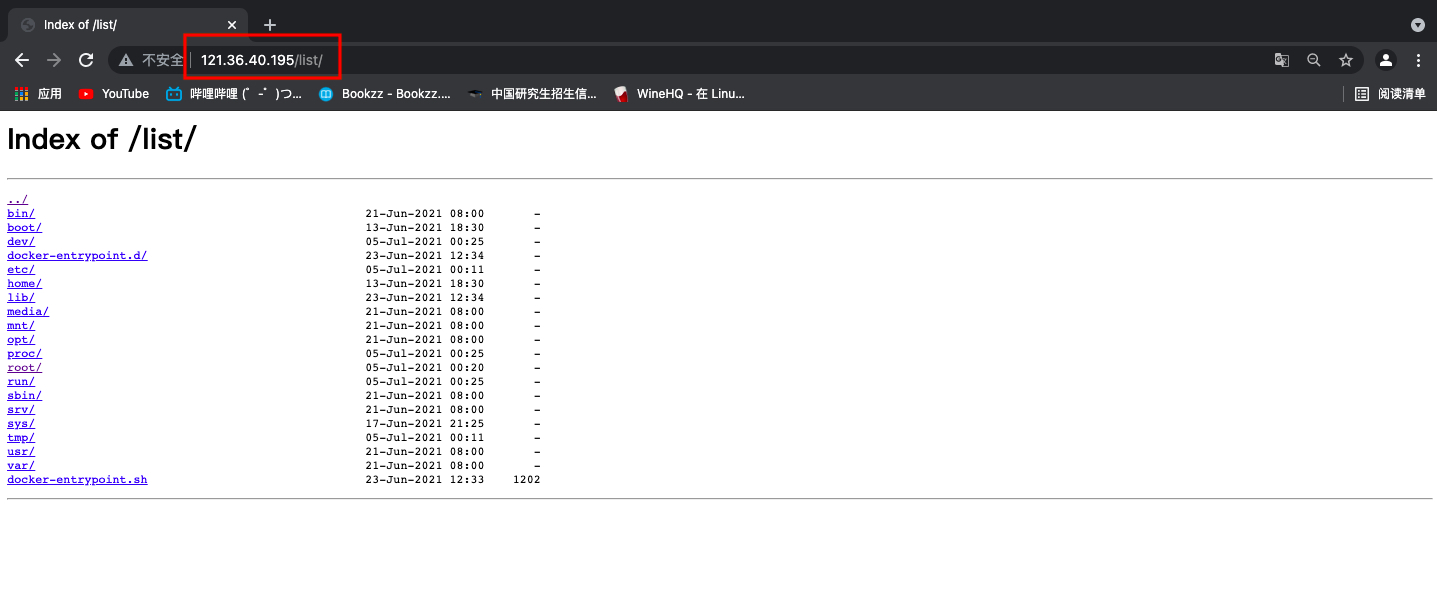
3.3如何防范和修复

    同SQL注入的防御。





1. **目标遍历漏洞**



1. 文件上传漏洞

截图

1. 日志不全或无操作日志漏洞

可以将下列代码放在任何需要打印日志的地方：

# 设置日志的记录等级

logging.basicConfig(level=logging.DEBUG) # 调试debug级

# 创建日志记录器，指明日志保存的路径、每个日志文件的最大大小、保存的日志文件个数上限

file\_log\_handler = RotatingFileHandler("logs/log", maxBytes=1024 \* 1024 \* 100, backupCount=10)

# 创建日志记录的格式 日志等级 输入日志信息的文件名 行数 日志信息

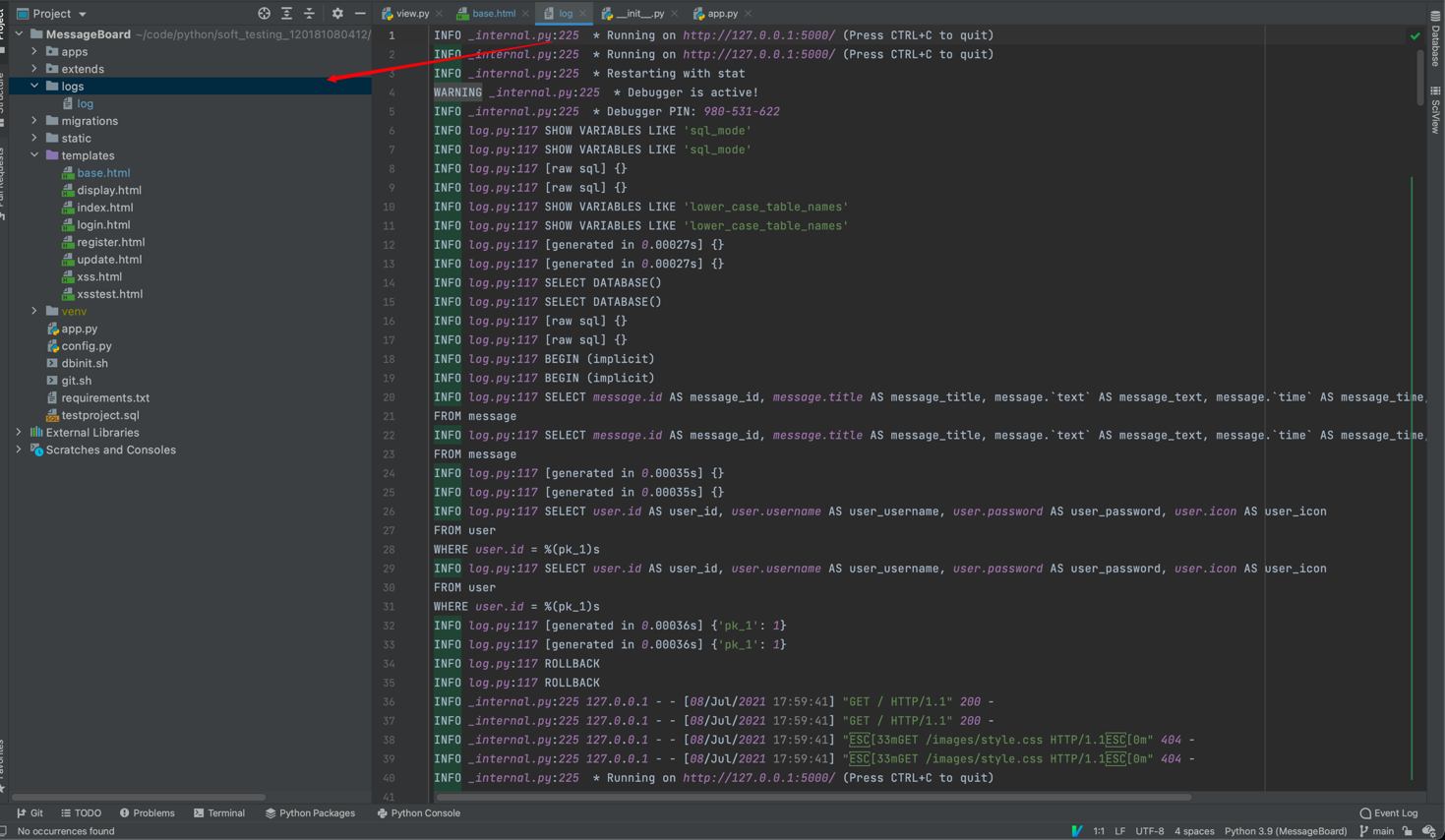
formatter = logging.Formatter('%(levelname)s %(filename)s:%(lineno)d %(message)s')

# 为刚创建的日志记录器设置日志记录格式

file\_log\_handler.setFormatter(formatter)

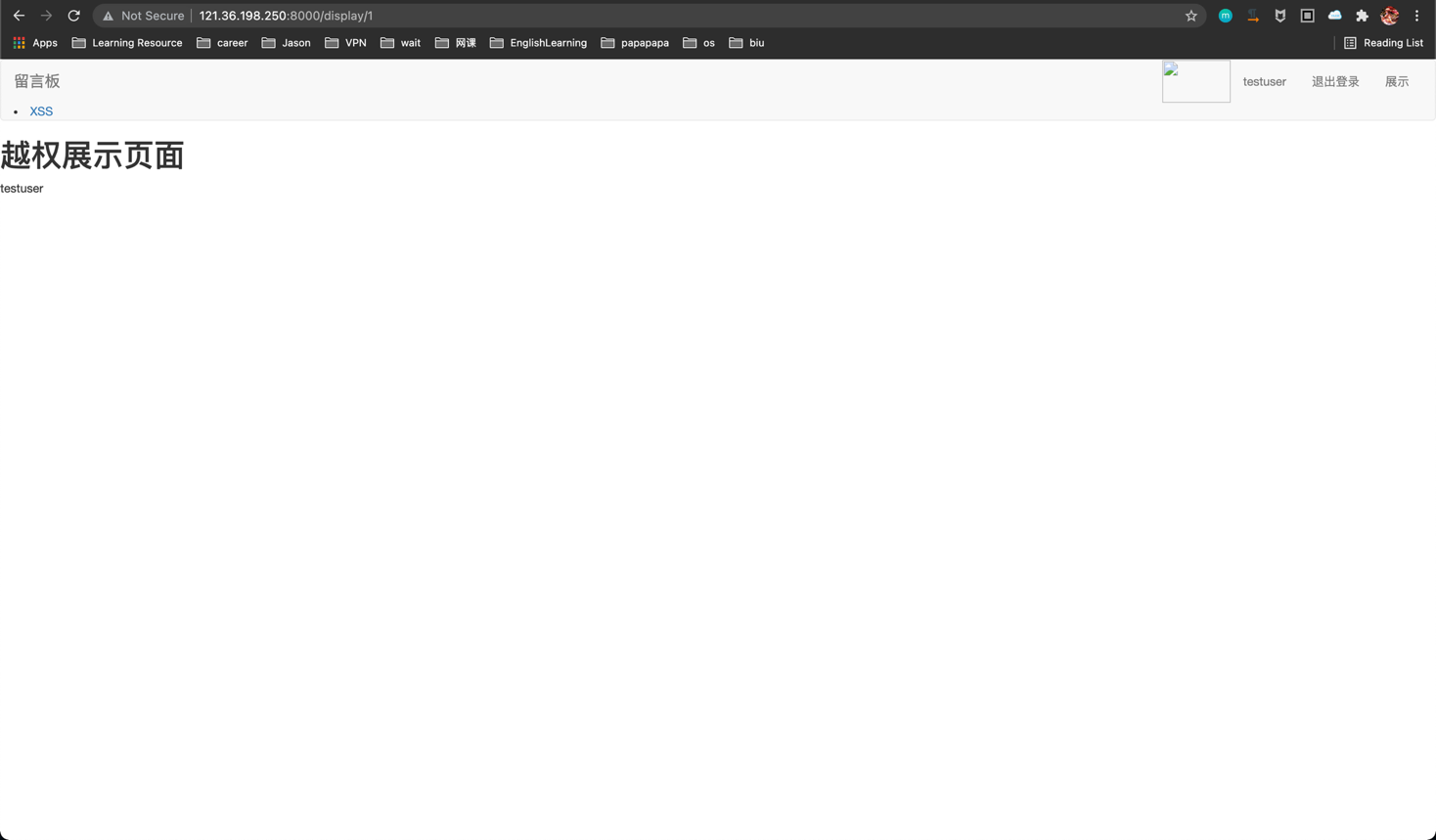
# 为全局的日志工具对象（flask app使用的）添加日志记录器

logging.getLogger().addHandler(file\_log\_handler)



1. 越权访问数据

用户可能根据URL中的数字来猜测其他用户的id，进而盗取其他用户的隐私数据



解决方案：

不在地址栏中显示，而采用cookie的方式直接获取当前用户id，不将数据暴露在外，具体实现为头像上传的逻辑

1. 敏感数据泄露或者传输时可见

截图（海王大佬）

1. **课程设计实验(大作业)中遇到的问题**

**常规MySQL安装过程太过繁琐，故采用docker容器技术安装，使用容器分配的虚拟ip连接MySQL数据库**

**XSS漏洞中script脚本可嵌入网页代码但无响应**

**SQL注入漏洞无法通过登录表单进行注入**

1. **课程设计（大作业）总结或结论可嵌入网页代码  
    通过本次课程设计，我们了解了在不借助框架的情况下如何进行相关漏洞的开发，以及其相应的解决方案，收获颇丰，不过还是有一些漏洞没有实现出来**