
ATTACK
DEFENSE
by PentesterAcademy

Name	Reversing Binaries I
URL	https://www.attackdefense.com/challengedetails?cid=1067
Type	Code Repository : Python PyPi

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

A binary is kept in the home directory of user root.

Objective: Recover the python code from the binary and retrieve the flag!

Solution:

Step 1: List the contents of the home directory of user root.

Commands: ls -l

```
root@attackdefense:~# ls -l
total 3896
-rw-r--r-- 1 root root   12667 Jun  6 15:52 pyinstxtractor.py
-rw-r--r-- 1 root root 3971315 Jun  6 16:28 sample.exe
root@attackdefense:~#
```

Step 2: Use pyinstxtractor.py to extract files from the binary.

Command: python pyinstxtractor.py sample.exe

```
root@attackdefense:~# python pyinstxtractor.py sample.exe
[*] Processing sample.exe
[*] Pyinstaller version: 2.1+
[*] Python version: 27
[*] Length of package: 3697907 bytes
[*] Found 18 files in CArchive
[*] Beginning extraction...please standby
[+] Possible entry point: pyiboot01_bootstrap
[+] Possible entry point: sample
[*] Found 194 files in PYZ archive
[*] Successfully extracted pyinstaller archive: sample.exe

You can now use a python decompiler on the pyc files within the extracted directory
root@attackdefense:~#
```

Step 3: Extracted files are stored in sample.exe_extracted directory.

Command: ls -l

```
root@attackdefense:~# ls -l
total 3900
-rw-r--r-- 1 root root   12667 Jun  6 15:52 pyinstxtractor.py
-rw-r--r-- 1 root root 3971315 Jun  6 16:28 sample.exe
drwxr-xr-x 3 root root    4096 Jun  6 18:13 sample.exe_extracted
root@attackdefense:~#
```

Step 4: Check the extracted files

Commands:
cd sample.exe_extracted
ls -l

```
root@attackdefense:~# cd sample.exe_extracted/
root@attackdefense:~/sample.exe_extracted# ls -l
total 8148
-rw-r--r-- 1 root root    1052 Jun  6 18:13  Microsoft.VC90.CRT.manifest
-rw-r--r-- 1 root root  642651 Jun  6 18:13  PYZ-00.pyz
drwxr-xr-x 2 root root   12288 Jun  6 18:13  PYZ-00.pyz_extracted
-rw-r--r-- 1 root root 1639424 Jun  6 18:13  _hashlib.pyd
-rw-r--r-- 1 root root   92672 Jun  6 18:13  bz2.pyd
-rw-r--r-- 1 root root  245760 Jun  6 18:13  msvcm90.dll
-rw-r--r-- 1 root root  854144 Jun  6 18:13  msvcp90.dll
-rw-r--r-- 1 root root  642176 Jun  6 18:13  msvcr90.dll
-rw-r--r-- 1 root root       0 Jun  6 18:13 'pyi-windows-manifest-filename sample.exe.manifest'
-rw-r--r-- 1 root root    5263 Jun  6 18:13  pyiboot01_bootstrap
-rw-r--r-- 1 root root    2739 Jun  6 18:13  pyimod01_os_path
-rw-r--r-- 1 root root   12687 Jun  6 18:13  pyimod02_archive
-rw-r--r-- 1 root root   23506 Jun  6 18:13  pyimod03_importers
-rw-r--r-- 1 root root 3428864 Jun  6 18:13  python27.dll
-rw-r--r-- 1 root root     504 Jun  6 18:13  sample
-rw-r--r-- 1 root root    1014 Jun  6 18:13  sample.exe.manifest
-rw-r--r-- 1 root root   11776 Jun  6 18:13  select.pyd
-rw-r--r-- 1 root root     254 Jun  6 18:13  struct
-rw-r--r-- 1 root root  692224 Jun  6 18:13  unicodedata.pyd
root@attackdefense:~/sample.exe_extracted#
```

Step 5: sample file is the one we are looking for (because of the same name as the binary).
This is a .pyc file but the magic number 03F30D0A00000000 (8 bytes) is missing.

Commands: hexedit sample

```
00000000   63 00 00 00  00 00 00 00  00 02 00 00  00 40 00 00  00 73 34 00  00 00 64 00  00 64 01 00
0000001C   6C 00 00 5A  00 00 64 02  00 61 01 00  64 03 00 84  00 00 5A 02  00 64 04 00  84 00 00 5A
00000038   03 00 64 05  00 47 48 65  03 00 83 00  00 01 64 01  00 53 28 06  00 00 00 69  FF FF FF FF
00000054   4E 74 10 00  00 00 35 64  30 35 36 39  30 34 61 33  63 33 30 33  62 32 63 01  00 00 00 02
00000070   00 00 00 02  00 00 00 43  00 00 00 73  17 00 00 00  74 00 00 7C  00 00 17 7D  01 00 64 01
0000008C   00 7C 01 00  17 47 48 64  00 00 53 28  02 00 00 00  4E 73 0C 00  00 00 48 65  72 65 20 69
000000A8   73 20 66 6C  61 67 28 01  00 00 00 74  05 00 00 00  66 6C 61 67  32 28 02 00  00 00 74 05
000000C4   00 00 00 66  6C 61 67 33  74 05 00 00  00 66 6C 61  67 31 28 00  00 00 00 28  00 00 00 00
000000E0   73 09 00 00  00 73 61 6D  70 6C 65 2E  70 79 74 0A  00 00 00 70  72 69 6E 74  5F 66 6C 61
000000FC   67 05 00 00  00 73 04 00  00 00 00 02  0A 01 63 00  00 00 00 00  00 00 00 02  00 00 00 43
00000118   00 00 00 73  0E 00 00 00  64 00 00 53  74 00 00 64  01 00 83 01  00 01 28 02  00 00 00 4E
00000134   74 10 00 00  00 36 62 66  36 31 32 34  64 63 64 30  62 37 64 32  35 28 01 00  00 00 52 04
00000150   00 00 00 28  00 00 00 00  28 00 00 00  00 28 00 00  00 00 73 09  00 00 00 73  61 6D 70 6C
0000016C   65 2E 70 79  74 07 00 00  00 73 74 61  67 65 5F 31  0B 00 00 00  73 04 00 00  00 00 01 04
00000188   01 73 12 00  00 00 50 65  72 66 6F 72  6D 69 6E 67  20 61 63 74  69 6F 6E 73  28 04 00 00
000001A4   00 74 02 00  00 00 6F 73  52 01 00 00  00 52 04 00  00 00 52 06  00 00 00 28  00 00 00 00
000001C0   28 00 00 00  00 28 00 00  00 00 73 09  00 00 00 73  61 6D 70 6C  65 2E 70 79  74 08 00 00
000001DC   00 3C 6D 6F  64 75 6C 65  3E 01 00 00  00 73 0A 00  00 00 0C 02  06 02 09 06  09 04 05 01
```

Step 6: Append the magic number to the “sample” file and save it as “sample.pyc”

Command: echo -e '03F30D0A00000000' | xxd -r -p | cat - sample > sample.pyc

```
root@attackdefense:~/sample.exe_extracted#
root@attackdefense:~/sample.exe_extracted# echo -e '03F30D0A00000000' | xxd -r -p | cat - sample > sample.pyc
root@attackdefense:~/sample.exe_extracted#
```

Step 7: Open the binary with hexedit to verify if the changes are correct.

Commands: hexedit sample.pyc

```
00000000   03 F3 0D 0A  00 00 00 00  63 00 00 00  00 00 00 00  00 02 00 00  00 40 00 00  00 73 34 00
0000001C   00 00 64 00  00 64 01 00  6C 00 00 5A  00 00 64 02  00 61 01 00  64 03 00 84  00 00 5A 02
00000038   00 64 04 00  84 00 00 5A  03 00 64 05  00 47 48 65  03 00 83 00  00 01 64 01  00 53 28 06
00000054   00 00 00 69  FF FF FF FF  4E 74 10 00  00 00 35 64  30 35 36 39  30 34 61 33  63 33 30 33
00000070   62 32 63 01  00 00 00 02  00 00 00 02  00 00 00 43  00 00 00 73  17 00 00 00  74 00 00 7C
0000008C   00 00 17 7D  01 00 64 01  00 7C 01 00  17 47 48 64  00 00 53 28  02 00 00 00  4E 73 0C 00
000000A8   00 00 48 65  72 65 20 69  73 20 66 6C  61 67 28 01  00 00 00 74  05 00 00 00  66 6C 61 67
000000C4   32 28 02 00  00 00 74 05  00 00 00 66  6C 61 67 33  74 05 00 00  00 66 6C 61  67 31 28 00
000000E0   00 00 00 28  00 00 00 00  73 09 00 00  00 73 61 6D  70 6C 65 2E  70 79 74 0A  00 00 00 70
000000FC   72 69 6E 74  5F 66 6C 61  67 05 00 00  00 73 04 00  00 00 00 02  0A 01 63 00  00 00 00 00
00000118   00 00 00 02  00 00 00 43  00 00 00 73  0E 00 00 00  64 00 00 53  74 00 00 64  01 00 83 01
00000134   00 01 28 02  00 00 00 4E  74 10 00 00  00 36 62 66  36 31 32 34  64 63 64 30  62 37 64 32
00000150   35 28 01 00  00 00 52 04  00 00 00 28  00 00 00 00  28 00 00 00  00 28 00 00  00 00 73 09
0000016C   00 00 00 73  61 6D 70 6C  65 2E 70 79  74 07 00 00  00 73 74 61  67 65 5F 31  0B 00 00 00
00000188   73 04 00 00  00 00 01 04  01 73 12 00  00 00 50 65  72 66 6F 72  6D 69 6E 67  20 61 63 74
000001A4   69 6F 6E 73  28 04 00 00  00 74 02 00  00 00 6F 73  52 01 00 00  00 52 04 00  00 00 52 06
000001C0   00 00 00 28  00 00 00 00  28 00 00 00  00 28 00 00  00 00 73 09  00 00 00 73  61 6D 70 6C
000001DC   65 2E 70 79  74 08 00 00  00 3C 6D 6F  64 75 6C 65  3E 01 00 00  00 73 0A 00  00 00 0C 02
000001F8   06 02 09 06  09 04 05 01
```

Step 8: Decompile .pyc file with uncompye6

Commands: uncompyle6 -o . sample.pyc

```
root@attackdefense:~/sample.exe_extracted# uncompyle6 -o . sample.pyc

# Successfully decompiled file
root@attackdefense:~/sample.exe_extracted#
```

Step 9: Check the code for flag value.

Commands: cat sample.py

```
root@attackdefense:~/sample.exe_extracted# cat sample.py
# uncompyle6 version 3.3.3
# Python bytecode 2.7 (62211)
# Decompiled from: Python 2.7.15rc1 (default, Nov 12 2018, 14:31:15)
# [GCC 7.3.0]
# Embedded file name: sample.py
import os
flag2 = '5d056904a3c303b2'

def print_flag(flag3):
    global flag2
    flag1 = flag2 + flag3
    print 'Here is flag' + flag1


def stage_1():
    return
    print_flag('6bf6124dcd0b7d25')
```

Step 10: Figure out the flag from code.

Flag: 5d056904a3c30303b26bf6124dcd0b7d25

References:

1. pypi (https://pypi.org)
2. pip (https://pypi.org/project/pip/)
3. Python Uncompyle6 (https://github.com/rocky/python-uncompyle6/)
4. Pyinstaller Extractor tool (https://sourceforge.net/projects/pyinstallerextractor)