Digital Forensics

Cujbă Mihai-Cătălin

- ## Introducere

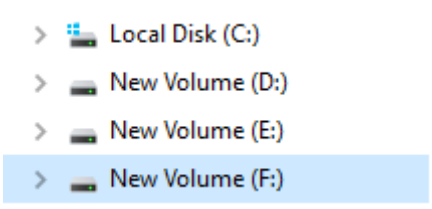
  Forensics reprezintă ramura de investigație a securității cibernetice. Aceasta se împarte în mai multe categorii, una dintre acestea fiind Digital Forensics.

  Această categorie reprezintă analiza datelor de pe echipamente sau componente fizice, precum HDD-uri sau stick-uri USB, analiza fișierelor sistemelor de operare sau a unor secțiuni de memorie.

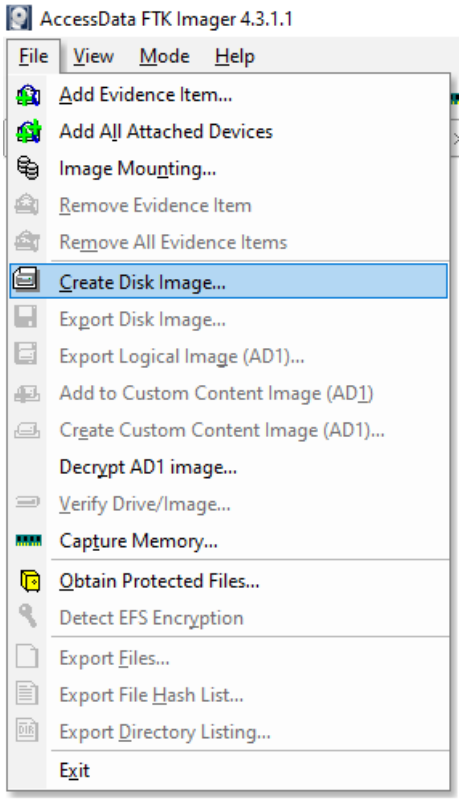
  Pentru a analiza componente fizice, trebuie să le facem o imagine pe care o putem folosi ulterior. În exemplul din laborator vom realiza o imagine a unui stick USB. De-asemenea, operațiunea este similară pentru orice partiție.

- ## Realizarea unei imagini pentru o partiție și analizarea acesteia

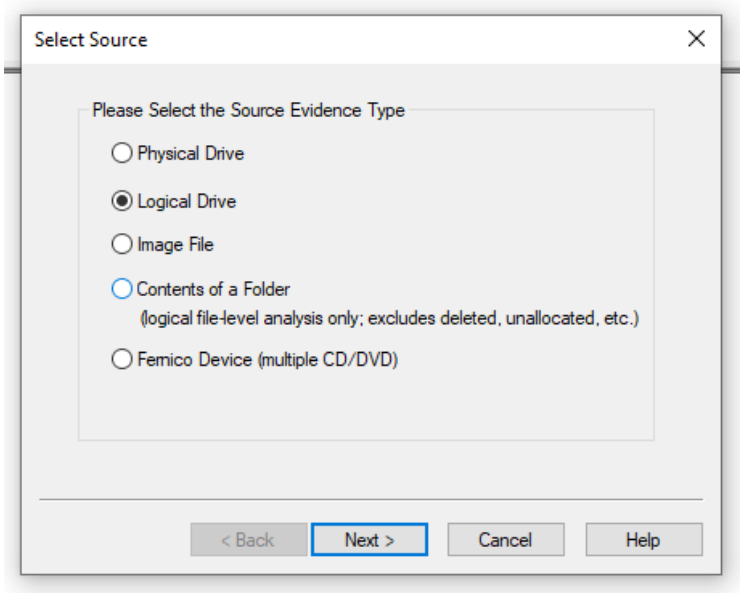
  Pasul 1: Descoperirea partiției căreia îi realizăm imaginea

  Pasul 2: Începem creearea imaginii: File -> Create Disk Image

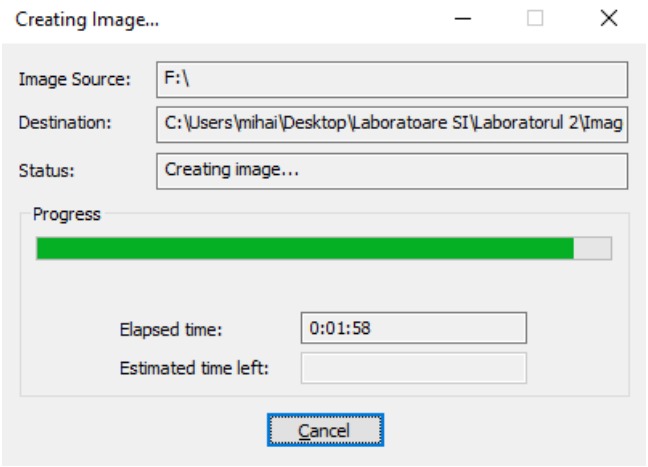
Pasul 3: Selectăm tipul sursei pe care o avem. În cazul nostru este vorba de o partiție, așa că alegem "Logical Drive". Daca voiam sa facem la HDD sau la tot stick-ul, selectam "Physical Drive".



Pasul 4: Selectăm partiția

Pasul 5: Add -> Selectăm tipul imaginii -> Alegem numele -> Finish

Pasul 6: Așteptăm

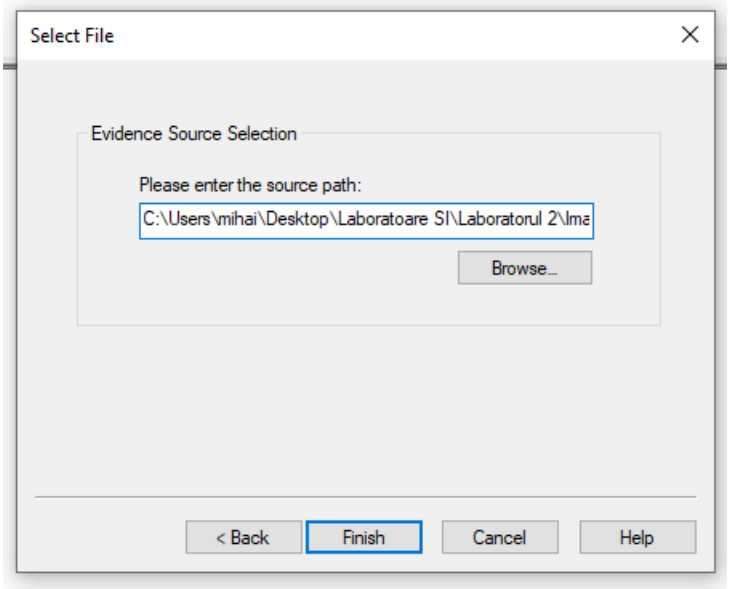


- Odată ce avem imaginea realizată, putem folosi FTKImager sau Autopsy pentru a vedea ce se află pe ele
  - FTKImager:

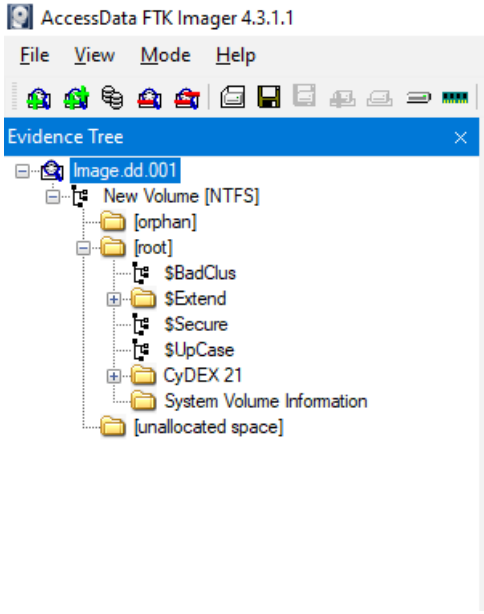
    Pasul 1: File -> Add Evidence Item

    Pasul 2: De data aceasta selectăm "Image File"

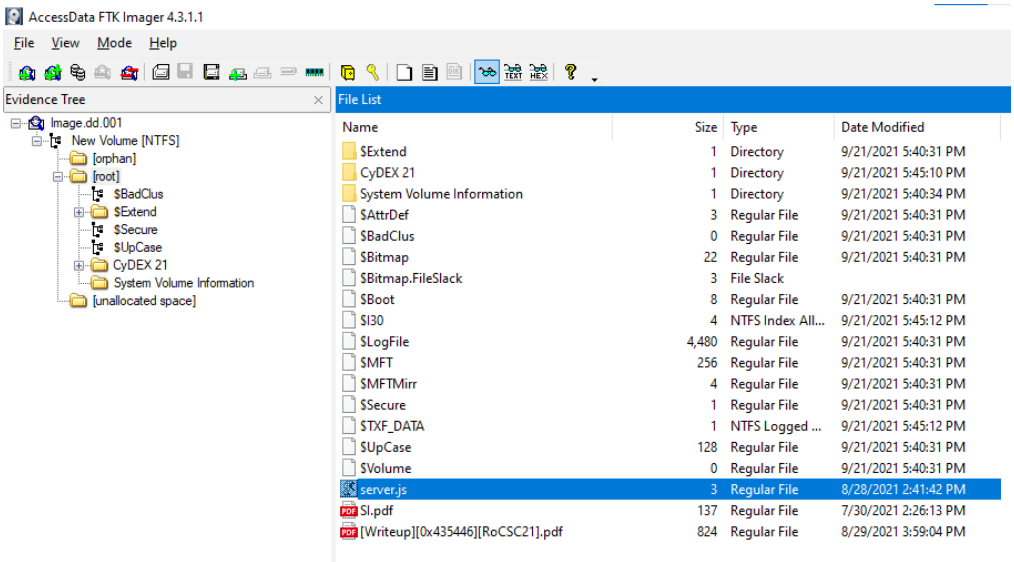
    Pasul 3: Selectăm calea către imagine



Pasul 4: Vizualizăm ierarhia de fișiere existente pe imaginea stick-ului USB



Pasul 5: Vizualizăm fișierele din imagine

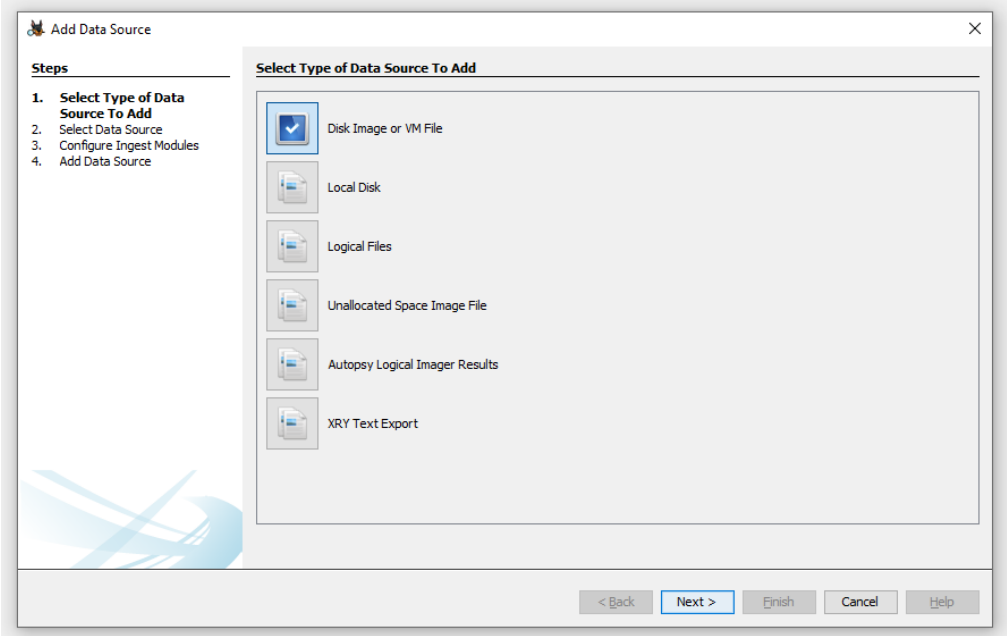


Pasul 6: De aici putem realiza o analiză asupra conținutului, putem căuta fișiere existente pe stick la momentul realizării imaginii, putem recupera fișiere șterse recent sau putem exporta întregul conținut pe mediul nostru de stocare

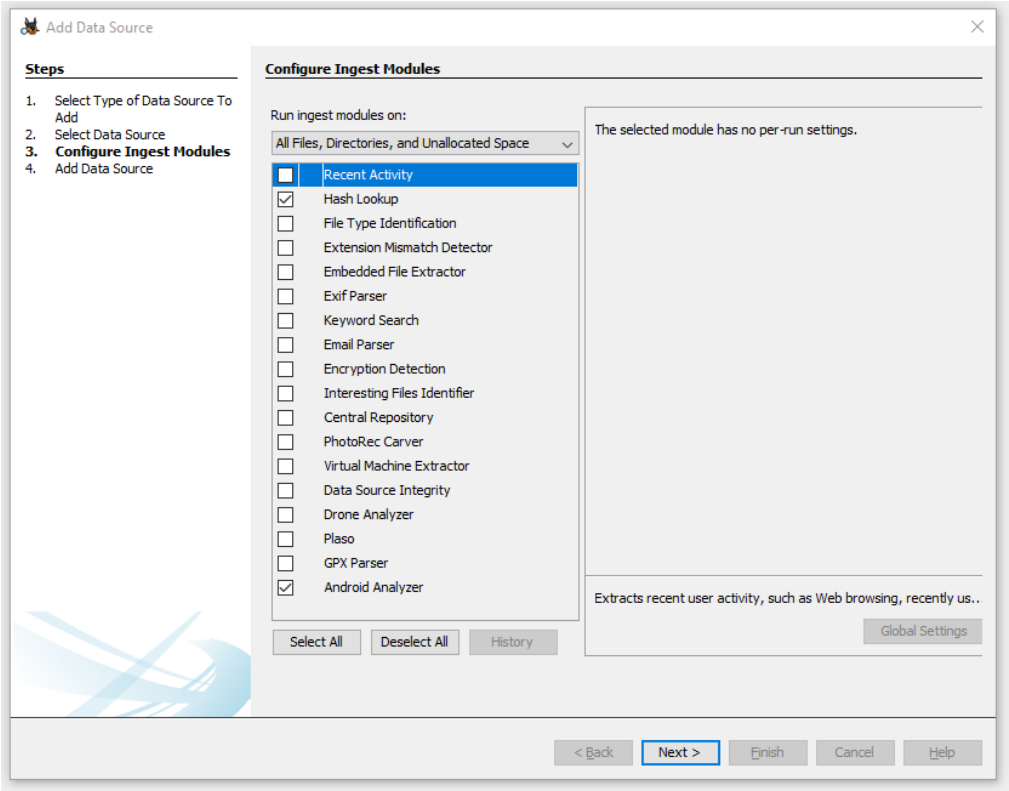
- Autopsy:

Pasul 1: New Case -> Next: Denumire caz -> Finish

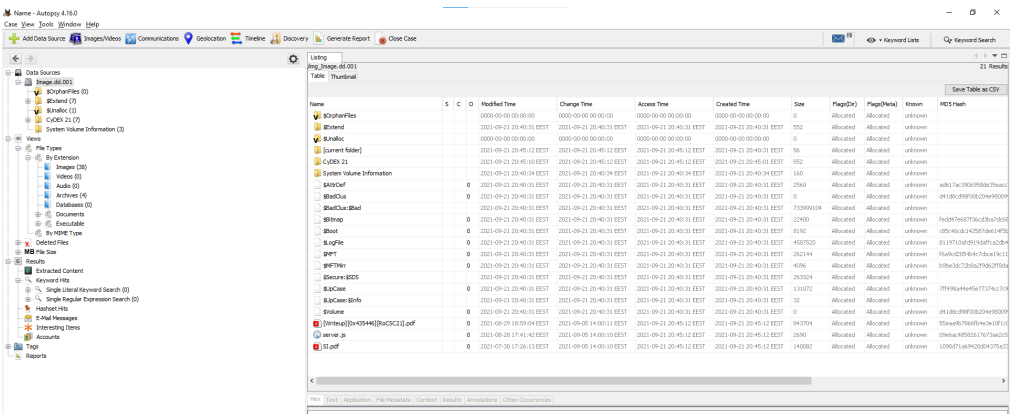
Pasul 2: Selectăm tipul sursei pe care o analizăm, în cazul nostru, Disk Image -> Selectăm sursa



Pasul 3: Selectăm modulele pe care vrem să le aplice programul în momentul analizării imaginii. În exemplul nostru am ales doar Hash Lookup și Android Analyser. -> Finish



Pasul 4: Analiza imaginii



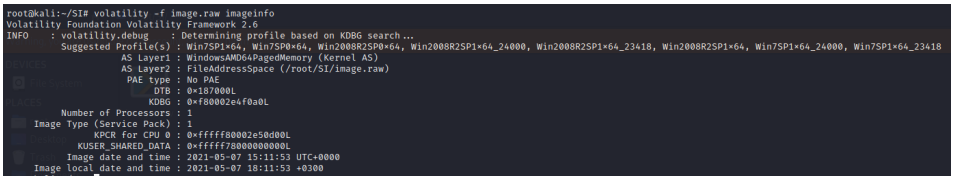
Autopsy aduce îmbunătățiri față de FTKImager prin modulele de căutare, modalitățile de filtrare a conținutului și rapoarte.

- # Analiza imaginii unei memorii volatile (RAM)

  - ## Volatility

    Volatility reprezintă unul dintre cele mai puternice tool-uri de analiză a memoriei volatile. Acesta permite verificarea artefactelor existente în memorie la momentul realizării imaginii.

    - Analiza unei imagini:
      - Pasul 1: Pentru început avem nevoie să cunoaștem sistemul de operare care rula pe sistemul respectiv
        - Comanda folosită: volatility -f image.raw imageinfo -> Win7SP1x64



- Pasul 2: După descoperirea profilului putem analiza fișierul. Vom începe prin listarea tuturor proceselor active.
    - Comanda folosită: volatility -f image.raw --profile Win7SP1x64 pslist

```
root@kali:~/SI# volatility -f image.raw --profile Win7SP1x64 pslist
Volatility Foundation Volatility Framework 2.6
Offset(V)          Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit
------------------ -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ ------------------------------
0xfffffa8010a649e0 System                    4      0     83      497 ------      0 2021-05-07 14:58:32 UTC+0000
0xfffffa80119be650 smss.exe                264      4      2       29 ------      0 2021-05-07 14:58:32 UTC+0000
0xfffffa8012038060 csrss.exe               336    328      9      383      0      0 2021-05-07 14:58:32 UTC+0000
0xfffffa8015065060 wininit.exe             384    328      3       74      0      0 2021-05-07 14:58:33 UTC+0000
0xfffffa8011ebc620 csrss.exe               392    376      9      223      1      0 2021-05-07 14:58:33 UTC+0000
0xfffffa8012721060 winlogon.exe            432    376      5      115      1      0 2021-05-07 14:58:33 UTC+0000
0xfffffa8013132b30 services.exe            476    384      9      199      0      0 2021-05-07 14:58:33 UTC+0000
0xfffffa80127523d0 lsass.exe               484    384      7      573      0      0 2021-05-07 14:58:33 UTC+0000
0xfffffa8012696b30 lsm.exe                 492    384     10      146      0      0 2021-05-07 14:58:33 UTC+0000
0xfffffa801281d7e0 svchost.exe             596    476     10      346      0      0 2021-05-07 14:58:33 UTC+0000
0xfffffa801285fb30 svchost.exe             664    476      8      257      0      0 2021-05-07 14:58:33 UTC+0000
0xfffffa80128754a0 svchost.exe             716    476     20      472      0      0 2021-05-07 14:58:33 UTC+0000
0xfffffa8012938b30 svchost.exe             824    476     19      435      0      0 2021-05-07 14:58:33 UTC+0000
0xfffffa8012980b30 svchost.exe             876    476     12      265      0      0 2021-05-07 14:58:33 UTC+0000
0xfffffa801299db30 svchost.exe             920    476     32      882      0      0 2021-05-07 14:58:33 UTC+0000
0xfffffa8012a60b30 audiodg.exe             968    716      6      129      0      0 2021-05-07 14:58:33 UTC+0000
0xfffffa8012a7eb30 svchost.exe            1004    476      6      107      0      0 2021-05-07 14:58:33 UTC+0000
0xfffffa8012b54b30 svchost.exe             536    476     14      368      0      0 2021-05-07 14:58:34 UTC+0000
0xfffffa8012c0d4a0 dwm.exe                1124    824      3       72      1      0 2021-05-07 14:58:34 UTC+0000
0xfffffa8012c68380 explorer.exe           1136   1116     44     1092      1      0 2021-05-07 14:58:34 UTC+0000
0xfffffa8012c8b060 spoolsv.exe            1180    476     12      262      0      0 2021-05-07 14:58:34 UTC+0000
0xfffffa8012cc9910 taskhost.exe           1244    476     10      188      1      0 2021-05-07 14:58:34 UTC+0000
0xfffffa8012ce5710 svchost.exe            1264    476     18      315      0      0 2021-05-07 14:58:34 UTC+0000
0xfffffa8012e574c0 svchost.exe            1092    476      6       93      0      0 2021-05-07 14:58:37 UTC+0000
0xfffffa8012c1c750 GoogleCrashHan         1532   2032      4       81      0      1 2021-05-07 14:58:39 UTC+0000
0xfffffa8012e42b30 GoogleCrashHan         1428   2032      4       74      0      0 2021-05-07 14:58:39 UTC+0000
0xfffffa8012d9eb30 SearchIndexer.         1816    476     11      620      0      0 2021-05-07 14:58:40 UTC+0000
0xfffffa8012c53360 chrome.exe             1120   1136      0 ------      1      0 2021-05-07 14:59:40 UTC+0000 2021-05-07 15:11:06 UTC+0000
0xfffffa8012ebdb30 mscorsvw.exe           1964    476      7       81      0      1 2021-05-07 15:00:41 UTC+0000
0xfffffa8010da8ae0 svchost.exe            2436    476      8      109      0      0 2021-05-07 15:00:43 UTC+0000
0xfffffa8010d89060 mscorsvw.exe           1884    476      7       76      0      0 2021-05-07 15:00:50 UTC+0000
0xfffffa8010cd8b30 sppsvc.exe             1088    476      4      140      0      0 2021-05-07 15:01:19 UTC+0000
0xfffffa8010f557c0 svchost.exe            2388    476     14      334      0      0 2021-05-07 15:01:26 UTC+0000
0xfffffa8012a8a060 taskeng.exe            2788    920      4       78      0      0 2021-05-07 15:08:34 UTC+0000
0xfffffa8010faab30 notepad.exe            2872   1136      1       61      1      0 2021-05-07 15:11:18 UTC+0000
0xfffffa8010e9cb30 SearchProtocol         2508   1816      8      278      0      0 2021-05-07 15:11:20 UTC+0000
0xfffffa80136b9060 SearchFilterHo         2384   1816      5       99      0      0 2021-05-07 15:11:20 UTC+0000
0xfffffa8010eef060 KeePass.exe            2192   1136      8      340      1      0 2021-05-07 15:11:24 UTC+0000
0xfffffa80128a3550 dllhost.exe            2044    596      6       83      1      0 2021-05-07 15:11:51 UTC+0000
0xfffffa8012f29060 dllhost.exe            2548    596      6       80      0      0 2021-05-07 15:11:51 UTC+0000
0xfffffa8010dfd060 DumpIt.exe             2252   1136      2       45      1      1 2021-05-07 15:11:51 UTC+0000
0xfffffa8010c2e060 conhost.exe            1488    392      2       50      1      0 2021-05-07 15:11:51 UTC+0000
```

    - Putem observa, pe lângă toate serviciile Windows, un proces ce aparține executabilului KeePass.exe. Acest utilitar este folosit la stocarea parolelor, astfel având posibilitatea să aflăm parolele utilizatorului care a folosit sistemul
- Pasul 3: Recuperarea parolelor din KeePass se poate realiza prin extragerea zonei de memorie alocate procesului și verificarea acesteia, în cazul în care parola de acces la fișierul cu parole a fost folosită înainte de realizarea imaginii
    - Comanda folosită: volatility -f image.raw --profile Win7SP1x64 memdump -p 2192 -D ./

```
root@kali:~/SI# volatility -f image.raw --profile Win7SP1x64 memdump -p 2192 -D ./
Volatility Foundation Volatility Framework 2.6
************************************************************************
Writing KeePass.exe [  2192] to 2192.dmp
```

- Pasul 4: Extragerea stringurilor din dump-ul de memorie și căutarea tag-ului XML <KeePass>
    - Comanda folosită: strings 2192.dmp | grep <KeePass>
    - Nu am primit niciun răspuns din partea acestei comenzi, deci parola nu a fost încă folosită. Am putea aborda două situații: Să încercăm sa facem bruteforce pe fișierul de parole, lucru care ne-ar costa mult timp și ar avea o posibilitate mare de eșec sau să căutăm parola în dump-ul de memorie.
- Pasul 5: Verificarea conținutului din notepad-urile deschise:
    - Comanda folosită: volatility -f image.raw --profile Win7SP1x64 notepad

```
root@kali:~/SI# volatility -f image.raw --profile Win7SP1x64 notepad
Volatility Foundation Volatility Framework 2.6
ERROR   : volatility.debug    : This command does not support the profile Win7SP1x64
```

    - Din ce se poate observa, comanda nu este valabilă pentru profilul imaginii noastre
- Pasul 6: Verificare clipboard-ului:
    - Comanda folosită: volatility -f image.raw --profile Win7SP1x64 clipboard

```
root@kali:~/SI# volatility -f image.raw --profile Win7SP1x64 clipboard
Volatility Foundation Volatility Framework 2.6
Session    WindowStation Format                  Handle Object             Data
---------- ------------- ------------ ---------------- ------------------ ----------------------------------------------------------------
         1 WinSta0       CF_UNICODETEXT            0x900cd 0xfffff900c2047370 mqDb*N6*(mAk3W)=
         1 WinSta0       CF_TEXT                0x7400000000 ----------
         1 WinSta0       CF_LOCALE                 0x110091 0xfffff900c1f75550
         1 WinSta0       0x0L                           0x0 ----------
```

- Din câte putm observa, exista un string copiat, "mqDb*N6*(mAk3W)=", ce poate fi folosit ca parola pentru fișier.

- Pasul 7.1: Căutarea fișierului cu parole
  - Comanda folosită: volatility -f image.raw --profile Win7SP1x64 filescan | egrep "*.kdbx"

```
root@kali:~/SI# volatility -f image.raw --profile Win7SP1x64 filescan | egrep "*.kdbx"
Volatility Foundation Volatility Framework 2.6
0x0000000052b0eaf0     16      0 R--r-- \Device\HarddiskVolume1\Users\Unbreakable\Desktop\Database.kdbx
0x00000000054212dc0      2      0 R--rwd \Device\HarddiskVolume1\Users\Unbreakable\Desktop\Database.kdbx
0x00000000543a0ae0      2      0 RW-rw- \Device\HarddiskVolume1\Users\Unbreakable\AppData\Roaming\Microsoft\Windows\Recent\Database.kdbx.lnk
```

  - De aici se poate vedea exact locația fișierului. Fiind un caz particular am putut căuta exact ceea ce aveam nevoie, dar sunt cazuri în care nu știm exact ce trebuie să găsim, așa că acest pas va avea încă o comandă prin care ne puteam da seama de fișierul dorit.

- Pasul 7.2: Verificarea comenzilor date folosind cmd.exe:
  - Comanda folosită: volatility -f image.raw --profile Win7SP1x64 cmdline

```
KeePass.exe pid:   2192
Command line : "C:\Program Files\KeePass Password Safe 2\KeePass.exe" "C:\Users\Unbreakable\Desktop\Database.kdbx"
```

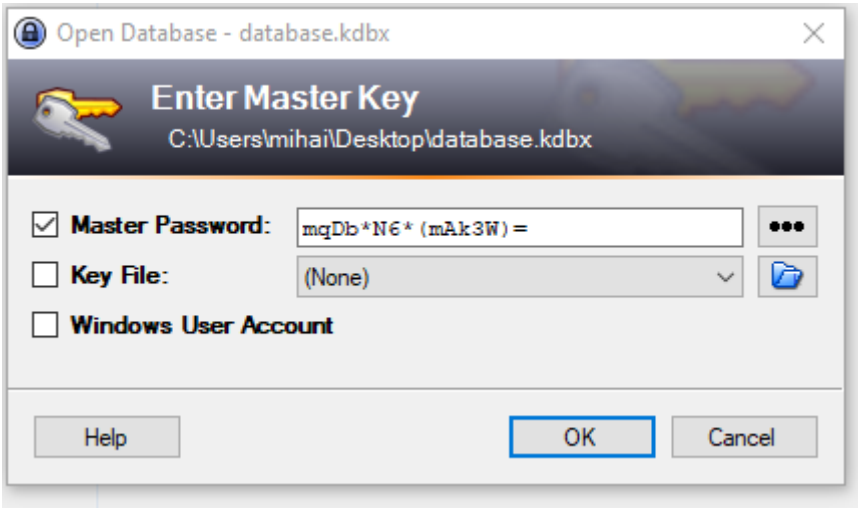
  - Putem observa că utilitarul KeePass a fost folosit din linie de comandă, iar aici putem vedea și locația fișierului

- Pasul 8: Extragerea fișierului din memorie
  - Comanda folosită: volatility -f image.raw --profile Win7SP1x64 dumpfiles -n --dump-dir=./ -Q 0x0000000052b0eaf0

```
root@kali:~/SI# volatility -f image.raw --profile Win7SP1x64 filescan | egrep "*.kdbx"
Volatility Foundation Volatility Framework 2.6
0x0000000052b0eaf0     16      0 R--r-- \Device\HarddiskVolume1\Users\Unbreakable\Desktop\Database.kdbx
0x00000000054212dc0      2      0 R--rwd \Device\HarddiskVolume1\Users\Unbreakable\Desktop\Database.kdbx
0x00000000543a0ae0      2      0 RW-rw- \Device\HarddiskVolume1\Users\Unbreakable\AppData\Roaming\Microsoft\Windows\Recent\Database.kdbx.lnk
root@kali:~/SI# volatility -f image.raw --profile Win7SP1x64 dumpfiles -n --dump-dir=./ -Q 0x0000000052b0eaf0
Volatility Foundation Volatility Framework 2.6
DataSectionObject 0x52b0eaf0   None   \Device\HarddiskVolume1\Users\Unbreakable\Desktop\Database.kdbx
```

  - Acum că avem atât fișierul cu parole cât și parola master cu care să accesăm conținutul, rămâne doar sa instalam KeePass și sa recuperam parolele.

- Pasul 9:
  - Introducem parola master



- # Steganografie
  - ## Fișiere polyglot

    Acestea sunt compuse din mai multe fișiere valide, forma lor fiind diferențiată de către formatul pe care îl au, în cadrul sistemelor de operare Windows sau de programele executabile ce le folosesc, în cadrul sistemelor Linux.

    Un fișier este diferențiat printr-un număr magic aflat la începutul conținutului, număr magic pe care sistemul de operare îl folosește pentru a-și da seama ce tip de fișier avem.

    Folosindu-ne de această informație, putem așeza două fișiere valide unul după celalalt în cadrul unuia nou pentru a încerca, spre exemplu, să facem bypass la niște filtre.

    Un tool special realizat pentru aceste operațiuni este Mitra (https://github.com/corkami/mitra).

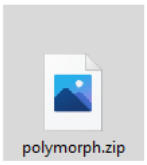
    - Mitra:
      - Comanda folosită: python3 mitra.py castravete.jpg ceva.zip

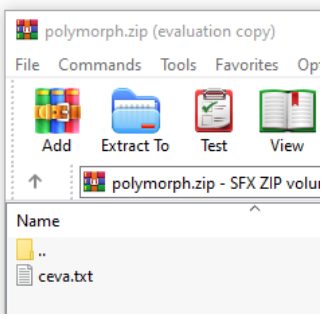
```
root@kali:~/SI/Mitra/mitra# python3 mitra.py  castravete.jpg ceva.zip
castravete.jpg
File 1: JFIF / JPEG File Interchange Format
ceva.zip
File 2: Zip

Stack: concatenation of File1 (type JPG) and File2 (type Zip)
Parasite: hosting of File2 (type Zip) in File1 (type JPG)
```

      - În imaginea de mai sus se poate observa output-ului utilitarului Mitra după ce am concatenat o imagine (castravete.jpg) cu o arhivă (ceva.zip)
      - Fișierul cu extensia .jpg:

      - Fișierul cu extensia .zip:





    Modalități de descoperire a fișierelor Polyglot:

    - Binwalk:

- Utilitarul binwalk are rolul de a verifica dacă în conținutul fișierelor există mai multe numere magice și de a afișa ceea ce găsește
- Comanda folosită: binwalk polyglot.jpg

```
root@kali:~/SI/Mitra/mitra# binwalk polyglot.jpg

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01
4150          0x1036          Zip archive data, at least v2.0 to extract, compressed size: 12, uncompressed size: 10, name: ceva.txt
4290          0x10C2          End of Zip archive, footer length: 22
```

- De-asemenea se pot extrage fișierele, folosind comanda: binwalk -e polyglot.jpg

- Foremost:
  - Asemenea binwalk, foremost scanează conținutul fișierelor și extrage tot ceea ce găsește
  - Comanda folosită: foremost polyglot.jpg

```
root@kali:~/SI/Mitra/mitra# foremost polyglot.jpg
Processing: polyglot.jpg
|foundat=ceva.txt
                N♦)-I,J♦♦
*|
root@kali:~/SI/Mitra/mitra# ls output/
audit.txt  jpg  zip
root@kali:~/SI/Mitra/mitra# ls output/*/
output/jpg/:
00000000.jpg

output/zip/:
00000008.zip
```

LSB (Least Significant Bit)

- Există foarte multe modalități custom pentru ascunderea informațiilor în LSB, dar acum vom vorbi despre utilitarul steghide (http://steghide.sourceforge.net/).
- Steghide:
  - Este folosit pentru ascunderea mesajelor atât în poze (jpg), cât și în fișiere audio
  - Comanda folosită: steghide embed -cf castravete.jpg -ef secret.txt

```
root@kali:~/SI/Mitra# steghide embed -cf castravete.jpg -ef secret.txt
Enter passphrase:
Re-Enter passphrase:
embedding "secret.txt" in "castravete.jpg"... done
```

  - În imaginea de mai sus se vede cum am ascuns fișierul "secret.txt" în imagine
  - Pentru a extrage secretul, trebuie sa folosim următoarea comandă și să băgăm parola pe care am ales-o la primul pas: steghide extract -sf castravete.jpg

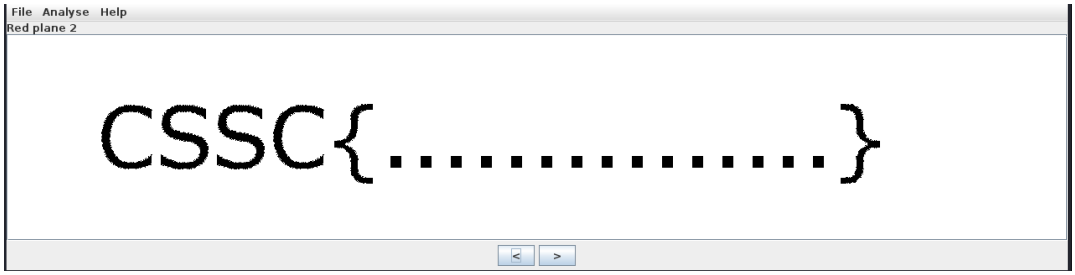
```
root@kali:~/SI/Mitra# steghide extract -sf castravete.jpg
Enter passphrase:
the file "secret.txt" does already exist. overwrite ? (y/n) y
wrote extracted data to "secret.txt".
```

Modificarea layerelor imaginilor

- O altă metodă de ascundere a mesajelor este inserarea unei imagini în cadrul layerelor inferioare din altă imagine.
- Pentru detectarea acestor imagini putem folosi utilitarul stegsolve (https://github.com/zardus/ctf-tools/tree/master/stegsolve) sau putem folosi un utilitar precum LSBViewer (https://github.com/0x435446/LSB-viewer/blob/master/LSB-viewer.py).
- Pentru acest exemplu vom folosi Stegsolve, un utilitar scris în java

- Imaginea folosită:

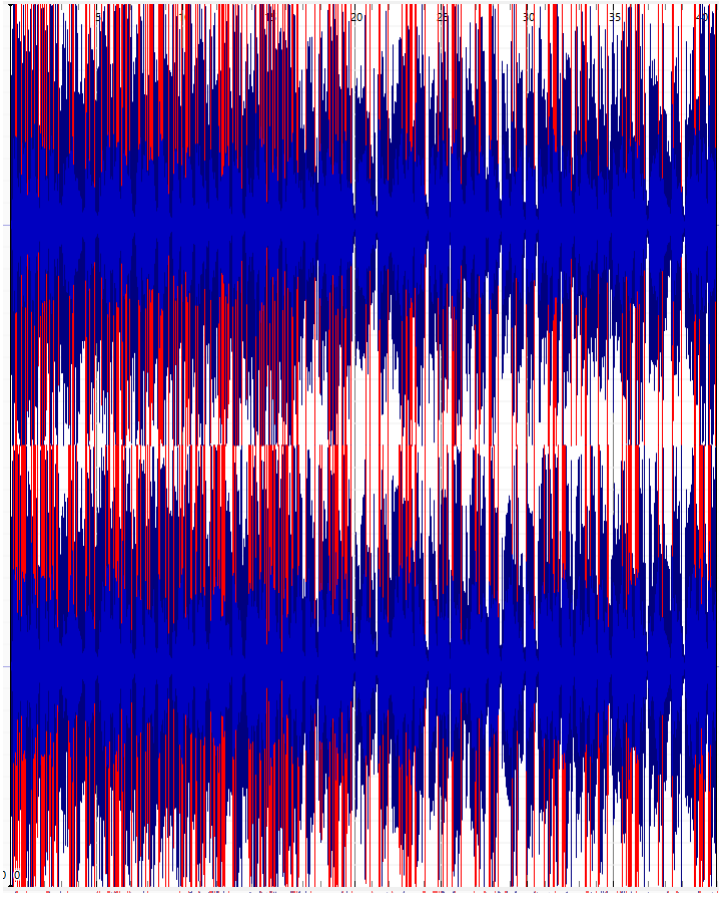
- Se poate observa că până la layer-ul 2, imaginea rămâne intactă



- Layer-ul 1:

Audio steganography

- Pentru audio vom folosi programul Sonic Visualizer, cu care vom deschide un fișier ce are ascuns un mesaj în spectogramă.
- Sunetul în Waveform:



- Spectograma sunetului:


HTB{str4wberry_milkshak3}