ДОДАТОК А

Головне меню розроблених програм

Після запуску розробленої програми BMP_CHECK.py користувачу буде запропоновано вказати абсолютний або відносний шлях до файлу зображення (рисунок А.1).

Рисунок А.1 – Запит шляху до файлу зображення

Після введення шляху, програма почне перевірку зображення за ключовими ознаками.

При перевірці оригінального зображення не буде виявлено жодних аномалій (рисунок А.2).

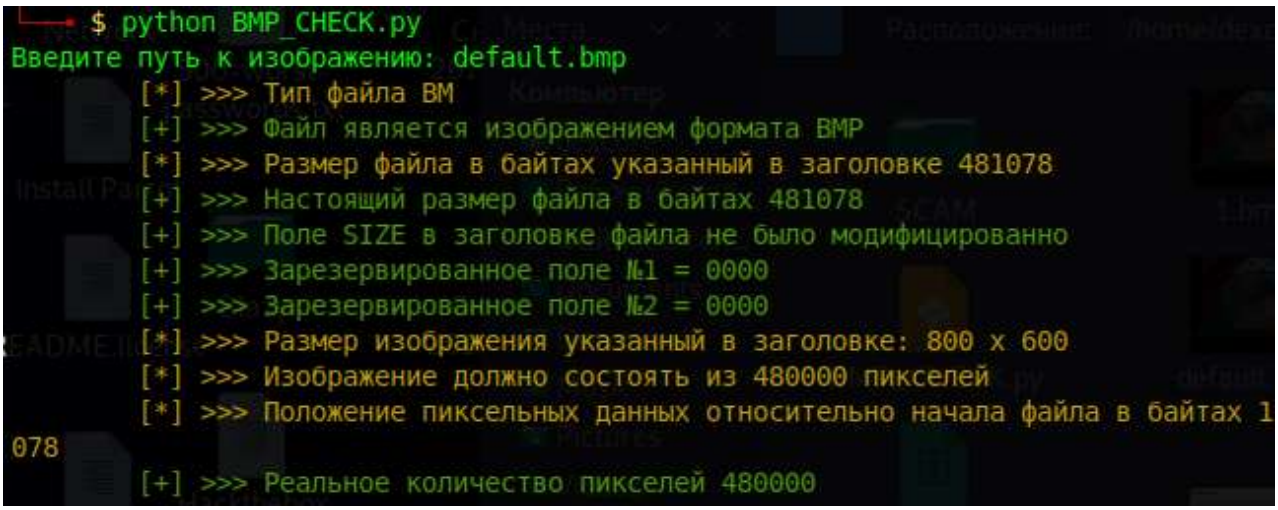


Рисунок А.2 – Перевірка оригінального зображення

При перевірці інфікованого зображення виявлені аномалії за ключовими ознаками будуть виділені червоним кольором (рисунок А.3).

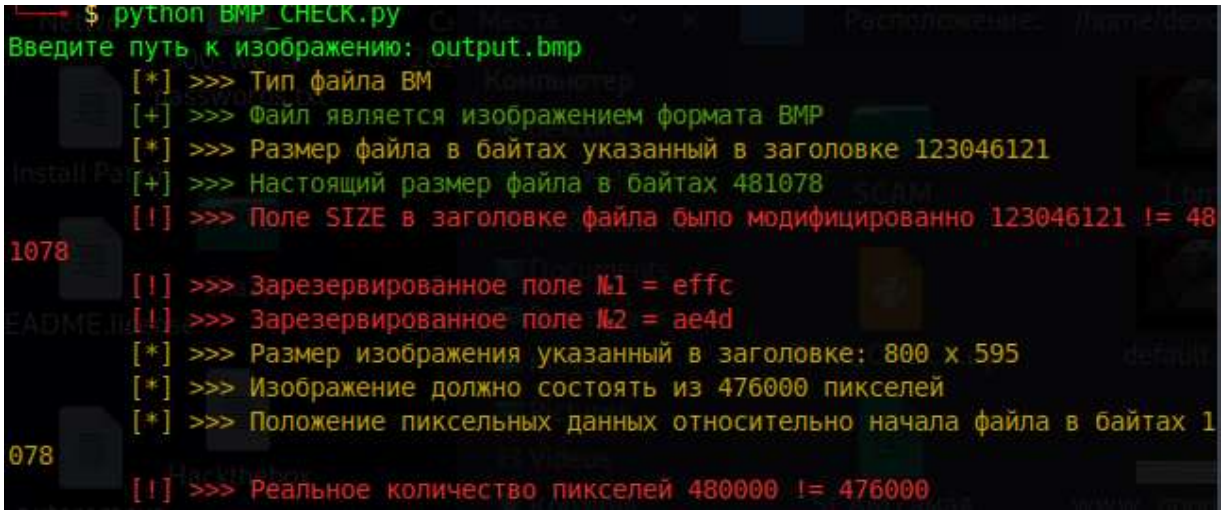


Рисунок А.3 – Перевірка інфікованого зображення

Після запуску розробленої програми HID.py вона функціонує, як фоновий процес. Для коректної роботи програми необхідна наявність кофігураційного файлу (рисунок А.4).

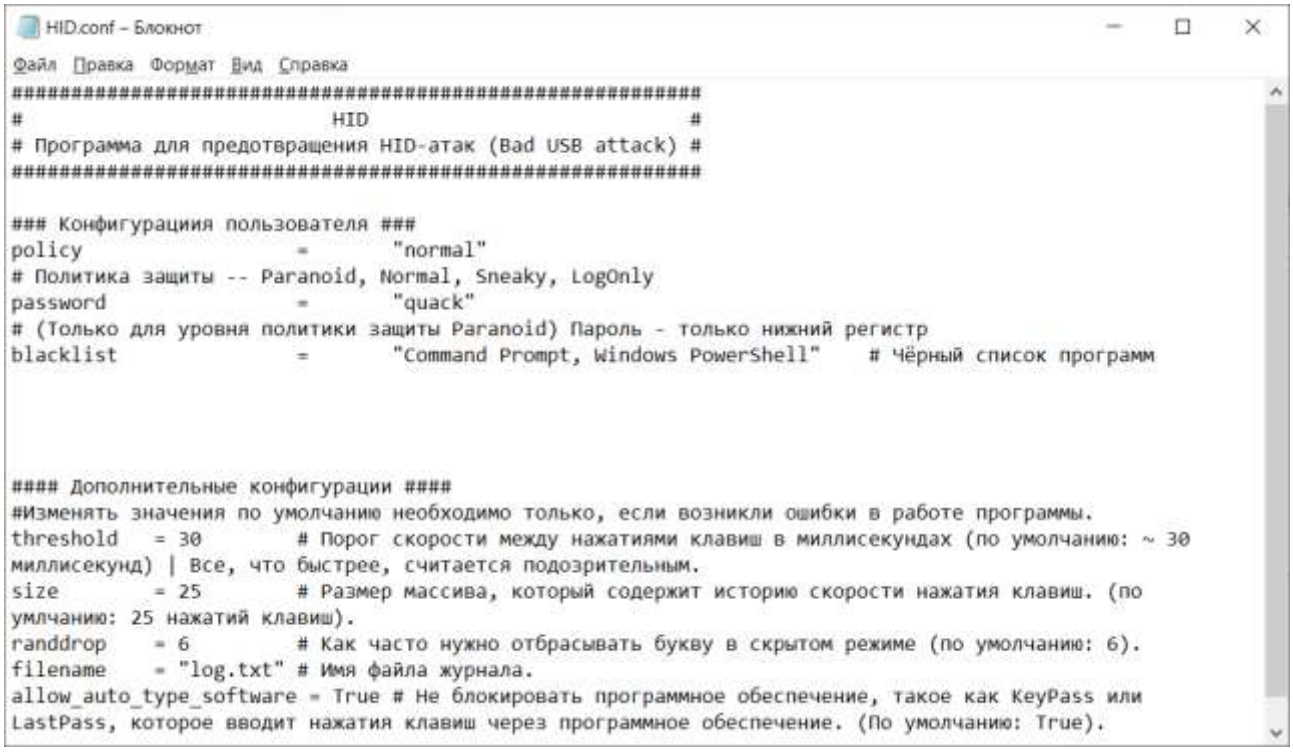


Рисунок А.4 – Конфігураційний файл для програми HID.py

В даному файлі можна задати оду з чотирьох політик, що будуть використовуватися, пароль для розблокування вводу. Крім того можна задати

граничне значення швидкості, ім’я лог файлу, кількість символів, що будуть перервані у разі вибору політики “Sneaky”.

ДОДАТОК Б

Детальні результати сканування вірусних зразків різними антивірусними засобами

В цьому додатку наведені детальні результати сканування (рисунок Б.1-Б.6, Б.8-Б.9, Б.11-Б.12 та рисунок Б.14-Б.21). На всіх етапах використовувався один і той самий вірус. Спочатку був згенерований та просканований шелл-код в форматі raw у 2019 році (рисунок Б.1) та у 2020 (рисунок Б.2). Потім цей самий вірус був згенерований та перевірений в форматі exe 2019 рік (рисунок Б.3) та 2020 рік (рисунок Б.4). Була перевірина оригінальна програма PuTTY 2019 рік (рисунок Б.5) та 2020 рік (рисунок Б.6). Після цього був видалений цифровий підпис програми (рисунок Б.7) та проведене сканування у 2019 році (рисунок Б.8) та 2020 році (рисунок Б.9). Також була створена нова порожня секція в оригінальній програмі (рисунок Б.10) та проведена перевірка 2019 рік (рисунок Б.11) та 2020 рік (рисунок Б.12). Після цього в оригінальну PuTTY був впроваджений шелл-код за допомого створення нової секції (рисунок Б.13), результати перевірки у 2019 році можна побачити на рисунку Б.14, та у 2020 році на рисуноку Б.15. Також в оригінальну програму був впроваджений вірус з використанням code cave, результати сканування 2019 року на рисунку Б.16, та 2020 року на рисуноку Б.17. Після цього було проаналізоване оригінальне зображення 2019 рік (рисунок Б.18), 2020 рік (рисунок Б.19) та зображення з впровадженим вірусом 2019 рік (рисунок Б.20) та 2020 рік (рисунок Б.21).

Рисунок Б.1 - Результат сканування шелл-код у форматі raw, що був згенерований за допомогою msfvenom 2019 рік

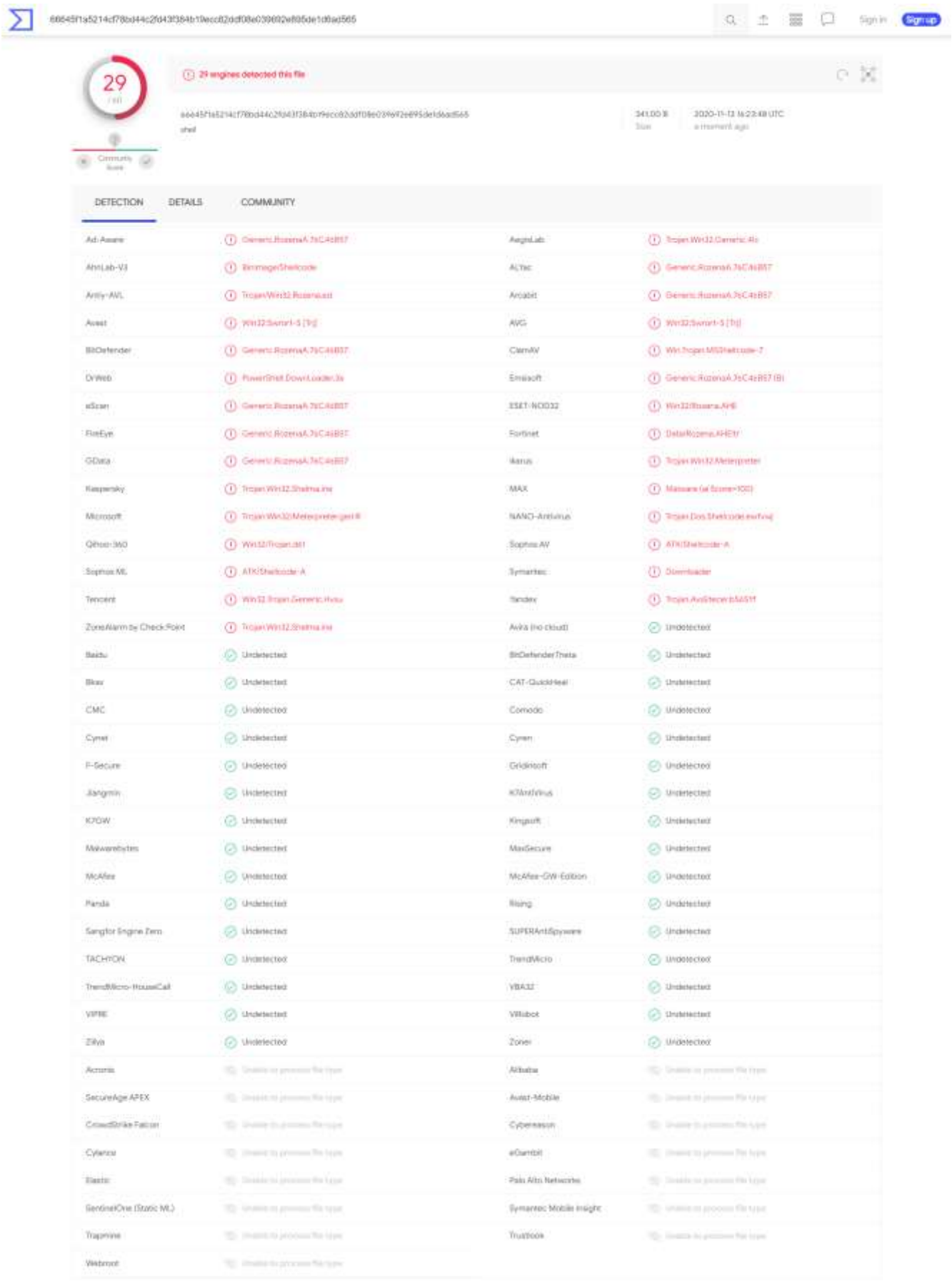


Рисунок Б.2 - Результат сканування шелл-код у форматі raw, що був згенерований за допомогою msfvenom 2020 рік

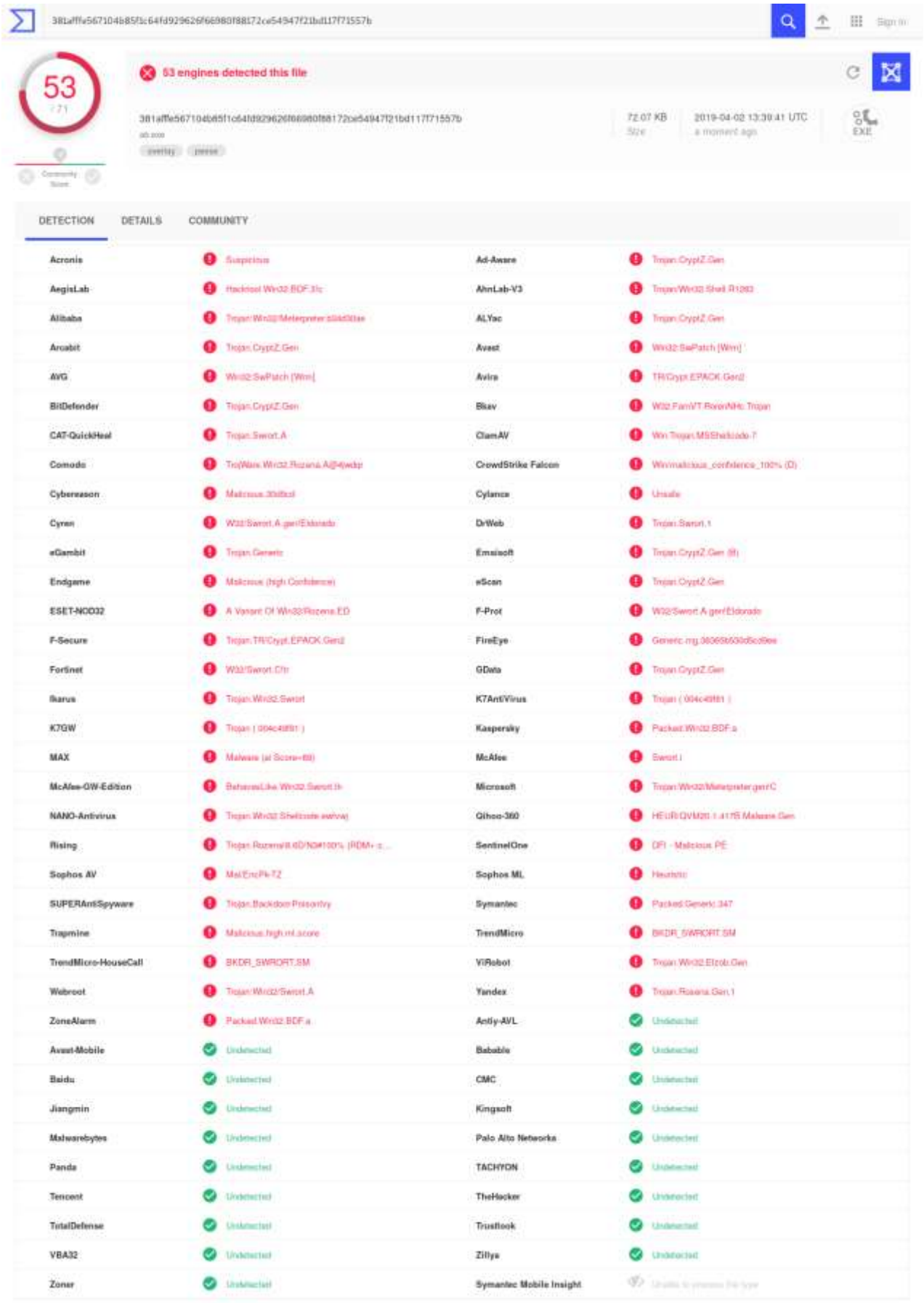


Рисунок Б.3 - Результат сканування шелл-код у форматі exe, що був згенерований за допомогою msfvenom 2019 рік

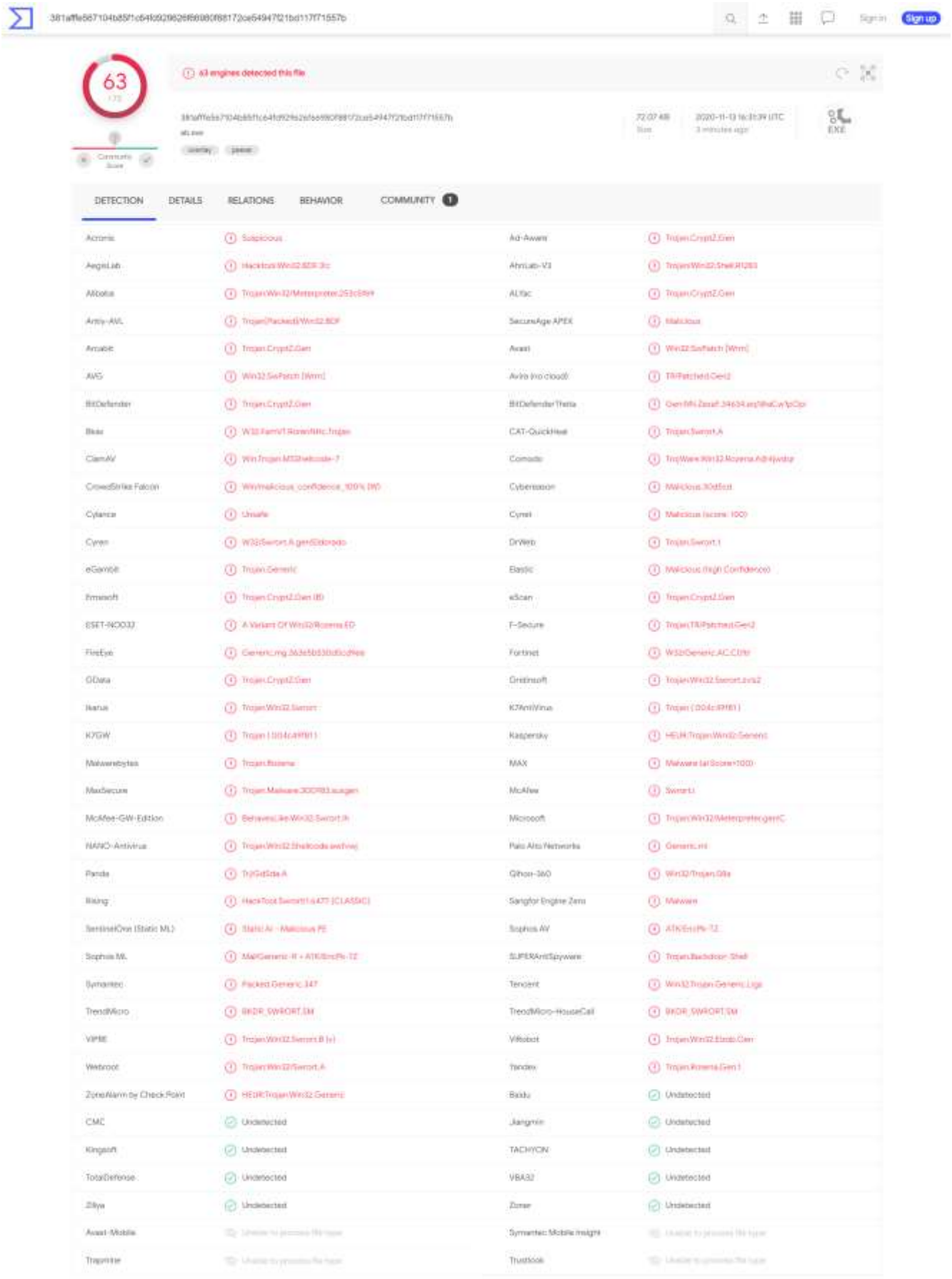


Рисунок Б.4 - Результат сканування шелл-код у форматі exe, що був згенерований за допомогою msfvenom 2020 рік

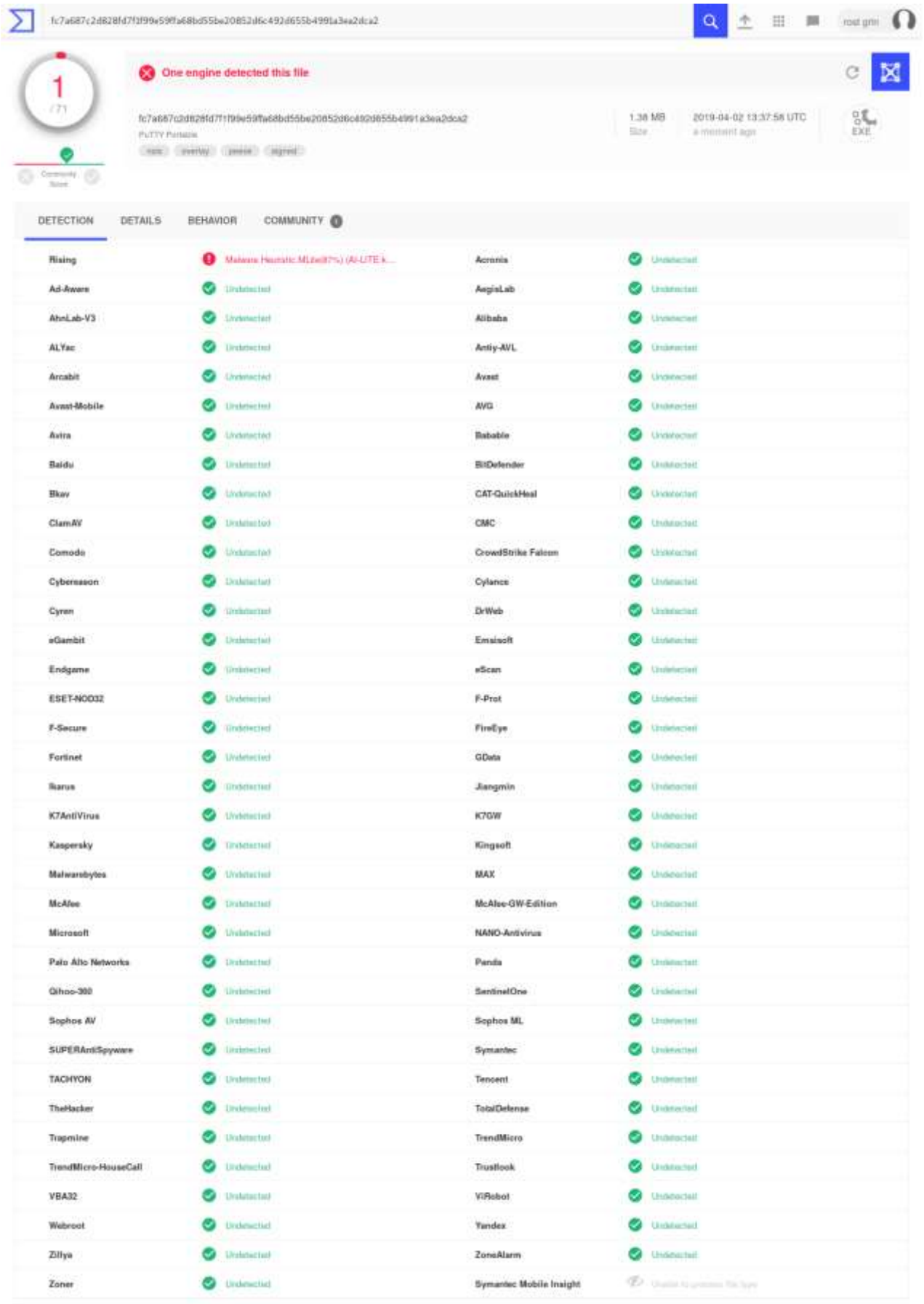


Рисунок Б.5 - Результат сканування оригінального PuTTY 2019 рік

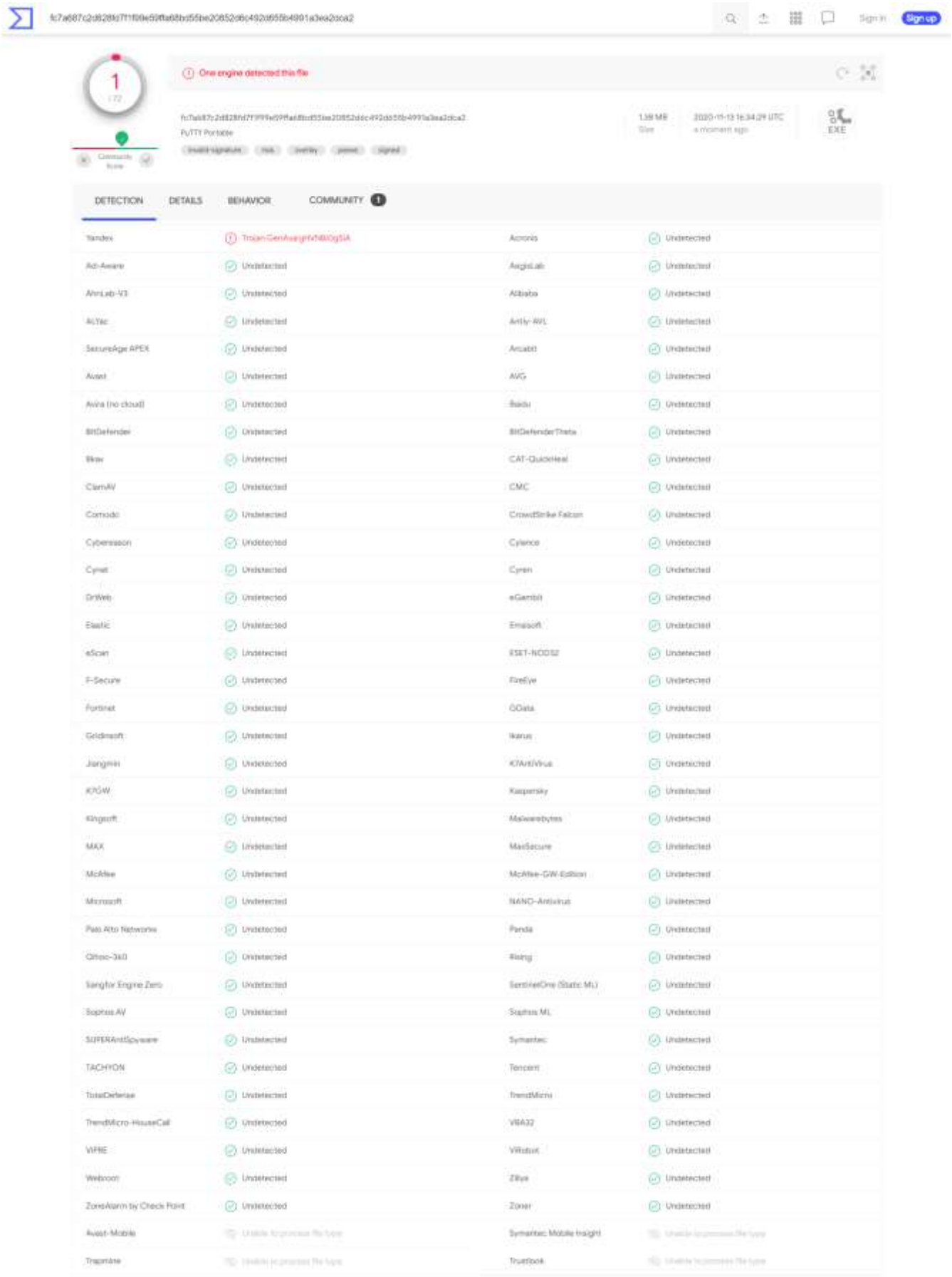


Рисунок Б.6 - Результат сканування оригінального PuTTY 2020 рік

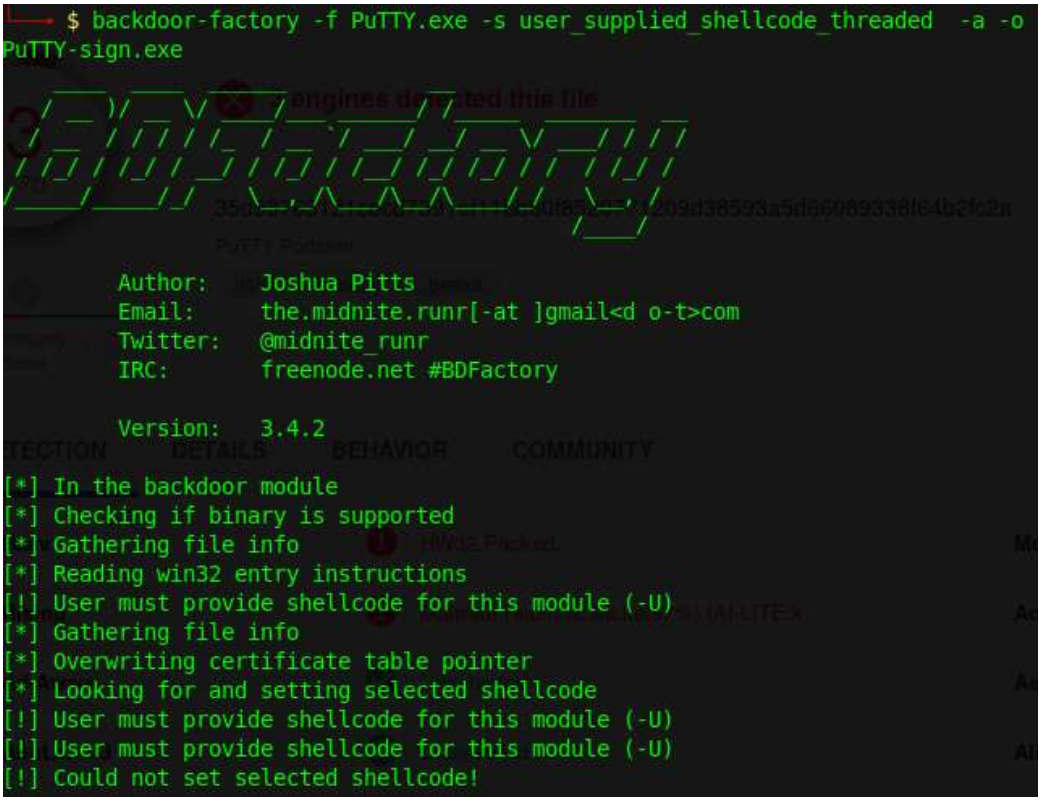


Рисунок Б.7 – Видалення цифрового підпису PuTTY

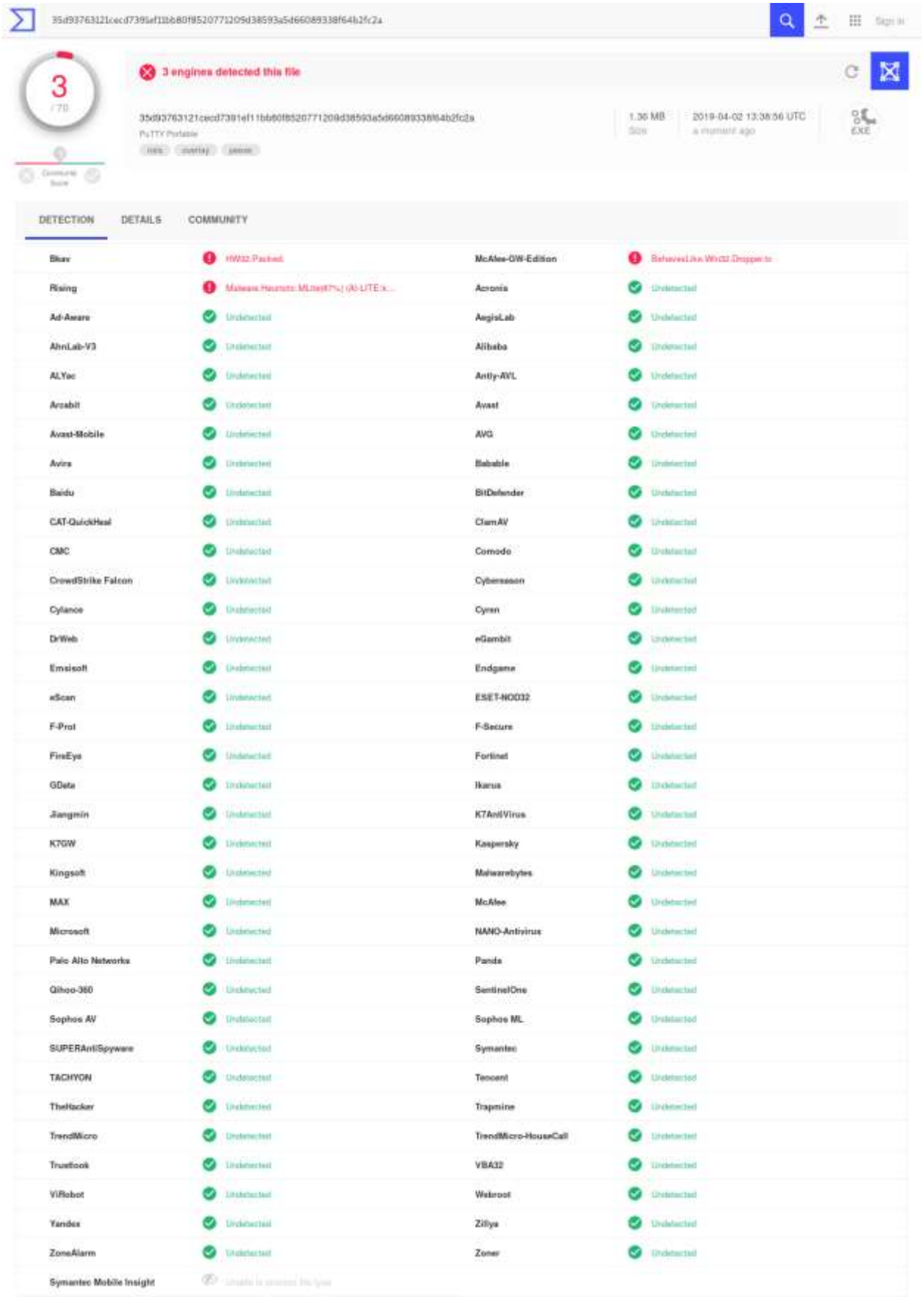


Рисунок Б.8 - Результат сканування оригінального PuTTY з видаленим цифровим підписом 2019 рік

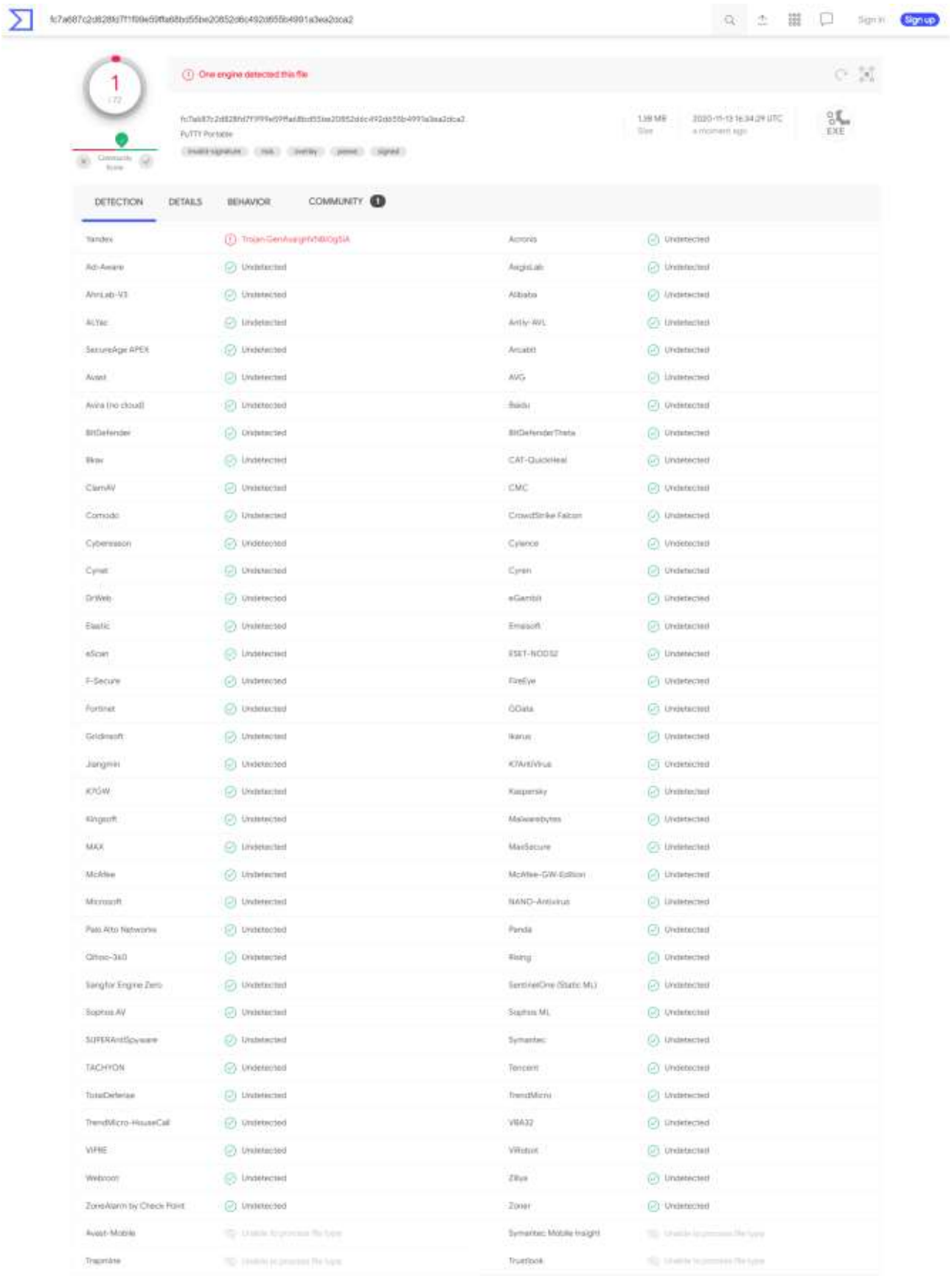


Рисунок Б.9 - Результат сканування оригінального PuTTY з видаленим цифровим підписом 2020 рік

Рисунок Б.10 – Створення нової порожньої секції в оригінальному PuTTY

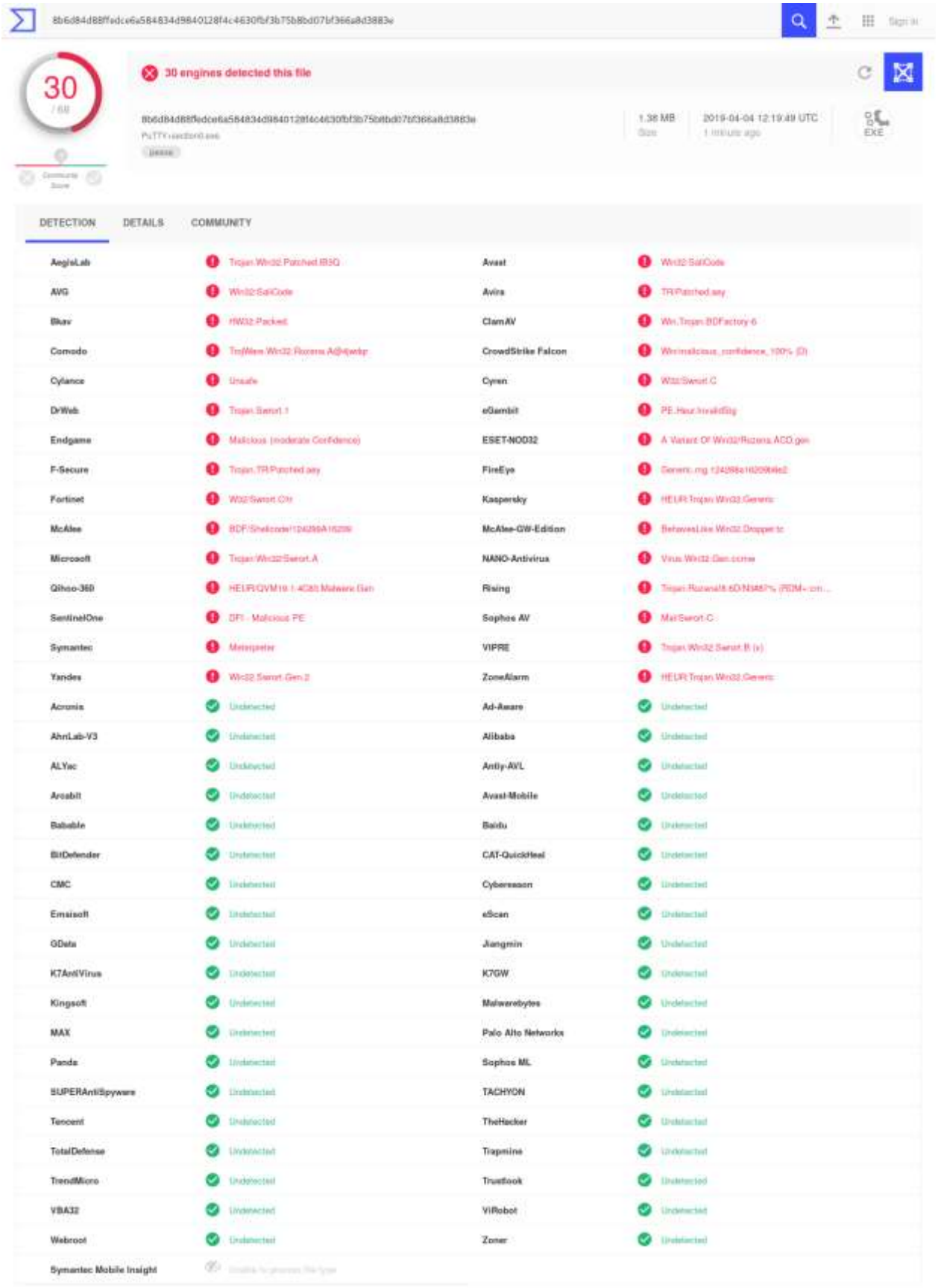


Рисунок Б.11 - Результат сканування оригінального PuTTY новою порожньою секцією 2019 рік

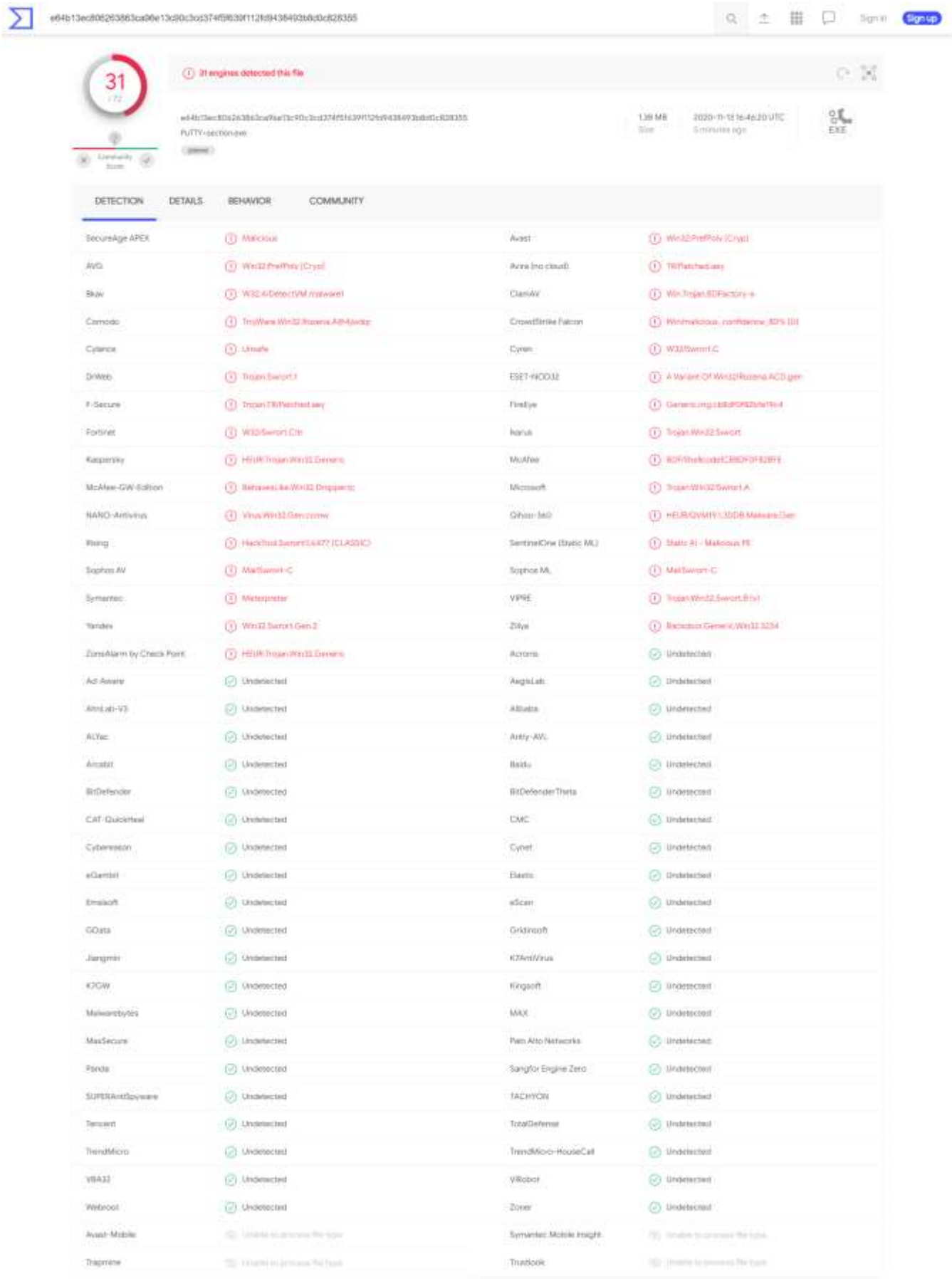


Рисунок Б.12 - Результат сканування оригінального PuTTY новою порожньою секцією 2020 рік

Рисунок Б.13 – Впровадження вірусу в нову порожню секцію

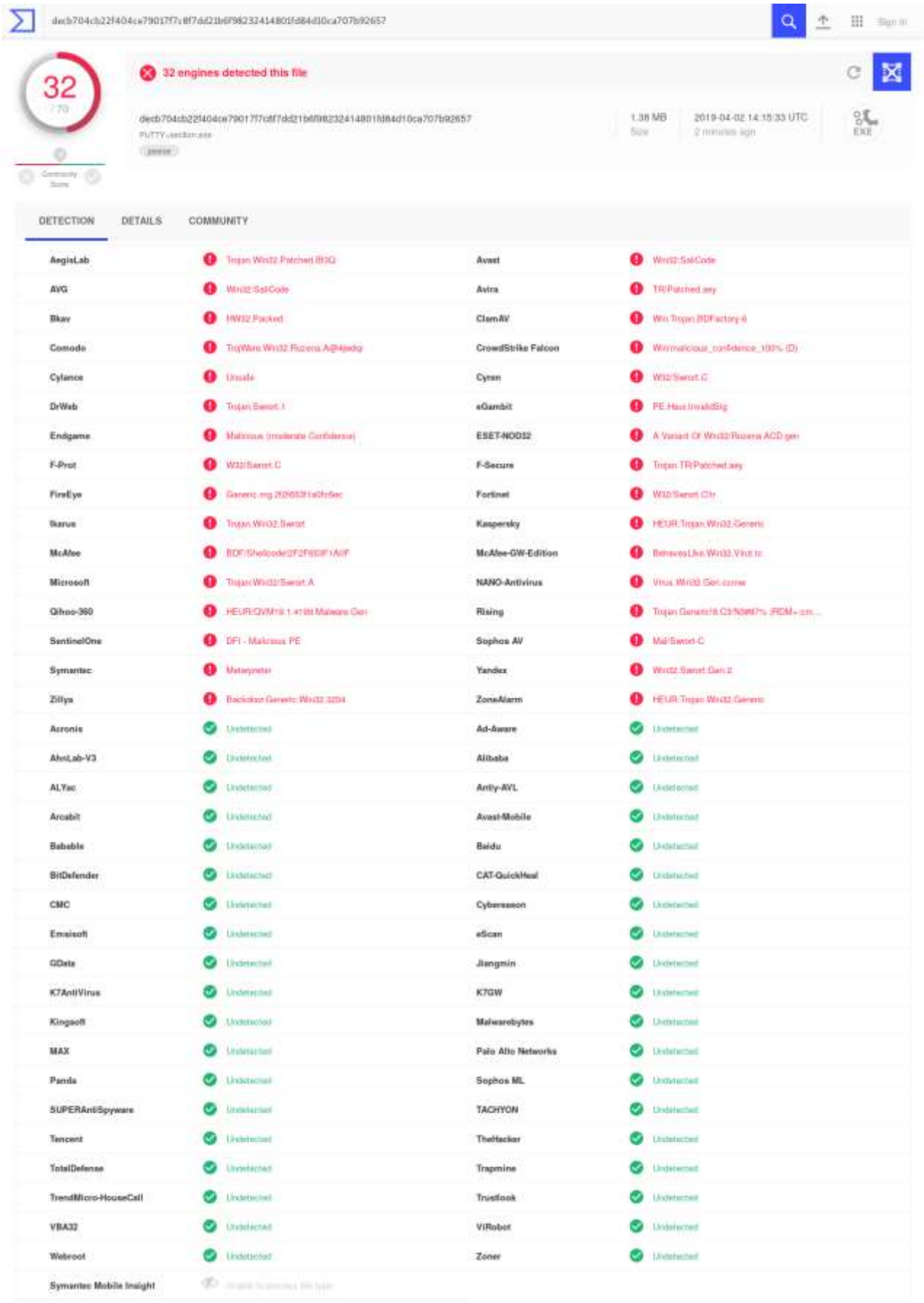


Рисунок Б.14 - Результат сканування PuTTY з вірусом, впровадженим в нову порожню секцію 2019 рік

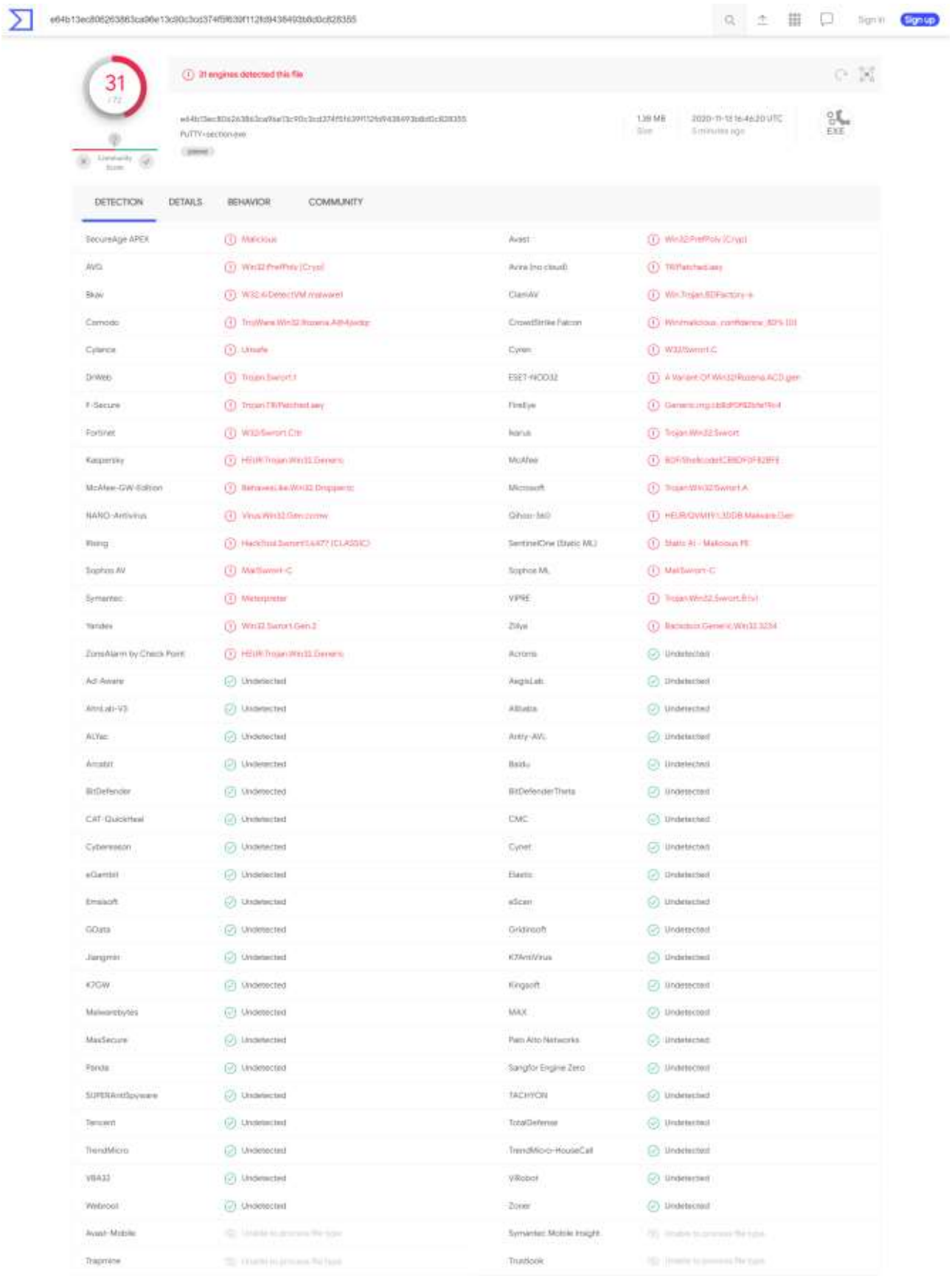


Рисунок Б.15 - Результат сканування PuTTY з вірусом, впровадженим в нову порожню секцію 2020 рік

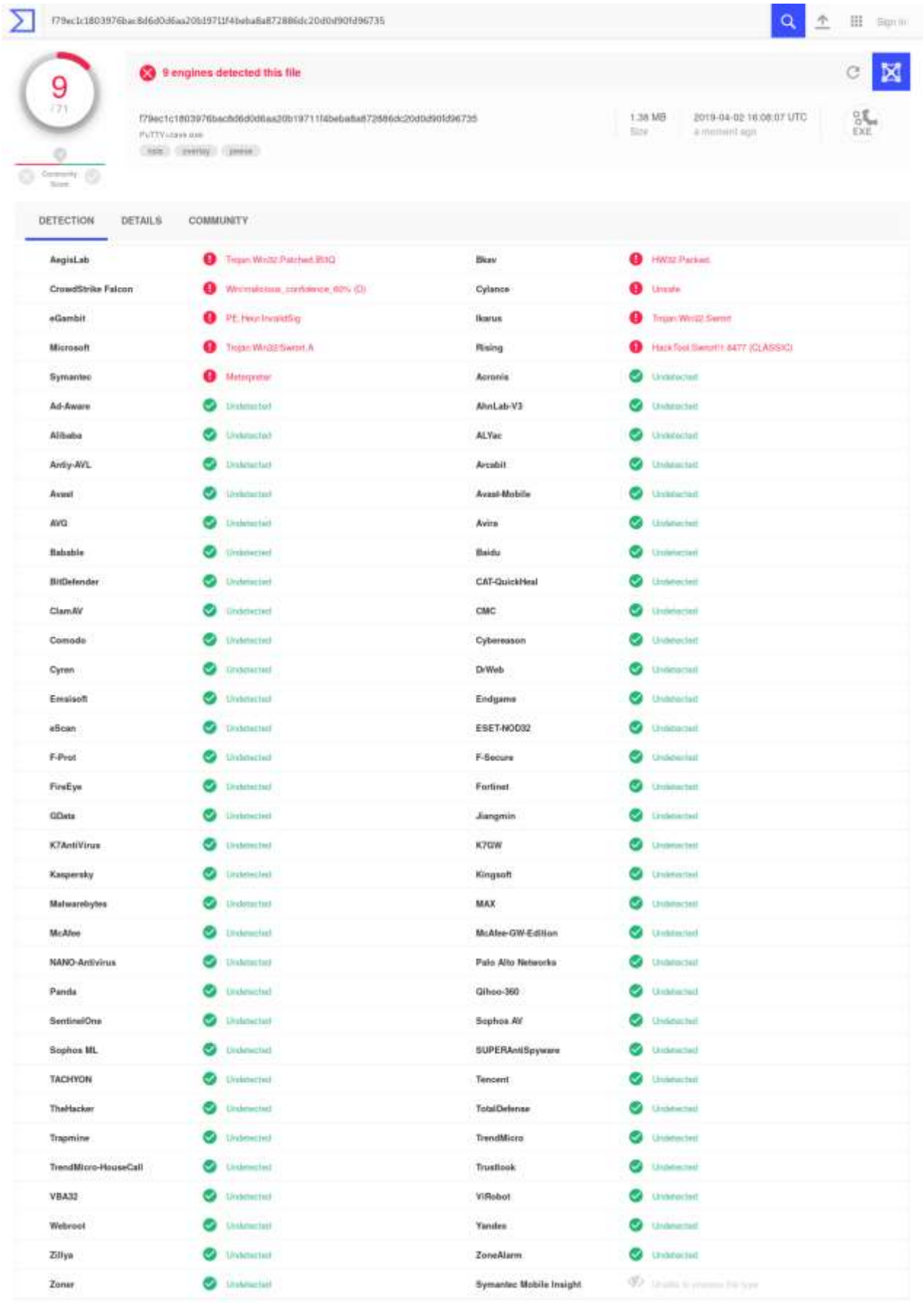


Рисунок Б.16 - Результат сканування PuTTY з вірусом впровадженим в code cave 2019 рік

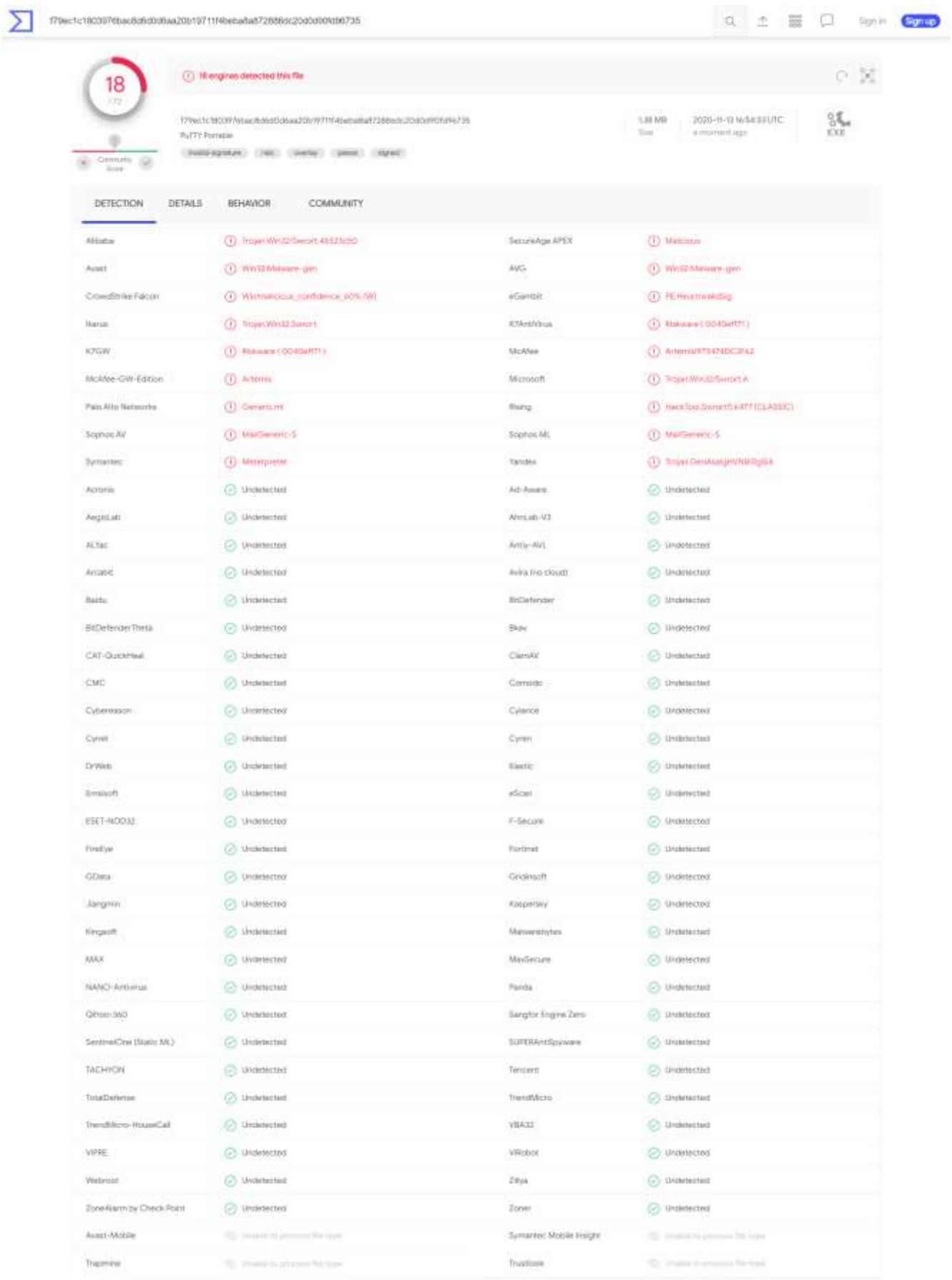


Рисунок Б.17 - Результат сканування PuTTY з вірусом впровадженим в code cave 2020 рік

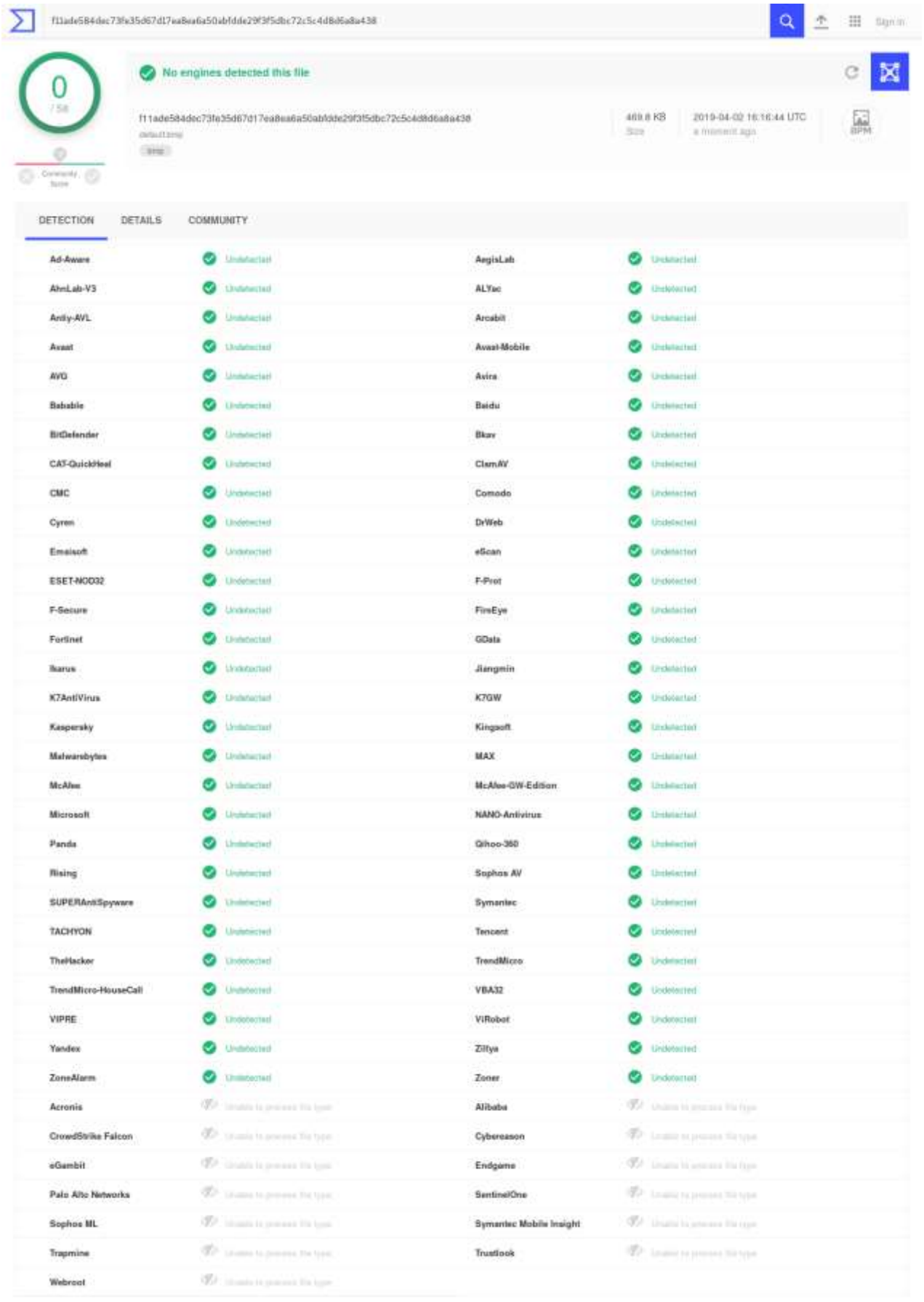


Рисунок Б.18 - Результат сканування оригінального зображення 2019 рік

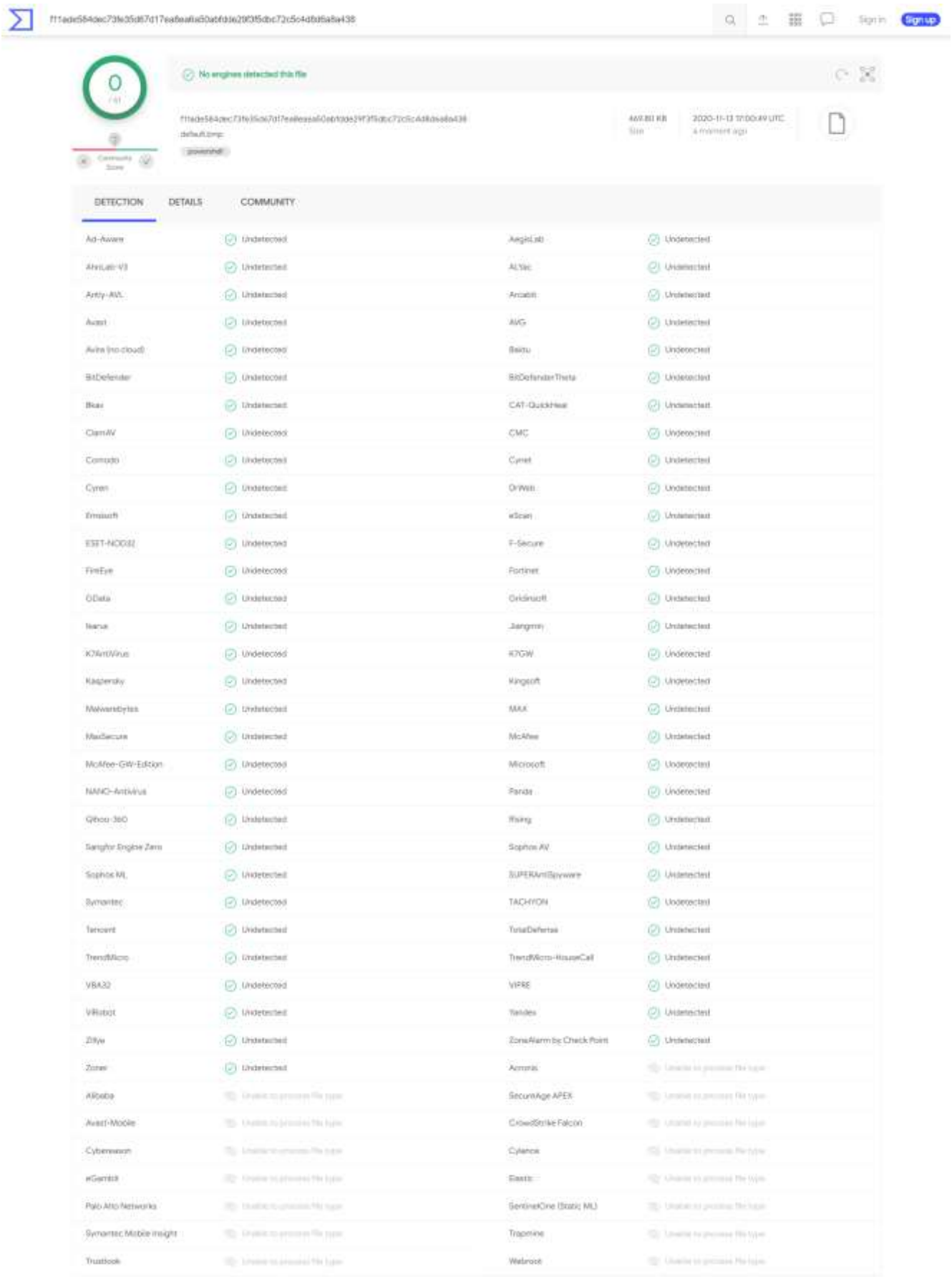


Рисунок Б.19 - Результат сканування оригінального зображення 2020 рік

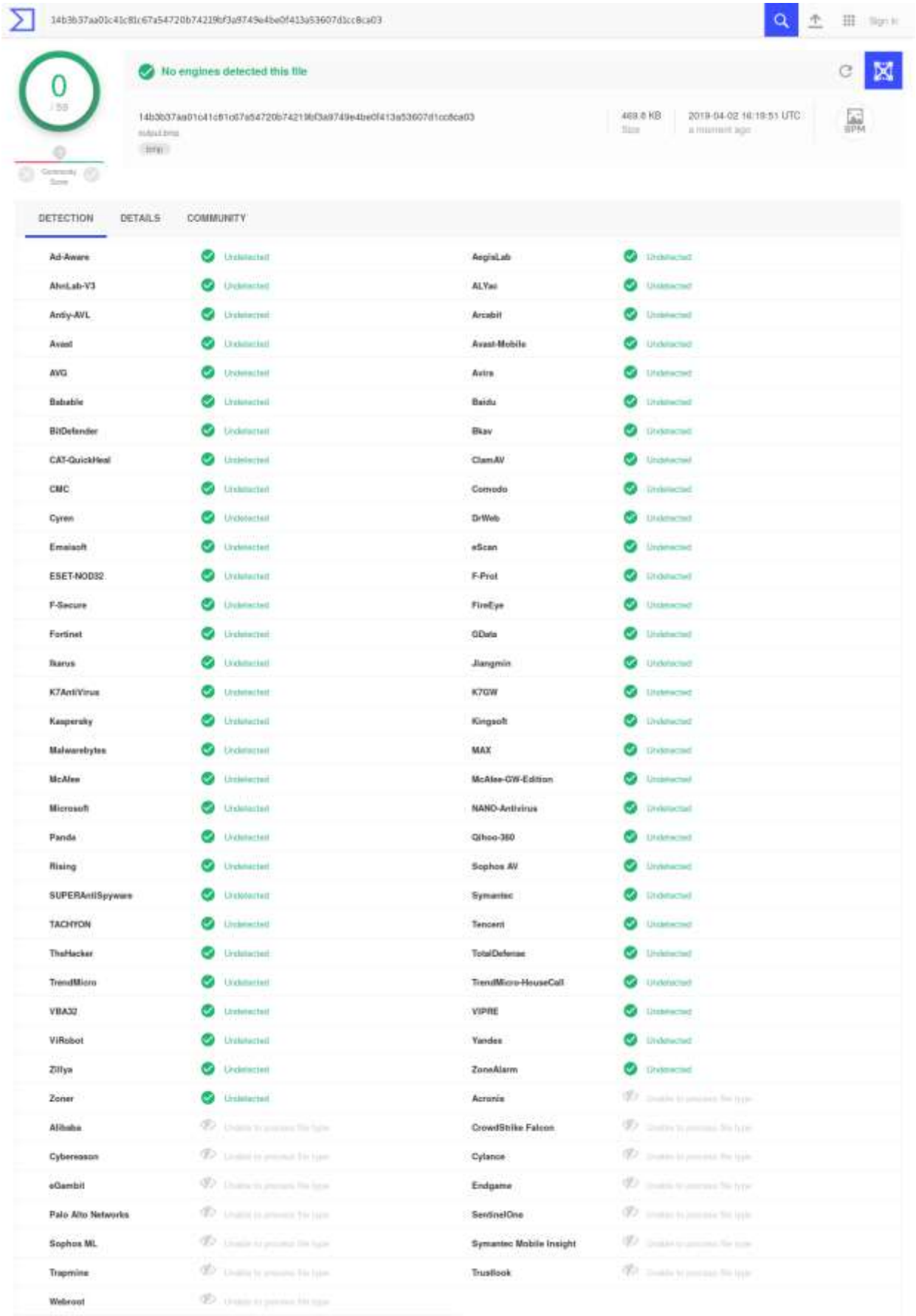


Рисунок Б.20 - Результат сканування зображення з прихованим вірусом 2019 рік

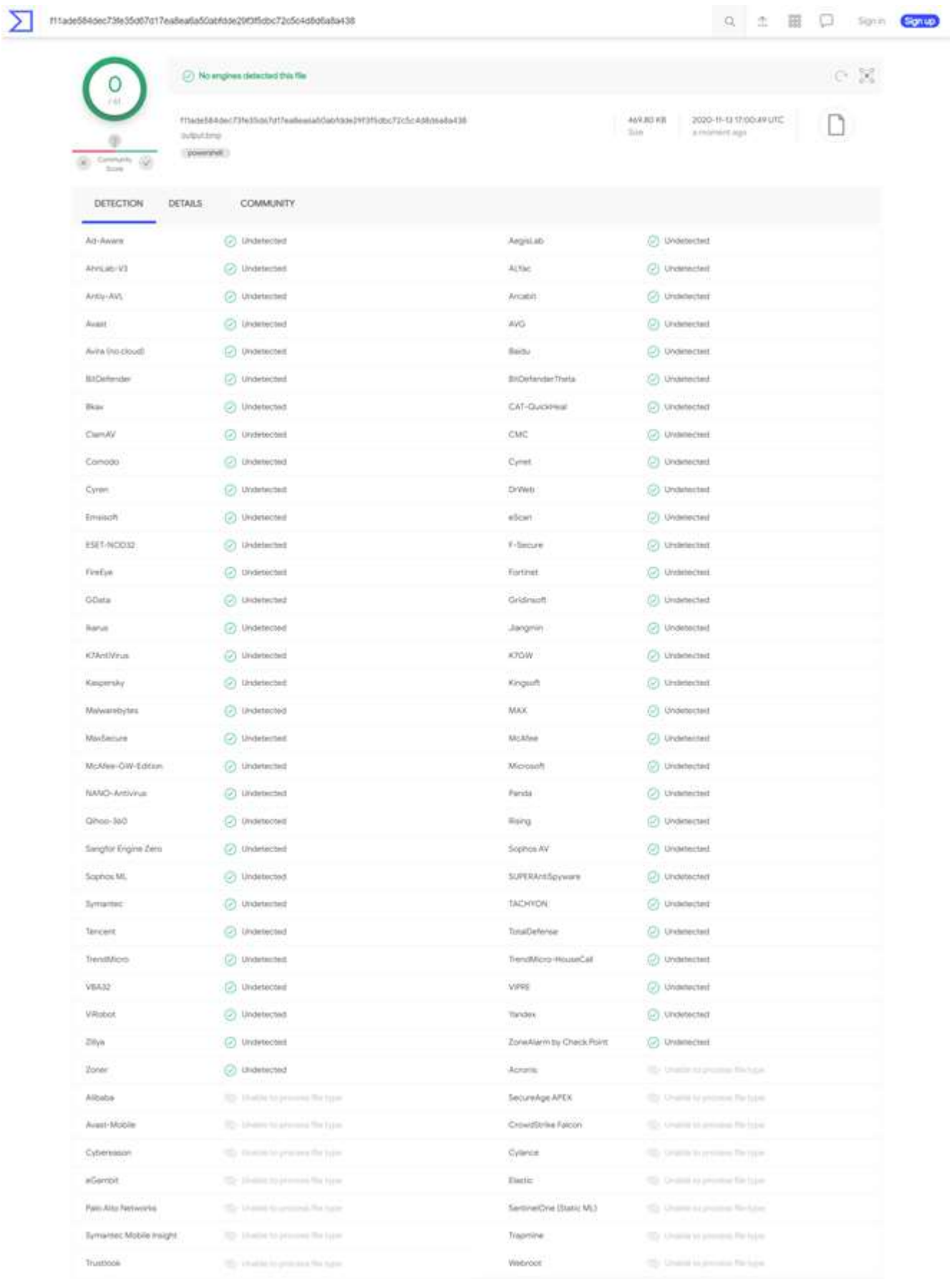


Рисунок Б.21 - Результат сканування зображення з прихованим вірусом 2020 рік

ДОДАТОК В

Детальна інформація про оригінальне зображення

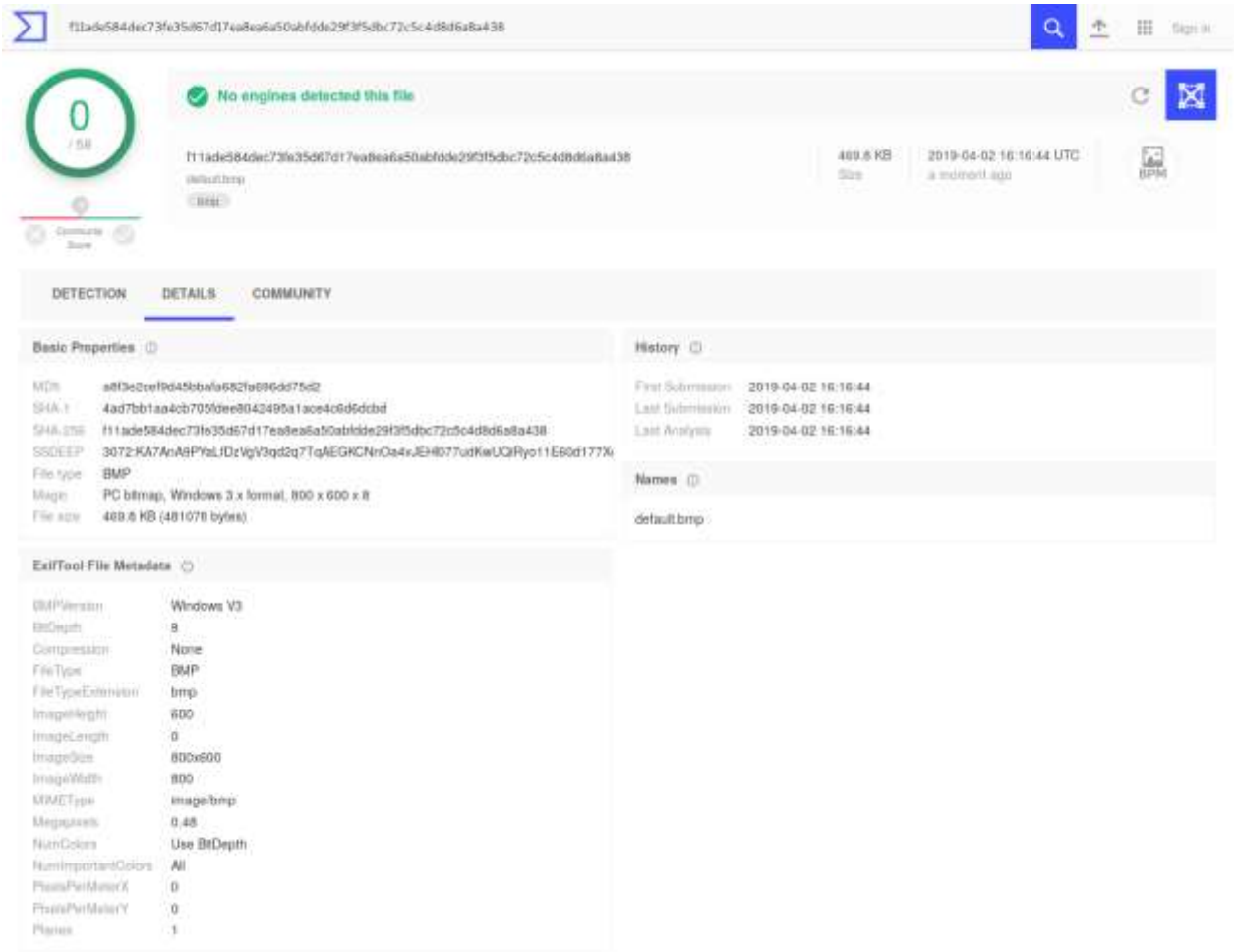


Рисунок В.1 - Детальна інформація про оригінальне зображення

ДОДАТОК Г

Детальна інформація про інфіковане зображення

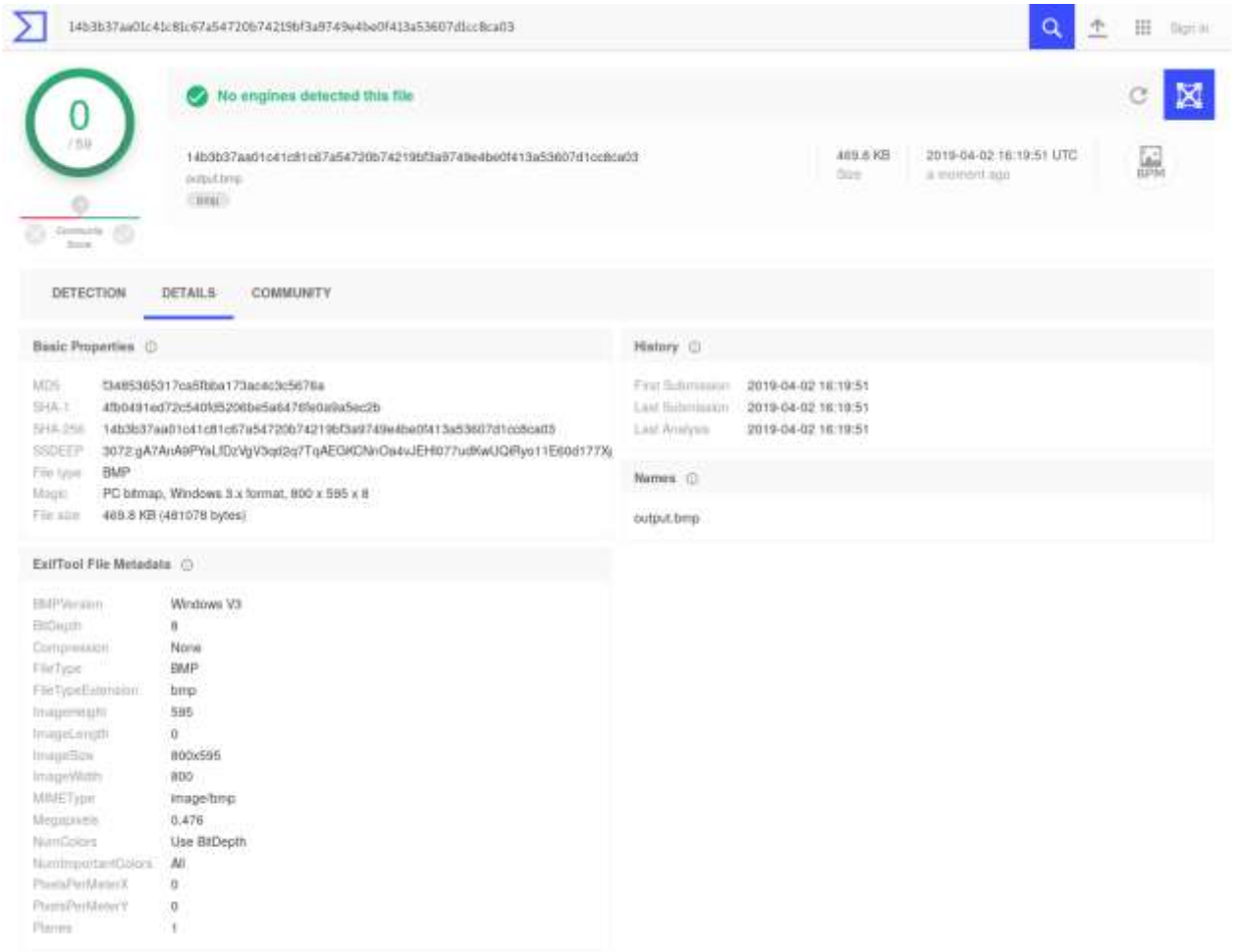


Рисунок Г.1 - Детальна інформація про інфіковане зображення

ДОДАТОК Д

Код програми BMP_CHECK.py

```
# -*- coding: utf-8 -*-
import os
import struct
import random
import time

def print_msg(msg): #Выводит сообщения и вспомагательную информацию
    print "\t\033[32m[+] >>> %s\033[00m" % msg

def print_error(error): #Выводит сообщения об ошибке во время работы модулей
    print "\t\033[91m[!] >>> %s\033[00m" % error

def print_info(info): #Выводит сообщения и вспомагательную информацию
    print "\t\033[33m[*] >>> %s\033[00m" % info


#convert string to hex
def toHex(s):
    lst = []
    for ch in s:
        hv = hex(ord(ch)).replace('0x', '')
        if len(hv) == 1:
            hv = '0'+hv
        lst.append(hv)

    return reduce(lambda x,y:x+y, lst)

image = {}

def Check(vars):
    data, size = get_file_data(vars) #Открытие файла для чтения
    if data:
        header, data = parse_image(data) #Считыввание данных из изображения
        image["header"] =  header #Заголовок файла
        image["data"] = data #Все остальное



        print_info("Тип файла %s " % header[0:2])
        if header[0:2] == "BM":
            print_msg("Файл является изображением формата BMP")
        else:
            print_error("Файл не является изображением формата BMP")
            exit()
```

```
        print_info("Размер файла в байтах указанный в заголовке %d" % struct.unpack("<i",
header[2:6])[0])
        print_msg("Настоящий размер файла в байтах %d" % size)
        if struct.unpack("<i", header[2:6])[0] == size:
            print_msg("Поле SIZE в заголовке файла не было модифицированно")
        else:
            print_error("Поле SIZE в заголовке файла было модифицированно %d != %d" %
(struct.unpack("<i", header[2:6])[0], size))

        HexRES1 = toHex(header[6:8])
        HexRES2 = toHex(header[8:10])
        if HexRES1 == "0000":
            print_msg("Зарезервированное поле №1 = %s" % HexRES1)
        else:
            print_error("Зарезервированное поле №1 = %s" % HexRES1)

        if HexRES2 == "0000":
            print_msg("Зарезервированное поле №2 = %s" % HexRES2)
        else:
            print_error("Зарезервированное поле №2 = %s" % HexRES2)

        image["width"] = header[18:22] #Ширина изображения в пикселях
        image["height"] = header[22:26] #Высота изображения в пикселях

        width = struct.unpack("<i", header[18:22])[0]
        #Распаковывает строку в формат int и получает значение ширины в пикселях
        #<  означает, что порядок цифр little-endian (от младшего к старшему или обратный)
        #i  означает, что тип int
        height = struct.unpack("<i", image["height"])[0]
        print_info("Размер изображения указанный в заголовке: %d x %d" % (width, height))
        print_info("Изображение должно состоять из %d пикселей" % (width * height))
        PixSTART = struct.unpack("<i", header[10:14])[0]
        print_info("Положение пиксельных данных относительно начала файла в байтах %d" %
PixSTART)
        Pixel = len(data[PixSTART:]) + 26
        if Pixel == width * height:
            print_msg("Реальное количество пикселей %d " % Pixel)
        else:
            print_error("Реальное количество пикселей %d != %d" % (Pixel, width * height))



def get_file_data(vars):
    data = ""
    path = os.getcwd()
    if os.path.exists(vars): #Проверка существования файла (относительный путь)
        data = open(vars, "rb").read()
        size = os.path.getsize(vars)
    elif os.path.exists(path + "/" + vars): #Проверка существования файла (полное указание пути)
        data = open(path + "/" + vars, "rb").read()
        size = os.path.getsize(path + "/" + vars)
```

```
        if data == "":
            print_error("Файл %s не найден" % vars)
            return False
    return data, size

def parse_image(data): #Получение данных из изображения
    header = data[:26] #Считывание заголока (первые 26 символов)
    data = data[(len(data) - 26) * -1:] #Считываются все оставшиеся символы
    return header, data

FileIn = raw_input("Введите путь к изображению: ")
#FileIn = "2.bmp"
Check(FileIn)
```

ДОДАТОК И

Слайди презентації

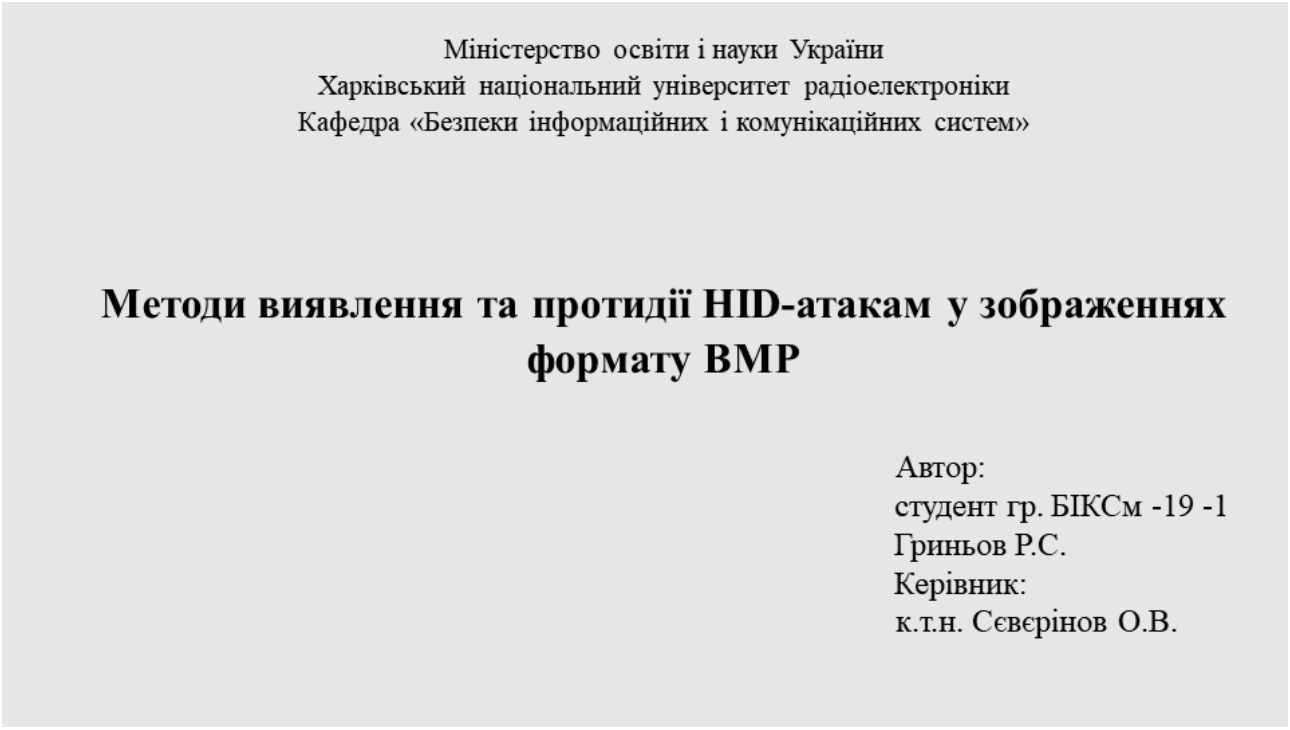


Рисунок И.1 – Слайд презентації № 1

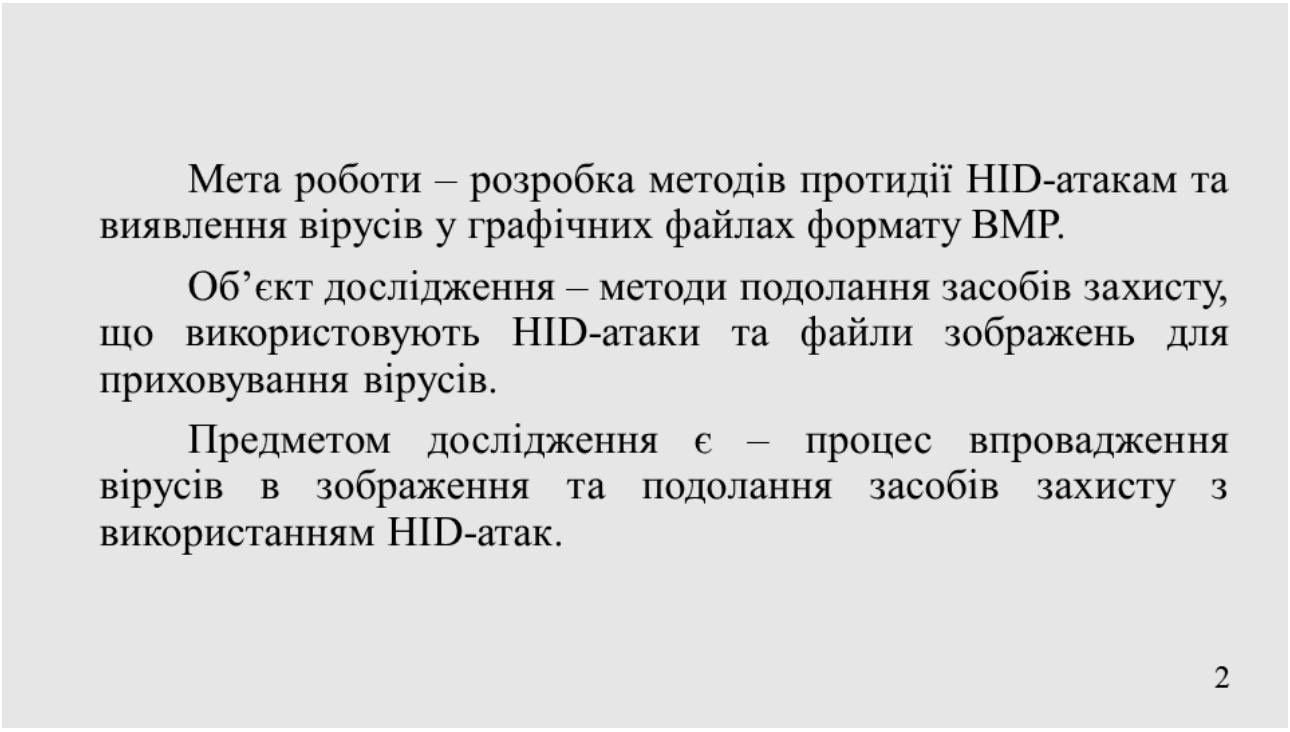


Рисунок И.2 – Слайд презентації № 2

Часткові задачі дослідження

1. Аналіз різноманітності та особливостей комп'ютерних вірусів
2. Аналіз способів та методів розповсюдження вірусів
3. Аналіз сучасних засобів захисту від комп'ютерних вірусів та вторгнень
4. Аналіз особливостей HID-атак та атак, що використовують файли зображень формату BMP
5. Розробка методів протидії HID-атакам та виявлення вірусів у графічних файлах формату BMP
6. Аналіз результатів випробувань

3

Рисунок И.3 – Слайд презентації № 3

Класифікація комп'ютерних вірусів

Комп'ютерні віруси можна класифікувати за наступними ознаками:

1. За деструктивним впливом
2. За способом зараження
3. За середовищем існування
4. За особливостями алгоритму

4

Рисунок И.4 – Слайд презентації № 4

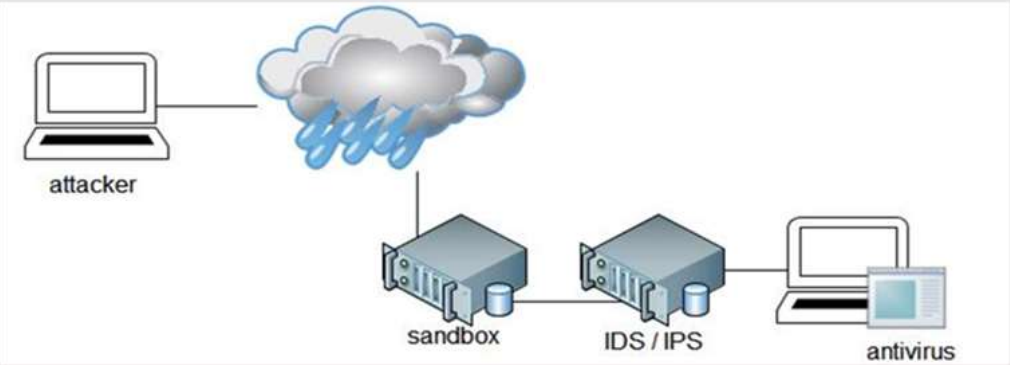
Способи розповсюдження вірусів

1. Претекстінг
2. Фішинг
3. Послуга за послугу
4. Дорожнє яблуко
5. Зворотня соціальна інженерія
6. Неліцензійне програмне забезпечення

5

Рисунок И.5 – Слайд презентації № 5

СУЧАСНІ ЗАСОБИ ЗАХИСТУ



Візуалізація шляху вірусу до цільового комп'ютера

6

Рисунок И.6 – Слайд презентації № 6

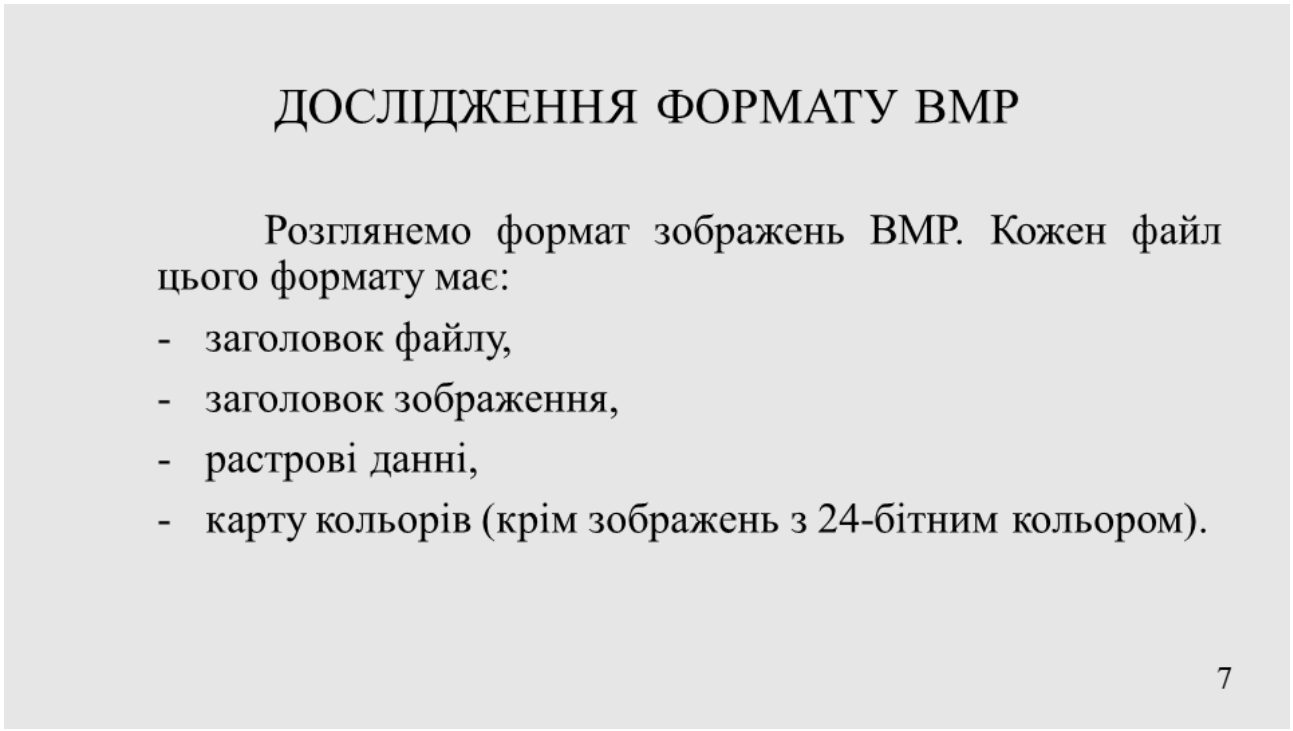


Рисунок И.7 – Слайд презентації № 7

Зміщення	Розмір (байт)	Ім'я	Опис
0	2	Type	Сигнатура формату. Використовується для ідентифікації формату. Має бути 4D42(hex)/424D(hex) (little-endian/big-endian). Після приведення до системи ASCII-символів має вигляд "BM".
2	4	Size	Розмір файлу в байтах.
6	2	Reserved 1	Зарезервоване поле має містити 0.
8	2	Reserved 2	Зарезервоване поле має містити 0.
10	4	OffsetBits	Положення піксельних даних відносно початку файлу (в байтах).

Рисунок И.8 – Слайд презентації № 8

Зміщення	Розмір (байт)	Ім'я	Опис
14	4	Size	Довжина заголовку.
18	4	Width	Ширина зображення в пікселях.
22	4	Height	Висота зображення в пікселях.
26	2	Planes	Кількість площин.
28	2	BitCount	Глибина кольору, біт на піксель (1, 4, 8, 24).
30	4	Compression	Тип компресії (0 – відсутня, 1 – RLE-8, 2 – RLE-4).
34	4	SizeImage	Розмір зображення, байт (включно з доповненням).
38	4	XpelsPerMeter	Горизонтальна роздільна здатність, пікселів на метр.
42	4	YpelsPerMeter	Вертикальна роздільна здатність, пікселів на метр.
46	4	ColorsUsed	Число кольорів, що використовується (0 – максимально можливе для даної глибини кольору).
50	4	ColorTable	Кількість основних кольорів.

Рисунок И.9 – Слайд презентації № 9

МЕТОДИ ПРОТИДІЇ

- перевірка зарезервованих полів;
- перевірка поля "Size" в заголовку файлу;
- отримання з заголовку зображення даних, що вказують на горизонтальну та вертикальну кількість пікселів;
- крім того можна використовувати методи, що дозволяють знаходити аномалії у зображеннях.

10

Рисунок И.10 – Слайд презентації № 10

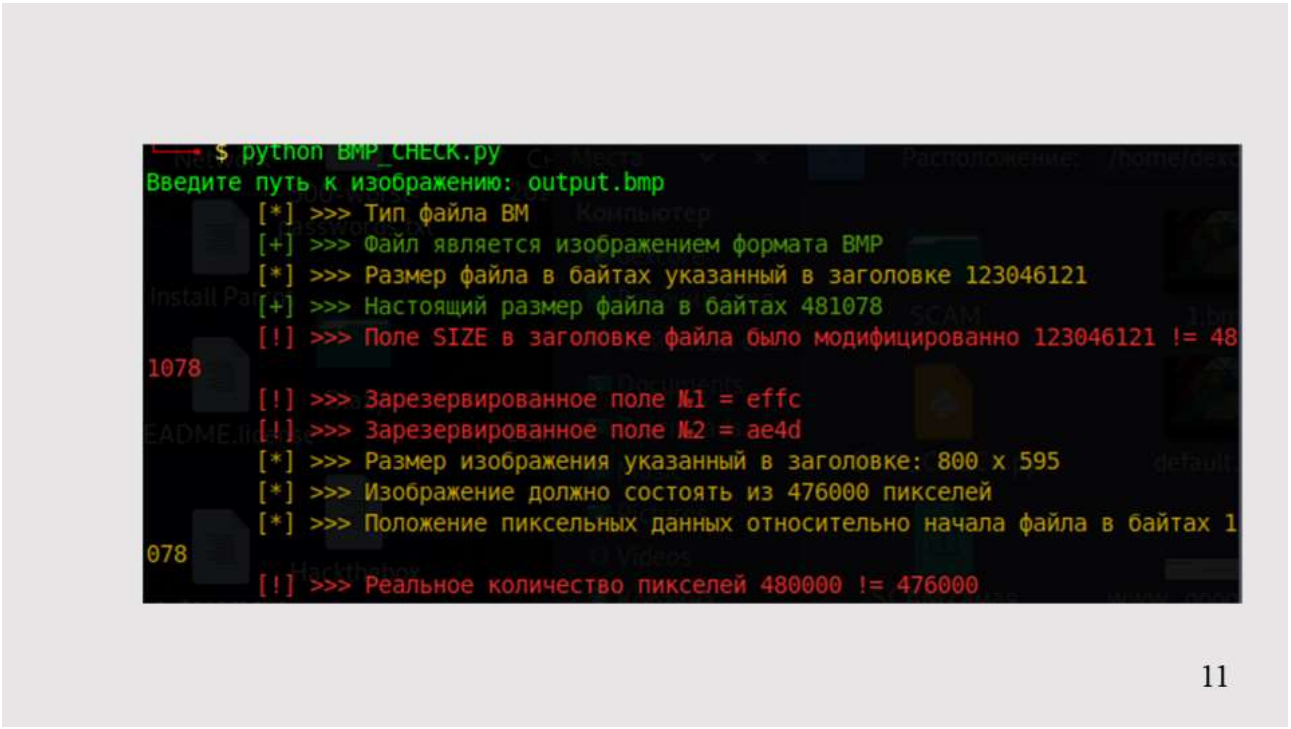


Рисунок И.11 – Слайд презентації № 11

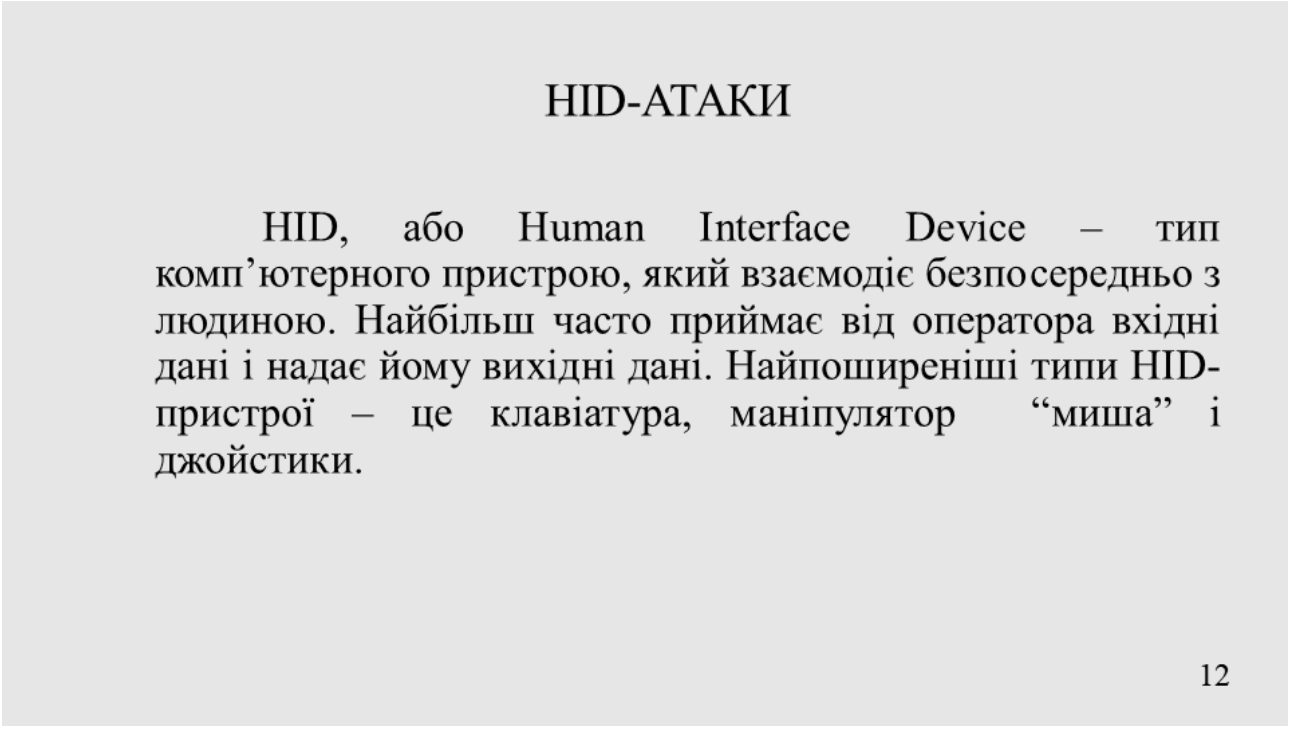


Рисунок И.12 – Слайд презентації № 12

13

Рисунок И.13 – Слайд презентації № 13

ІСНУЮЧІ ПРИСТРОЇ ДЛЯ ПРОВЕДЕННЯ HID-АТАК

USB Rubber Ducky

Bash bunny

USBHarpoon та O.MG Cable

14

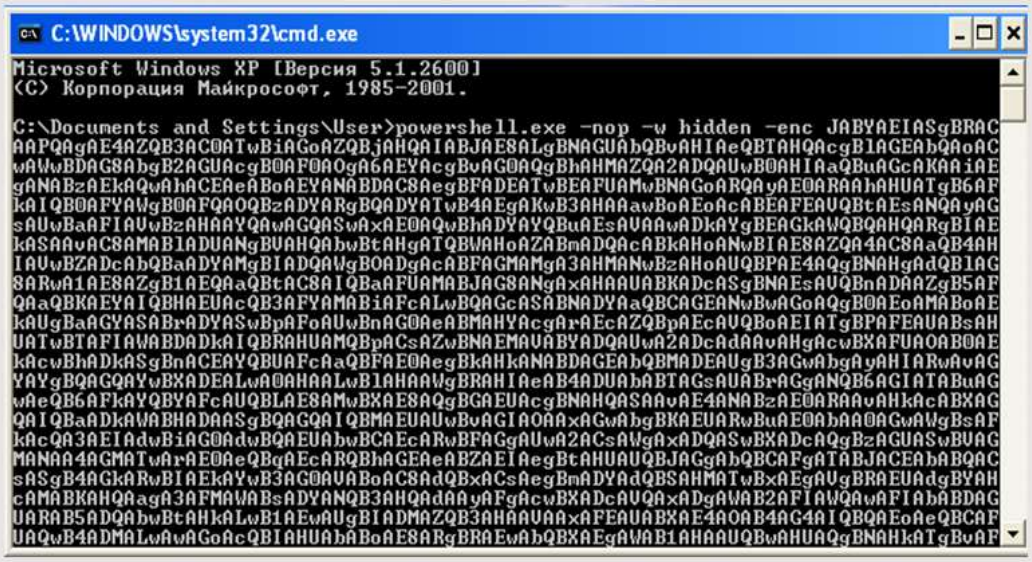
Рисунок И.14 – Слайд презентації № 14

МЕТОДИ ПРОТИДІЇ

- заборона на встановлення з'ємних пристроїв;
- використовувати список довірених пристроїв;
- заборона фізичного доступу до USB-портів;
- найбільш вдалим та ефективним є використання евристичних методів для виявлення і блокування HID-атак.

15

Рисунок И.15 – Слайд презентації № 15



16

Рисунок И.16 – Слайд презентації № 16

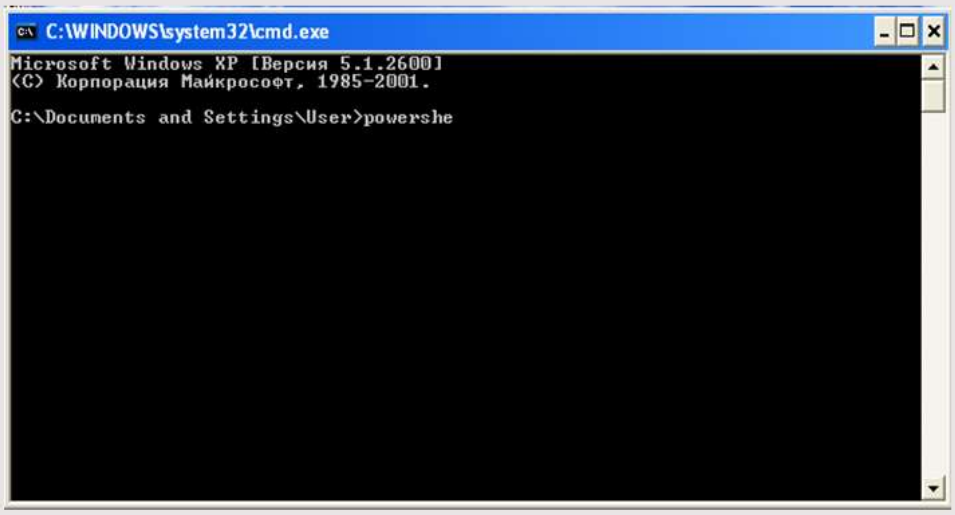


Рисунок И.17 – Слайд презентації № 17

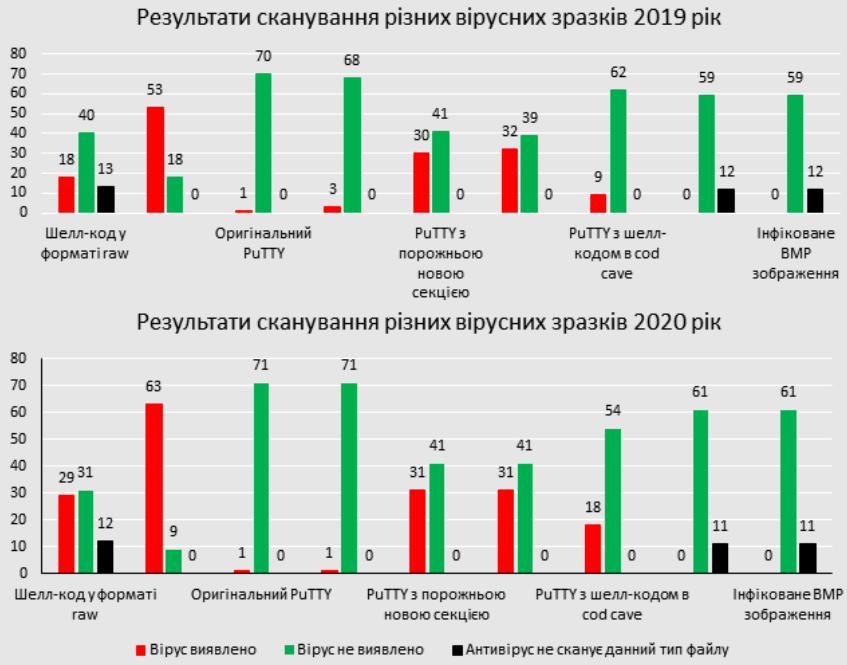


Рисунок И.18 – Слайд презентації № 18

Висновки

В рамках дипломної роботи були розроблені методи протидії HID-атакам та виявлення вірусів у графічних файлах формату BMP. В роботі були проаналізовані:

- різних типів комп'ютерних вірусів, способів їх розповсюдження та подолання засобів захисту;
- сучасних засобів захисту від комп'ютерних вірусів та вторгнень;
- особливості файлів формату BMP;
- особливості HID-атак;
- існуючих пристроїв для проведення HID-атак;
- розроблена програма та проаналізовані результати.

19

Рисунок И.19 – Слайд презентації № 19

ВІДОМІСТЬ МАГІСТРСЬКОЇ АТЕСТАЦІЙНОЇ РОБОТИ

№	Позначення	Найменування	Дод. відомості
		Текстові документи	
1	ГЮІК.ХХХХХ1412Ст05ПЗ	Пояснювальна записка	123 с.
		Графічні документи	
2		Слайд-презентація	19 слайдів
3		Інші документи	
		Електронна версія ПЗ	1
		Рецензія	1 с.
		Відгук керівника	1 с.

					ГЮІК.ХХХХХХ1412Ст05ВД	
Змін.	Арк	Номер докум.	Підп	Дата		
Розроб.		Гриньов Р.С.			Методи виявлення та протидії HID-атакам у зображеннях формату BMP	Літ: У; Аркуш; Аркушів
Перевір.		Сєвєрінов О.В.				
Н.контр.		Лихограй О.М.				ХНУРЕ Кафедра БІТ
Затв.		Халімов Г.З.				