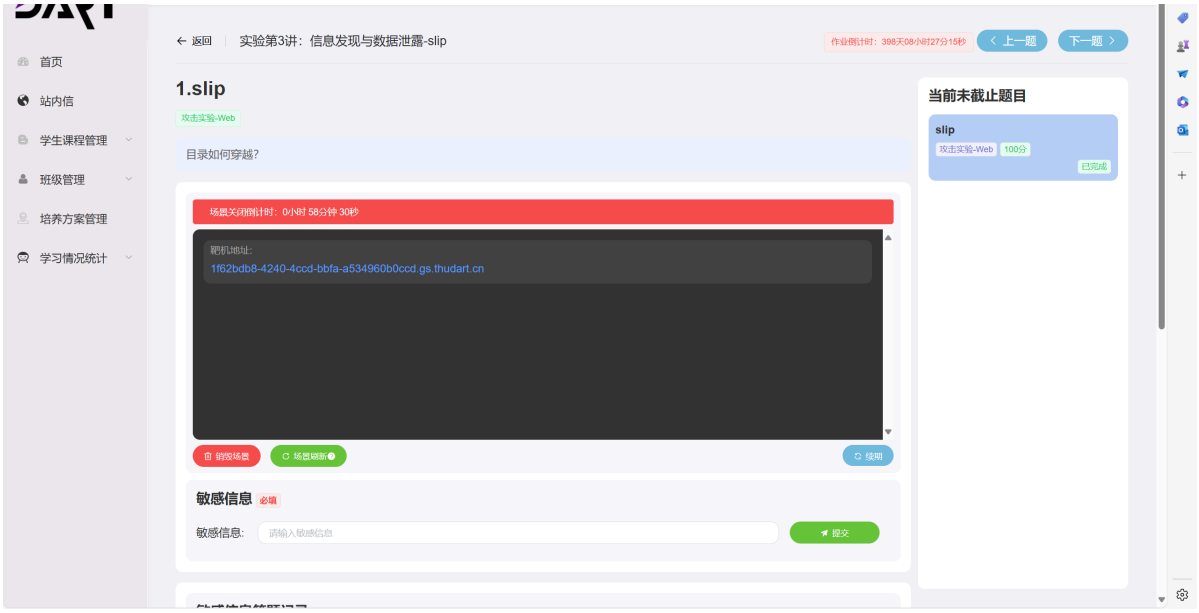
网络安全与对抗 第三讲 Slip

首先看下题目提示，猜测是目录遍历漏洞



目录遍历漏洞讲解

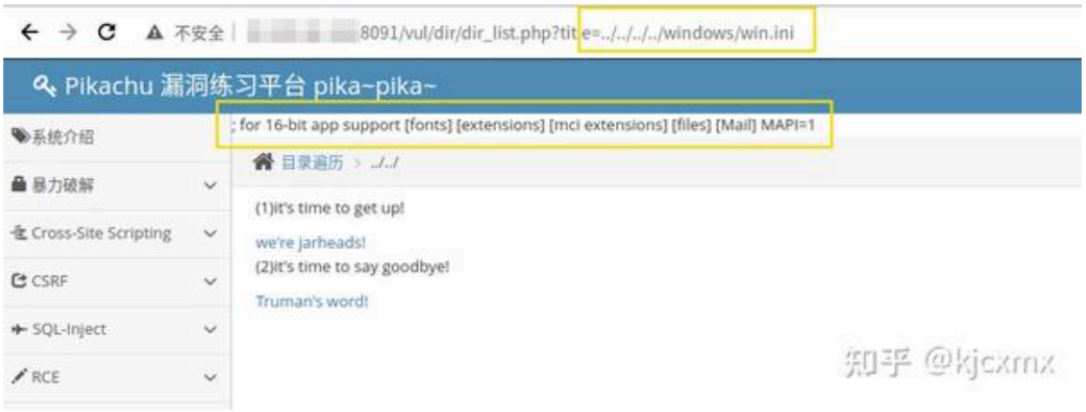
一、什么是目录遍历漏洞

目录遍历Directory traversal（也称文件路径遍历、目录穿越、路径遍历✦、路径穿越）是一种允许攻击者在未授权的状态下读取应用服务上任意文件的安全漏洞。这包括应用代码✦、数据、凭证以及操作系统的敏感文件。在有些情况下，攻击者还可能对服务器里的文件进行任意写入，更改应用数据甚至完全控制服务器。

二、目录遍历漏洞成因

程序系统在实现上没有过滤用户输入的../之类的目录跳转符，允许攻击者通过提交目录跳转符来遍历服务器上的任意文件。 比如： `http://www.test.com/index.php?file=image1.jpg` 当服务器处理传送过来的image1.jpg文件名后，Web应用程序会自动添加完整的路径，比如：`c://test/static/imgs/image1.jpg` ，然后web系统将读取的内容返回给攻击者。 若对文件名称的安全性验证不足，攻击者会使用 `../../../ect/passwd` 的文件名，将会导致访问非授权文件资源。

四、漏洞攻击+利用手法



直接使用../目录穿越

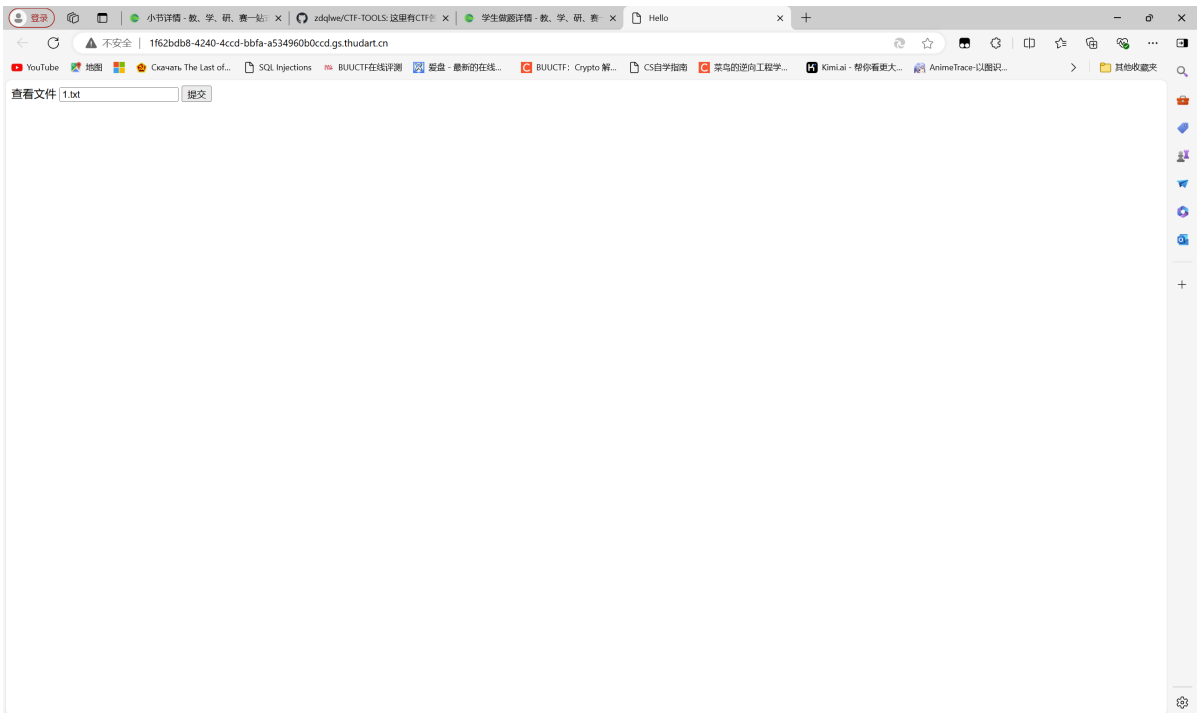
比如： `http://www.test.com/index.php?file=image1.jpg` ，服务器拼接成 `c://test/static/imgs/image1.jpg`

```
payload:
http://www.test.com/index.php?file=../imgs/image1.jpg
http://www.test.com/index.php?file=../../../../windows/win.ini
http://www.test.com/index.php?file=../../../../windows/win.ini%00.jpg
```

简而言之就是，程序系统没有过滤'../'这样的跳转字符，攻击者在提交文件名的时候就可以在文件名中插入目录跳转字符来查看程序中所有目录下的文件，例如一个网站想显示一个图片，那么payload可能就是http://www.test.com/index.php?file=images.jpg,那么攻击者只需要将提交的文件名改为../imgs/images.jpg就可以达到查看imgs目录的目的，甚至可以改为根目录等敏感目录查看敏感文件的目的

接下来我们继续看题目

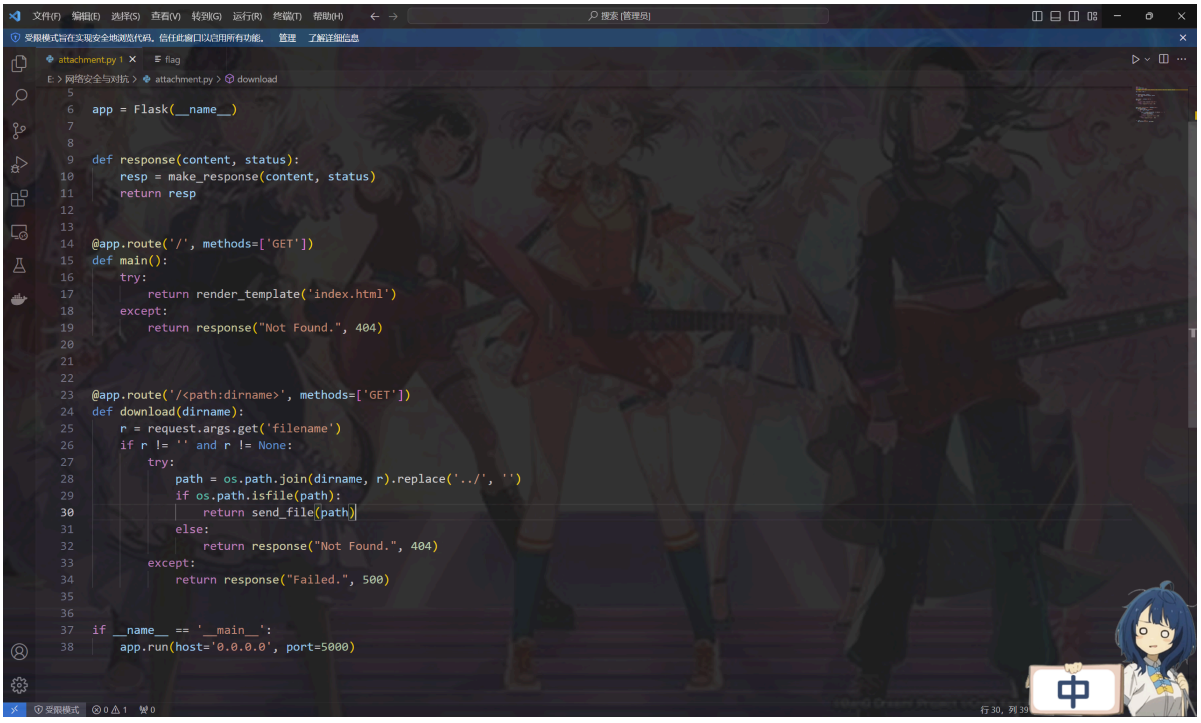
进入靶机发现这么一个提交文件名以查看文件名的框



先随便提交一个文件试试，回显是there is nothing，那么我们就看看源码（直接看页面源码啥都没有，包含解题关键的源码放在附件里面）

```
there is nothing
```

通过阅读下面python源码<关键地方replace('../',' ')>我们可知为了防止目录穿越漏洞，该页面过滤了../字符，发现../字符直接替换为空

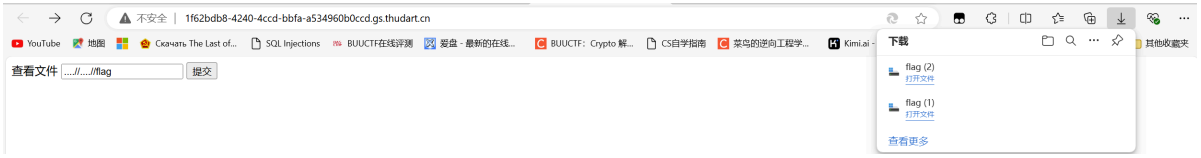


```python
app = Flask(__name__)

def response(content, status):
    resp = make_response(content, status)
    return resp

@app.route('/', methods=['GET'])
def main():
    try:
        return render_template('index.html')
    except:
        return response("Not Found.", 404)


@app.route('/<path:dirname>', methods=['GET'])
def download(dirname):
    r = request.args.get('filename')
    if r != '' and r != None:
        try:
            path = os.path.join(dirname, r).replace('../', '')
            if os.path.isfile(path):
                return send_file(path)
            else:
                return response("Not Found.", 404)
        except:
            return response("Failed.", 500)


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

既然这么处心积虑要把返回上级目录的命令ban掉，我猜他flag肯定就在上级目录或者上上级目录，管他的，试就完事了

因为../会被ban掉，使用../../那么都会被ban掉，起不到绕过作用，但是....//的话中间的../被ban掉了，剩下的..和/又能组合成新的../达到绕过过滤的效果

纳尼，上一级目录都没有吗，那就试试上上级目录喵



欧克，这下flag就出来了