

Лабораторна робота №6

Аналіз конфігурації

Виконав:

Студент 3 курсу ФТІ

групи ФІ-92

Поночевний Назар Юрійович

Варіант 6

Мета роботи

Отримати навички аналізу налаштувань та середовища виконання ШПЗ для задач реагування на інциденти.

Завдання 1:

Створіть парсер конфігурації з пам'яті Вашої системи з лабораторної роботи 4. Впевніться, що парсер працює після застосування UPX [124] та MPRESS [100] на виконуваному файлі зразку;

Трошки модифікуємо файл з методички для пошуку, наприклад, IP адреси сервера, який контролює клієнта:

```
from winappdbg import System
from re import findall

s = System()
s.request_debug_privileges()
s.scan_processes()

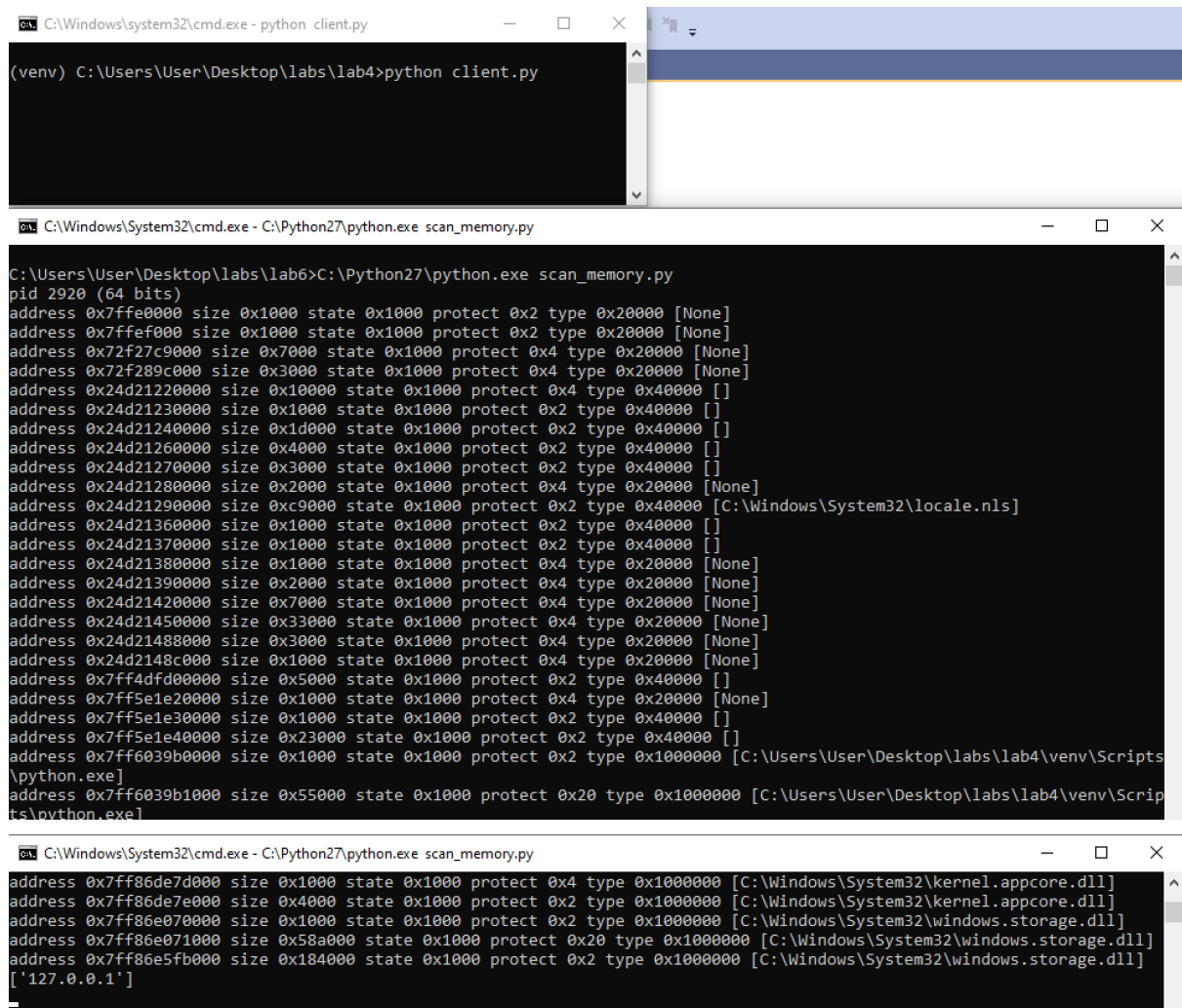
for p, path in s.find_processes_by_filename("python.exe"):
    pid = p.get_pid()
    bits = p.get_bits()
    print "pid %d (%d bits)" % (pid, bits)

    mmap = p.get_memory_map()
    mapf = p.get_mapped_filenames(mmap)

    for m in mmap:
        a = m.BaseAddress
        fn = mapf.get(a, None)

        if m.has_content():
            print "address 0x%x size 0x%x state 0x%x protect 0x%x type 0x%x [%s]" % (a,
m.RegionSize, m.State, m.Protect, m.Type, fn)
            d = p.read(a, m.RegionSize)
            cc = findall(r"\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b", d[:2])
            if len(cc) > 0:
                print cc
                raw_input()
```

Тепер запусимо клієнта і спробуємо отримати адресу спарсивши конфігурацію пам'яті процесу:



```
C:\Windows\system32\cmd.exe - python client.py
(venv) C:\Users\User\Desktop\labs\lab4>python client.py

C:\Windows\System32\cmd.exe - C:\Python27\python.exe scan_memory.py
C:\Users\User\Desktop\labs\lab6>C:\Python27\python.exe scan_memory.py
pid 2920 (64 bits)
address 0x7ffe0000 size 0x1000 state 0x1000 protect 0x2 type 0x20000 [None]
address 0x7ffef000 size 0x1000 state 0x1000 protect 0x2 type 0x20000 [None]
address 0x72f27c9000 size 0x7000 state 0x1000 protect 0x4 type 0x20000 [None]
address 0x72f289c000 size 0x3000 state 0x1000 protect 0x4 type 0x20000 [None]
address 0x24d2122000 size 0x1000 state 0x1000 protect 0x4 type 0x40000 []
address 0x24d2123000 size 0x1000 state 0x1000 protect 0x2 type 0x40000 []
address 0x24d2124000 size 0x1d000 state 0x1000 protect 0x2 type 0x40000 []
address 0x24d2126000 size 0x4000 state 0x1000 protect 0x2 type 0x40000 []
address 0x24d2127000 size 0x3000 state 0x1000 protect 0x2 type 0x40000 []
address 0x24d2128000 size 0x2000 state 0x1000 protect 0x4 type 0x20000 [None]
address 0x24d2129000 size 0xc9000 state 0x1000 protect 0x2 type 0x40000 [C:\Windows\System32\locale.nls]
address 0x24d2136000 size 0x1000 state 0x1000 protect 0x2 type 0x40000 []
address 0x24d2137000 size 0x1000 state 0x1000 protect 0x2 type 0x40000 []
address 0x24d2138000 size 0x1000 state 0x1000 protect 0x4 type 0x20000 [None]
address 0x24d2139000 size 0x2000 state 0x1000 protect 0x4 type 0x20000 [None]
address 0x24d2142000 size 0x7000 state 0x1000 protect 0x4 type 0x20000 [None]
address 0x24d2145000 size 0x33000 state 0x1000 protect 0x4 type 0x20000 [None]
address 0x24d2148000 size 0x3000 state 0x1000 protect 0x4 type 0x20000 [None]
address 0x24d2148c000 size 0x1000 state 0x1000 protect 0x4 type 0x20000 [None]
address 0x7ff4dfd0000 size 0x5000 state 0x1000 protect 0x2 type 0x40000 []
address 0x7ff5e1e20000 size 0x1000 state 0x1000 protect 0x4 type 0x20000 [None]
address 0x7ff5e1e30000 size 0x1000 state 0x1000 protect 0x2 type 0x40000 []
address 0x7ff5e1e40000 size 0x23000 state 0x1000 protect 0x2 type 0x40000 []
address 0x7ff6039b0000 size 0x1000 state 0x1000 protect 0x2 type 0x1000000 [C:\Users\User\Desktop\labs\lab4\venv\Scripts\python.exe]
address 0x7ff6039b1000 size 0x55000 state 0x1000 protect 0x20 type 0x1000000 [C:\Users\User\Desktop\labs\lab4\venv\Scripts\python.exe]

address 0x7ff86de7d000 size 0x1000 state 0x1000 protect 0x4 type 0x1000000 [C:\Windows\System32\kernel.appcore.dll]
address 0x7ff86de7e000 size 0x4000 state 0x1000 protect 0x2 type 0x1000000 [C:\Windows\System32\kernel.appcore.dll]
address 0x7ff86e070000 size 0x1000 state 0x1000 protect 0x2 type 0x1000000 [C:\Windows\System32\windows.storage.dll]
address 0x7ff86e071000 size 0x58a000 state 0x1000 protect 0x20 type 0x1000000 [C:\Windows\System32\windows.storage.dll]
address 0x7ff86e5fb000 size 0x184000 state 0x1000 protect 0x2 type 0x1000000 [C:\Windows\System32\windows.storage.dll]
['127.0.0.1']
```

Завдання 2:

Проаналізуйте 1-2 антивіруси з лабораторії розділу 3.3.2 за допомогою методів з розділу 6.3.3. Знайдіть ім'я системи, ім'я користувача, список процесів, список файлів на робочому столі, перші 32 байти notepad.exe. Для пришвидшення роботи рекомендується використати 256 зразків з theZoo [58], VirusShare [125] або інших джерел [126] для отримання 1 байту за запит;

1) Знайти ім'я системи

Згенеруємо зразки та запусимо сканування у KAV:

VIEW

Bytes

FORMAT

Binary

GROUP BY

None

```

00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000001111000
011010000110010001110110011001
10011010000

```

ENCODE DECODE

Base64

VARIANT

Base64 (RFC 3548, RFC 4648)

→ Encoded 28 chars

VIEW

Text

```

AAAAAAAAAAAAAAAAAB4aGR2ZmgA

```

Base64*

copy clear

```

AAAAAAAAAAAAAAAAAB4aGR2ZmgA

```

Detect Character Encoding

38 possible encodings from 39 were found. Note that each decoded text is limited to 500 characters.

Charset	Decoded Text
UTF-8	xhdvfh
ASCII	xhdvfh
EUC-JP	xhdvfh
SJIS	xhdvfh
eucJP-win	xhdvfh
SJIS-win	xhdvfh
CP932	xhdvfh
ISO-2022-JP	xhdvfh
GB18030	xhdvfh
Windows-1252	xhdvfh
Windows-1254	xhdvfh
ISO-8859-1	xhdvfh
ISO-8859-2	xhdvfh
ISO-8859-3	xhdvfh
ISO-8859-4	xhdvfh
ISO-8859-5	xhdvfh
ISO-8859-6	xhdvfh
ISO-8859-7	xhdvfh
ISO-8859-8	xhdvfh
ISO-8859-9	xhdvfh
ISO-8859-10	xhdvfh

Далі просто “розвернемо” рядок, бо у звіті останні “біти” були першими:

```

In [11]: string = str("xhdvfh")

In [12]: string[::-1]
Out[12]: 'hfvdx'

```

- Знайти ім'я користувача
Замість GetComputerNameA напишемо GetUserNameA, згенеруємо зразки і запусимо сканування у KAV:

Base64*

AAAAAAAAAHJvdGFydHkpbm1tZEE=

copy clear download

Detect Character Encoding

38 possible encodings from 39 were found. Note that each decoded text is limited to 500 characters.

Charset	Decoded Text
UTF-8	rotartsinimdA
ASCII	rotartsinimdA
EUC-JP	rotartsinimdA
SJIS	rotartsinimdA
eucJP-win	rotartsinimdA
SJIS-win	rotartsinimdA
CP932	rotartsinimdA
ISO-2022-JP	rotartsinimdA
GB18030	rotartsinimdA
Windows-1252	rotartsinimdA
Windows-1254	rotartsinimdA
ISO-8859-1	rotartsinimdA
ISO-8859-2	rotartsinimdA
ISO-8859-3	rotartsinimdA
ISO-8859-4	rotartsinimdA
ISO-8859-5	rotartsinimdA
ISO-8859-6	rotartsinimdA
ISO-8859-7	rotartsinimdA
ISO-8859-8	rotartsinimdA
ISO-8859-9	rotartsinimdA
ISO-8859-10	rotartsinimdA

Далі просто “розвернемо” рядок, бо у звіті останні “біти” були першими:

```
In [8]: user_name = "rotartsinimdA"

In [9]: user_name[::-1]
Out[9]: 'Administrator'
```