python_exe解压包 提取码:bbxb

Pyinstaller介绍

PyInstaller是一个开源的Python程序打包工具，它能够将Python应用程序打包成独立的执行文件，无需
依赖Python解释器即可在多种操作系统上运行。这对于分发Python应用程序非常有用，因为用户无需安装
Python或应用程序所依赖的库，就可以直接运行程序。

经过 PyInstaller 打包的 .exe 文件不能直接用 IDA Pro 顺利分析，因为 PyInstaller 将 Python 字
节码和依赖打包成机器码执行文件，而 IDA Pro 主要用于分析机器码，对字节码的直接分析支持有限，且
打包过程中可能包含的混淆和加密技术增加了分析的难度。

Pyinstaller解包方法（羊城杯login）

- pyc文件恢复
- 反编译pyc文件

pyc文件恢复

在将python文件打包成exe文件的过程中，会抹去pyc文件前面的部分信息，所以在反编译之前需要检查并添
加上这部分信息。
抹去的信息内容可以从struct文件中获取，利用010editor添加这部分信息。

方法一（利用pyinstxtractor.py）：

```
# 执行下面命令，会生成 target.exe_extracted 文件，里面会有pyc入口文件
python pyinstxtractor.py target.exe
```

方法二（archive_viewer.py）：

```
# 执行以下命令，会显示由 PyInstaller 打包的 .exe 文件中的内容，然后利用 x filename 命令提取文件，并命名为.pyc文件。
pos：文件在 .exe 文件中的起始位置。
length：文件在 .exe 文件中的长度（压缩后的长度）。
uncompressed：文件解压后的原始长度。
iscompressed：一个标志，指示文件是否被压缩。
type：文件类型。
name：文件或模块的名称。

python archive_viewer.py target.exe
```

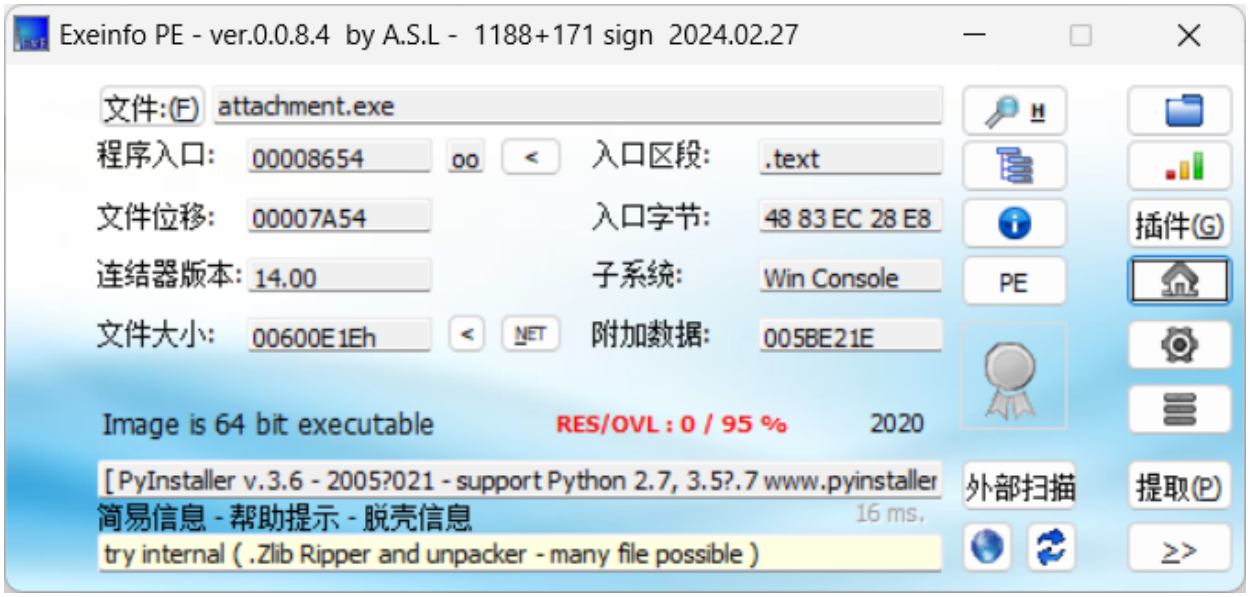
方法三：
直接使用 EasyPythonDecompiler 工具，可以直接查看 .exe 文件中的内容，并提取出 .pyc 文件。

反编译pyc文件

```
# 安装 uncompyle6 库
pip install uncompyle6
# 反编译pyc文件
uncompyle6 login.pyc
```

演示案例：羊城杯login

- 利用 exeinfo 查看文件的PE信息
- 利用 pyinstxtractor.py 解包pyc文件
- 利用 uncompyle6 反编译pyc文件
- 根据反编译文件 逆向flag




```
# 可以看到，是经过pyinstaller打包的exe文件，python版本为3.6，所以先恢复pyc文件
# 这里用第一种方法，利用Aconada中的python3.6虚拟环境执行以下命令，由于环境正确所
# 以这里不需要额外去对照struct.pyc文件去修复login.pyc文件，如果没有生成login.pyc
# 文件，就利用 010editor 工具对比struct.pyc文件和login.pyc文件的头部信息，为
login.pyc
# 并添加上缺少的头部信息
python pyinstxtractor.py login.exe

# 接下来进入到生成的文件夹中，可以看到login.pyc文件，用uncompyle6反编译
uncompyle6 login.pyc

源码如下：
import sys
input1 = input("input something:")
```

```
if len(input1) != 14:
    print("Wrong length!")
    sys.exit()
else:
    code = []
    for i in range(13):
        code.append(ord(input1[i]) ^ ord(input1[i + 1]))

    code.append(ord(input1[13]))
    a1 = code[2]
    a2 = code[1]
    a3 = code[0]
    a4 = code[3]
    a5 = code[4]
    a6 = code[5]
    a7 = code[6]
    a8 = code[7]
    a9 = code[9]
    a10 = code[8]
    a11 = code[10]
    a12 = code[11]
    a13 = code[12]
    a14 = code[13]
    if (a1 * 88 + a2 * 67 + a3 * 65 - a4 * 5 + a5 * 43 + a6 * 89 + a7 * 25 +
a8 * 13 - a9 * 36 + a10 * 15 + a11 * 11 + a12 * 47 - a13 * 60 + a14 * 29 ==
22748) & (a1 * 89 + a2 * 7 + a3 * 12 - a4 * 25 + a5 * 41 + a6 * 23 + a7 * 20
- a8 * 66 + a9 * 31 + a10 * 8 + a11 * 2 - a12 * 41 - a13 * 39 + a14 * 17 ==
7258) & (a1 * 28 + a2 * 35 + a3 * 16 - a4 * 65 + a5 * 53 + a6 * 39 + a7 * 27
+ a8 * 15 - a9 * 33 + a10 * 13 + a11 * 101 + a12 * 90 - a13 * 34 + a14 * 23
== 26190) & (a1 * 23 + a2 * 34 + a3 * 35 - a4 * 59 + a5 * 49 + a6 * 81 + a7
* 25 + (a8 << 7) - a9 * 32 + a10 * 75 + a11 * 81 + a12 * 47 - a13 * 60 + a14
* 29 == 37136) & (a1 * 38 + a2 * 97 + a3 * 35 - a4 * 52 + a5 * 42 + a6 * 79
+ a7 * 90 + a8 * 23 - a9 * 36 + a10 * 57 + a11 * 81 + a12 * 42 - a13 * 62 -
a14 * 11 == 27915) & (a1 * 22 + a2 * 27 + a3 * 35 - a4 * 45 + a5 * 47 + a6 *
49 + a7 * 29 + a8 * 18 - a9 * 26 + a10 * 35 + a11 * 41 + a12 * 40 - a13 * 61
+ a14 * 28 == 17298) & (a1 * 12 + a2 * 45 + a3 * 35 - a4 * 9 - a5 * 42 + a6
* 86 + a7 * 23 + a8 * 85 - a9 * 47 + a10 * 34 + a11 * 76 + a12 * 43 - a13 *
44 + a14 * 65 == 19875) & (a1 * 79 + a2 * 62 + a3 * 35 - a4 * 85 + a5 * 33 +
a6 * 79 + a7 * 86 + a8 * 14 - a9 * 30 + a10 * 25 + a11 * 11 + a12 * 57 - a13
* 50 - a14 * 9 == 22784) & (a1 * 8 + a2 * 6 + a3 * 64 - a4 * 85 + a5 * 73 +
a6 * 29 + a7 * 2 + a8 * 23 - a9 * 36 + a10 * 5 + a11 * 2 + a12 * 47 - a13 *
64 + a14 * 27 == 9710) & (a1 * 67 - a2 * 68 + a3 * 68 - a4 * 51 - a5 * 43 +
a6 * 81 + a7 * 22 - a8 * 12 - a9 * 38 + a10 * 75 + a11 * 41 + a12 * 27 - a13
* 52 + a14 * 31 == 13376) & (a1 * 85 + a2 * 63 + a3 * 5 - a4 * 51 + a5 * 44
+ a6 * 36 + a7 * 28 + a8 * 15 - a9 * 6 + a10 * 45 + a11 * 31 + a12 * 7 - a13
* 67 + a14 * 78 == 24065) & (a1 * 47 + a2 * 64 + a3 * 66 - a4 * 5 + a5 * 43
+ a6 * 112 + a7 * 25 + a8 * 13 - a9 * 35 + a10 * 95 + a11 * 21 + a12 * 43 -
a13 * 61 + a14 * 20 == 27687) & (a1 * 89 + a2 * 67 + a3 * 85 - a4 * 25 + a5
* 49 + a6 * 89 + a7 * 23 + a8 * 56 - a9 * 92 + a10 * 14 + a11 * 89 + a12 *
47 - a13 * 61 - a14 * 29 == 29250) & (a1 * 95 + a2 * 34 + a3 * 62 - a4 * 9 -
```

```
a5 * 43 + a6 * 83 + a7 * 25 + a8 * 12 - a9 * 36 + a10 * 16 + a11 * 51 + a12
* 47 - a13 * 60 - a14 * 24 == 15317):
        print("flag is GWHT{md5(your_input)}")
        print("Congratulations and have fun!")
    else:
        print("Sorry,plz try again...")
```

解密脚本

```
import hashlib
from z3 import *
a1=Int('a1')
a2=Int('a2')
a3=Int('a3')
a4=Int('a4')
a5=Int('a5')
a6=Int('a6')
a7=Int('a7')
a8=Int('a8')
a9=Int('a9')
a10=Int('a10')
a11=Int('a11')
a12=Int('a12')
a13=Int('a13')
a14=Int('a14')
s=Solver()
s.add(a1 * 88 + a2 * 67 + a3 * 65 - a4 * 5 + a5 * 43 + a6 * 89 + a7 * 25 +
a8 * 13 - a9 * 36 + a10 * 15 + a11 * 11 + a12 * 47 - a13 * 60 + a14 * 29 ==
22748)
s.add(a1 * 89 + a2 * 7 + a3 * 12 - a4 * 25 + a5 * 41 + a6 * 23 + a7 * 20 -
a8 * 66 + a9 * 31 + a10 * 8 + a11 * 2 - a12 * 41 - a13 * 39 + a14 * 17 ==
7258)
s.add(a1 * 28 + a2 * 35 + a3 * 16 - a4 * 65 + a5 * 53 + a6 * 39 + a7 * 27 +
a8 * 15 - a9 * 33 + a10 * 13 + a11 * 101 + a12 * 90 - a13 * 34 + a14 * 23 ==
26190)
s.add(a1 * 23 + a2 * 34 + a3 * 35 - a4 * 59 + a5 * 49 + a6 * 81 + a7 * 25 +
a8 * 128 - a9 * 32 + a10 * 75 + a11 * 81 + a12 * 47 - a13 * 60 + a14 * 29 ==
37136)
s.add(a1 * 38 + a2 * 97 + a3 * 35 - a4 * 52 + a5 * 42 + a6 * 79 + a7 * 90 +
a8 * 23 - a9 * 36 + a10 * 57 + a11 * 81 + a12 * 42 - a13 * 62 - a14 * 11 ==
27915)
s.add(a1 * 22 + a2 * 27 + a3 * 35 - a4 * 45 + a5 * 47 + a6 * 49 + a7 * 29 +
a8 * 18 - a9 * 26 + a10 * 35 + a11 * 41 + a12 * 40 - a13 * 61 + a14 * 28 ==
17298)
s.add(a1 * 12 + a2 * 45 + a3 * 35 - a4 * 9 - a5 * 42 + a6 * 86 + a7 * 23 +
a8 * 85 - a9 * 47 + a10 * 34 + a11 * 76 + a12 * 43 - a13 * 44 + a14 * 65 ==
19875)
s.add(a1 * 79 + a2 * 62 + a3 * 35 - a4 * 85 + a5 * 33 + a6 * 79 + a7 * 86 +
```

```
a8 * 14 - a9 * 30 + a10 * 25 + a11 * 11 + a12 * 57 - a13 * 50 - a14 * 9 ==
22784)
s.add(a1 * 8 + a2 * 6 + a3 * 64 - a4 * 85 + a5 * 73 + a6 * 29 + a7 * 2 + a8
* 23 - a9 * 36 + a10 * 5 + a11 * 2 + a12 * 47 - a13 * 64 + a14 * 27 == 9710)
s.add(a1 * 67 - a2 * 68 + a3 * 68 - a4 * 51 - a5 * 43 + a6 * 81 + a7 * 22 -
a8 * 12 - a9 * 38 + a10 * 75 + a11 * 41 + a12 * 27 - a13 * 52 + a14 * 31 ==
13376)
s.add(a1 * 85 + a2 * 63 + a3 * 5 - a4 * 51 + a5 * 44 + a6 * 36 + a7 * 28 +
a8 * 15 - a9 * 6 + a10 * 45 + a11 * 31 + a12 * 7 - a13 * 67 + a14 * 78 ==
24065)
s.add(a1 * 47 + a2 * 64 + a3 * 66 - a4 * 5 + a5 * 43 + a6 * 112 + a7 * 25 +
a8 * 13 - a9 * 35 + a10 * 95 + a11 * 21 + a12 * 43 - a13 * 61 + a14 * 20 ==
27687)
s.add(a1 * 89 + a2 * 67 + a3 * 85 - a4 * 25 + a5 * 49 + a6 * 89 + a7 * 23 +
a8 * 56 - a9 * 92 + a10 * 14 + a11 * 89 + a12 * 47 - a13 * 61 - a14 * 29 ==
29250)
s.add(a1 * 95 + a2 * 34 + a3 * 62 - a4 * 9 - a5 * 43 + a6 * 83 + a7 * 25 +
a8 * 12 - a9 * 36 + a10 * 16 + a11 * 51 + a12 * 47 - a13 * 60 - a14 * 24 ==
15317)
if s.check()==sat:
        print(s.model())
#a2 = 24,a13 = 88,a14 = 33,a6 = 43,a9 = 52,
#a5 = 104,a12 = 74,a7 = 28,a1 = 119,a10 = 108,
#a11 = 88,a8 = 91,a4 = 7,a3 = 10
s=[119,24,10,7,104,43,28,91,52,108,88,74,88,33]
index=[2,1,0,3,4,5,6,7,9,8,10,11,12,13]
data=[0]*14
flag=''
for i in range(len(index)):
        data[i]=s[index[i]]
for i in range(len(index)-2,-1,-1):
        data[i]^=data[i+1]
for i in range(len(data)):
        flag+=chr(data[i])
m=hashlib.md5()
m.update(flag.encode('utf-8'))
print('flag{'+m.hexdigest()+'}')
```