



Mission Name

DumperClearance

History Background

Ethan and Claire hack their clearance status to grant them a global departure with their own recon car.

Technical High-Level Overview

A memory dump from a computer connected to a system that manages authorizations is provided to the player. This memory dump contains a password that allow to modify Claire and Ethan permissions to fly. The password is located on the Google Chrome database, but in this scenario, all artifacts must be extracted from the memory.

Short Description

Your goal is to analyse a memory dump from a computer connected to Skytech Flight Authorization System and get a password related to an email. Keep in mind, password could be in anywhere.

Mission Description

Your goal is to analyse a memory dump from a computer connected to Skytech Flight Authorization System and get a password. This memory dump contains a password to an email that allow to modify Claire and Ethan permissions to fly. You should analyse memory deeply, in order to get the unencrypted password.

Location

- SYLVARCON | PORT 2 | INTERNATIONAL TRANSIT ZONE

Tools

- Volatility 2 -> follow these steps, all, including python3:
<https://netsidetech.ca/2021/02/07/how-to-install-volatility-in-kali/>
- Download PIP from wget <https://bootstrap.pypa.io/pip/2.7/get-pip.py>
- <https://github.com/tijldeneut/dpapilab-ng> -> remove **construct==2.5.5-reupload package.**
- Wxhexeditor: <https://howtostall.co/es/wxhexeditor>
- SQLite viewer tool to read the database extracted.
- Pillow Python Module: pip2 install Pillow

Questions

Which is the PID that has encrypted_key key?

- 3464

To provide you the memory evidence, Which program was used?

- Dumpit

Which IP address of the memory provided?

- 192.168.211.163

Hints

1. Use Volatility2 and get all processes running.
2. Use Volatility2, plugin editbox, and extract all the necessary files from the memory.
3. Use any script which allow you decrypt DPAPI blobs.

Write Up

It's mandatory to use Volatility 2.

```
$ vol.py --info
Volatility Foundation Volatility Framework 2.6.1

Profiles
_____
VistaSP0x64      - A Profile for Windows Vista SP0 x64
VistaSP0x86      - A Profile for Windows Vista SP0 x86
VistaSP1x64      - A Profile for Windows Vista SP1 x64
VistaSP1x86      - A Profile for Windows Vista SP1 x86
```

Figure 1

Why volatility2? Editbox plugin it's mandatory to extract a key from Notepad. Using Volatility3 could increase the level of the challenge. First of all, identify Profile:

```
vol.py -f /mnt/hgfs/Maquina/C2-M1/evidence/memory imageinfo
```

```
(kali㉿kali)-[~/mnt/hgfs]
$ vol.py -f /mnt/hgfs/Maquina/C2-M1/evidence/memory imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug : Determining profile based on KDBG search ...
Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64, Win2008R2SP1x64_23418, Win2008R2SP1x64, Win7SP1x64_23418
                      AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
                      AS Layer2 : FileAddressSpace (/mnt/hgfs/Maquina/C2-M1/evidence/memory)
                      PAE type : No PAE
                      DTB : 0x187000L
                      KDBG : 0xf80002a4f0a0L
Number of Processors : 1
Image Type (Service Pack) : 1
                      KPCR for CPU 0 : 0xfffff80002a50d00L
                      KUSER_SHARED_DATA : 0xfffff78000000000L
Image date and time : 2021-07-17 16:42:44 UTC+0000
Image local date and time : 2021-07-17 18:42:44 +0200
```

Figure 2

You must install Pillow module for Volatility:

```
pip2 install Pillow
```

```
vol.py -f /mnt/hgfs/Maquina/C2-M1/evidence/memory --profile=Win7SP1x64_23418 screenshot --dump-dir
/mnt/hgfs/Maquina/C2-M1/evidence/screenshots
```

Image captured from the memory:

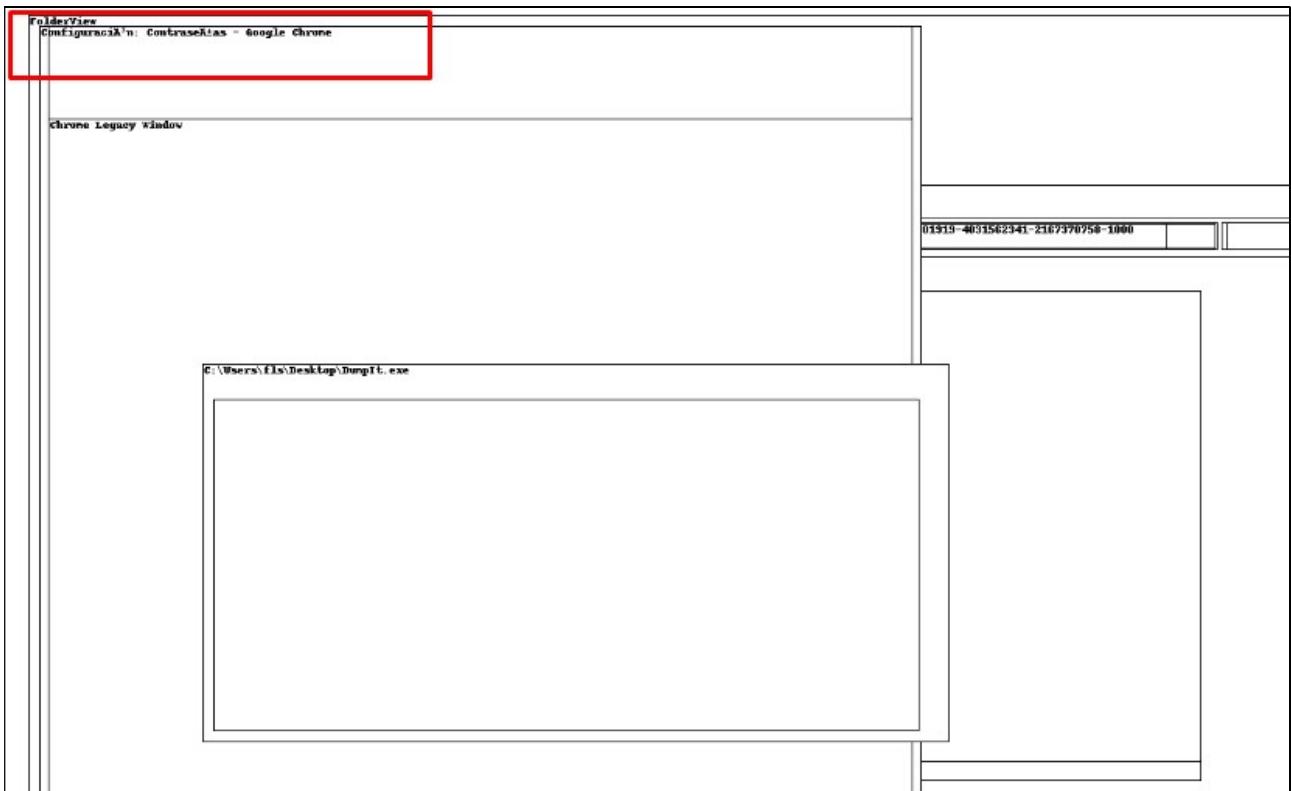


Figure 3

Then, to get windows opened on Windows, player should launch the following command:

```
vol.py -f /mnt/hgfs/Maquina/C2-M1/evidence/memory --profile=Win7SP1x64_23418 windows > /home/kali/windows.txt  
cat /home/kali/windows.txt
```

```
$ cat /home/kali/windows.txt | grep Chrome  
Window Handle: #b01c4 at 0xfffff900c062b640, Name: C:\Users\fls\AppData\Local\Google\Chrome\User Data\Local State - Notepad++  
ClassAtom: 0xc1b8, Class: Chrome_WidgetWin_0  
SuperClassAtom: 0xc1b8, SuperClass: Chrome_WidgetWin_0  
ClassAtom: 0xc1c6, Class: Chrome_SystemMessageWindow  
SuperClassAtom: 0xc1c6, SuperClass: Chrome_SystemMessageWindow  
ClassAtom: 0xc1c7, Class: Chrome_StatusTrayWindow  
SuperClassAtom: 0xc1c7, SuperClass: Chrome_StatusTrayWindow  
ClassAtom: 0xc1b8, Class: Chrome_WidgetWin_0  
SuperClassAtom: 0xc1b8, SuperClass: Chrome_WidgetWin_0  
Window Handle: #c03b6 at 0xfffff900c0691230, Name: Configuración: Contraseñas - Google Chrome  
ClassAtom: 0xc1e5, Class: Chrome_WidgetWin_1  
SuperClassAtom: 0xc1e5, SuperClass: Chrome_WidgetWin_1  
Window Handle: #a0464 at 0xfffff900c0691290, Name: Chrome Legacy Window  
ClassAtom: 0xc1e2, Class: Chrome_RenderWidgetHostHWND  
SuperClassAtom: 0xc1e2, SuperClass: Chrome_RenderWidgetHostHWND
```

Figure 4

To get windows opened on Windows, player should launch the following command:

```
vol.py -f /mnt/hgfs/Maquina/C2-M1/evidence/memory --profile=Win7SP1x64_23418 editbox
```

```
undoBuf      : _____
"encrypted_key": "RFBBUEkAAAAA0Iyd3wEV0RGMegDAT8KX6wEAAAD0Gypmun6JTIHLFMJ/whWIAAAAIAAAAABmAAAAAQAAIAAAAAd7Na
Zjz94R0NG8sVMAAAAHfQoncOHWyEVLygtlSbdoAWubEIgy3+uYhdu+PO3yQPu5SYbGihG3qasKPXJtxFyEAAAABZzHmD2PmYg3ag+48XQS
lrrHsc*****
Wnd Context   : 1\WinSta\Default
Process ID    : 1372
ImageFileName  : notepad++.exe
IsWow64       : Yes
atom_class     : 6.0.7601.17514!Edit
value-of WndExtra : 0x2ad5268
nChars        : 0
selStart       : 0
selEnd         : 0
isPwdControl  : False
```

Figure 5

If player performs a search on Google related to encrypted_key, could identify:

- https://chromium.googlesource.com/chromium/src/+/HEAD/docs/user_data_dir.md

```
RFBBUEkAAAAA0Iyd3wEV0RGMegDAT8KX6wEAAAD0Gypmun6JTIHLFMJ/whWIAAAA
AAIAAAAABmAAAAAQAAIAAAAAd7NaChenx91Eq2EdykPy1wnIU/61MQt1C3cQZpet
IAAAAAA6AAAAAAgAAIAAAAOhSIha1fZ3cSejUfMWQZcGLNYTPPQXRZjz94R0NG8sV
MAAAAHfQoncOHWyEVLygtlSbdoAWubEIgy3+uYhdu+PO3yQPu5SYbGihG3qasKPXJtxFyE
AAAABZzHmD2PmYg3ag+48XQS
lrrHscBfV/4GAFbSGA4/4IU0PU4CSj4vCbc0fcPFXDUtndA
5IZFQqubiW7p/Cxbm8A
```

Multiple projects to decrypt passwords or cookies based on Chrome, but all relies on CryptUnprotectData(), this function only works on Windows live system, not post mortem system.

chrome local state encrypted_key

Aproximadamente 22.600 resultados (0,59 segundos)

Quizás quisiste decir: chrome local state **encryption key**

<https://stackoverflow.com/questions/Traducir esta página>

Chrome 80 how to decode cookies - Stack Overflow

26 feb 2020 — The **encrypted key** starts with the ASCII encoding of DPAPI (i.e. ... path = r'%LocalAppData%\Google\Chrome\User Data\Local State' path ...)

2 respuestas - Mejor respuesta: Since Chrome version 80 and higher, cookies are encrypted u...

Chrome 80 Password File Decryption in Python - Stack Overflow 8 abr 2020

DPAPI fails with CryptographicException when trying to ... 10 mar 2020

Chrome 80 how to decode cookies python To powershell... 26 abr 2020

Chrome ver80+ Login Data decrypting problem - Stack Overflow 6 abr 2021

Más resultados de stackoverflow.com

<https://gist.github.com/GramThanos/Traducir esta página>

Decrypt Chrome Cookies File (Python 3) - Windows · GitHub

Local/Google/Chrome/User Data/Local State", r") as file: encrypted_key = json.loads(file.read())['os_crypt']['encrypted_key']. encrypted_key ...

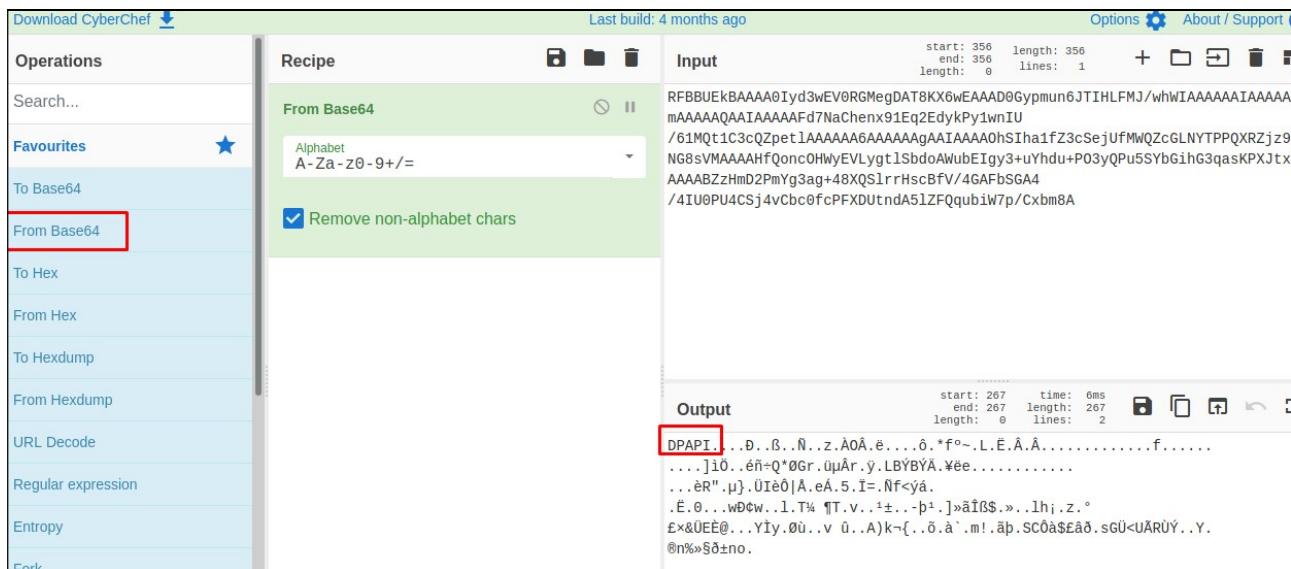
Otras personas también buscan

- decrypt chrome cookies
- read chrome cookies python
- how to read chrome cookies file
- chrome cookie dpapi
- chrome cookie encrypted_value decryptor
- dpapi mimikatz

Figure 6

This function only works on the same machine that was encrypted the password. So, to unencrypt password Chrome, we are going to use: <https://github.com/tijldeneut/dpapilab-ng>

First of all, encrypted_key got it from Volatility , must be deciphered, using CyberChef:

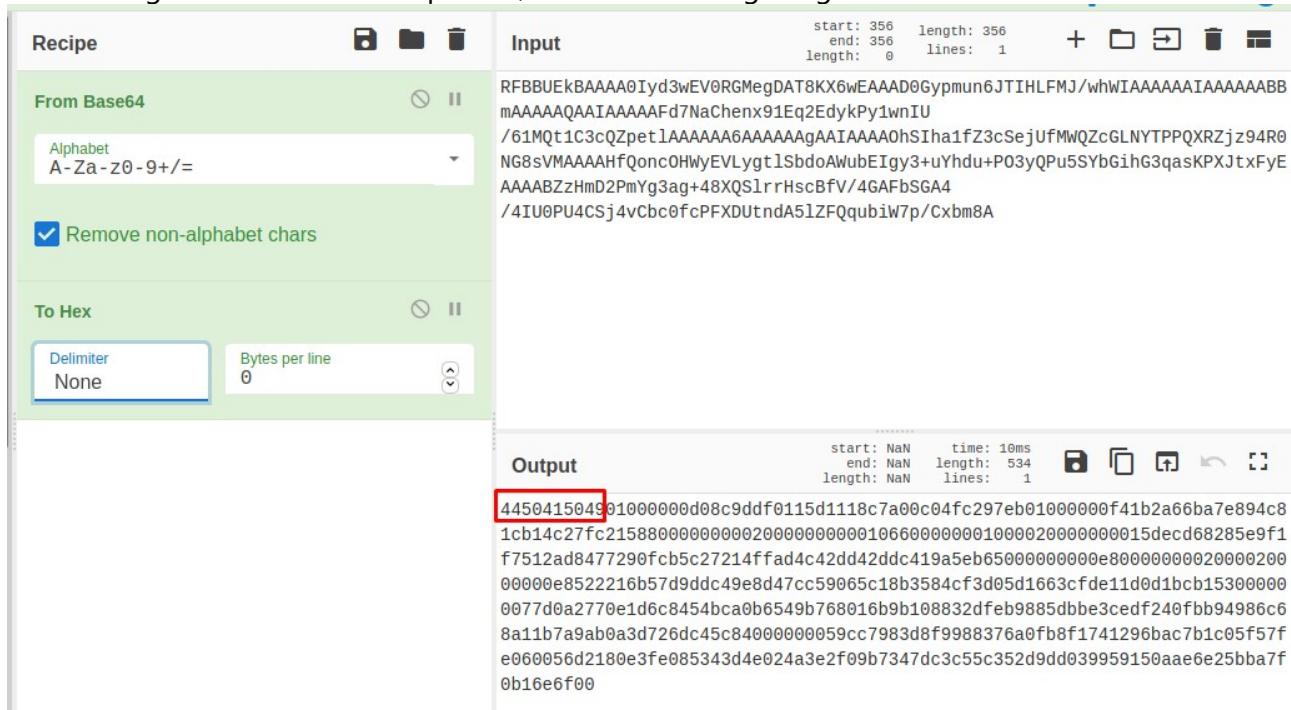


The screenshot shows the CyberChef interface with the following details:

- Operations:** A sidebar with various options like To Base64, From Base64 (highlighted with a red box), To Hex, From Hex, etc.
- Recipe:** Set to "From Base64" with the "Alphabet" dropdown set to "A-Za-z0-9+=". A checkbox for "Remove non-alphabet chars" is checked.
- Input:** A large block of base64 encoded data starting with "RFBBUEkBAAAA0Iyd3wEV0RG...".
- Output:** The decrypted result starting with "DPAPI...".

Figure 7

Above image shows base64 deciphered, and the following image to HEX:

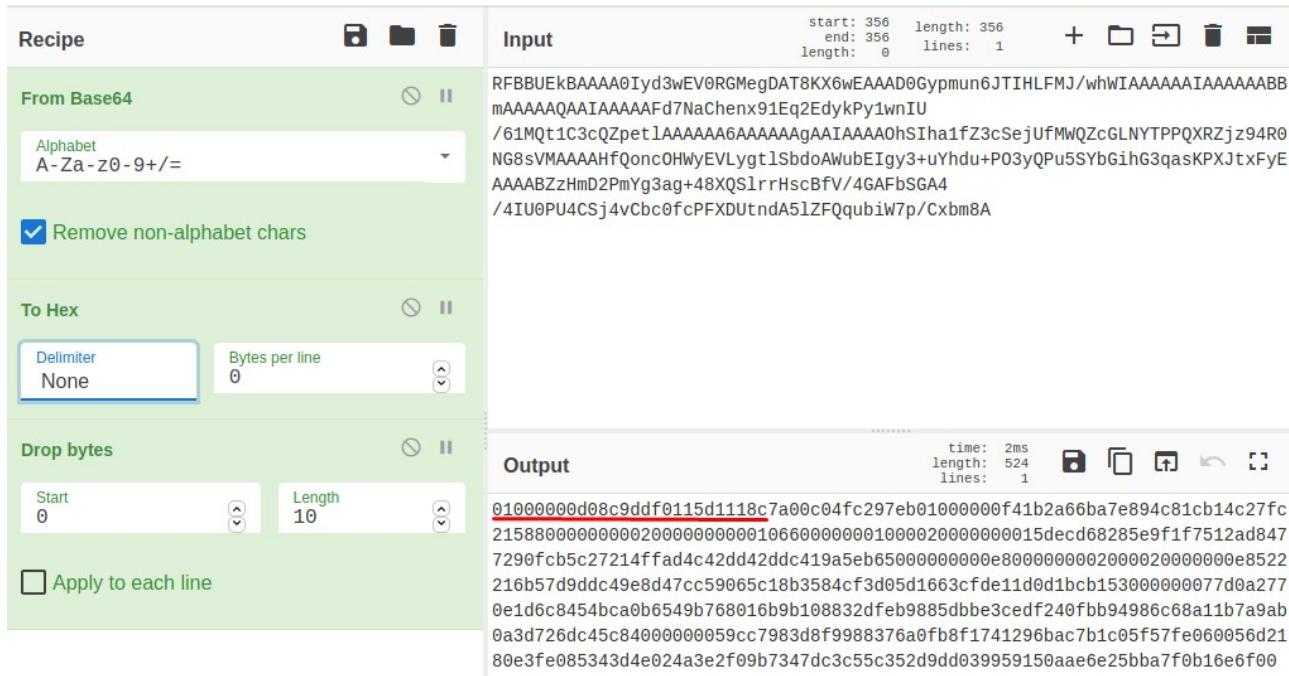


The screenshot shows the CyberChef interface with the following details:

- Recipe:** Set to "From Base64" with the "Alphabet" dropdown set to "A-Za-z0-9+=". A checkbox for "Remove non-alphabet chars" is checked.
- To Hex:** A section with a "Delimiter" dropdown set to "None" and a "Bytes per line" dropdown set to "0".
- Input:** The same base64 encoded data as in Figure 7.
- Output:** The resulting hex dump starting with "445041504901000000d08c9ddf0115d1118c7a00c04fc297eb01000000f41b2a66ba7e894c8".

Figure 8

We have to remove 5 bytes, related to DPAPI ASCII word:



Recipe

From Base64

Alphabet: A-Za-z0-9+=

Remove non-alphabet chars

To Hex

Delimiter: None

Bytes per line: 0

Drop bytes

Start: 0

Length: 10

Apply to each line

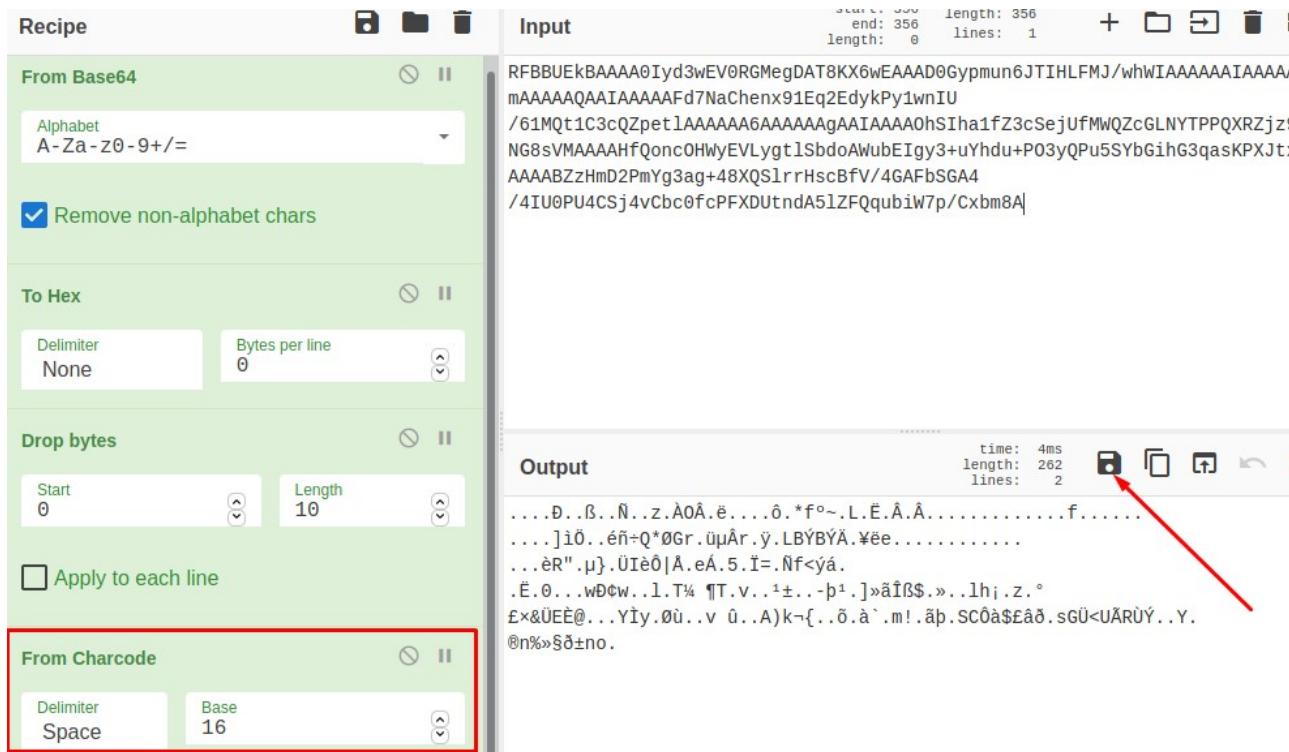
Input

```
RFBBUEkBAAA0Iyd3wEV0RGMeGDAT8KX6wEAAAD0Gypmun6JTIHLFMJ/whwIAAAAAAIAAAAAB
mAAAAAQAAIAAAAAd7NaChenx91Eq2EdykPy1wnIU
/61MQt1C3cQZpet1AAAAA6AAAAAgAIAAA0hSIha1fZ3cSejUfMWQZcGLNYTPPQRZjz94R0
NG8sVMAAAAHfQoncOHwyEVLygtlSbdoAwubEIgy3+uYhdu+P03yQPu5SYbGihG3qasKPXJtxFyE
AAAABZZHmD2PmYg3ag+48XQSlrrHscBfV/4GAFbSGA4
/4IU0PU4CSj4vCbc0fcPFXDUtnA51ZFQqubiW7p/Cxbm8A
```

Output

```
01000000d08c9ddf0115d1118c7a00c04fc297eb01000000f41b2a66ba7e894c81cb14c27fc
2158800000000020000000000106600000001000020000000015decfd68285e9f1f7512ad847
7290fcbb5c27214ffad4c42dd42dc419a5eb65000000000e8000000002000020000000e8522
216b57d9ddc49e8d47cc5906c518b3584cf3d05d1663cfde11d0d1bcb153000000077d0a277
0e1d6c8454bca0b6549b768016b9b108832dfb9885dbbe3cedf240fb94986c68a11b7a9ab
0a3d726dc45c8400000059cc7983d8f9988376a0fb8f1741296bac7b1c05f57fe060056d21
80e3fe085343d4e024a3e2f09b7347dc3c55c352d9dd039959150aae6e25bba7f0b16e6f00
```

Figure 9



Recipe

From Base64

Alphabet: A-Za-z0-9+=

Remove non-alphabet chars

To Hex

Delimiter: None

Bytes per line: 0

Drop bytes

Start: 0

Length: 10

Apply to each line

From Charcode

Delimiter: Space

Base: 16

Input

```
RFBBUEkBAAA0Iyd3wEV0RGMeGDAT8KX6wEAAAD0Gypmun6JTIHLFMJ/whwIAAAAAAIAAAAAB
mAAAAAQAAIAAAAAd7NaChenx91Eq2EdykPy1wnIU
/61MQt1C3cQZpet1AAAAA6AAAAAgAIAAA0hSIha1fZ3cSejUfMWQZcGLNYTPPQRZjz94R0
NG8sVMAAAAHfQoncOHwyEVLygtlSbdoAwubEIgy3+uYhdu+P03yQPu5SYbGihG3qasKPXJtxFyE
AAAABZZHmD2PmYg3ag+48XQSlrrHscBfV/4GAFbSGA4
/4IU0PU4CSj4vCbc0fcPFXDUtnA51ZFQqubiW7p/Cxbm8A
```

Output

```
....ð..þ..ñ..z.ÀoÀ.ë...ò.*f°~.L.È.À.À.....f.....
....]ið..éñ=Q*ØGr.üþÙr.ÿ.LBÝBÝ.¥ée.....
...èR".µ}.ÜtèØ|À.eÁ.5.Í=.Nf<ýá.
.È.0....wÐ¢w..l.T% ¶T.v..±...-þ¹.]»åíß$.»..lh;.z.º
Ex&ÜEÈ@...YìY.Øù..v Ú..A)k¬{..ð.à` .m!.äþ.SCÒà$£âð.sGÜ<UÃRÙÝ..Y.
@n%»§ð±no.
```

Figure 10

Finally, to charcode and download file generated. Install WxhEditor to check the real DPAPI header:

```
sudo apt-get install wxhexeditor
```

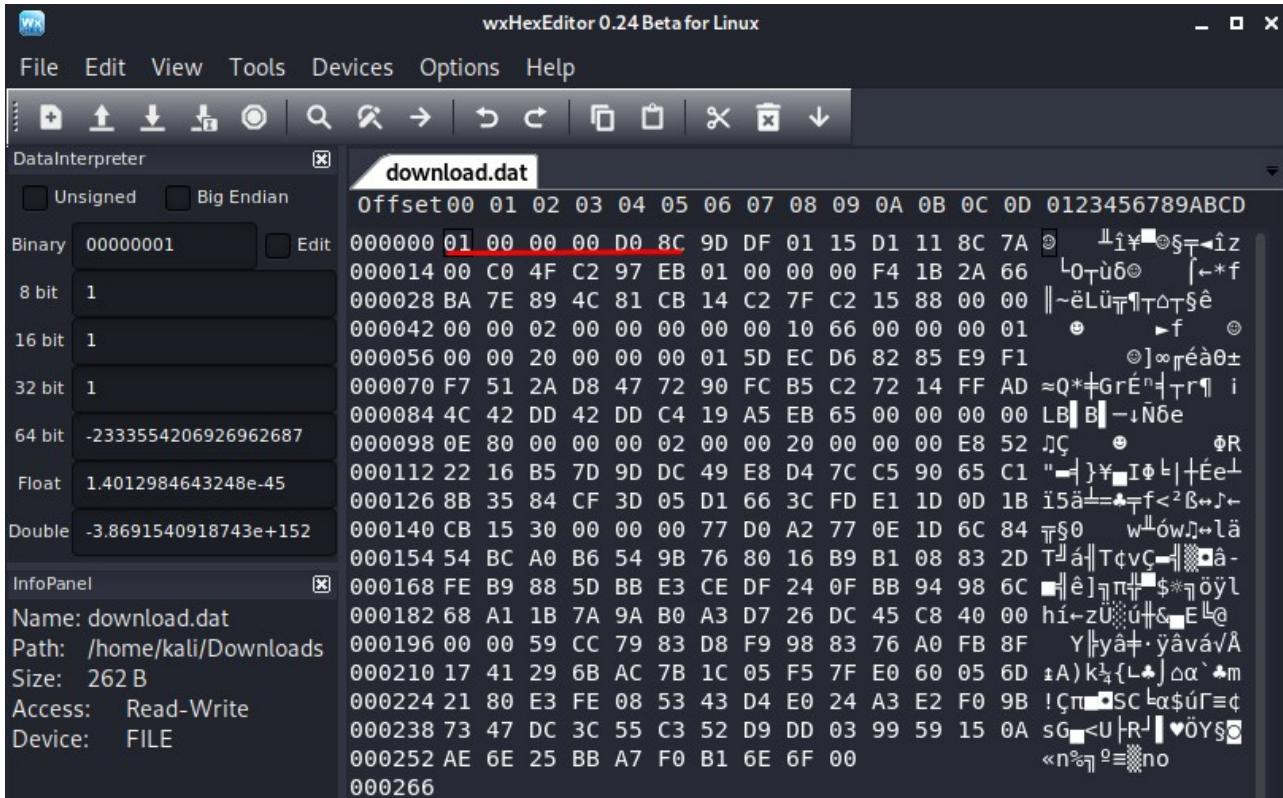


Figure 11

Above image, shows the header of a DPAPI blob. Next step would be to install the following project:

- <https://github.com/tijldeneut/dpapilab-ng>

```
sudo apt update && sudo apt install -y python3-pip git
python3 -m pip install wheel pytz pycryptodomex python-registry
git clone https://github.com/tijldeneut/dpapilab-ng
cd
#Edit requirements.txt and delete construct==2.5.5-reupload
sudo python3 -m pip install -r requirements.txt
sudo cp -rp *.py /usr/bin/
```

Launch the following command in order to check DPAPI blob data:

```
blobinfo.py /home/kali/Downloads/download.dat
```

```
(kali㉿kali)-[~]
$ blobinfo.py /home/kali/Downloads/download.dat
DPAPI BLOB
version      = 1
provider     = df9d8cd0-1501-11d1-8c7a-00c01fc297eb
mkey         = 662a1bf4-7eba-4c89-81cb-14c27fc21588
flags        = 0x0
descr        = b'\x00'
cipherAlgo   = AES-256 [0x6610]
hashAlgo     = sha512 [0x800e]
salt          = 015decd68285e9f1f7512ad8477290fcb5c27214ffad4c42dd42ddc419a5eb65
hmac          = e8522216b57d9ddc49e8d47cc59065c18b3584cf3d05d1663cfde11d0d1bcb15
cipher        = 77d0a2770e1d6c8454bca0b6549b768016b9b108832dfeb9885dbbe3cedf240fb94
sign          = 59cc7983d8f9988376a0fb8f1741296bac7b1c05f57fe060056d2180e3fe085343d4
```

Figure 12

At this point we need to extract

- Master key from memory
- Identify the SID of the user
- Locate Password of the user
- SECURITY and SYSTEM hives
- Sqlite Login Data from Chrome

List Master Key offset:

```
vol.py -f /mnt/hgfs/Maquina/C2-M1/evidence/memory --profile=Win7SP1x64_23418 filescan | grep "Protect"
```

```
vol.py -f /mnt/hgfs/Maquina/C2-M1/evidence/memory --profile=Win7SP1x64_23418 filescan | grep "Protect"
0x000000001e7f6870    17    0 RWD-- \Device\HarddiskVolume1\Windows\ServiceProfiles\NetworkService\AppData\Roaming\Microsoft\SoftwareProtectionPlatform\tokens.dat
0x0000000007b59d500    2    1 R--rwd \Device\HarddiskVolume1\Users\f1s\AppData\Roaming\Microsoft\Protect
0x0000000007c39910    18    1 RW--rwd \Device\HarddiskVolume1\Windows\System32\winevt\Logs\Microsoft-Windows-NetworkAccess\Protect%4Operational.evtx
0x0000000007d0106e0    16    0 R--r-d \Device\HarddiskVolume1\Users\f1s\AppData\Roaming\Microsoft\Protect\$S-1-5-21-480501919-4031562341-2167370758-1000\662a1bf4-7eba-4c89-81cb-14c27fc21588
0x0000000007d51c20    2    1 R--rwd \Device\HarddiskVolume1\Users\f1s\AppData\Roaming\Microsoft\Protect\$S-1-5-21-480501919-4031562341-2167370758-1000
0x0000000007d643810    1    1 R--r-- \Device\HarddiskVolume1\Windows\ServiceProfiles\NetworkService\AppData\Roaming\Microsoft\SoftwareProtectionPlatform\tokens.dat
0x0000000007e289d10   18    1 RW--r-- \Device\HarddiskVolume1\Windows\System32\winevt\Logs\Microsoft-Windows-NetworkAccess\Protect%4WHC.evtx
0x0000000007e7535f0    1    1 _____ \Device\Mailslot\ProtectedPrefix\NetworkService
0x0000000007e753740    1    1 _____ \Device\NamedPipe\ProtectedPrefix\NetworkService
0x0000000007e753890    1    1 _____ \Device\Mailslot\ProtectedPrefix\LocalService
0x0000000007e7539e0    1    1 _____ \Device\NamedPipe\ProtectedPrefix\LocalService
0x0000000007e753b30    1    1 _____ \Device\Mailslot\ProtectedPrefix\Administrators
0x0000000007e753c80    1    1 _____ \Device\NamedPipe\ProtectedPrefix\Administrators
0x0000000007e753d00    1    1 _____ \Device\Mailslot\ProtectedPrefix
0x0000000007e753f20    1    1 _____ \Device\NamedPipe\ProtectedPrefix
0x0000000007e759450    2    1 R--rwd \Device\HarddiskVolume1\Users\f1s\AppData\Roaming\Microsoft\Protect\$S-1-5-21-480501919-4031562341-2167370758-1000
0x0000000007fa43f20    2    0 W--w- \Device\HarddiskVolume1\Windows\ServiceProfiles\NetworkService\AppData\Roaming\Microsoft\SoftwareProtectionPlatform\Cache\cache.dat
0x0000000007fc62dd0    2    1 R--rwd \Device\HarddiskVolume1\Users\f1s\AppData\Roaming\Microsoft\Protect
```

Figure 13

SID could be extracted from path related to the MASTER KEY:

- S-1-5-21-480501919-4031562341-2167370758-1000



Get Master Key using previous offsets:

```
vol.py -f /mnt/hgfs/Maquina/C2-M1/evidence/memory --profile=Win7SP1x64_23418 dumpfiles -Q  
0x000000007d0106e0 -D ./master -u -n
```

```
(kali㉿kali)-[~/evidences]  
$ vol.py -f /mnt/hgfs/Maquina/C2-M1/evidence/memory --profile=Win7SP1x64_23418 dumpfiles -Q 0x000000007d0106e0 -D ./master -u -n  
Volatility Foundation Volatility Framework 2.6  
DataSectionObject 0x7d0106e0 None \Device\HarddiskVolume1\Users\f1s\AppData\Roaming\Microsoft\Protect\5-1-5-21-480501919-4031562341-2167370758-1000\662a1bf4-7eba-4c89-81cb-14c27fc21588
```

List Google Chrome SQLite databases:

```
vol.py -f /mnt/hgfs/Maquina/C2-M1/evidence/memory --profile=Win7SP1x64_23418 filescan | grep "Login Data"
```

```
(kali㉿kali)-[~/evidences]  
$ vol.py -f /mnt/hgfs/Maquina/C2-M1/evidence/memory --profile=Win7SP1x64_23418 filescan | grep "Login Data"  
Volatility Foundation Volatility Framework 2.6  
0x000000007e2f7070 2 1 RW-rw- \Device\HarddiskVolume1\Users\f1s\AppData\Local\Google\Chrome\User Data\Default\Login Data  
0x000000007fc679c0 2 1 RW-rw- \Device\HarddiskVolume1\Users\f1s\AppData\Local\Google\Chrome\User Data\Default\Login Data For Account
```

Figure 14

Get Sqlite Database:

```
vol.py -f /mnt/hgfs/Maquina/C2-M1/evidence/memory --profile=Win7SP1x64_23418 dumpfiles -Q  
0x000000007e2f7070 -D ./sqlite/ -u -n
```

```
(kali㉿kali)-[~/evidences]  
$ vol.py -f /mnt/hgfs/Maquina/C2-M1/evidence/memory --profile=Win7SP1x64_23418 dumpfiles -Q 0x000000007e2f7070 -D ./sqlite/ -u -n  
Volatility Foundation Volatility Framework 2.6  
DataSectionObject 0x7e2f7070 None \Device\HarddiskVolume1\Users\f1s\AppData\Local\Google\Chrome\User Data\Default>Login Data  
SharedCacheMap 0x7e2f7070 None \Device\HarddiskVolume1\Users\f1s\AppData\Local\Google\Chrome\User Data\Default>Login Data
```

Figure 15

Get all windows registry hives:

```
vol.py -f /mnt/hgfs/Maquina/C2-M1/evidence/memory --profile=Win7SP1x64_23418 dumpregistry -D  
registry
```

```
(kali㉿kali)-[~/evidences]  
$ vol.py -f /mnt/hgfs/Maquina/C2-M1/evidence/memory --profile=Win7SP1x64_23418 dumpregistry -D registry  
Volatility Foundation Volatility Framework 2.6  
*****  
Writing out registry: registry.0xfffff8a0012503e0.SAM.reg  
  
*****  
*****  
*****  
***** Writing python registry construct==2.5.5-reupload-88 git  
*****  
*****  
***** Writing out registry: registry.0xfffff8a001c05010.UsrClassdat.reg  
*****  
*****  
*****  
***** Writing out registry: registry.0xfffff8a000052410.HARDWARE.reg  
*****  
*****  
*****  
***** Writing out registry: registry.0xfffff8a000024010.SYSTEM.reg  
*****  
***** In any case, if you do not want to receive a request. Any contribution is much appreciated.  
*****  
*****  
***** Writing out registry: registry.0xfffff8a001b0a010.SyScachehve.reg  
*****
```

Figure 16

Dump the current password of the logged user:

```
vol.py -f /mnt/hgfs/Maquina/C2-M1/evidence/memory --profile=Win7SP1x64_23418 lsadump
```

```
(kali㉿kali)-[~/evidences]
$ vol.py -f /mnt/hgfs/Maquina/C2-M1/evidence/memory --profile=Win7SP1x64_23418 lsadump
Volatility Foundation Volatility Framework 2.6
DefaultPassword
0x00000000 12 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .
0x00000010 61 00 62 00 79 00 67 00 75 00 72 00 6c 00 36 00 a.b.y.g.u.r.l.6.
0x00000020 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 9.... .
DPAPI_SYSTEM
0x00000000 2c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ,.... .
0x00000010 01 00 00 00 ff 45 a6 c5 75 19 c8 23 19 a4 64 7d ..E..u..#..d}
0x00000020 20 89 8b 02 ff d7 fb 91 60 57 44 d2 5b d1 b9 a9 .....`WD.[ ...
0x00000030 87 40 83 e7 29 e5 48 b3 bb 7a cd fc 00 00 00 00 .@..).H..z.....
```

Figure 17

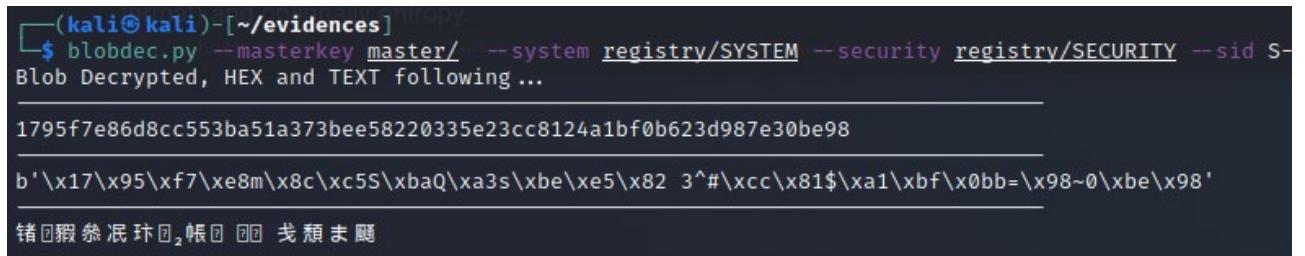
It's mandatory to rename all hives to their real names, removing the beginning of the files:

```
(kali㉿kali)-[~/evidences]
$ tree
.
└── master
    └── file.None.0xfffffa800464ec90 662a1bf4-7eba-4c89-81cb-14c27fc21588.dat
└── registry
    ├── registry.0xffff8a0000d250.no_name.reg
    ├── registry.0xffff8a000024010.SYSTEM.reg
    ├── registry.0xffff8a000052410.HARDWARE.reg
    ├── registry.0xffff8a0000f6410.DEFAULT.reg
    ├── registry.0xffff8a0000f25010.BCD.reg
    ├── registry.0xffff8a0000f38010.SOFTWARE.reg
    ├── registry.0xffff8a0012503e0.SAM.reg
    ├── registry.0xffff8a0012eb010.ntuserdat.reg
    ├── registry.0xffff8a00138f010.NTUSERDAT.reg
    ├── registry.0xffff8a0013fe010.NTUSERDAT.reg
    ├── registry.0xffff8a001b0a010.Sysscachehve.reg
    ├── registry.0xffff8a001c05010.UusrClassdat.reg
    └── registry.0xffff8a0063cf010.SECURITY.reg
└── sqlite
    ├── file.None.0xfffffa8002ef55e0.Login Data.dat
    └── file.None.0xfffffa80038aa2e0.Login Data.vacb
```

Figure 18

At this point, we have to use all files extracted from memory and use to decrypt the DPAPI blob:

```
blobdec.py --masterkey master/ --system registry/SYSTEM --security registry/SECURITY --sid S-1-5-21-480501919-4031562341-2167370758-1000 --password abygurl69 /home/kali/Downloads/download.dat
```



```
(kali㉿kali)-[~/evidences]
$ blobdec.py --masterkey master/ --system registry/SYSTEM --security registry/SECURITY --sid S-1-5-21-480501919-4031562341-2167370758-1000 --password abygurl69 /home/kali/Downloads/download.dat
Blob Decrypted, HEX and TEXT following ...
1795f7e86d8cc553ba51a373bee58220335e23cc8124a1bf0b623d987e30be98
b'\x17\x95\xF7\xE8\x6D\x8C\xC5\x53\xBA\x51\xA3\x73\xBE\xE5\x82\x20\x33\x5E\x23\xCC\x81\x24\xA1\xBF\x0B\x62\x3D\x98\x7E\x30\xBE\x98'
错@猶參混环@帳@ @ 戈頬ま鷗
```

Figure 19

We could use **chrome-edge-opera-dec.py**, but we don't have the whole content of Local State Chrome file, so we need to modify the following script: https://github.com/ohyicong/decrypt-chrome-passwords/blob/main/decrypt_chrome_password.py



```
def get_secret_key():
    try:
        #Remove suffix DPAPI
        secret_key =
            b'\x17\x95\xF7\xE8\x6D\x8C\xC5\x53\xBA\x51\xA3\x73\xBE\xE5\x82\x20\x33\x5E\x23\xCC\x81\x24\xA1\xBF\x0B\x62\x3D\x98\x7E\x30\xBE\x98'
        return secret_key
    except Exception as e:
        print("%s"%str(e))
        print("[ERR] Chrome secretkey cannot be found")
        return None

def decrypt_payload(cipher, payload):
    return cipher.decrypt(payload)
```

Figure 20

Now, we have to insert decrypted password on hex format in **decrypt_chrome_password.py**

We need to open Login Data Sqlite database from Google Chrome (previously extracted from memory):

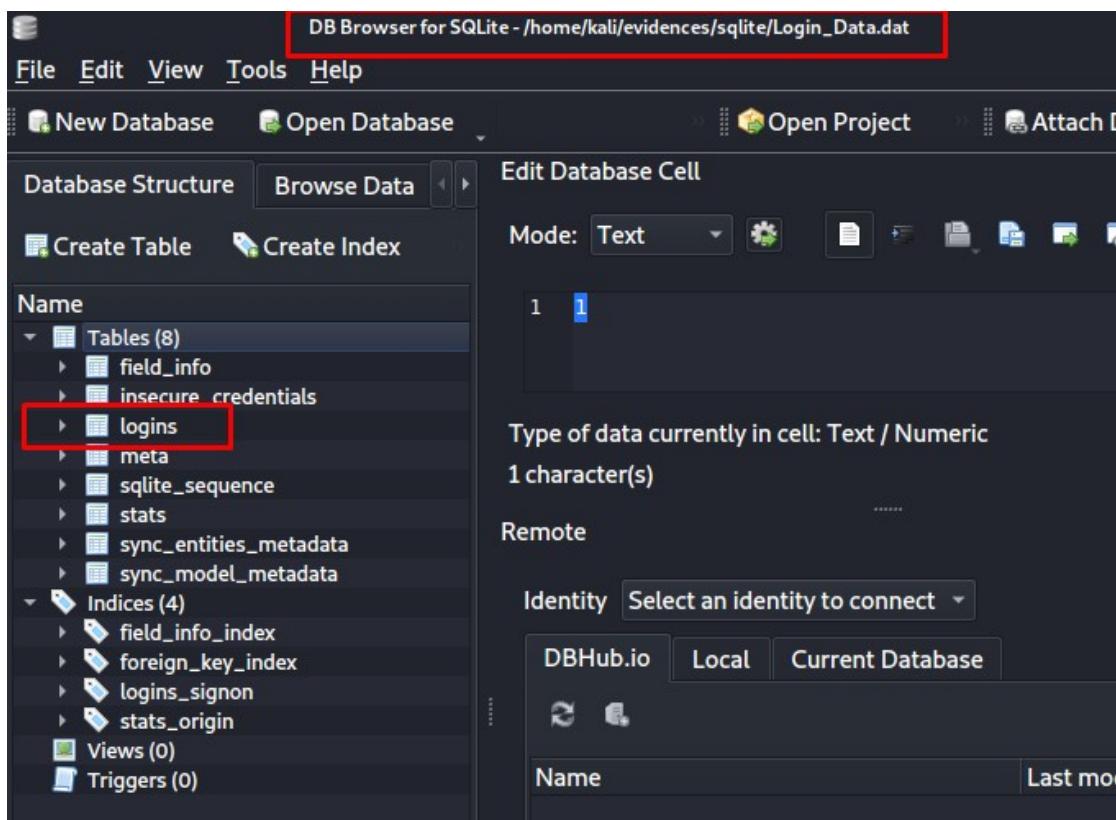


Figure 21

Select logins table, an extract the datablob:

Database Structure							
Table: logins							
rigin_url	action_url	username_element	username_value	password_element	password_value	submit_element	
1.protonmail.com/login	https://account.protonmail.com/	username	skytech_flight_authorization@protonmail.c...	password	BLOB		

Figure 22

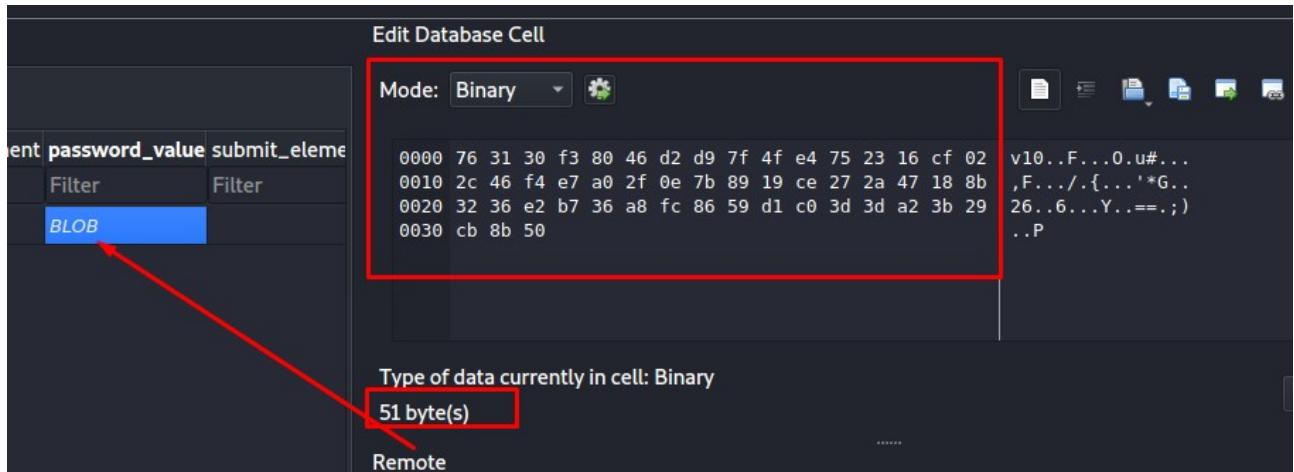


Figure 23

The data blob must be inserted in **decrypt_chrome_password.py**

```

if __name__ == '__main__':
    try:
        secret_key = get_secret_key()
        ciphertext =
b'\x76\x31\x30\xF3\x80\x46\xD2\xD9\x7F\x4F\xE4\x75\x23\x16\xCF\x02\x20\xA0\x2F\x0E\x7B\x89\x19\xCE\x27\x2A\x47\x18\x8B\x32\x36\xE2\xB7\x36\x59\xD1\xC0\x3D\x3D\xA2\x3B\x29\xCB\x8B\x50'
        decrypted_password = decrypt_password(ciphertext, secret_key)
        print("\nPassword: %s\n"%decrypted_password)
    except Exception as e:
        print("[ERR] %s" %str(e))

```

Figure 24

```
python3 /mnt/hgfs/Maquina/decrypt.py
```

```

(kali㉿kali)-[~/evidences]
$ python3 /mnt/hgfs/Maquina/decrypt.py

Password: USHAuCs7oqW32fpjLAPQ

```

Figure 25

Finally the password is extracted:

- USHAuCs7oqW32fpjLAPQ



Decrypt.py (not to provide players, just for reviewer)

```
import os
import re
import sys
import json
import base64
from Cryptodome.Cipher import AES
import shutil

def get_secret_key():
    try:
        # Remove suffix DPAPI
        secret_key =
b'\x17\x95\xF7\xE8\x6D\x8C\xC5\x53\xBA\x51\xA3\x73\xBE\xE5\x82\x20\x33\x5E\x23\xCC\x81\x24
\xA1\xBF\x0B\x62\x3D\x98\x7E\x30\xBE\x98'
        return secret_key
    except Exception as e:
        print("[ERR] Chrome secretkey cannot be found: %s" % str(e))
        return None

def decrypt_payload(cipher, payload):
    return cipher.decrypt(payload)

def generate_cipher(aes_key, iv):
    return AES.new(aes_key, AES.MODE_GCM, iv)

def decrypt_password(ciphertext, secret_key):
    try:
        # (3-a) Initialisation vector for AES decryption
        initialisation_vector = ciphertext[3:15]
        # (3-b) Get encrypted password by removing suffix bytes (last 16 bits)
        # Encrypted password is 192 bits
        encrypted_password = ciphertext[15:-16]
        # (4) Build the cipher to decrypt the ciphertext
        cipher = generate_cipher(secret_key, initialisation_vector)
        decrypted_pass = decrypt_payload(cipher, encrypted_password)
        decrypted_pass = decrypted_pass.decode()
        return decrypted_pass # Return the decrypted password
    except Exception as e:
        print("[ERR] %s" % str(e))
        return None
```



```
if __name__ == '__main__':
    try:
        secret_key = get_secret_key()
        ciphertext =
b'\x76\x31\x30\xf3\x80\x46\xd2\xd9\x7f\x4f\xe4\x75\x23\x16\xcf\x02\x2c\x46\xf4\xe7\xa0\x2f\x0e\
\x7b\x89\x19\xce\x27\x2a\x47\x18\x8b\x32\x36\xe2\xb7\x36\xa8\xfc\x86\x59\xd1\xc0\x3d\x3d\xa2\
\x3b\x29\xcb\x8b\x50'
        decrypted_password = decrypt_password(ciphertext, secret_key)
        if decrypted_password:
            print("\nPassword: %s\n" % decrypted_password)
    except Exception as e:
        print("[ERR] %s" % str(e))
```



Flag Information

flag{USHAuCs7oqW32fpjLAPQ!}