

Análisis forense de un volcado de memoria volátil (RAM) utilizando Volatility.

Daniel López Gala

12 de Diciembre de 2022

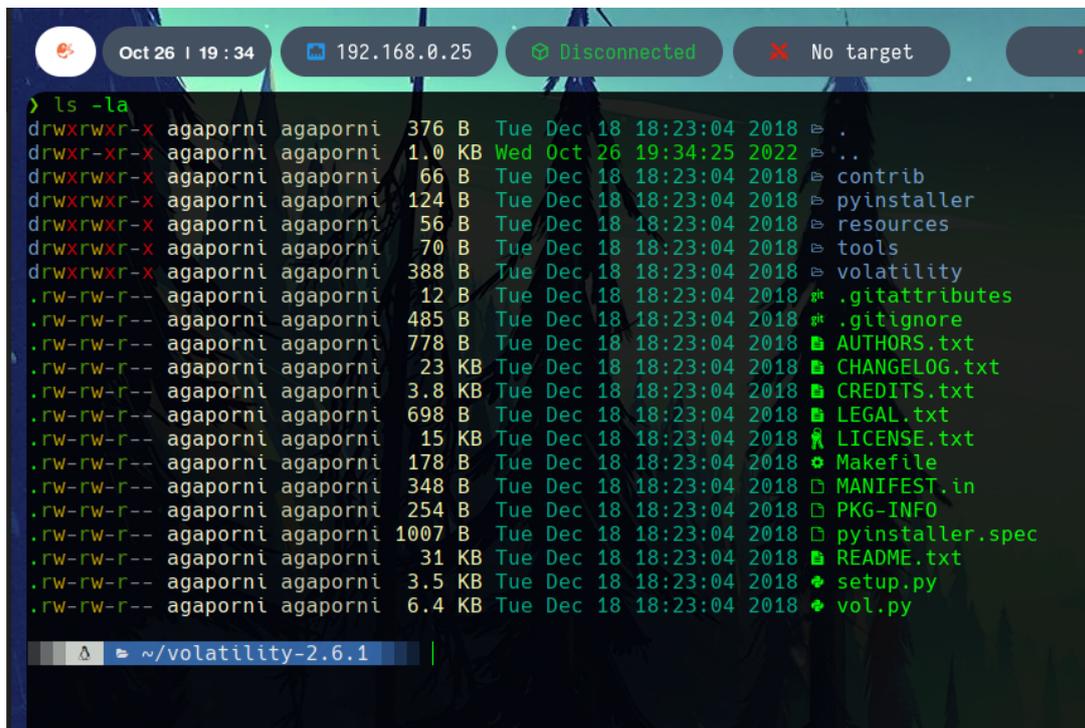
1. Introducción.

Se trata de realizar el análisis de memoria RAM de un *dump* de un sistema con Ubuntu cuya extracción se realizará con [LiMe](#), un módulo LKM que permite adquisiciones de memoria volátil de Linux y sistemas basados en Linux, como Android.

El análisis se realizará utilizando [Volatility](#), un framework con herramientas forenses para la extracción de evidencias digitales a partir de muestras de memoria volátil (RAM).

2. Instalación de Volatility.

La versión 2.6.1 de Volatility se puede descargar desde [este enlace](#). Una vez descargada y descomprimida, podremos ver los siguientes archivos.



```
> ls -la
drwxrwxr-x agaporni agaporni 376 B Tue Dec 18 18:23:04 2018 .
drwxr-xr-x agaporni agaporni 1.0 KB Wed Oct 26 19:34:25 2022 ..
drwxrwxr-x agaporni agaporni 66 B Tue Dec 18 18:23:04 2018 contrib
drwxrwxr-x agaporni agaporni 124 B Tue Dec 18 18:23:04 2018 pyinstaller
drwxrwxr-x agaporni agaporni 56 B Tue Dec 18 18:23:04 2018 resources
drwxrwxr-x agaporni agaporni 70 B Tue Dec 18 18:23:04 2018 tools
drwxrwxr-x agaporni agaporni 388 B Tue Dec 18 18:23:04 2018 volatility
-rw-rw-r-- agaporni agaporni 12 B Tue Dec 18 18:23:04 2018 *.gitattributes
-rw-rw-r-- agaporni agaporni 485 B Tue Dec 18 18:23:04 2018 *.gitignore
-rw-rw-r-- agaporni agaporni 778 B Tue Dec 18 18:23:04 2018 AUTHORS.txt
-rw-rw-r-- agaporni agaporni 23 KB Tue Dec 18 18:23:04 2018 CHANGELOG.txt
-rw-rw-r-- agaporni agaporni 3.8 KB Tue Dec 18 18:23:04 2018 CREDITS.txt
-rw-rw-r-- agaporni agaporni 698 B Tue Dec 18 18:23:04 2018 LEGAL.txt
-rw-rw-r-- agaporni agaporni 15 KB Tue Dec 18 18:23:04 2018 LICENSE.txt
-rw-rw-r-- agaporni agaporni 178 B Tue Dec 18 18:23:04 2018 Makefile
-rw-rw-r-- agaporni agaporni 348 B Tue Dec 18 18:23:04 2018 MANIFEST.in
-rw-rw-r-- agaporni agaporni 254 B Tue Dec 18 18:23:04 2018 PKG-INFO
-rw-rw-r-- agaporni agaporni 1007 B Tue Dec 18 18:23:04 2018 pyinstaller.spec
-rw-rw-r-- agaporni agaporni 31 KB Tue Dec 18 18:23:04 2018 README.txt
-rw-rw-r-- agaporni agaporni 3.5 KB Tue Dec 18 18:23:04 2018 setup.py
-rw-rw-r-- agaporni agaporni 6.4 KB Tue Dec 18 18:23:04 2018 vol.py
```

Figura 1: Volatility 2.6.1

Como se indica en el repositorio, Volatility funciona con Python 2, ya que existe una nueva versión de Volatility llamada Volatility 3 que utiliza Python 3, pero es incompatible con muchos módulos de análisis y da bastantes problemas en general, por lo que se sigue utilizando la versión antigua. Por tanto, debemos instalar Python 2 tal y como se indica [aquí](#).

```
sudo apt install python
```

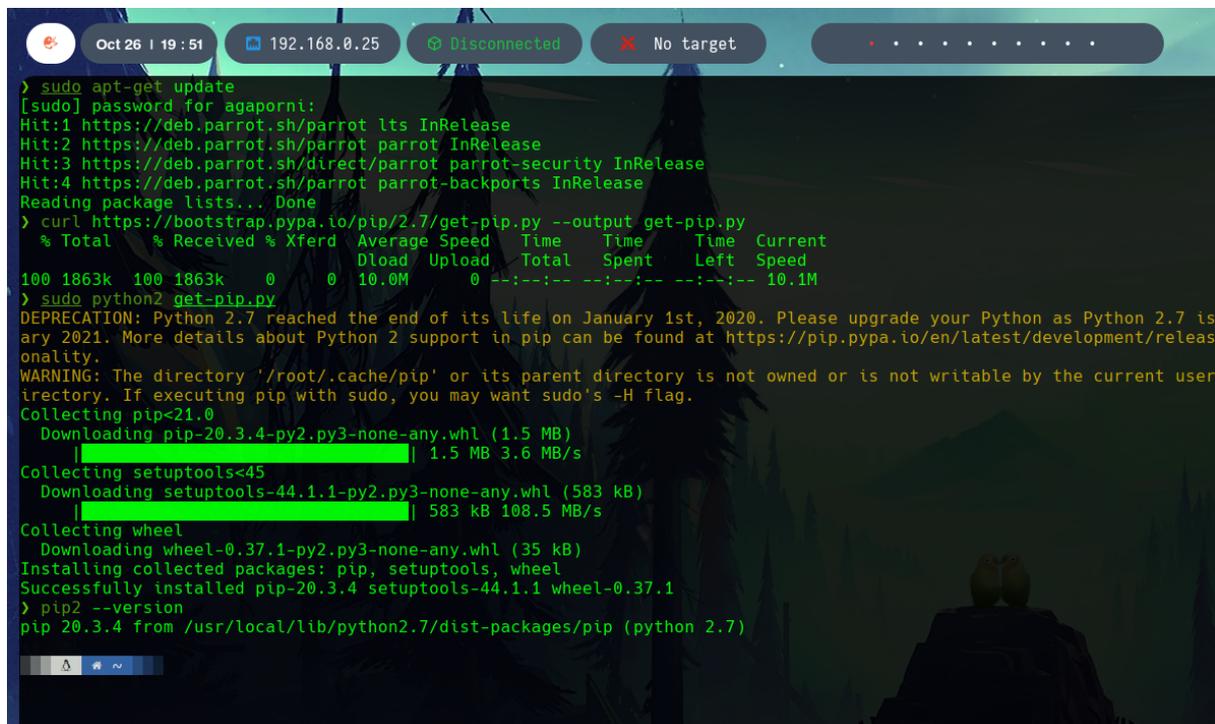
```
sudo apt install curl
```

```
curl https://bootstrap.pypa.io/pip/2.7/get-pip.py --output get-pip.py
```

```
sudo python2 get-pip.py
```

```
pip2 --version
```

Continuamos con el proceso de instalación de Volatility ejecutando `sudo python2 get-pip.py` para obtener pip de Python 2.



```
> sudo apt-get update
[sudo] password for agaporni:
Hit:1 https://deb.parrot.sh/parrot lts InRelease
Hit:2 https://deb.parrot.sh/parrot parrot InRelease
Hit:3 https://deb.parrot.sh/direct/parrot parrot-security InRelease
Hit:4 https://deb.parrot.sh/parrot parrot-backports InRelease
Reading package lists... Done
> curl https://bootstrap.pypa.io/pip/2.7/get-pip.py --output get-pip.py
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload  Total   Spent    Left  Speed
100 1863k  100 1863k    0     0  10.0M      0  --:--:-- --:--:-- --:--:--  10.1M
> sudo python2 get-pip.py
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 is
ary 2021. More details about Python 2 support in pip can be found at https://pip.pypa.io/en/latest/development/releas
onality.
WARNING: The directory '/root/.cache/pip' or its parent directory is not owned or is not writable by the current user
irectory. If executing pip with sudo, you may want sudo's -H flag.
Collecting pip<21.0
  Downloading pip-20.3.4-py2.py3-none-any.whl (1.5 MB)
    |#####| 1.5 MB 3.6 MB/s
Collecting setuptools<45
  Downloading setuptools-44.1.1-py2.py3-none-any.whl (583 kB)
    |#####| 583 kB 108.5 MB/s
Collecting wheel
  Downloading wheel-0.37.1-py2.py3-none-any.whl (35 kB)
Installing collected packages: pip, setuptools, wheel
Successfully installed pip-20.3.4 setuptools-44.1.1 wheel-0.37.1
> pip2 --version
pip 20.3.4 from /usr/local/lib/python2.7/dist-packages/pip (python 2.7)
```

Figura 2: Instalación de get-pip.py

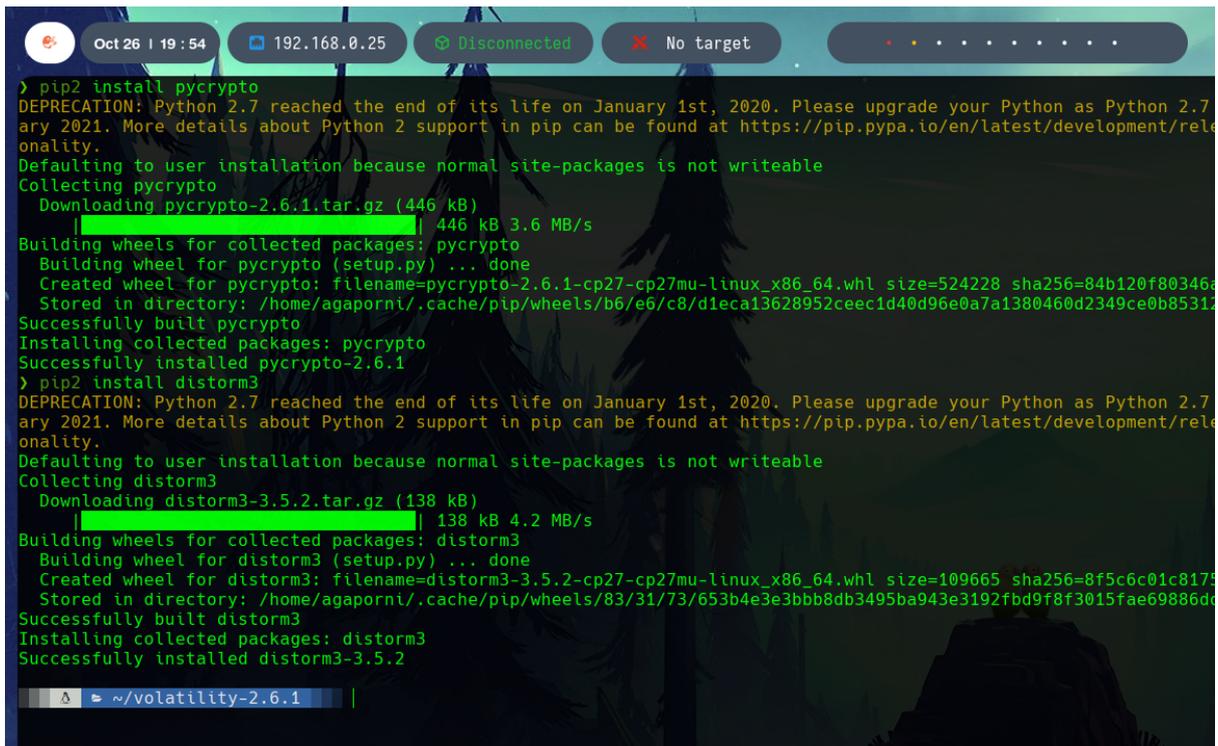
A continuación instalamos las dependencias necesarias de Volatility.

```
pip2 install --upgrade setuptools
```

```
sudo apt-get install python-dev
```

```
pip2 install pycrypto
```

```
pip2 install distorm3
```



```
Oct 26 | 19:54 | 192.168.0.25 | Disconnected | No target
> pip2 install pycrypto
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7
ary 2021. More details about Python 2 support in pip can be found at https://pip.pya.io/en/latest/development/rel
onality.
Defaulting to user installation because normal site-packages is not writeable
Collecting pycrypto
  Downloading pycrypto-2.6.1.tar.gz (446 kB)
    | 446 kB 3.6 MB/s
Building wheels for collected packages: pycrypto
  Building wheel for pycrypto (setup.py) ... done
  Created wheel for pycrypto: filename=pycrypto-2.6.1-cp27-cp27mu-linux_x86_64.whl size=524228 sha256=84b120f80346
  Stored in directory: /home/agaporni/.cache/pip/wheels/b6/e6/c8/d1eca13628952ceec1d40d96e0a7a1380460d2349ce0b8531
Successfully built pycrypto
Installing collected packages: pycrypto
Successfully installed pycrypto-2.6.1
> pip2 install distorm3
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7
ary 2021. More details about Python 2 support in pip can be found at https://pip.pya.io/en/latest/development/rel
onality.
Defaulting to user installation because normal site-packages is not writeable
Collecting distorm3
  Downloading distorm3-3.5.2.tar.gz (138 kB)
    | 138 kB 4.2 MB/s
Building wheels for collected packages: distorm3
  Building wheel for distorm3 (setup.py) ... done
  Created wheel for distorm3: filename=distorm3-3.5.2-cp27-cp27mu-linux_x86_64.whl size=109665 sha256=8f5c6c01c817
  Stored in directory: /home/agaporni/.cache/pip/wheels/83/31/73/653b4e3e3bbb8db3495ba943e3192fbd9f8f3015fae69886d
Successfully built distorm3
Installing collected packages: distorm3
Successfully installed distorm3-3.5.2
~/volatility-2.6.1
```

Figura 3: Instalación de las dependencias de Volatility.

Ahora tenemos dos opciones, dirigirnos al directorio de Volatility y ejecutar `./vol.py [comando]` para utilizar la herramienta, o realizar una instalación completa para poder usarlo desde cualquier directorio y poder eliminar el repositorio. Optamos por la segunda opción.

```
sudo python setup.py install
```

3. Creación de un volcado LiMe.

Es posible extraer la memoria de **forma local** con un comando del tipo `sudo insmod lime-3.16.0-77-generic.ko "path=/home/marcos/Evidencias/MemLub1404 format=raw"`, o de forma remota con `sudo insmod LiME/src/lime-3.16.0-77-generic.ko "path=tcp:4444 format=raw"`.

Esto crearía un puerto de escucha desde el que se podría descargar el volcado de memoria a través de una máquina expuesta a la misma red. En el artículo citado hay un oneliner que permite obtener los hashes adecuados para adjuntarlo en un informe pericial.

```
time ncat 192.168.1.33 && cp MemLubuntu1404 MemLubuntu1404.bak && ls -l
→ && sha1sum MemLubuntu1404 MemLubuntu1404.bak > HashMemLubuntu.txt &&
→ cat HashMemLubuntu.txt
```

El primer paso es descargar la herramienta en el sistema.

```
git clone https://github.com/504ensicsLabs/LiME.git
```

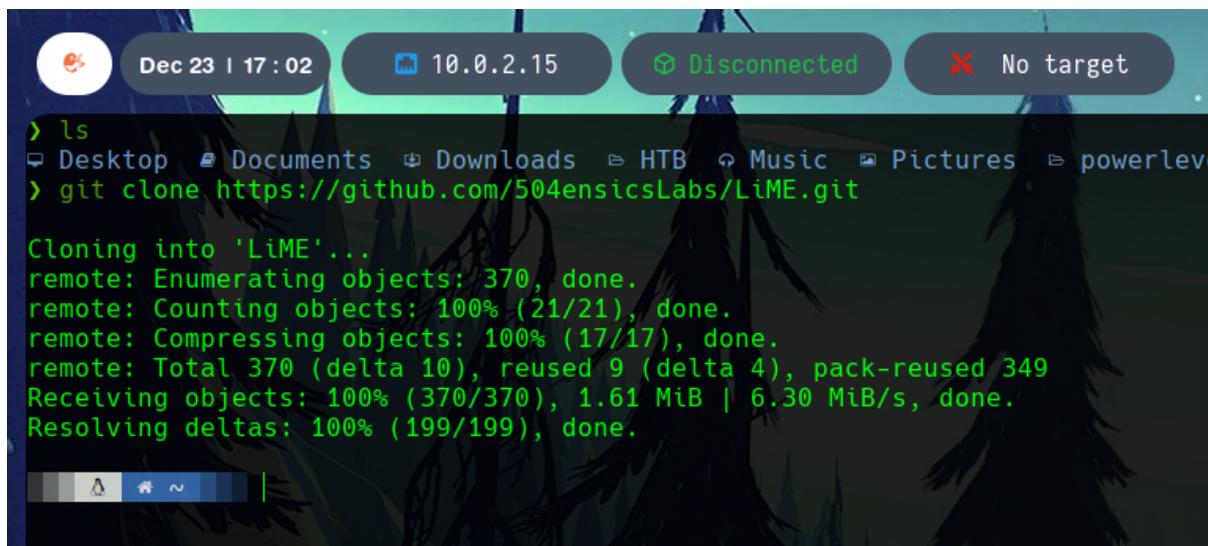


Figura 4: Instalación de LiMe.

A continuación, simplemente accediendo al directorio src de la herramienta y compilando podremos generar el módulo de nuestro sistema, que se corresponde al archivo con formato .ko generado, como podemos ver en la figura 5.

```
cd LiME/src  
make
```

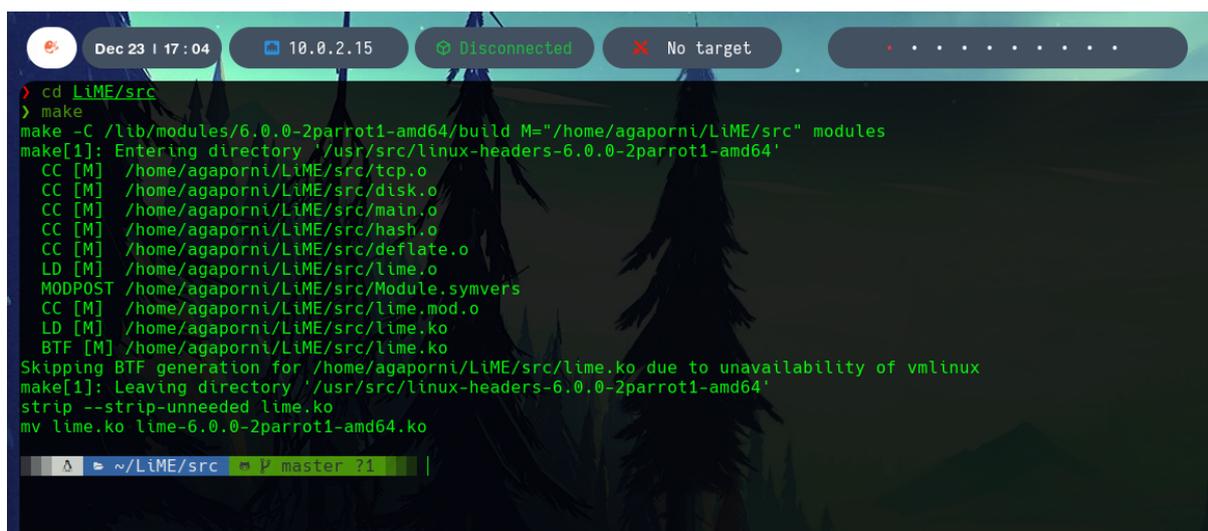


Figura 5: Extracción del volcado de memoria.

4. Análisis del volcado.

Ahora comenzaremos con el análisis de un volcado de memoria. Simularemos un ejemplo práctico e intentaremos obtener información relevante a partir de un volcado LiMe como el que acabamos de crear.

4.1. Situación inicial.

Se nos ofrece un fichero .zip que contiene un volcado de memoria de un sistema Linux.

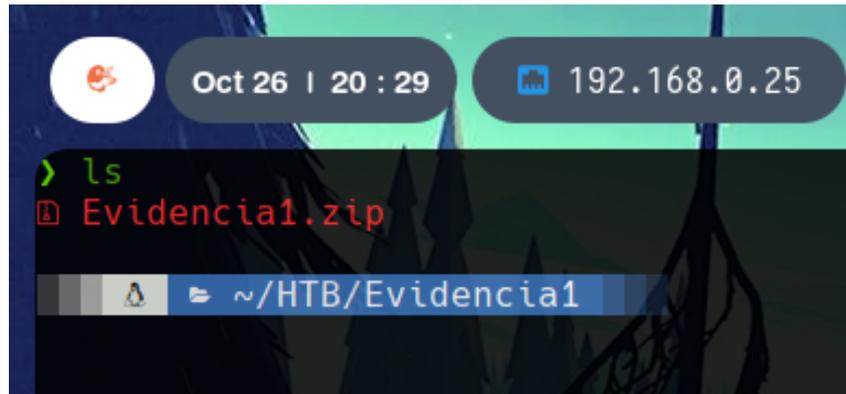


Figura 6: Evidencia.

En este comprimido nos encontramos el volcado de la evidencia en formato .lime y un archivo module.dwarf junto al directorio boot que contiene el archivo System.map del sistema de donde se ha extraído la evidencia. Esto será necesario para la creación del profile.

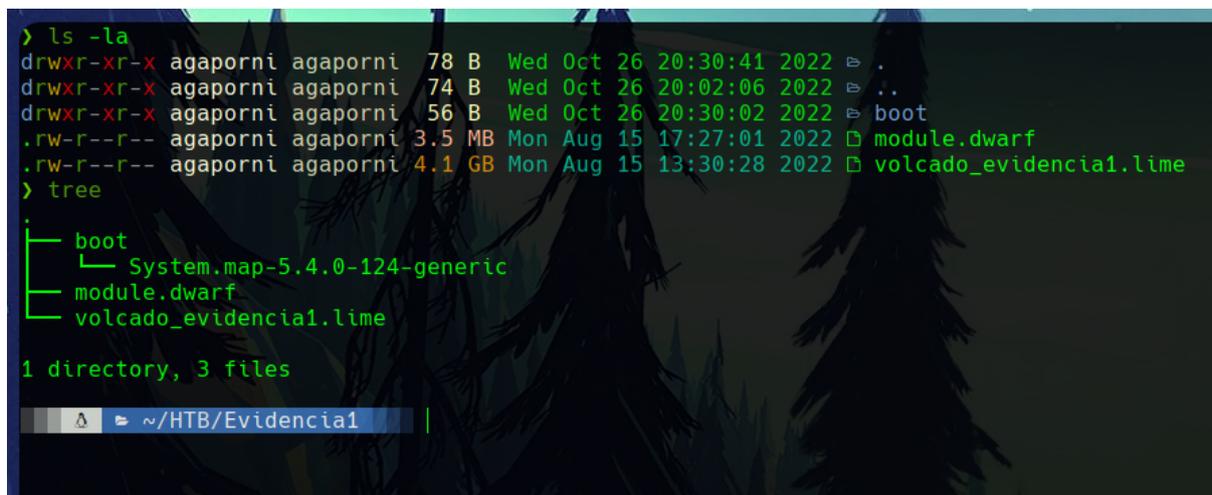
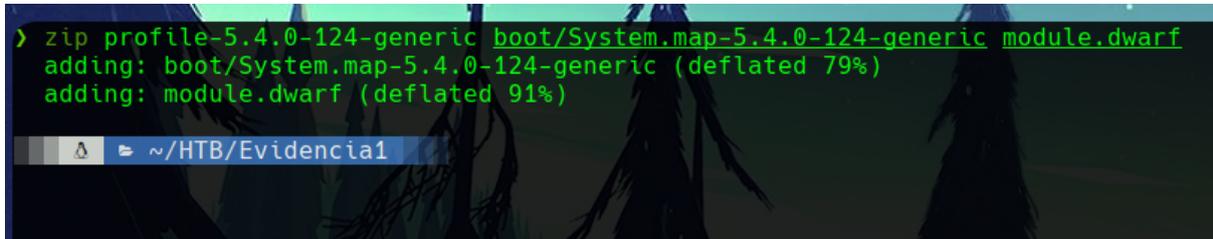


Figura 7: Contenido de la evidencia.

4.2. Creación del perfil.

A continuación tenemos que crear un perfil de volatility con el System.map y el archivo module.dwarf. Esto puede hacerse con el comando zip de la siguiente manera.

```
zip profile-5.8.0-38-generic System.map-5.8.0-38-generic module.dwarf
```

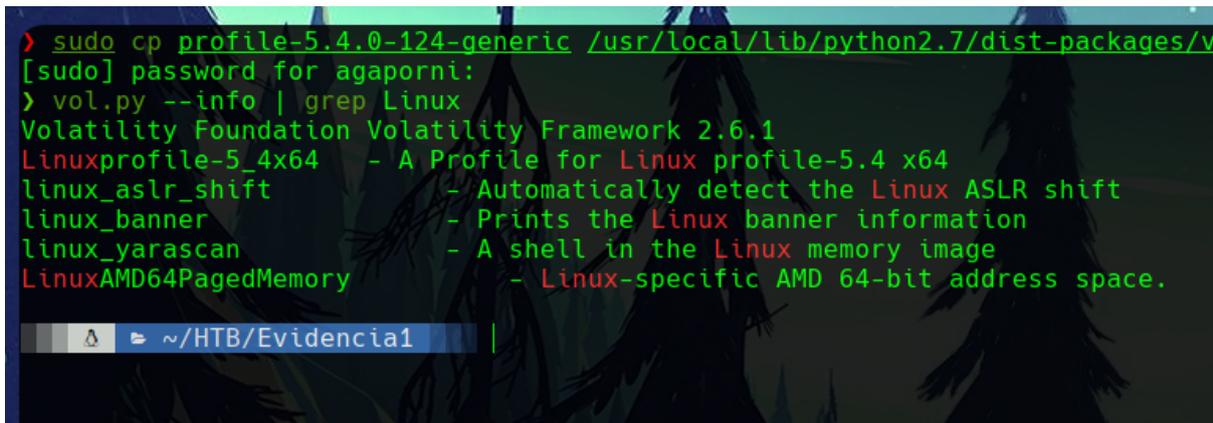


```
> zip profile-5.4.0-124-generic boot/System.map-5.4.0-124-generic module.dwarf
adding: boot/System.map-5.4.0-124-generic (deflated 79%)
adding: module.dwarf (deflated 91%)
~ /HTB/Evidencia1
```

Figura 8: Creación del perfil.

Ahora hay que copiar el profile a la carpeta `volatility/plugins/overlays/linux/`.

En nuestro caso, como hemos instalado volatility con el setup, la ruta de los plugins estará en el directorio `/usr/local/lib/python2.7/dist-packages/`, concretamente en el paquete `/usr/local/lib/python2.7/dist-packages/volatility-2.6.1-py2.7.egg/volatility/plugins/overlays/linux/`.



```
> sudo cp profile-5.4.0-124-generic /usr/local/lib/python2.7/dist-packages/v
[sudo] password for agaporni:
> vol.py --info | grep Linux
Volatility Foundation Volatility Framework 2.6.1
Linuxprofile-5_4x64 - A Profile for Linux profile-5.4 x64
linux_aslr_shift - Automatically detect the Linux ASLR shift
linux_banner - Prints the Linux banner information
linux_yarascan - A shell in the Linux memory image
LinuxAMD64PagedMemory - Linux-specific AMD 64-bit address space.
~ /HTB/Evidencia1
```

Figura 9: Instalación del perfil.

Podemos verificar que el perfil está correctamente instalado utilizando Volatility con el comando `--info` y filtrando por Linux.

```
vol.py --info | grep Linux
```

Este comando muestra los perfiles de Linux que están instalados en la herramienta. Por defecto, Volatility contiene algunos perfiles de las distribuciones de Linux y los kernels más frecuentes, pero en casos como este es necesario añadirlo manualmente.

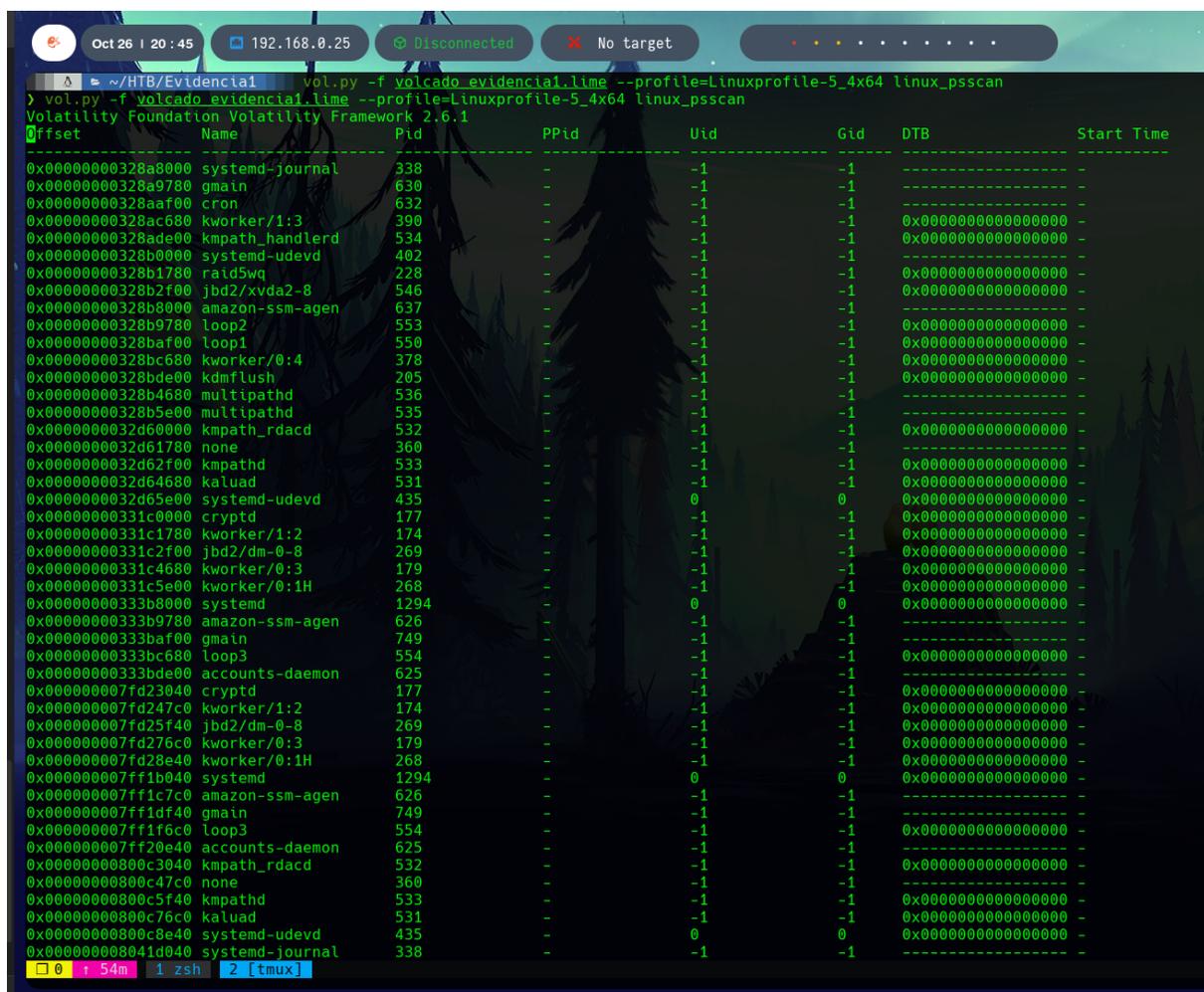
4.3. Procesos en ejecución.

Para utilizar la herramienta harán falta, por norma general, 3 parámetros. El primero, el archivo del volcado .lime, seguido del identificador del perfil que estamos usando, y por último el plugin que queremos aplicar para analizar la imagen.

```
vol.py -f volcado_evidencia1.lime --profile=Linuxprofile-5_4x64 linux_psscan
```

Para identificar los procesos de un sistema se puede utilizar el plugin *psscan*, que recorre la lista que apunta a *PsActiveProcessHead*. El plugin *pstree* los enumera con la misma técnica pero los imprime con estructura de árbol. Por último, podemos utilizar *psscan* que además de lo anterior muestra el ID del offset, procesos inactivos, u otros ocultos. Es decir, realiza un escaneo en el volcado.

En nuestro ejemplo, utilizando *psscan* podemos identificar los siguientes procesos.



```
Oct 26 | 20:45 | 192.168.0.25 | Disconnected | No target
~/HTB/Evidencia1 | vol.py -f volcado_evidencia1.lime --profile=Linuxprofile-5_4x64 linux_psscan
> vol.py -f volcado_evidencia1.lime --profile=Linuxprofile-5_4x64 linux_psscan
Volatility Foundation Volatility Framework 2.6.1
Offset      Name          Pid    PPid    Uid      Gid      DTB      Start Time
-----
0x00000000328a8000 systemd-journal 338    -       -1       -1       -         -
0x00000000328a9780 gmain         630    -       -1       -1       -         -
0x00000000328aaf00 cron          632    -       -1       -1       -         -
0x00000000328ac680 kworker/1:3   390    -       -1       -1       0x0000000000000000 -
0x00000000328ade00 kmpath_handlerd 534    -       -1       -1       0x0000000000000000 -
0x00000000328b0000 systemd-udev  402    -       -1       -1       -         -
0x00000000328b1780 raid5wq      228    -       -1       -1       0x0000000000000000 -
0x00000000328b2f00 jbd2/xvda2-8 546    -       -1       -1       0x0000000000000000 -
0x00000000328b8000 amazon-ssm-agen 637    -       -1       -1       -         -
0x00000000328b9780 loop2        553    -       -1       -1       0x0000000000000000 -
0x00000000328baf00 loop1        550    -       -1       -1       0x0000000000000000 -
0x00000000328bc680 kworker/0:4   378    -       -1       -1       0x0000000000000000 -
0x00000000328bde00 kdmflush     205    -       -1       -1       0x0000000000000000 -
0x00000000328b4680 multipathd   536    -       -1       -1       -         -
0x00000000328b5e00 multipathd   535    -       -1       -1       -         -
0x0000000032d60000 kmpath_rdacd  532    -       -1       -1       0x0000000000000000 -
0x0000000032d61780 none         360    -       -1       -1       -         -
0x0000000032d62f00 kmpathd      533    -       -1       -1       0x0000000000000000 -
0x0000000032d64680 kaluad       531    -       -1       -1       0x0000000000000000 -
0x0000000032d65e00 systemd-udev  435    0       0       0       0x0000000000000000 -
0x00000000331c0000 cryptd       177    -       -1       -1       0x0000000000000000 -
0x00000000331c1780 kworker/1:2   174    -       -1       -1       0x0000000000000000 -
0x00000000331c2f00 jbd2/dm-0-8  269    -       -1       -1       0x0000000000000000 -
0x00000000331c4680 kworker/0:3   179    -       -1       -1       0x0000000000000000 -
0x00000000331c5e00 kworker/0:1H  268    -       -1       -1       0x0000000000000000 -
0x00000000333b8000 systemd      1294   0       0       0       0x0000000000000000 -
0x00000000333b9780 amazon-ssm-agen 626    -       -1       -1       -         -
0x00000000333baf00 gmain        749    -       -1       -1       -         -
0x00000000333bc680 loop3        554    -       -1       -1       0x0000000000000000 -
0x00000000333bde00 accounts-daemon 625    -       -1       -1       -         -
0x000000007fd23040 cryptd       177    -       -1       -1       0x0000000000000000 -
0x000000007fd247c0 kworker/1:2   174    -       -1       -1       0x0000000000000000 -
0x000000007fd25f40 jbd2/dm-0-8  269    -       -1       -1       0x0000000000000000 -
0x000000007fd276c0 kworker/0:3   179    -       -1       -1       0x0000000000000000 -
0x000000007fd28e40 kworker/0:1H  268    -       -1       -1       0x0000000000000000 -
0x000000007ff1b040 systemd      1294   0       0       0       0x0000000000000000 -
0x000000007ff1c7c0 amazon-ssm-agen 626    -       -1       -1       -         -
0x000000007ff1df40 gmain        749    -       -1       -1       -         -
0x000000007ff1f6c0 loop3        554    -       -1       -1       0x0000000000000000 -
0x000000007ff20e40 accounts-daemon 625    -       -1       -1       -         -
0x00000000800c3040 kmpath_rdacd  532    -       -1       -1       0x0000000000000000 -
0x00000000800c47c0 none         360    -       -1       -1       -         -
0x00000000800c5f40 kmpathd      533    -       -1       -1       0x0000000000000000 -
0x00000000800c76c0 kaluad       531    -       -1       -1       0x0000000000000000 -
0x00000000800c8e40 systemd-udev  435    0       0       0       0x0000000000000000 -
0x000000008041d040 systemd-journal 338    -       -1       -1       -         -
```

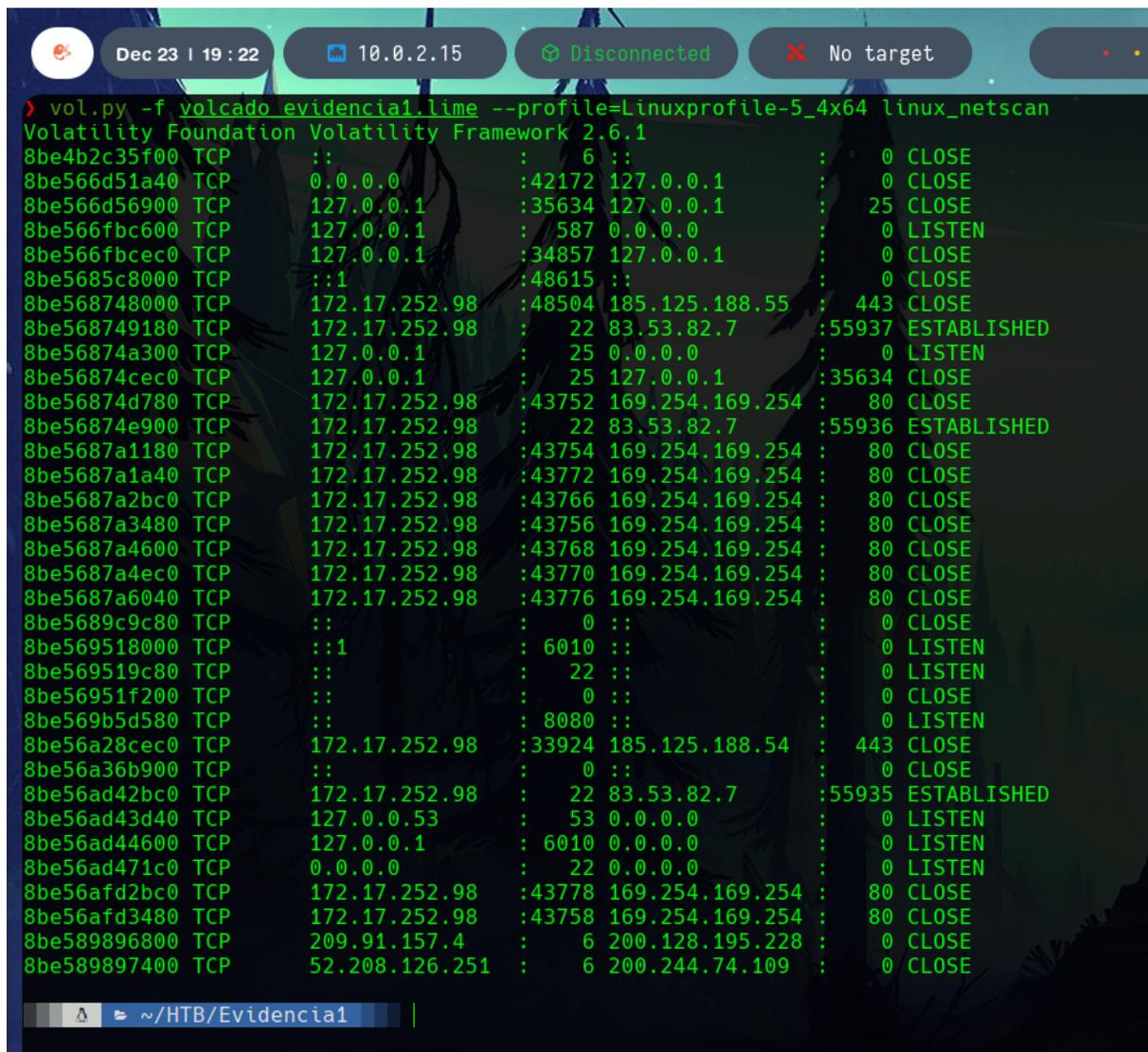
Figura 10: Procesos en ejecución.

4.4. Análisis de sockets.

Podemos utilizar algunas herramientas para recuperar información sobre las conexiones del sistema. En concreto, podemos usar *netscan* para realizar un escaneo TCP y UDP en la propia máquina. Esto nos permite identificar qué servicios pueden estar haciendo uso de determinadas conexiones. El comando sería el siguiente

```
vol.py -f volcado_evidencia1.lime --profile=Linuxprofile-5_4x64 linux_netscan
```

Y como podemos ver, existe una comunicación relativa al puerto 8080 que muy seguramente se trate de un servidor web.



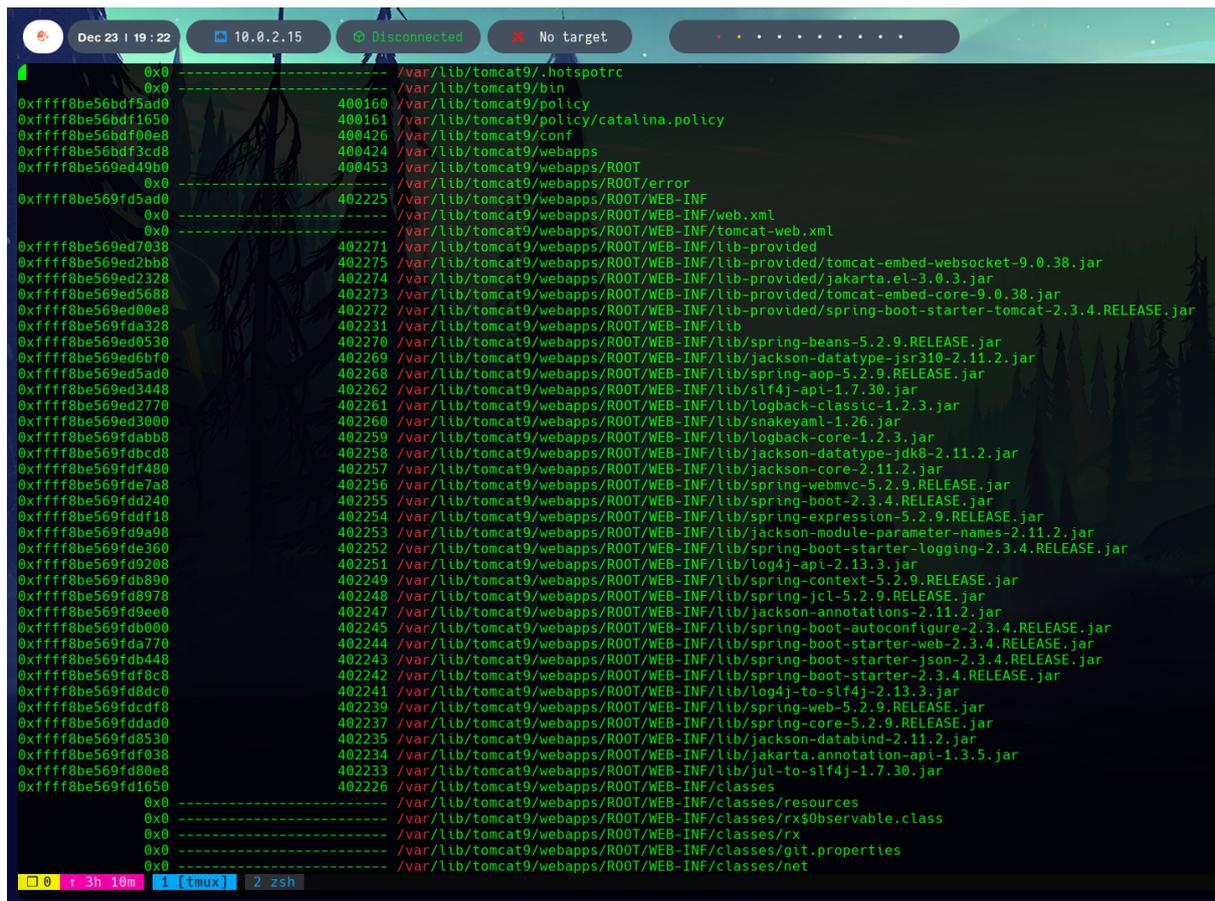
```
> vol.py -f volcado_evidencia1.lime --profile=Linuxprofile-5_4x64 linux_netscan
Volatility Foundation Volatility Framework 2.6.1
8be4b2c35f00 TCP      ::      : 6 ::      : 0 CLOSE
8be566d51a40 TCP      0.0.0.0 :42172 127.0.0.1 : 0 CLOSE
8be566d56900 TCP      127.0.0.1 :35634 127.0.0.1 : 25 CLOSE
8be566fbc600 TCP      127.0.0.1 : 587 0.0.0.0 : 0 LISTEN
8be566fbcec0 TCP      127.0.0.1 :34857 127.0.0.1 : 0 CLOSE
8be5685c8000 TCP      :::1    :48615 ::      : 0 CLOSE
8be568748000 TCP      172.17.252.98 :48504 185.125.188.55 : 443 CLOSE
8be568749180 TCP      172.17.252.98 : 22 83.53.82.7 :55937 ESTABLISHED
8be56874a300 TCP      127.0.0.1 : 25 0.0.0.0 : 0 LISTEN
8be56874cece0 TCP      127.0.0.1 : 25 127.0.0.1 :35634 CLOSE
8be56874d780 TCP      172.17.252.98 :43752 169.254.169.254 : 80 CLOSE
8be56874e900 TCP      172.17.252.98 : 22 83.53.82.7 :55936 ESTABLISHED
8be5687a1180 TCP      172.17.252.98 :43754 169.254.169.254 : 80 CLOSE
8be5687a1a40 TCP      172.17.252.98 :43772 169.254.169.254 : 80 CLOSE
8be5687a2bc0 TCP      172.17.252.98 :43766 169.254.169.254 : 80 CLOSE
8be5687a3480 TCP      172.17.252.98 :43756 169.254.169.254 : 80 CLOSE
8be5687a4600 TCP      172.17.252.98 :43768 169.254.169.254 : 80 CLOSE
8be5687a4ec0 TCP      172.17.252.98 :43770 169.254.169.254 : 80 CLOSE
8be5687a6040 TCP      172.17.252.98 :43776 169.254.169.254 : 80 CLOSE
8be5689c9c80 TCP      ::      : 0 ::      : 0 CLOSE
8be569518000 TCP      :::1    : 6010 ::      : 0 LISTEN
8be569519c80 TCP      ::      : 22 ::      : 0 LISTEN
8be56951f200 TCP      ::      : 0 ::      : 0 CLOSE
8be569b5d580 TCP      ::      : 8080 ::      : 0 LISTEN
8be56a28cec0 TCP      172.17.252.98 :33924 185.125.188.54 : 443 CLOSE
8be56a36b900 TCP      ::      : 0 ::      : 0 CLOSE
8be56ad42bc0 TCP      172.17.252.98 : 22 83.53.82.7 :55935 ESTABLISHED
8be56ad43d40 TCP      127.0.0.53 : 53 0.0.0.0 : 0 LISTEN
8be56ad44600 TCP      127.0.0.1 : 6010 0.0.0.0 : 0 LISTEN
8be56ad471c0 TCP      0.0.0.0 : 22 0.0.0.0 : 0 LISTEN
8be56afd2bc0 TCP      172.17.252.98 :43778 169.254.169.254 : 80 CLOSE
8be56afd3480 TCP      172.17.252.98 :43758 169.254.169.254 : 80 CLOSE
8be589896800 TCP      209.91.157.4 : 6 200.128.195.228 : 0 CLOSE
8be589897400 TCP      52.208.126.251 : 6 200.244.74.109 : 0 CLOSE
```

Figura 11: Sockets.

4.5. Recuperar un archivo.

Ahora podemos explorar el directorio `/var` para tratar de encontrar información relevante acerca del servidor web que podría estar alojándose en este sistema. Para ello, se puede utilizar el plugin `linux_enumerate_files` y filtrar por el directorio que estamos buscando de la siguiente manera.

```
vol.py -f volcado_evidencial.lime --profile=Linuxprofile-5_4x64  
→ linux_enumerate_files | grep /var
```



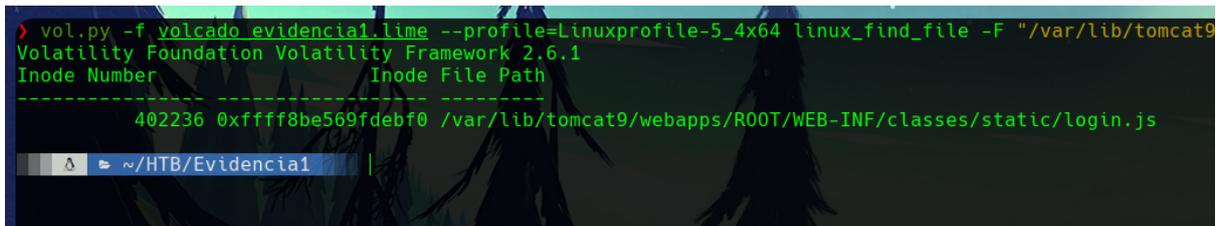
```
Dec 23 | 19:22 | 10.0.2.15 | Disconnected | No target  
-----  
0x0 /var/lib/tomcat9/.hotspotrc  
0x0 /var/lib/tomcat9/bin  
0xffff8be56bd15ad0 400160 /var/lib/tomcat9/policy  
0xffff8be56bd16500 400161 /var/lib/tomcat9/policy/catalina.policy  
0xffff8be56bd100e8 400426 /var/lib/tomcat9/conf  
0xffff8be56bd13cd8 400424 /var/lib/tomcat9/webapps  
0xffff8be569ed49b0 400453 /var/lib/tomcat9/webapps/ROOT  
0x0 /var/lib/tomcat9/webapps/ROOT/error  
0xffff8be569fd5ad0 402225 /var/lib/tomcat9/webapps/ROOT/WEB-INF  
0x0 /var/lib/tomcat9/webapps/ROOT/WEB-INF/web.xml  
0x0 /var/lib/tomcat9/webapps/ROOT/WEB-INF/tomcat-web.xml  
0xffff8be569ed7038 402271 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib-provided  
0xffff8be569ed2bb8 402275 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib-provided/tomcat-embed-websocket-9.0.38.jar  
0xffff8be569ed2328 402274 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib-provided/jakarta.el-3.0.3.jar  
0xffff8be569ed5688 402273 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib-provided/tomcat-embed-core-9.0.38.jar  
0xffff8be569ed00e8 402272 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib-provided/spring-boot-starter-tomcat-2.3.4.RELEASE.jar  
0xffff8be569fda328 402231 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib  
0xffff8be569ed0530 402270 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/spring-beans-5.2.9.RELEASE.jar  
0xffff8be569ed6bf0 402269 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/jackson-datatype-jsr310-2.11.2.jar  
0xffff8be569ed5ad0 402268 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/spring-aop-5.2.9.RELEASE.jar  
0xffff8be569ed3448 402262 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/slf4j-api-1.7.30.jar  
0xffff8be569ed2770 402261 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/logback-classic-1.2.3.jar  
0xffff8be569ed3000 402260 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/snakeyaml-1.26.jar  
0xffff8be569fdab8 402259 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/logback-core-1.2.3.jar  
0xffff8be569fdbc8 402258 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/jackson-datatype-jdk8-2.11.2.jar  
0xffff8be569fd480 402257 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/jackson-core-2.11.2.jar  
0xffff8be569fde7a8 402256 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/spring-webmvc-5.2.9.RELEASE.jar  
0xffff8be569fd240 402255 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/spring-boot-2.3.4.RELEASE.jar  
0xffff8be569fd18 402254 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/spring-expression-5.2.9.RELEASE.jar  
0xffff8be569fd9a98 402253 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/jackson-module-parameter-names-2.11.2.jar  
0xffff8be569fde360 402252 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/spring-boot-starter-logging-2.3.4.RELEASE.jar  
0xffff8be569fd9208 402251 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/log4j-api-2.13.3.jar  
0xffff8be569fdb890 402249 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/spring-context-5.2.9.RELEASE.jar  
0xffff8be569fd8978 402248 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/spring-jcl-5.2.9.RELEASE.jar  
0xffff8be569fd9e00 402247 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/jackson-annotations-2.11.2.jar  
0xffff8be569fdb000 402245 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/spring-boot-autoconfigure-2.3.4.RELEASE.jar  
0xffff8be569fda770 402244 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/spring-boot-starter-web-2.3.4.RELEASE.jar  
0xffff8be569fdb448 402243 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/spring-boot-starter-json-2.3.4.RELEASE.jar  
0xffff8be569fd8c0 402242 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/spring-boot-starter-2.3.4.RELEASE.jar  
0xffff8be569fd8dc0 402241 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/log4j-to-slf4j-2.13.3.jar  
0xffff8be569fcd4f8 402239 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/spring-web-5.2.9.RELEASE.jar  
0xffff8be569fd4d30 402237 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/spring-core-5.2.9.RELEASE.jar  
0xffff8be569fd8530 402235 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/jackson-databind-2.11.2.jar  
0xffff8be569fd6038 402234 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/jakarta.annotation-api-1.3.5.jar  
0xffff8be569fd80e8 402233 /var/lib/tomcat9/webapps/ROOT/WEB-INF/lib/jul-to-slf4j-1.7.30.jar  
0xffff8be569fd1650 402226 /var/lib/tomcat9/webapps/ROOT/WEB-INF/classes  
0x0 /var/lib/tomcat9/webapps/ROOT/WEB-INF/classes/resources  
0x0 /var/lib/tomcat9/webapps/ROOT/WEB-INF/classes/rx$observable.class  
0x0 /var/lib/tomcat9/webapps/ROOT/WEB-INF/classes/rx  
0x0 /var/lib/tomcat9/webapps/ROOT/WEB-INF/classes/git.properties  
0x0 /var/lib/tomcat9/webapps/ROOT/WEB-INF/classes/net
```

Figura 12: Directorio `/var`.

Gracias a este reconocimiento, podemos ver el directorio `/var/lib/tomcat9`, que se corresponde a un servidor web Tomcat. Podemos intentar recuperar algún archivo relevante del servidor.

Como sabemos que la máquina hosteaba un servidor de Tomcat, buscaremos el inode del archivo `login.js` en el volcado utilizando el plugin `linux_find_file` y la ruta del archivo `login.js`.

```
vol.py -f volcado_evidencia1.lime --profile=Linuxprofile-5_4x64  
↳ linux_find_file -F  
↳ "/var/lib/tomcat9/webapps/ROOT/WEB-INF/classes/static/login.js"
```



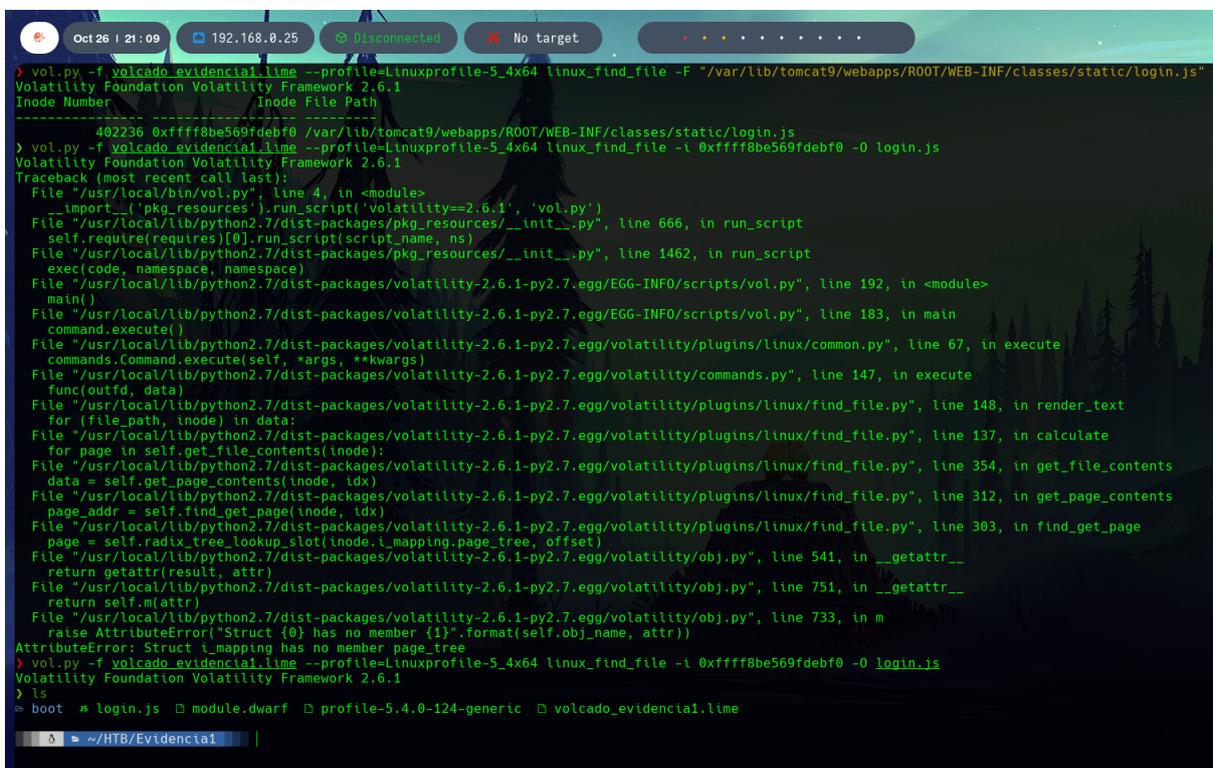
```
> vol.py -f volcado_evidencia1.lime --profile=Linuxprofile-5_4x64 linux_find_file -F "/var/lib/tomcat9/webapps/ROOT/WEB-INF/classes/static/login.js"  
Volatility Foundation Volatility Framework 2.6.1  
-----  
Inode Number      Inode File Path  
-----  
402236 0xffff8be569fdeb0 /var/lib/tomcat9/webapps/ROOT/WEB-INF/classes/static/login.js  
~/.HTB/Evidencia1
```

Figura 13: Inode del archivo `login.js` del servidor Tomcat.

Como se puede ver, obtenemos el número de inode y su dirección en hexadecimal. Esto nos permite utilizar el mismo plugin con la flag `-i` para recuperar el contenido del archivo, y la flag `-O` para indicar dónde queremos que lo guarde.

```
vol.py -f volcado_evidencia1.lime --profile=Linuxprofile-5_4x64  
↳ linux_find_file -i 0xffff8be569fdeb0 -O login.js
```

No obstante, al ejecutarlo obtuve el siguiente error.



```
> vol.py -f volcado_evidencia1.lime --profile=Linuxprofile-5_4x64 linux_find_file -F "/var/lib/tomcat9/webapps/ROOT/WEB-INF/classes/static/login.js"  
Volatility Foundation Volatility Framework 2.6.1  
-----  
Inode Number      Inode File Path  
-----  
402236 0xffff8be569fdeb0 /var/lib/tomcat9/webapps/ROOT/WEB-INF/classes/static/login.js  
> vol.py -f volcado_evidencia1.lime --profile=Linuxprofile-5_4x64 linux_find_file -i 0xffff8be569fdeb0 -O login.js  
Traceback (most recent call last):  
  File "/usr/local/bin/vol.py", line 4, in <module>  
    __import__('pkg_resources').run_script('volatility==2.6.1', 'vol.py')  
  File "/usr/local/lib/python2.7/dist-packages/pkg_resources/__init__.py", line 666, in run_script  
    self.require(requires)[0].run_script(script_name, ns)  
  File "/usr/local/lib/python2.7/dist-packages/pkg_resources/__init__.py", line 1462, in run_script  
    exec(code, namespace, namespace)  
  File "/usr/local/lib/python2.7/dist-packages/volatility-2.6.1-py2.7.egg/EGG-INF0/scripts/vol.py", line 192, in <module>  
    main()  
  File "/usr/local/lib/python2.7/dist-packages/volatility-2.6.1-py2.7.egg/EGG-INF0/scripts/vol.py", line 183, in main  
    command.execute()  
  File "/usr/local/lib/python2.7/dist-packages/volatility-2.6.1-py2.7.egg/volatility/plugins/linux/common.py", line 67, in execute  
    commands.Command.execute(self, *args, **kwargs)  
  File "/usr/local/lib/python2.7/dist-packages/volatility-2.6.1-py2.7.egg/volatility/commands.py", line 147, in execute  
    func(outfd, data)  
  File "/usr/local/lib/python2.7/dist-packages/volatility-2.6.1-py2.7.egg/volatility/plugins/linux/find_file.py", line 148, in render_text  
    for file_path, inode in data:  
  File "/usr/local/lib/python2.7/dist-packages/volatility-2.6.1-py2.7.egg/volatility/plugins/linux/find_file.py", line 137, in calculate  
    for page in self.get_file_contents(inode):  
  File "/usr/local/lib/python2.7/dist-packages/volatility-2.6.1-py2.7.egg/volatility/plugins/linux/find_file.py", line 354, in get_file_contents  
    data = self.get_page_contents(inode, idx)  
  File "/usr/local/lib/python2.7/dist-packages/volatility-2.6.1-py2.7.egg/volatility/plugins/linux/find_file.py", line 312, in get_page_contents  
    page_addr = self.find_get_page(inode, idx)  
  File "/usr/local/lib/python2.7/dist-packages/volatility-2.6.1-py2.7.egg/volatility/plugins/linux/find_file.py", line 303, in find_get_page  
    page = self.radix_tree_lookup_slot(inode.i_mapping.page_tree, offset)  
  File "/usr/local/lib/python2.7/dist-packages/volatility-2.6.1-py2.7.egg/volatility/obj.py", line 541, in __getattr__  
    return getattr(result, attr)  
  File "/usr/local/lib/python2.7/dist-packages/volatility-2.6.1-py2.7.egg/volatility/obj.py", line 751, in __getattr__  
    return self.m[attr]  
  File "/usr/local/lib/python2.7/dist-packages/volatility-2.6.1-py2.7.egg/volatility/obj.py", line 733, in m  
    raise AttributeError("Struct {0} has no member {1}".format(self.obj_name, attr))  
AttributeError: Struct i_mapping has no member page_tree  
> vol.py -f volcado_evidencia1.lime --profile=Linuxprofile-5_4x64 linux_find_file -i 0xffff8be569fdeb0 -O login.js  
Volatility Foundation Volatility Framework 2.6.1  
> ls  
boot # login.js  module.dwarf  profile-5.4.0-124-generic  volcado_evidencia1.lime  
~/.HTB/Evidencia1
```

Figura 14: Error y solución en Volatility.

Puede solucionarlo modificando el script `linux_find_file.py` a mano. En el repositorio de Volatility se encuentra el script actualizado. En la imagen anterior se ve al final la ejecución del mismo comando, pero sin ningún error.

Ahora podemos leer el archivo, aunque viene en hexadecimal porque proviene directamente del volcado.

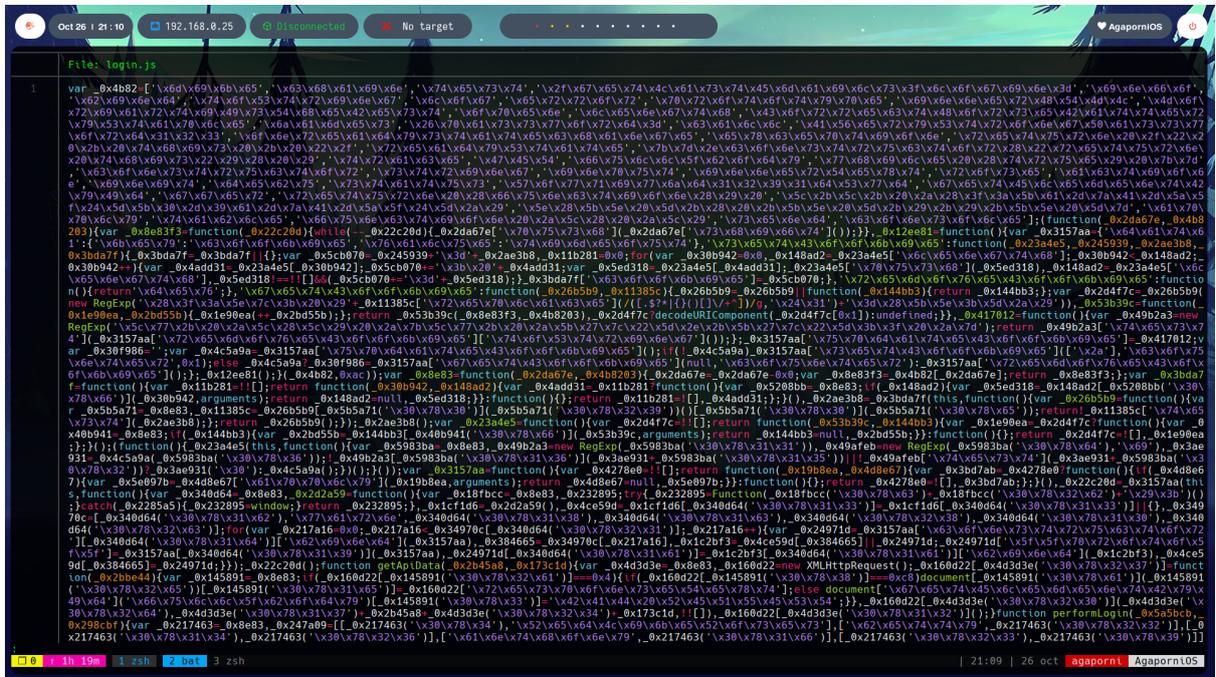


Figura 15: Archivo login.js

Es sencillo deducir que en la variable declarada al comienzo del archivo se almacenará toda la información relativa al login, pues contiene las variables necesarias del propio archivo js. Con un editor de texto podemos convertir correctamente el vector para quedarnos con las strings en hexadecimal.

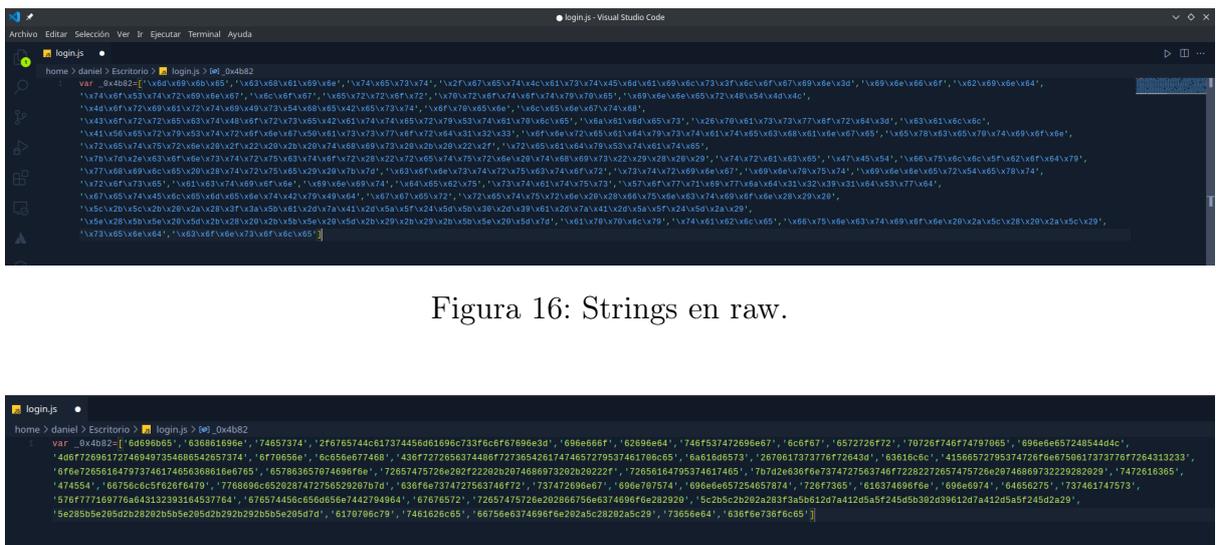


Figura 16: Strings en raw.

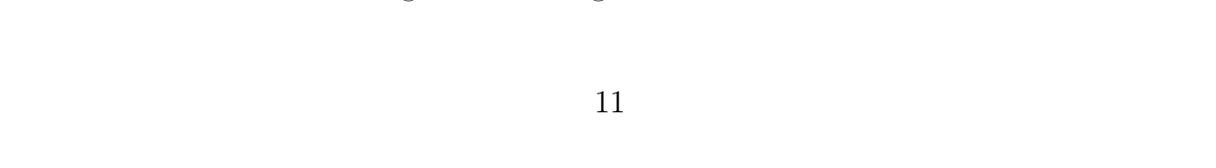
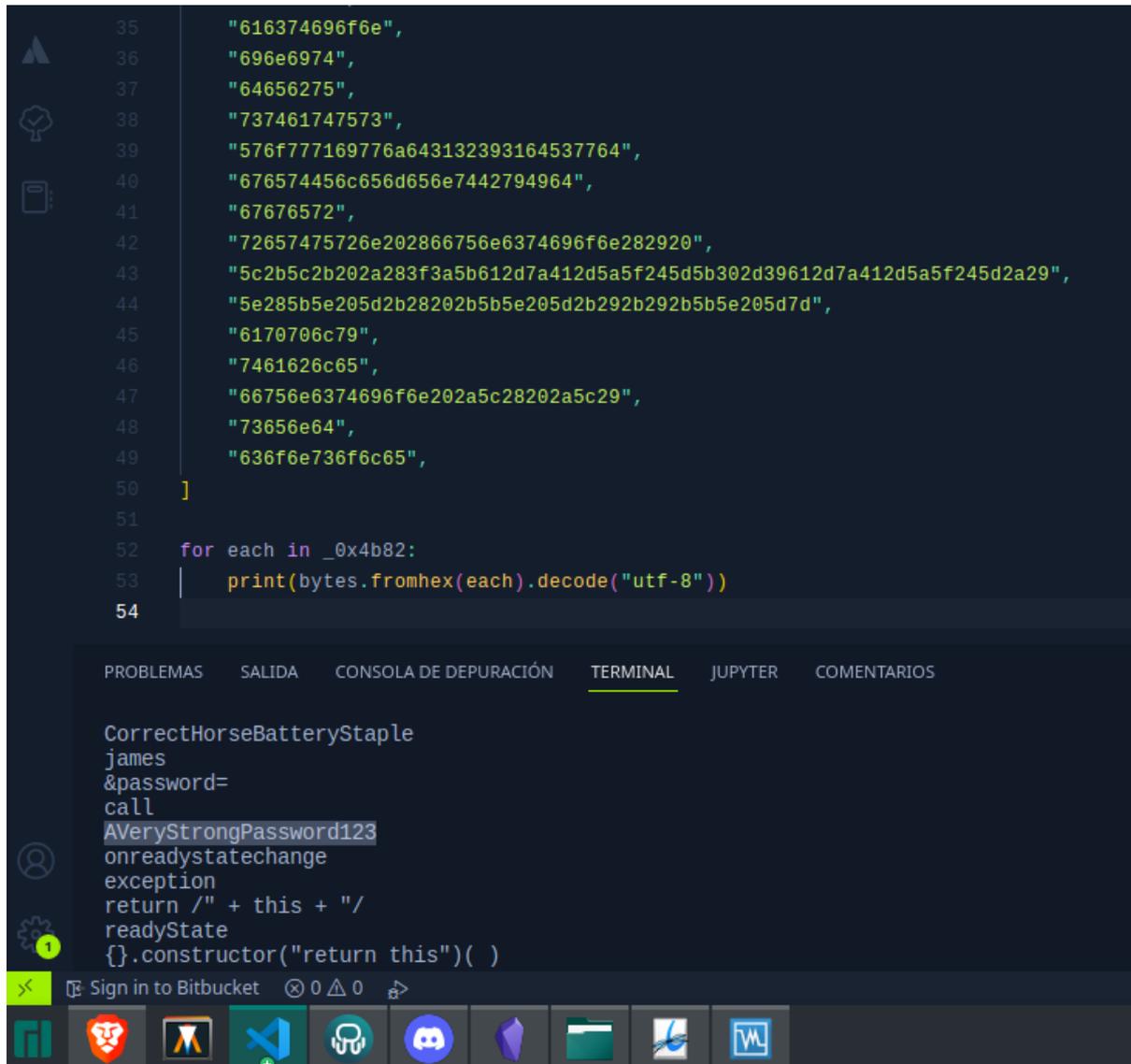


Figura 17: Strings en hexadecimal.

Hay muchas maneras de procesarlas, pero ya que tenía el VS code abierto y la sintaxis del vector es la misma que en Python, con un pequeño bucle podemos convertir todas las variables en hexadecimal a una string normal para leer su contenido. El script es el siguiente.

```
for each in _0x4b82:  
    print(bytes.fromhex(each).decode("utf-8"))
```

Y el resultado, como podemos ver, desvela que una de las variables es la contraseña del servidor web Tomcat.



```
35     "6163746966e",  
36     "696e6974",  
37     "64656275",  
38     "737461747573",  
39     "576f777169776a643132393164537764",  
40     "676574456c656d656e7442794964",  
41     "67676572",  
42     "72657475726e202866756e63746966e282920",  
43     "5c2b5c2b202a283f3a5b612d7a412d5a5f245d5b302d39612d7a412d5a5f245d2a29",  
44     "5e285b5e205d2b28202b5b5e205d2b292b292b5b5e205d7d",  
45     "6170706c79",  
46     "7461626c65",  
47     "66756e63746966e202a5c28202a5c29",  
48     "73656e64",  
49     "636f6e736f6c65",  
50 ]  
51  
52 for each in _0x4b82:  
53     print(bytes.fromhex(each).decode("utf-8"))  
54
```

CorrectHorseBatteryStaple
james
&password=
call
AVeryStrongPassword123
onreadystatechange
exception
return /" + this + "/
readyState
{}.constructor("return this")()

Figura 18: Contraseña de Tomcat.

Obviamente, esto es sólo un ejemplo didáctico de lo que se puede hacer con Volatility. Es una herramienta muy compleja y muy utilizada en análisis forense, ya que permite extraer pruebas periciales que sirven para demostrar qué ocurre o que ocurrió en un determinado sistema informático.