***Documentatie***

***Tematica :***

Proiectul prezent este reprezentat de o aplicatie, la pornirea careia un spyware de tip keylogger este lansat. El actioneaza in background si este inofensiv sistemului.

***Atributii spyware:***

1. Inregistrarea tastaturii, ceea ce face este sa preia inputul de la tastatura, prelucrarea datelor si scrierea lor in fisier.

2. Inregistrareaza ecranul prin intermediul capturilor de ecran, salvate apoi drept poze in formatul dorit.

***Partea non-malitioasa ( aplicatia masca ) :***

Este realizata cu ajutorul framwork-ului open source, dezvoltat de Google : Flutter.

***Partea malitioasa ( spyware-ul ) :***

Este alcatuita din doua scripturi dezvoltate in Python.

***Motivarea tehnologiilor alese :***

***Front-end :***

- crossplatform: disponibil atat pe desktop, web, dar si mobile pe ambele sisteme principale – Android & iOS

- tehnologie dezvoltata si intretinuta de Google, ceea ce duce la calitate inalta a bibliotecilor existente, exemplu:

Metoda de lansare a unor noi procese

-> <https://api.dart.dev/stable/2.7.0/dart-io/Process-class.html>

- este stabil, data de lansare este 2017

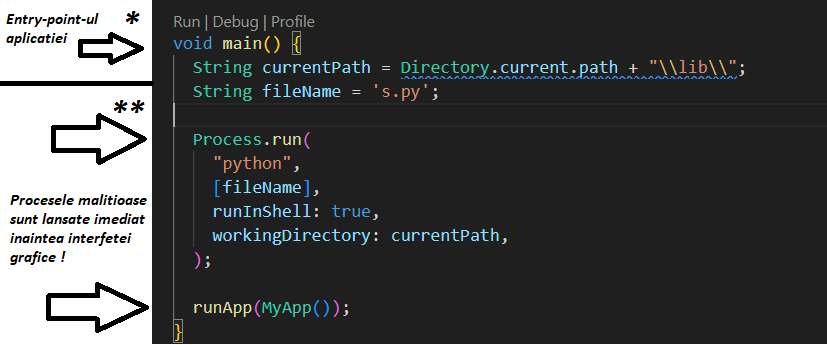
- experienta personala cu Flutter

***Back-end :***

* limbaj de programare lightweight
* permite accentuarea partii de algoritmica in defavoarea structurilor de date si implementarii structurilor de date folosite, lucru potrivit mai ales pentru scripturile de dimensiuni mici.
* experienta personala cu Python
* popularitatea sa pentru scripting, lucru care se reflecta in cantitatea de documentatie si tutoriale existente, disponibile gratuit.

***Modalitatea de implementare & Vulnerabilitatea exploatata :***

La pornirea aplicatiei se lanseaza cele 2 scripturi cu intentii malitioase, urmarindu-se mascarea lor printre cele necesare aplicatiei.

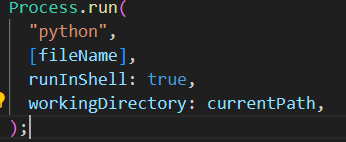


*Biblioteca folosita pentru lansarea proceselor:*

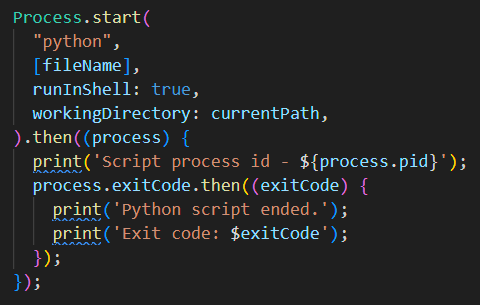
<https://api.dart.dev/stable/2.7.0/dart-io/Process-class.html>

Procesele pot fi indepedente sau dependente de aplicatia care le-a lansat, exemplu corespunzator bibliotecii folosite :

1. Lansarea unui proces independent :



2. Lansarea unui proces dependent :



Persistenta urmatoare inchiderii aplicatiei constanta intr-o eroare comuna in programare. Cele 2 procese se considera resurse externe care nu sunt gestionate ‘corespunzator’, nu exista eliberarea lor.

Astfel, se obtine comportamentul unui memory leak.

What is a memory leak ?

<https://en.wikipedia.org/wiki/Memory_leak>

***Flow / Demers :***

1. La pornirea aplicatiei, se lanseaza un proces independent care ruleaza scriptul pentu inregirarea ecranului .

2. Acest subproces, la randul sau, lanseaza un proces care ruleaza scriptul ce se ocupa cu inregistrarea tastelor.

3. Fisierul main.dart este cel care contine fisierul main pentru aplicatia flutter.

La fiecare rulare, se creeaza folderul numit ‘ss’ in locatia fisierului principal ( “lib\\ss”).

In interiorul folderului ‘ss’, se creeaza un fisier txt, ‘log.txt’ ( “lib\\ss\\log.txt”).

Resursele generate se gasesc in “ProiectFlutter\lib\ss”.

!!! Fiecare script se poate oprii in diferite modalitati.

Cele implementate sunt :

* Scriptul keylogger se opreste prin apasarea tastei ESC ;
* Scriptul ce inregistreaza se opreste dupa consumarea celor 150 secunde, fiind configurat cu 1 screenshot / secunda ;

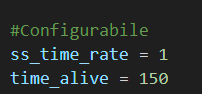
***Fisierele cheie ( explicate in 3 puncte continut, configurabilitate, implementare ) :***

*Script screenshot-uri ( s.py ) :*

1. Contine scriptul pentu inregirarea ecranului sub forma de screenshot-uri.

2. Configurabile:

* Rata screenshot-uri/secunda, initial 1.
* Timpul real de executie al programului, initial 150 secunde.

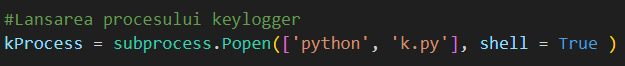


3. Contine 3 etape – initializare, lansarea altui proces, executia logicii :

Initializare :

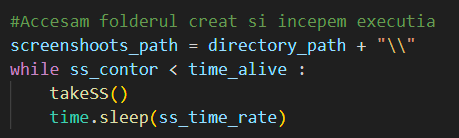


Lanseaza procesul ce ruleaza fisierul k.py :

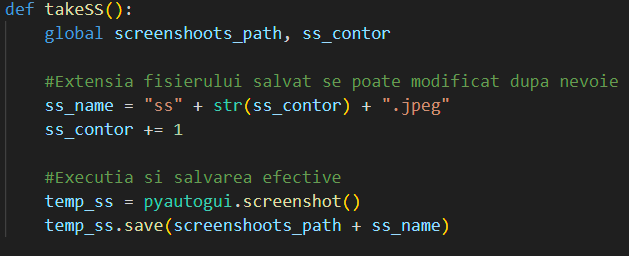


Parametrul shell este folosit doar daca se doreste un proces independent.

Executia logicii :



Functia ‘takeSS’ :

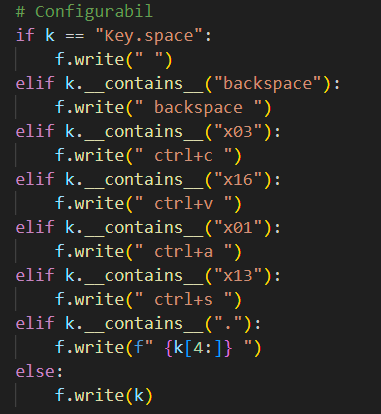


*Script screenshot-uri ( k.py ) :*

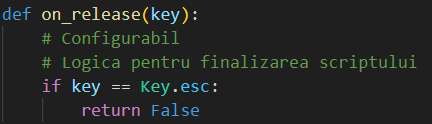
1. Contine scriptul keylogger.

2. Configurabile nu exista posibilitati ‘setari’, insa :

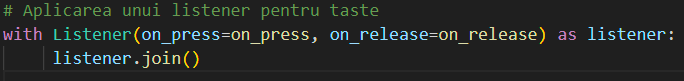
* Se pot modela: fiecare combinatie de taste, initial sunt aplicate parsare si scriere in clar doar pentru: backspace, ctrl+c, ctrl+v, ctrl+a, ctrl+s, space.



* Logica pentru finalizarea scriptului



3.Contine logica necesara functionarii scriptului:



*Entry-point-ul in aplicatia Flutter ( main.dart ) :*

1. Contine bootstrapper-ul atat pentru aplicatia flutter, cat si pentru primul script, dar si lansarea acestuia.

2. Configurabile:

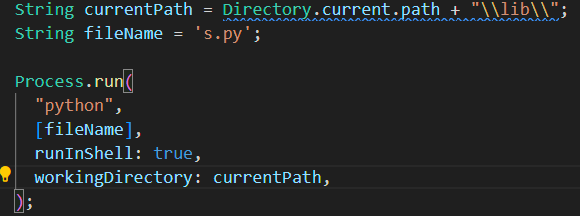
-Nume fisier executat

-Path in functie de cel al proiectului flutter.

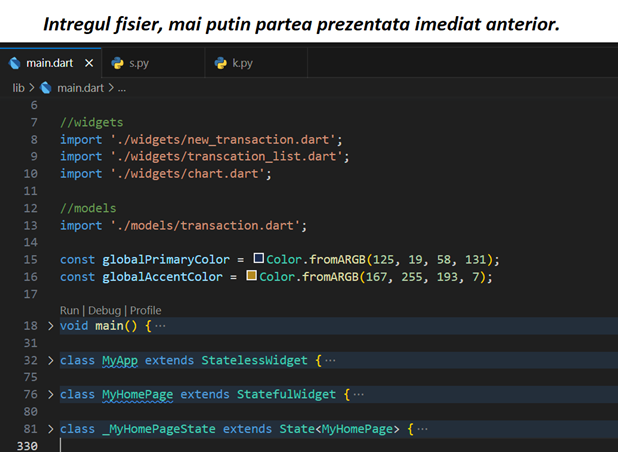


3. Contine 2 etape - initializarea si lansarea sciptului malitios, ruleaza aplicatia flutter.

Partea malitioasa :



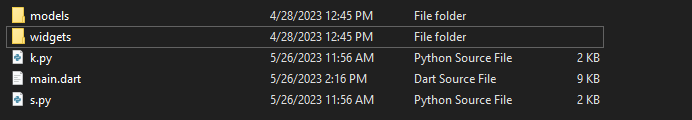
Partea non-malitioasa :



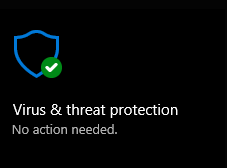
***Ghid de utilizare :***

0. Starea initiala:

\* Fisiere existente:



\*\* Antivirus pornit:



\*\*\* Pregatiti enviroment-ul :

- Instalati Flutter, VS Code, apoi configurati VS Code pentru el

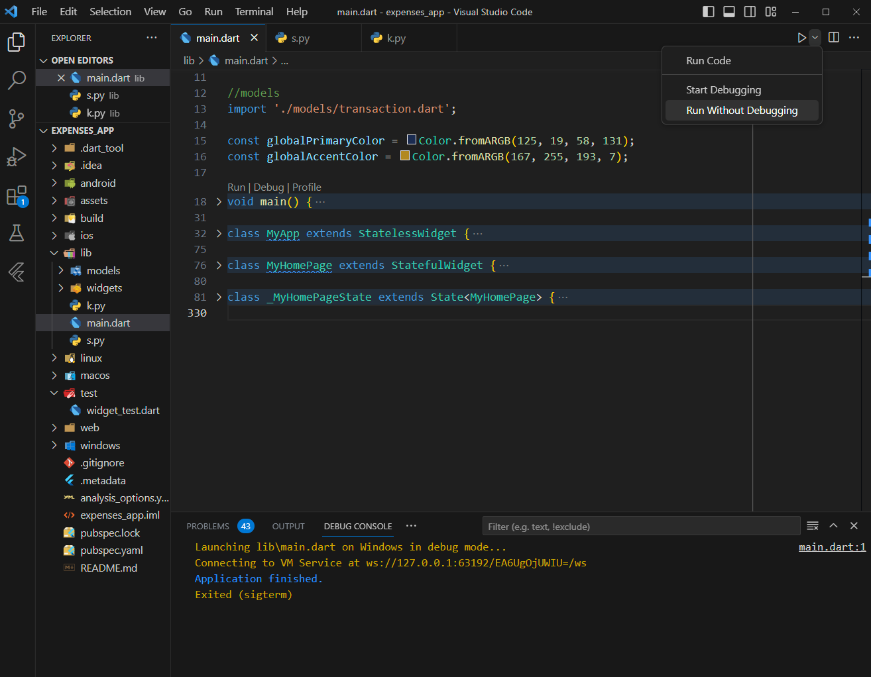
<https://www.youtube.com/watch?v=0SRvmcsRu2w>

- Instalati python, configurati vs code pentru el:

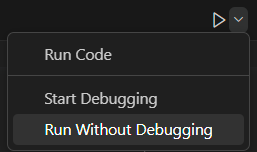
<https://www.youtube.com/watch?v=cUAK4x_7thA&pp=ygUjaG93IHRvIGNvbmZpZ3VyZSB2cyBjb2RlIGZvciBweXRob24%3D>

1. Lansati aplicatia:

\* Deschideti proiectul Flutter in VS Code :

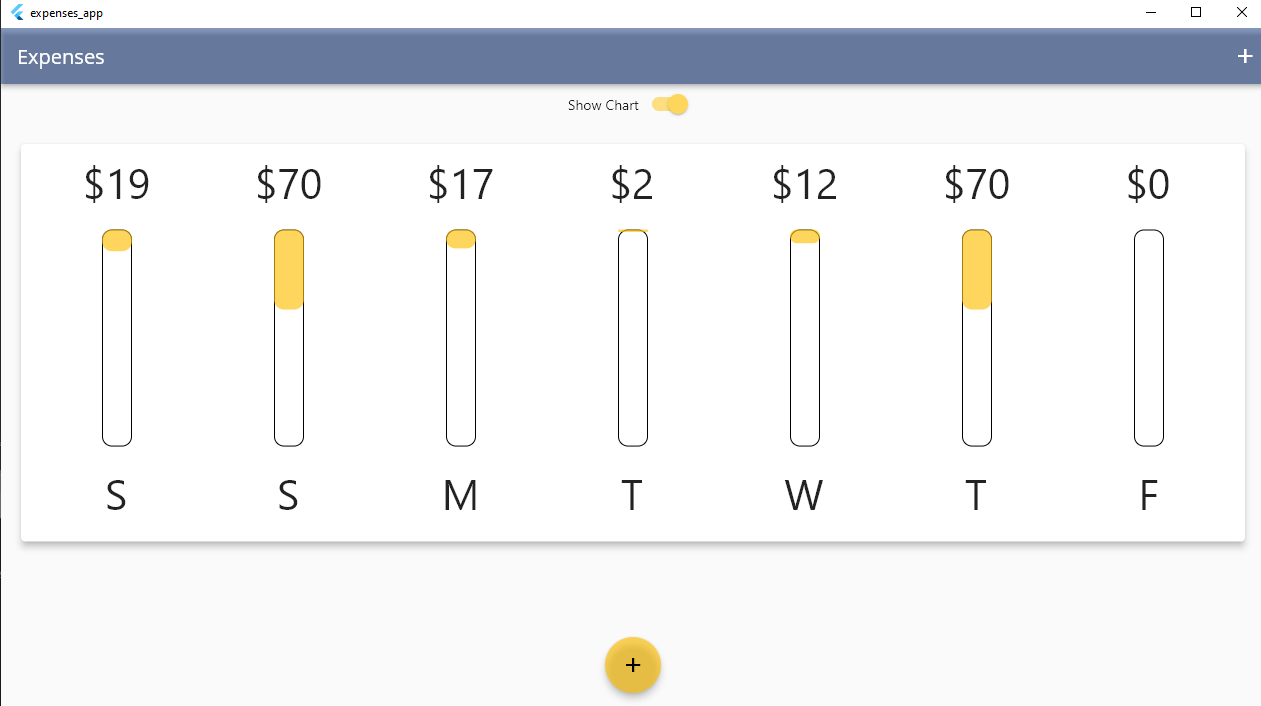


\* Lansati aplicatia fara a face debug :

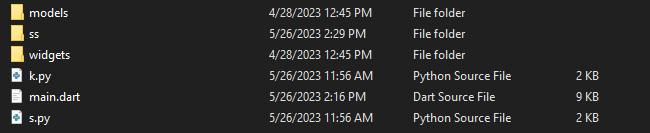


2. Vizualizati rezultatul :

\* Aplicatia este afisata pe ecran:



\*\* Noua stare a fisierelor :



\*\*\* Datele capturate :

