

# 模版注入漏洞介绍

感觉ssti漏洞和sql注入和xss漏洞也是大同小异

## Flask漏洞

flask代码不严谨很危险

SSTI

可能造成任意文件读取和RCE远程控制后台系统

漏洞成因：

渲染模板时，没有严格控制对用户的输入；

使用了危险的模板，导致用户可以和flask程序进行交互。

flask是基于python开发的一种web服务器，那么也就意味着如果用户可以和flask进行交互的话，就可以执行python的代码，比如eval，system，file等等等等之类的函数。

这里的str只会被输出在页面上而不会执行自己本身的意思

## 漏洞演示

```
from flask import Flask,request,render_template_string
app = Flask(__name__)
@app.route('/',methods = ['GET'])
def index():
    str = request.args.get('ben')
    html_str = ""
    <html>
    <head></head>
    <body>{{str}}</body>
    </html>
    ""
    return render_template_string(html_str,str=str)
if __name__ == '__main__':
    app.debug = True
    app.run('127.0.0.1','8080')
```

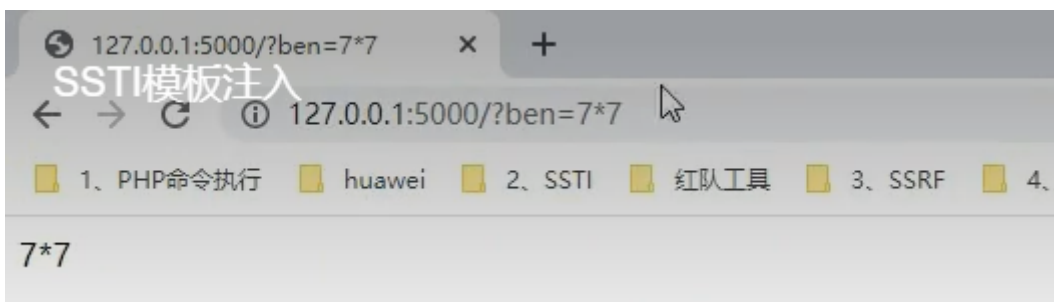
导入函数

GET方式获取ben的值，赋值给str

str值通过render\_template\_string加载到body中间

str是被{{}}包括起来的，会被预先渲染转义，然后才输出，不会被渲染执行；

本来应该是输出49的，但是因为被预先渲染转义了。就会直接输出而不是渲染执行



这样的话就可能存在漏洞，我们稍微修改一下代码

## 漏洞演示

```
from importlib.resources import contents
import time
from flask import Flask, request, render_template_string
app = Flask(__name__)
@app.route('/', methods = ['GET'])
def index():
    str = request.args.get('ben')
    html_str = ""
    <html>
    <head></head>
    <body>{0}</body>
    </html>
    """.format(str)
    return render_template_string(html_str)
if __name__ == '__main__':
    app.debug = True
    app.run('127.0.0.1', '8080')
```

str值通过format()函数填充到body中间

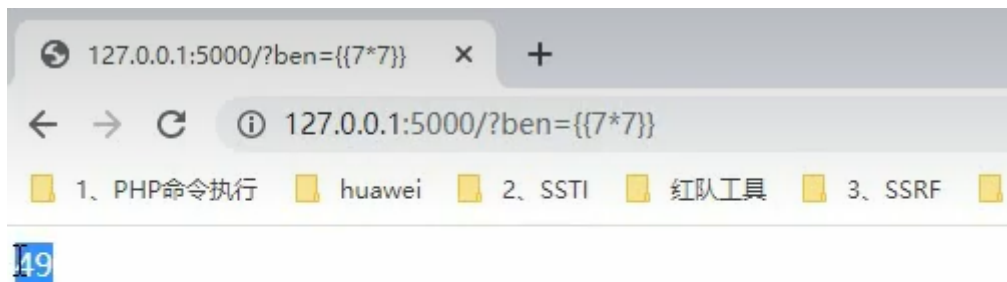
{ }里可以定义任何参数

return render\_template\_string会把{}内的字符串当成代码指令

127.0.0.1:8080/?ben={{7\*7}}

{{7\*7}}会被当成指令执行

原来的str是会先通过render\_template\_string 进行转义后再填充到body中间，但是经过代码修改以后会直接被format函数填充到body中间，此时render\_template\_string函数就会将str当做代码渲染执行(说白了就是后面的代码中str没有预先被渲染转义再输出)



jinja的语言:

控制结构 {% %}

变量取值 {{ }}

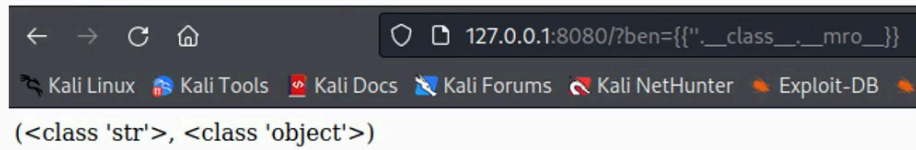
注释 {# #}

# 漏洞演示

{{7\*7}}可用来检测漏洞

```
127.0.0.1:8080/?ben={{'.__class__.__mro__'}}
```

当成指令来执行

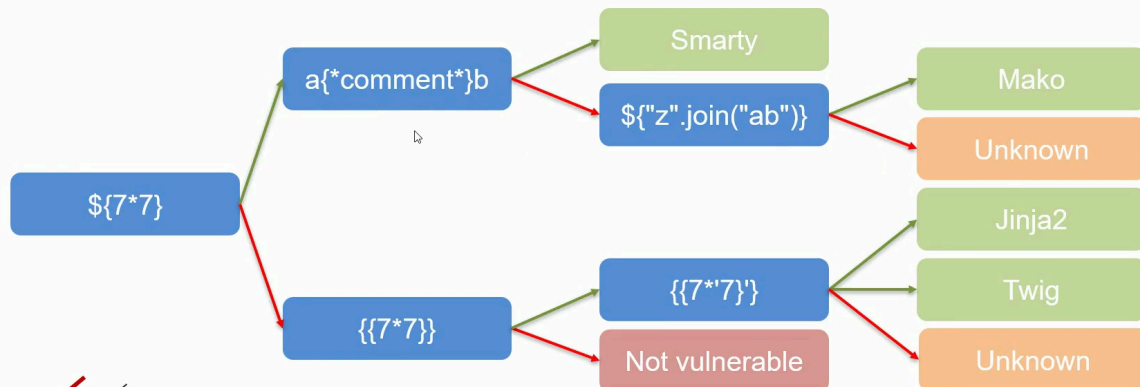


没有对提交的字符串进行足够严格的过滤，  
可能会当成代码指令执行。

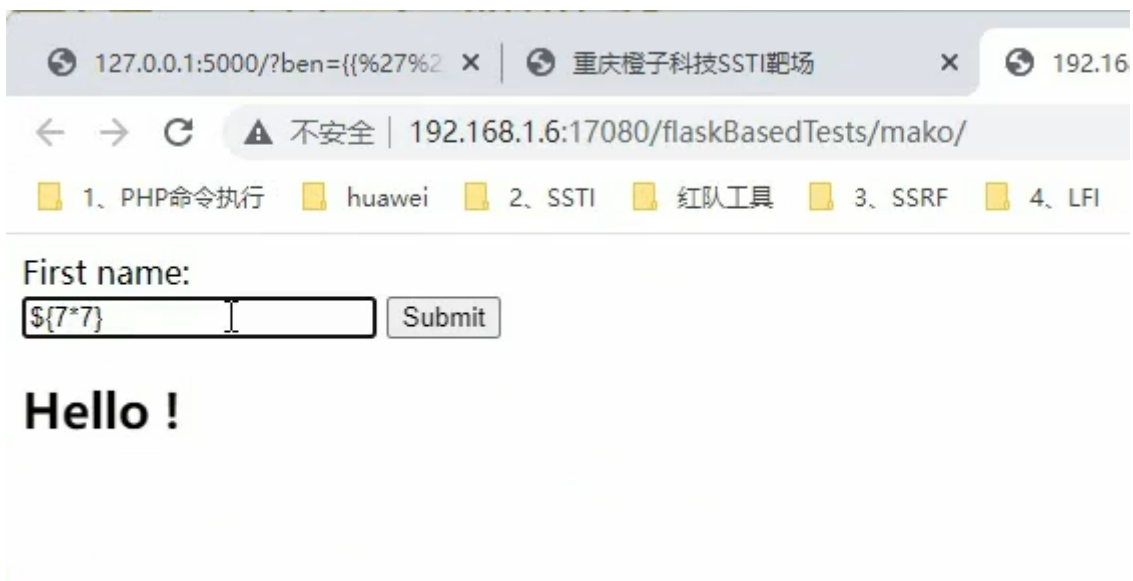
辨别不同类型模版注入的命令方法，在输入框通过输入以下蓝图中的命令，如果能执行，那么就是改命令绿色线后面的类型模版注入

## SSTI

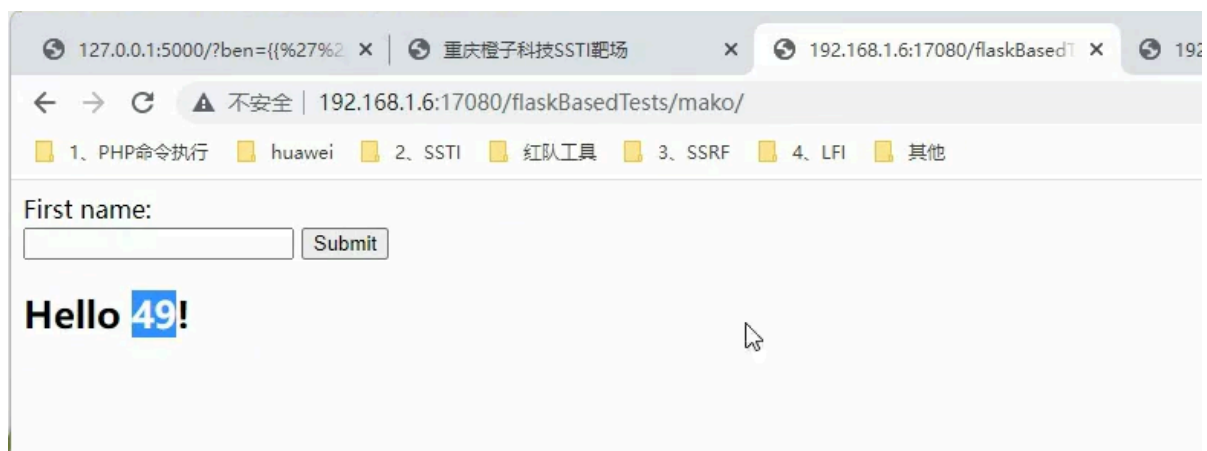
服务器端模板注入(Server-Side Template Injection),  
实际上也是一种注入漏洞。



比如我敲一个



点击submit给我回显49，说明存在ssti漏洞·1但我不知道是什么类型。那么输入其他命令试试



解下来我输入第二个命令，点击submit



发现命令没有被执行，那么就不是Smarty类型注入，是mako类型注入

