filecarving: Programa para encontrar archivos en un binario

La herramienta tiene varias opciones para poder utilizarla. En particular, la única verdaderamente necesaria es –f pues sin el archivo no se puede realizar ninguna acción.

```
root@debian:/home/lilium/Documents/filecarving# python filecarving.py --help
Usage: filecarving.py [options]

Options:
  -h, --help            show this help message and exit
  -f FILE_N, --file=FILE_N
                        Archivo para busqueda
  -c CONF, --config_file=CONF
                        Archivo de configuracion
  -d DIRECTORY, --directory=DIRECTORY
                        Directorio donde se guardaran los archivos encontrados
```

El archivo de configuración contiene comentarios con # y con *, solo que estos últimos ayudan a conocer la sección del tipo del archivo del que se trata.
Cada línea del archivo de configuración válida debe de iniciar por el código en hexadecimal de los bytes del magic number del tipo de imágenes a la que se refiere. Inmediatamente algún carácter de salto de línea por lo menos y una etiqueta “size=” seguida del tamaño que pude tener el archivo (Se pueden utilizar KB, MB o GB siempre y cuando se especifique con estas etiquetas). Después de otro salto de línea, se debe de especificar a qué tipo de archivo corresponde dicho magic number.

```
#configuration file
* Image : gif, png, jpg
#\x47\x49\x46\x38\x37\x61       size=10MB GIF
\x89\x50\x4e\x47\x0d\x0a\x1a\x0a size=1MB PNG
\xff\xd8\xff\xe0\x00\x10        size=200KB JPG
\xff\xd8\xff\xdb                size=200KB JPG
\xff\xd8\xff\xee                size=200KB JPG
* Compress : zip
\x50\x4b\x03\x04                size=10MB ZIP
* Executable : exe
\x4d\x5a\x90\x00                size=10MB EXE
~
```

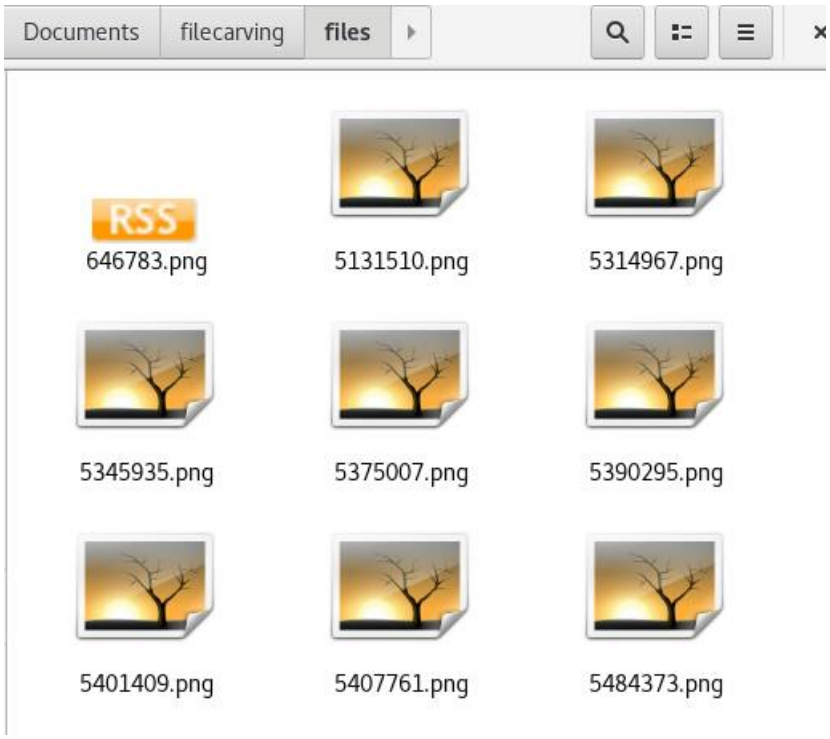
Para ejecutarlo, en este caso se crea un directorio llamado “files” donde se guardarán las imágenes (el de defecto es el actual). Se utiliza la captura de trafico juego.pcap como ejemplo y como archivo de configuración se escoge por defecto el filecarving.cnf que está en el mismo directorio.

```
root@debian:/home/lilium/Documents/filecarving# python filecarving.py -f juego.pcap -d files/
Found PNG Signature #1 at 646783
Found PNG Signature #2 at 5131510
Found PNG Signature #3 at 5314967
Found PNG Signature #4 at 5345935
Found PNG Signature #5 at 5375007
Found PNG Signature #6 at 5390295
Found PNG Signature #7 at 5401409
Found PNG Signature #8 at 5407761
Found PNG Signature #9 at 5484373
Found PNG Signature #10 at 5507033
Found PNG Signature #11 at 5526317
Found PNG Signature #12 at 5544793
Found PNG Signature #13 at 5557351
Found PNG Signature #14 at 5584004
Found PNG Signature #15 at 5622002
```

Una vez que terminó de recoger y guardar las imágenes se pueden ver dentro del directorio especificado

```
root@debian:/home/lilium/Documents/filecarving# ls files
5131510.png  5375007.png  5407761.png  5526317.png  5584004.png  5654394.png  5734624.png  6026510.png
5314967.png  5390295.png  5484373.png  5544793.png  5622002.png  5693050.png  5766151.png  6208546.png
5345935.png  5401409.png  5507033.png  5557351.png  5644417.png  5710663.png  5916605.png  646783.png
```

Sin embargo, se muestra que solo una que otra fue recolectada adecuadamente y se puede visualizar.



Comentario

Aunque funciona correctamente en el caso de las imágenes, hace falta que se pruebe de forma adecuada con más tipos y más archivos que contengan otros archivos para corregir el error. Esta prueba solo se hizo sobre una muestra de tráfico que a su vez contiene tanto petición como respuesta, por lo que el valor del magic numbres puede estar presente a pesar de que no se trate de un archivo de imagen, por eso hay tantas imágenes imposibles de visualizar (pues en un inicio nos e trata de imágenes). Sin embargo, cumple el cometido de encontrar muchas que sí lo son y recuperarlas.

Referencias

- https://www.devdungeon.com/content/working-binary-data-python#is-jpg
- https://asecuritysite.com/forensics/magic