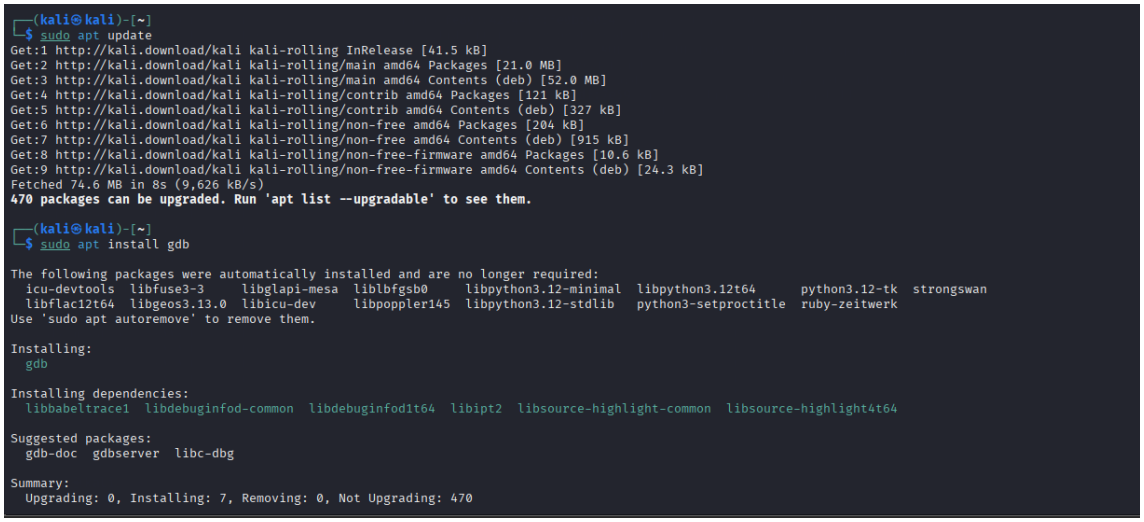
MEMORY ENCRYPT

We open a terminal, create a Python cod using nano, this code Will requests a password and retains it in memory until the second enter.

In a second window, we install requirements in order to do the next step:



```
┌──(kali㉿kali)-[~]
└─$ sudo apt install dump

The following packages were automatically installed and are no longer required:
  icu-devtools  libfuse3-3     libglapi-mesa  liblbfgsb0     libpython3.12-minimal  libpython3.12t64      python3.12-tk  strongswan
  libflac12t64  libgeos3.13.0  libicu-dev     libpoppler145  libpython3.12-stdlib   python3-setproctitle  ruby-zeitwerk
Use 'sudo apt autoremove' to remove them.

Installing:
  dump

Summary:
  Upgrading: 0, Installing: 1, Removing: 0, Not Upgrading: 470
  Download size: 158 kB
  Space needed: 384 kB / 62.4 GB available

Get:1 http://mirror.es.cdn-perfprod.com/kali kali-rolling/main amd64 dump amd64 0.4b49-2 [158 kB]
Fetched 158 kB in 1s (240 kB/s)
Selecting previously unselected package dump.
(Reading database ... 417928 files and directories currently installed.)
Preparing to unpack .../dump_0.4b49-2_amd64.deb ...
Unpacking dump (0.4b49-2) ...
Setting up dump (0.4b49-2) ...
update-alternatives: using /usr/sbin/rmt-dump to provide /usr/sbin/rmt (rmt) in auto mode
Processing triggers for man-db (2.13.0-1) ...
Processing triggers for kali-menu (2025.2.0) ...
Scanning processes...
Scanning linux images...
```

Verify if the Python is running:

```
┌──(kali㉿kali)-[~]
└─$ ps -ef|grep cod_no

kali      166598  164170  0 10:41 pts/3    00:00:00 python cod_no_seguro.py
kali      171719  167427  0 10:51 pts/5    00:00:00 grep --color=auto cod_no
```

We take the proces ID and run the following commands to find the range of directions of the heap zone:

```
┌──(kali㉿kali)-[~]
└─$ sudo cat /proc/164170/maps|grep heap

[sudo] password for kali:
5590d63af000-5590d656e000 rw-p 00000000 00:00 0                          [heap]

┌──(kali㉿kali)-[~]
└─$ sudo gdb -p 164170
```

```
Start Addr         End Addr           Size               Offset             Perms File
0×000055909dd37000 0×000055909dd4e000 0×17000            0×0                r--p  /usr/bin/zsh
0×000055909dd4e000 0×000055909ddec000 0×9e000            0×17000            r-xp  /usr/bin/zsh
0×000055909ddec000 0×000055909de07000 0×1b000            0×b5000            r--p  /usr/bin/zsh
0×000055909de07000 0×000055909de09000 0×2000             0×cf000            r--p  /usr/bin/zsh
0×000055909de09000 0×000055909de0f000 0×6000             0×d1000            rw-p  /usr/bin/zsh
0×000055909de0f000 0×000055909de23000 0×14000            0×0                rw-p
0×00005590d63af000 0×00005590d656e000 0×1bf000           0×0                rw-p  [heap]
0×00007f0fb9600000 0×00007f0fb98ad000 0×2ad000           0×0                r--s  /usr/share/zsh/functions/Completion/Unix.zwc
0×00007f0fb98d0000 0×00007f0fb98d3000 0×3000             0×0                r--p  /usr/lib/x86_64-linux-gnu/zsh/5.9/zsh/computil.so
0×00007f0fb98d3000 0×00007f0fb98e1000 0×e000             0×3000             r-xp  /usr/lib/x86_64-linux-gnu/zsh/5.9/zsh/computil.so
0×00007f0fb98e1000 0×00007f0fb98e3000 0×2000             0×11000            r--p  /usr/lib/x86_64-linux-gnu/zsh/5.9/zsh/computil.so
```

We run this comando to create an exacto copy of a memory spcae:

```
(gdb) dump memory /tmp/output.bin 0×00005590d63af000 0×00005590d656e000
(gdb) quit
A debugging session is active.

        Inferior 1 [process 164170] will be detached.

Quit anyway? (y or n) y
Detaching from program: /usr/bin/zsh, process 164170
[Inferior 1 (process 164170) detached]
```

With the bin as output file, we use the strings method to convert it to .txt

```
┌──(kali㉿kali)-[~]
└─$ strings /tmp/output.bin > /tmp/output.txt
```

To increase security, we install croptography:

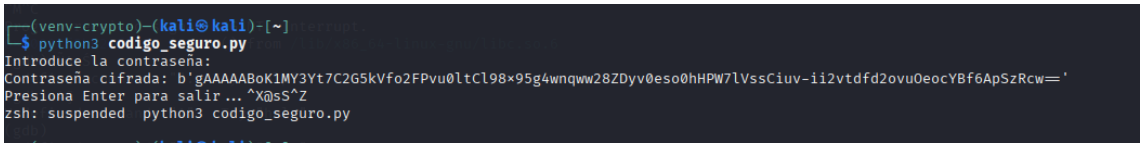
```
└─$ sudo apt install python3-venv  # Instala venv si no lo tienes
python3 -m venv ~/venv-crypto
source ~/venv-crypto/bin/activate
pip install cryptography

python3-venv is already the newest version (3.13.2-2).
python3-venv set to manually installed.
The following packages were automatically installed and are no longer required:
  icu-devtools  libfuse3-3     libglapi-mesa  liblbfgsb0     libpython3.12-minimal  libpython3.12t64      python3.12-tk  strongswan
  libflac12t64  libgeos3.13.0  libicu-dev     libpoppler145  libpython3.12-stdlib   python3-setproctitle  ruby-zeitwerk
Use 'sudo apt autoremove' to remove them.

Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 470
Collecting cryptography
  Downloading cryptography-45.0.2-cp311-abi3-manylinux_2_34_x86_64.whl.metadata (5.7 kB)
Collecting cffi≥1.14 (from cryptography)
  Downloading cffi-1.17.1-cp313-cp313-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting pycparser (from cffi≥1.14→cryptography)
  Downloading pycparser-2.22-py3-none-any.whl.metadata (943 bytes)
Downloading cryptography-45.0.2-cp311-abi3-manylinux_2_34_x86_64.whl (4.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.5/4.5 MB 45.6 MB/s eta 0:00:00
Downloading cffi-1.17.1-cp313-cp313-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (479 kB)
Downloading pycparser-2.22-py3-none-any.whl (117 kB)
Installing collected packages: pycparser, cffi, cryptography
Successfully installed cffi-1.17.1 cryptography-45.0.2 pycparser-2.22
```

Then, I create another Python file, called archivo_seguro where I put this code :

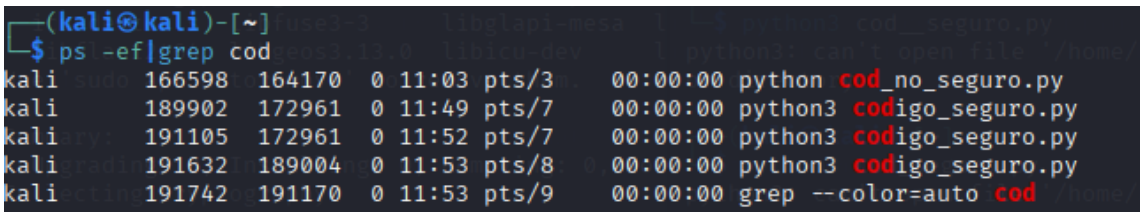
```
from cryptography.fernet import Fernet

import getpass

# Esto generará una clave segura para cifrar/descifrar la contraseña

key = Fernet.generate_key()

cipher_suite = Fernet(key)

def main():

    # Usamos getpass para que la contraseña no se muestre al escribirla

    password = getpass.getpass("Introduce la contraseña: ")

    # Ciframos la contraseña
```

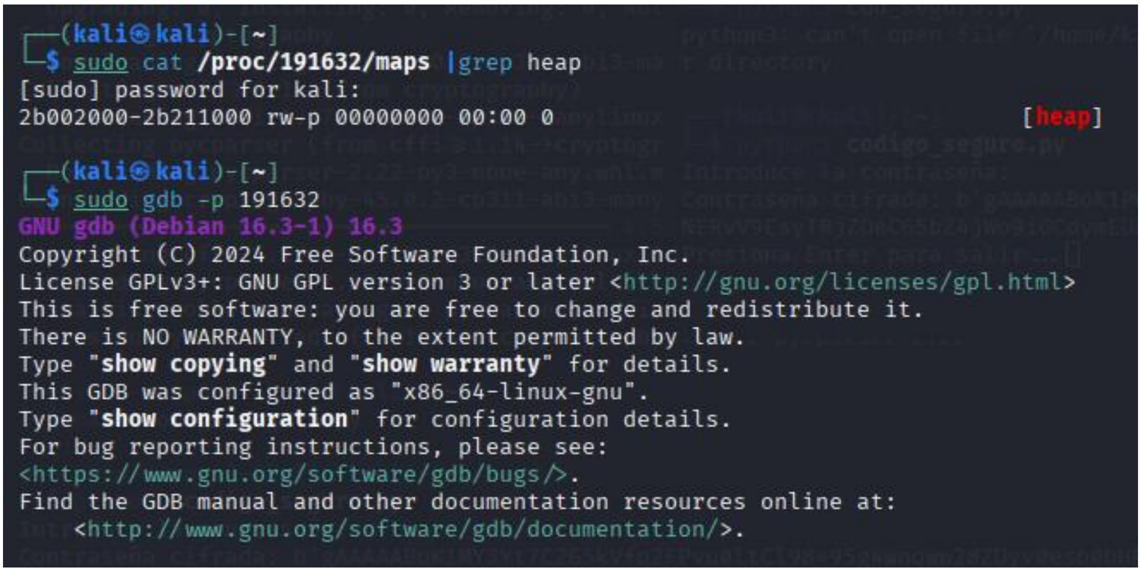
```
    encrypted_password = cipher_suite.encrypt(password.encode('utf-8'))
    # Limpiamos la memoria del texto claro
    password = None
    # Para fines de demostración, mostramos la contraseña cifrada
    print(f"Contraseña cifrada: {encrypted_password}")
    # Deciframos la contraseña (por ejemplo, si necesitas usarla más tarde)
    decrypted_password = cipher_suite.decrypt(encrypted_password).decode('utf-8')
    # Muestra la contraseña descifrada (por motivos de demostración, no hacer esto en aplicaciones reales)
    #print(f"Contraseña descifrada: {decrypted_password}")
    # Limpia la memoria
    decrypted_password = None
    encrypted_password = None
    input("Presiona Enter para salir...")
if __name__ == "__main__":
    main()
```

And when I run it the password isn't visible and it appears hashed



Now I try the previous steps with the secure code running:





Now the password appears hashed: