SEGURIDAD EN SISTEMAS OPERATIVOS

4º Grado en Informática – Complementos de Ing. del Software

Curso 2017-18

Práctica [1]

Sesión [6]

Autor[1]: Iván Rodríguez Millán

Ejercicio 1.

En primer lugar mostramos el dispositivo y su partición con fdisk -l:

```
Disposit.  Inicio Comienzo   Final Sectores Tamaño Id Tipo
/dev/sdb1           7272 4095999  4088728     2G  6 FAT16
```

Seguido de esto desmontamos la unidad con:

umount /dev/sdb1

Después realizamos el cifrado completo de la partición sdb1:

```
[root@ivancito ivancito]# cryptsetup -c aes -h sha256 -y -s 256 luksFormat /dev/sdb1

WARNING!
========
Esto sobreescribirá los datos en /dev/sdb1 de forma irrevocable.

Are you sure? (Type uppercase yes): YES
Introduzca la frase contraseña:
Verifique la frase contraseña:
```

Una vez esté cifrado, abrimos el dispositivo con la aplicación cryptsetup y lo montamos:

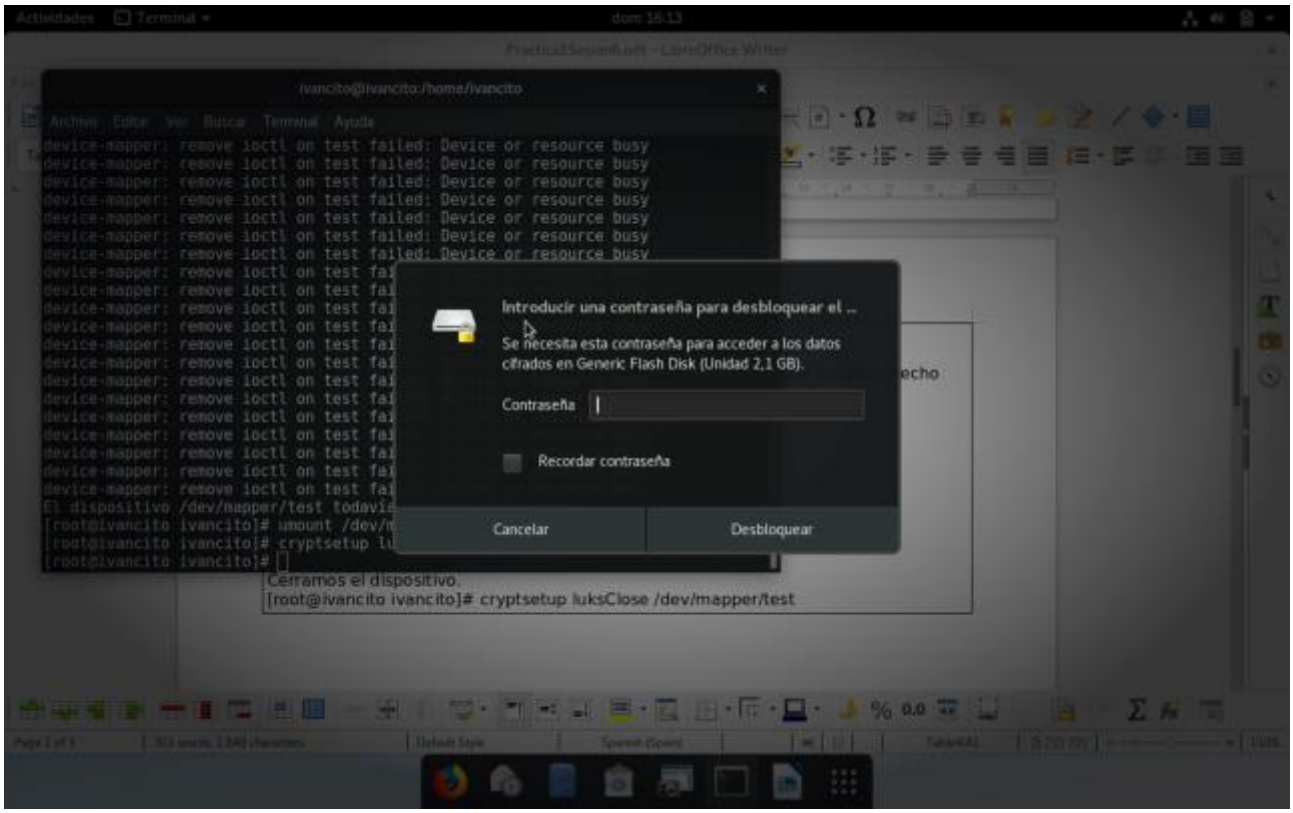
```
[root@ivancito ivancito]# cryptsetup luksOpen /dev/sdb1 test
Introduzca la frase contraseña de /dev/sdb1:
```

[1] Como autor declaro que los contenidos del presente documento son originales y elaborados por mi. De no cumplir con este compromiso, soy consciente de que, de acuerdo con la "Normativa de evaluación y de calificaciones de los estudiantes de la Universidad de Granada" esto "conllevará la calificación numérica de cero … independientemente del resto de calificaciones que el estudiante hubiera obtenido ..."

En los siguientes pasos creamos el sistema de ficheros, montamos el USB usando la unidad mapeada y comprobamos que la unidad se ha montado correctamente copiando un fichero .txt en ella. Después desmontamos el dispositivo y lo cerramos.

```
[root@ivancito ivancito]# mkfs /dev/mapper/test
mke2fs 1.43.4 (31-Jan-2017)
Se está creando un sistema de ficheros con 510579 bloques de 4k y 127744 nodos-i
UUID del sistema de ficheros: 2b172564-d205-42fb-aef5-0996a3e2614b
Respaldo del superbloque guardado en los bloques:
        32768, 98304, 163840, 229376, 294912

Reservando las tablas de grupo: hecho
Escribiendo las tablas de nodos-i: hecho
Escribiendo superbloques y la información contable del sistema de ficheros: hecho

Montamos el USB.
[root@ivancito ivancito]# mount /dev/mapper/test /mnt/prueba

Copiamos un documento a la carpeta prueba.
[root@ivancito Documentos]# cp ficheroIvan.txt /mnt/prueba/

Desmontamos el dispositivo.
[root@ivancito ivancito]# umount /dev/mapper/test

Cerramos el dispositivo.
[root@ivancito ivancito]# cryptsetup luksClose /dev/mapper/test
```

Para finalizar, extraemos el pendrive y volvemos a insertarlo para comprobar que todo ha ido correctamente. Para verificar que todo ha ido bien, se muestra la siguiente imagen que nos sale al insertar el pendrive.



Ejercicio 2.

Una vez tenemos instalado el paquete STEGHIDE procedemos con los siguientes pasos.

Vamos a ocultar el siguiente mensaje que se encuentra en el fichero “ficheroIvan.txt”:

Esto es un ejercicio de SSO

Para ello en primer lugar vamos a ocultar el archivo con la orden:

```
[root@ivancito Documentos]# steghide embed -cf pingu.jpg -ef ficheroIvan.txt
Anotar salvoconducto:
Re-ingresar salvoconducto:
adjuntando "ficheroIvan.txt" en "pingu.jpg"... hecho
```

Nos ha pedido introducir una palabra, en nuestro caso ha sido prueba.

Después de esto podremos extraer el archivo con la siguiente orden:

```
[root@ivancito Documentos]# steghide extract -sf pingu.jpg
Anotar salvoconducto:
anot<U+FFFD> los datos extra<U+FFFD>dos e/"ficheroIvan.txt".
```

Para finalizar como diferencias podemos ver que el nuevo portador tiene mayor tamaño, esto se puede comprobar con la orde du

```
[root@ivancito Documentos]# du -sh pingu.jpg
32K     pingu.jpg
[root@ivancito Documentos]# du -sh ../Descargas/pingu.jpg
28K     ../Descargas/pingu.jpg
```

Otra cosa que podemos hacer es comparar la firma usando md5sum

```
[root@ivancito Documentos]# md5sum ../Descargas//pingu.jpg
75cfd764ce424ade5b02634623a1f044  ../Descargas//pingu.jpg

[root@ivancito Documentos]# md5sum pingu.jpg
dcdf3a24fa752838cdb909785f1d36f5  pingu.jpg
```

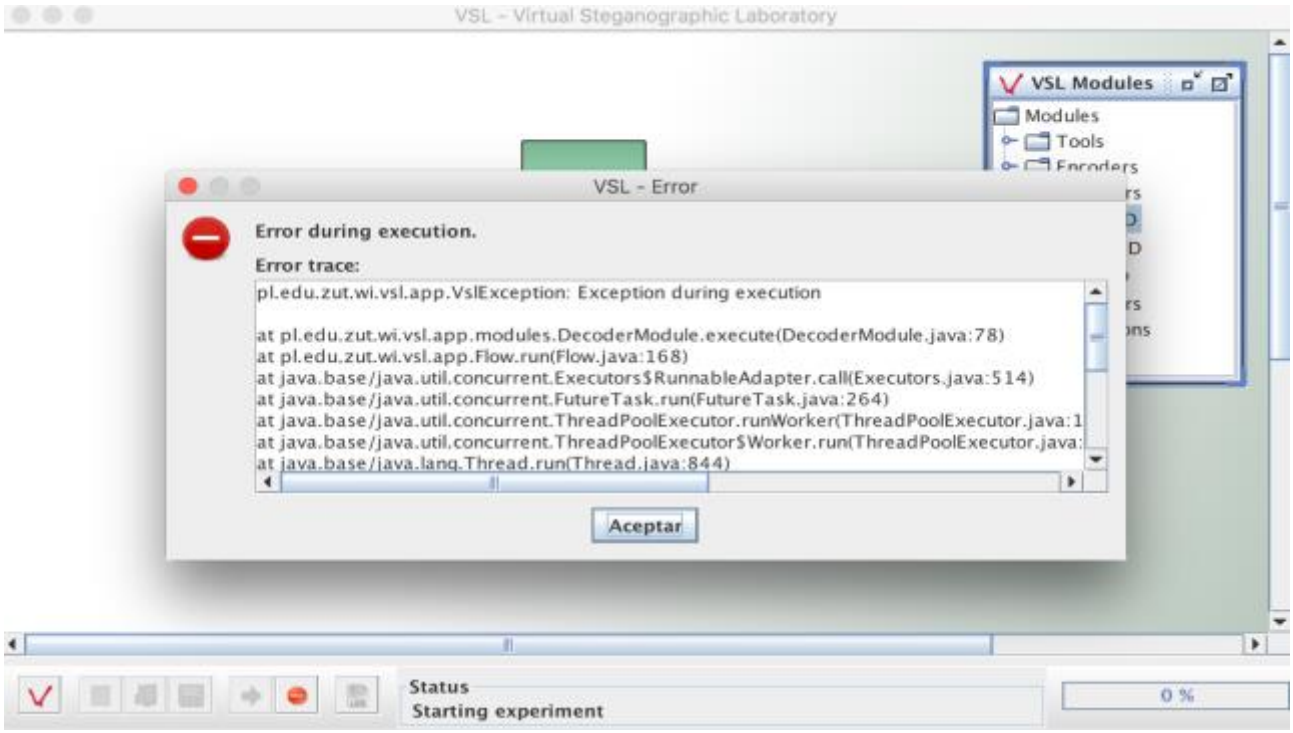
Vemos que como era de preveer la firma es diferente. Esto nos da un indicativo de que ha sido modificada la imagen.

Ejercicio 3.

Esta parte se va a realizar fuera de la máquina virtual de Fedora 26, por el hecho de que me da fallo al ejecutar el .jar. Por ello la voy a realizar en el SO macOS High Sierra.

También cabe matizar que se produce un error en el propio programa descargado si intentamos descifrar el mensaje oculto en la imagen, habiendo cifrado esta con los pasos del ejercicio anterior.

El error es el siguiente:

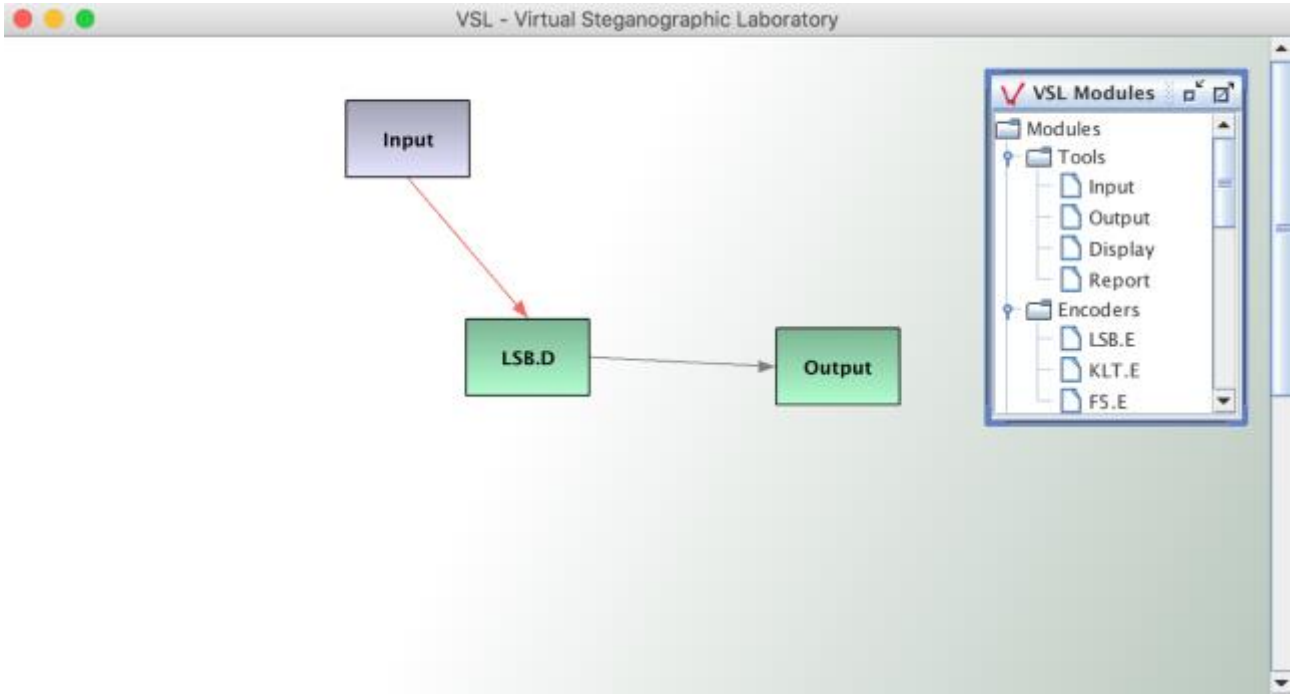


Por ello para mostrar como funciona el programa he optado por cifrar el mensaje con la herramienta y descifrarlo con la misma ya que lo veía mejor que no hacer nada. Por ello en primer lugar ciframos con la herramienta el mensaje en la imagen y procedemos con los siguientes pasos.

Una vez tenemos descargado el programa **visual steganalysis laboratory** lo ejecutamos con la orden:

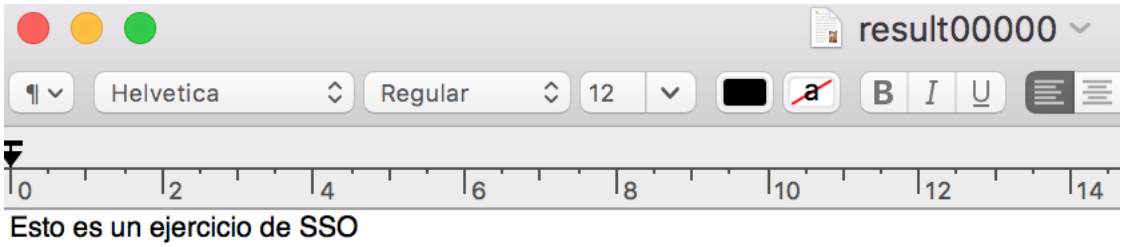
java -jar vsl-app-1.1.jar

Para hacer el proceso inverso al del paso 2 y obtener el fichero hacemos lo siguiente:



En donde input representa al nodo que carga la imagen, el nodo LSB.D que sirve para hacer el paso inverso, y por último el nodo output sirve para indicar la ruta donde queremos sacar el fichero de resultados.

Una vez ejecutado obtenemos el siguiente fichero:



Bibliografía

http://manpages.ubuntu.com/manpages/yakkety/es/man1/du.1.html

http://www.ubuntu-es.org/node/4377 - .WhrkQPaCE1I

http://steghide.sourceforge.net/

https://help.ubuntu.com/community/HowToMD5SUM

http://vsl.sourceforge.net/