

# SEGURIDAD EN SISTEMAS OPERATIVOS

4º Grado en Informática – Complementos de Ing. del Software  
Curso 2019-20

**Práctica [1].** Administración de la seguridad en Linux.

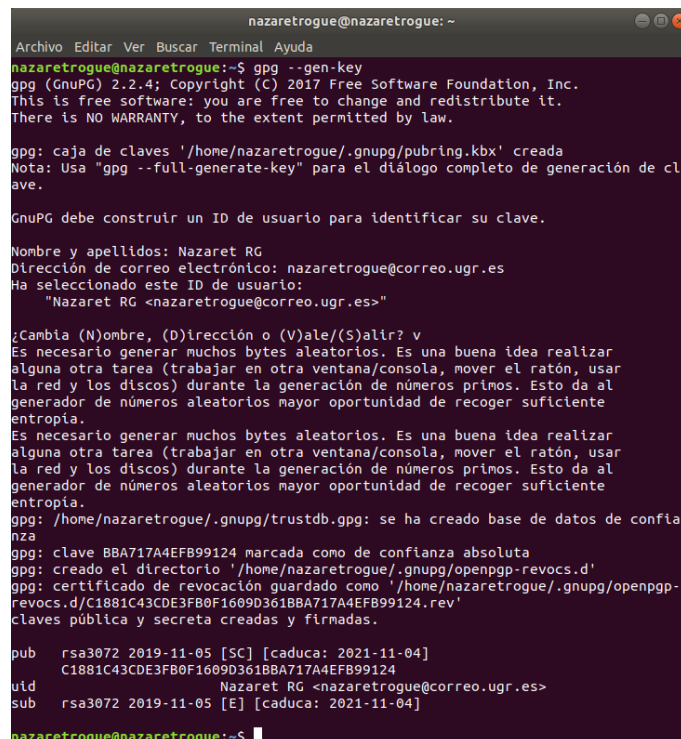
**Sesión [5].** Cifrado de archivos.

**Autor<sup>1</sup>:** Nazaret Román Guerrero

## Ejercicio 1.

- a) Utiliza la herramienta gpg para crear unas claves personales.
  - b) Una vez creadas, utilízalas para cifrar y descifrar un archivo.
  - c) Agrúpate con un compañero e intercambiar un archivo cifrado por cada uno que el otro debe descifrar.
- a) Antes de comenzar, podemos comprobar si tenemos instalado gpg con el comando `gpg --version`. En el sistema que estoy utilizando está instalado.

Para generar la clave utilizamos el comando `gpg --gen-key`, como se muestra en la imagen.



```
nazaretroque@nazaretroque: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
nazaretroque@nazaretroque:~$ gpg --gen-key  
gpg (GnuPG) 2.2.4; Copyright (C) 2017 Free Software Foundation, Inc.  
This is free software: you are free to change and redistribute it.  
There is NO WARRANTY, to the extent permitted by law.  
  
gpg: caja de claves '/home/nazaretroque/.gnupg/pubring.kbx' creada  
Nota: Usa "gpg --full-generate-key" para el diálogo completo de generación de clave.  
  
GnuPG debe construir un ID de usuario para identificar su clave.  
  
Nombre y apellidos: Nazaret RG  
Dirección de correo electrónico: nazaretroque@correo.ugr.es  
Ha seleccionado este ID de usuario:  
"Nazaret RG <nazaretroque@correo.ugr.es>"  
  
¿Cambia (N)ombre, (D)irección o (V)ale/(S)alir? v  
Es necesario generar muchos bytes aleatorios. Es una buena idea realizar alguna otra tarea (trabajar en otra ventana/console, mover el ratón, usar la red y los discos) durante la generación de números primos. Esto da al generador de números aleatorios mayor oportunidad de recoger suficiente entropía.  
Es necesario generar muchos bytes aleatorios. Es una buena idea realizar alguna otra tarea (trabajar en otra ventana/console, mover el ratón, usar la red y los discos) durante la generación de números primos. Esto da al generador de números aleatorios mayor oportunidad de recoger suficiente entropía.  
gpg: /home/nazaretroque/.gnupg/trustdb.gpg: se ha creado base de datos de confianza  
gpg: clave BBA717A4EFB99124 marcada como de confianza absoluta  
gpg: creado el directorio '/home/nazaretroque/.gnupg/openpgp-revocs.d'  
gpg: certificado de revocación guardado como '/home/nazaretroque/.gnupg/openpgp-revocs.d/C1881C43CDE3FB0F1609D361BBA717A4EFB99124.rev'  
claves pública y secreta creadas y firmadas.  
  
pub   rsa3072 2019-11-05 [SC] [caduca: 2021-11-04]  
      C1881C43CDE3FB0F1609D361BBA717A4EFB99124  
uid           Nazaret RG <nazaretroque@correo.ugr.es>  
sub   rsa3072 2019-11-05 [E] [caduca: 2021-11-04]  
nazaretroque@nazaretroque:~$
```

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 ...”

Nos pedirá un nombre y correo electrónico para dar un ID a la clave, una contraseña. Por defecto el algoritmo utilizado es el RSA, como se puede observar en la antepenúltima línea, donde indica el algoritmo; como se puede leer en el segundo párrafo, si queremos que salga el diálogo completo en el que también nos pregunta el algoritmo de cifrado hay que utilizar otra opción extra en el comando: `--full-generate-key` (se muestra en el apartado c).

- b) Vamos a crear un archivo, `archivo.txt`, que contenga un mensaje. Para ello usamos la orden `echo` y creamos el mensaje, tal y como se ve aquí:

```
nazaretroque@nazaretroque: ~
Archivo Editar Ver Buscar Terminal Ayuda
nazaretroque@nazaretroque:~$ echo "Vamos a cifrar un mensaje" >> archivo.txt
nazaretroque@nazaretroque:~$ cat archivo.txt
Vamos a cifrar un mensaje
nazaretroque@nazaretroque:~$
```

Tras crear el archivo, ciframos con la clave que hemos generado antes. El destinatario del mensaje es el correo que está relacionado con la clave pública con la que vamos a cifrar, es decir, en este caso el mío.

```
nazaretroque@nazaretroque: ~
Archivo Editar Ver Buscar Terminal Ayuda
nazaretroque@nazaretroque:~$ gpg --armor --recipient nazaretroque@correo.ugr.es --encrypt archivo.txt
gpg: comprobando base de datos de confianza
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: nivel: 0 validez: 1 firmada: 0 confianza: 0-, 0q, 0n, 0m, 0f, 1u
gpg: siguiente comprobación de base de datos de confianza el: 2021-11-04
nazaretroque@nazaretroque:~$
```

Ahora que lo hemos cifrado, procedemos a descifrarlo. Primero comprobamos que está el archivo cifrado. En efecto, hay un archivo llamado `archivo.txt.asc` que es cifrado que se ha creado con la orden anterior. Ahora desciframos. Para ello se necesita la clave privada de la persona a la que iba dirigido el archivo. Utilizando la orden `gpg --decrypt archivo.txt.asc`, se nos pedirá la contraseña de la clave privada, y una vez que la introduzcamos, se descifrá y saldrá el mensaje en pantalla:

```
nazaretroque@nazaretroque: ~
Archivo Editar Ver Buscar Terminal Ayuda
nazaretroque@nazaretroque:~$ ls -l
total 52
-rw-r--r-- 1 nazaretroque nazaretroque 26 nov 6 11:13 archivo.txt
-rw-r--r-- 1 nazaretroque nazaretroque 736 nov 6 11:15 archivo.txt.asc
drwxr-xr-x 2 nazaretroque nazaretroque 4096 sep 20 13:18 Descargas
drwxr-xr-x 2 nazaretroque nazaretroque 4096 sep 20 13:18 Documentos
drwxr-xr-x 2 nazaretroque nazaretroque 4096 sep 20 13:18 Escritorio
-rw-r--r-- 1 nazaretroque nazaretroque 8980 sep 20 13:02 examples.desktop
drwxr-xr-x 2 nazaretroque nazaretroque 4096 sep 20 13:18 Imágenes
drwxr-xr-x 2 nazaretroque nazaretroque 4096 sep 20 13:18 Música
drwxr-xr-x 2 nazaretroque nazaretroque 4096 sep 20 13:18 Plantillas
drwxr-xr-x 2 nazaretroque nazaretroque 4096 sep 20 13:18 Público
drwxr-xr-x 2 nazaretroque nazaretroque 4096 sep 20 13:18 Vídeos
nazaretroque@nazaretroque:~$ gpg --decrypt archivo.txt.asc
gpg: cifrado con clave de 3072 bits RSA, ID 3EAF1FD5ADF383E4, creada el 2019-11-05
"Nazaret RG <nazaretroque@correo.ugr.es>"
Vamos a cifrar un mensaje
nazaretroque@nazaretroque:~$
```

- c) Para llevar a cabo este ejercicio, es necesario crear un nuevo usuario, pepe, que generará las claves pública y privada y luego enviará la clave pública a mi usuario personal para cifrar el archivo.

Tras crear el usuario con `adduser pepe` y logearnos en la sesión de Pepe con `login pepe`, generamos las claves. Esta vez mostramos el procedimiento completo de generación de la clave: con el algoritmo RSA, con tamaño 1024 y que no caduca nunca. La persona a la que pertenece la llave es Pepe Pepito.

```
pepe@nazaretroque: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
pepe@nazaretroque:~$ gpg --full-gen-key  
gpg (GnuPG) 2.2.4; Copyright (C) 2017 Free Software Foundation, Inc.  
This is free software: you are free to change and redistribute it.  
There is NO WARRANTY, to the extent permitted by law.  
  
gpg: caja de claves '/home/pepe/.gnupg/pubring.kbx' creada  
Por favor seleccione tipo de clave deseado:  
  (1) RSA y RSA (por defecto)  
  (2) DSA y ElGamal  
  (3) DSA (sólo firmar)  
  (4) RSA (sólo firmar)  
Su elección: 1  
Las claves RSA pueden tener entre 1024 y 4096 bits de longitud.  
¿De qué tamaño quiere la clave? (3072) 1024  
El tamaño requerido es de 1024 bits  
Por favor, especifique el período de validez de la clave.  
  0 = la clave nunca caduca  
  <n> = la clave caduca en n días  
  <n>w = la clave caduca en n semanas  
  <n>m = la clave caduca en n meses  
  <n>y = la clave caduca en n años  
¿Validez de la clave (0)? 0  
La clave nunca caduca  
¿Es correcto? (s/n) s  
  
GnuPG debe construir un ID de usuario para identificar su clave.  
Nombre y apellidos: Pepe Pepito  
Dirección de correo electrónico: pepito@gmail.com  
Comentario:  
Ha seleccionado este ID de usuario:  
  "Pepe Pepito <pepito@gmail.com>"  
  
¿Cambia (N)ombre, (C)omentario, (D)irección o (V)ale/(S)alir? v  
Es necesario generar muchos bytes aleatorios. Es una buena idea realizar  
alguna otra tarea (trabajar en otra ventana/consola, mover el ratón, usar  
la red y los discos) durante la generación de números primos. Esto da al  
generador de números aleatorios mayor oportunidad de recoger suficiente  
entropía.  
Es necesario generar muchos bytes aleatorios. Es una buena idea realizar  
alguna otra tarea (trabajar en otra ventana/consola, mover el ratón, usar  
la red y los discos) durante la generación de números primos. Esto da al  
generador de números aleatorios mayor oportunidad de recoger suficiente  
entropía.  
gpg: /home/pepe/.gnupg/trustdb.gpg: se ha creado base de datos de confianza  
gpg: clave ED2040B6D7A6876F marcada como de confianza absoluta  
gpg: creado el directorio '/home/pepe/.gnupg/openpgp-revocs.d'  
gpg: certificado de revocación guardado como '/home/pepe/.gnupg/openpgp-revocs.d/10AAA3939637  
D500F163FF24ED2040B6D7A6876F.rev'
```

Ahora hay que exportar la clave pública a un archivo para enviársela a la persona que va a cifrar el mensaje, en este caso, a mi usuario personal. Se puede enviar por cualquier canal seguro, por ejemplo, telegram.

```
pepe@nazaretroque: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
pepe@nazaretroque:~$ gpg --armor --export --output pepe-pubkey.gpg pepito  
pepe@nazaretroque:~$ ls -l  
total 16  
-rw-r--r-- 1 pepe pepe 8980 nov  6 20:58 examples.desktop  
-rw-rw-r-- 1 pepe pepe 1042 nov  6 21:11 pepe-pubkey.gpg  
pepe@nazaretroque:~$
```

Desde mi usuario personal, ahora podemos ver que tenemos la clave pública de Pepe.

```
nazaretroque@nazaretroque: ~
Archivo Editar Ver Buscar Terminal Ayuda
nazaretroque@nazaretroque:~$ ls -l
total 64
-rw-r--r-- 1 nazaretroque nazaretroque  26 nov  6 21:16 archivo_c.txt
-rw-r--r-- 1 nazaretroque nazaretroque 386 nov  6 21:16 archivo_c.txt.asc
-rw-r--r-- 1 nazaretroque nazaretroque  26 nov  6 11:13 archivo.txt
-rw-r--r-- 1 nazaretroque nazaretroque 736 nov  6 11:15 archivo.txt.asc
drwxr-xr-x 2 nazaretroque nazaretroque 4096 nov  6 21:47 Descargas
drwxr-xr-x 2 nazaretroque nazaretroque 4096 sep 20 13:18 Documentos
drwxr-xr-x 2 nazaretroque nazaretroque 4096 sep 20 13:18 Escritorio
-rw-r--r-- 1 nazaretroque nazaretroque 8980 sep 20 13:02 examples.desktop
drwxr-xr-x 2 nazaretroque nazaretroque 4096 sep 20 13:18 Imágenes
drwxr-xr-x 2 nazaretroque nazaretroque 4096 sep 20 13:18 Música
-rw-rw-r-- 1 nazaretroque nazaretroque 1042 nov  6 21:47 pepe-pubkey.gpg
drwxr-xr-x 2 nazaretroque nazaretroque 4096 sep 20 13:18 Plantillas
drwxr-xr-x 2 nazaretroque nazaretroque 4096 sep 20 13:18 Público
drwxr-xr-x 2 nazaretroque nazaretroque 4096 sep 20 13:18 Vídeos
nazaretroque@nazaretroque:~$
```

Importamos dicha clave a nuestro sistema para poder encriptar ficheros con ella.

```
nazaretroque@nazaretroque: ~
Archivo Editar Ver Buscar Terminal Ayuda
nazaretroque@nazaretroque:~$ gpg --import pepe-pubkey.gpg
gpg: clave ED2040B6D7A6876F: clave pública "Pepe Pepito <pepito@gmail.com>" impo
rtada
gpg: Cantidad total procesada: 1
gpg:          importadas: 1
nazaretroque@nazaretroque:~$
```

Una vez que tenemos la clave, procedemos a crear y mostrar el archivo.

```
nazaretroque@nazaretroque: ~
Archivo Editar Ver Buscar Terminal Ayuda
nazaretroque@nazaretroque:~$ echo "Archivo cifrado para Pepe" >> archivo_c.txt
nazaretroque@nazaretroque:~$ cat archivo_c.txt
Archivo cifrado para Pepe
nazaretroque@nazaretroque:~$
```

Ahora lo encriptamos, igual que en el segundo apartado.

```
nazaretroque@nazaretroque: ~
Archivo Editar Ver Buscar Terminal Ayuda
nazaretroque@nazaretroque:~$ gpg --armor --recipient pepito@gmail.com --encrypt
archivo_c.txt
gpg: D38BC7DFCC7525A0: No hay seguridad de que esta clave pertenezca realmente
al usuario que se nombra
sub rsa1024/D38BC7DFCC7525A0 2019-11-06 Pepe Pepito <pepito@gmail.com>
Huella clave primaria: 10AA A393 9637 D500 F163 FF24 ED20 40B6 D7A6 876F
Huella de subclave: F716 EADA 5086 CA39 C894 DF62 D38B C7DF CC75 25A0

No es seguro que la clave pertenezca a la persona que se nombra en el
identificador de usuario. Si *realmente* sabe lo que está haciendo,
puede contestar sí a la siguiente pregunta.

¿Usar esta clave de todas formas? (s/N) s
nazaretroque@nazaretroque:~$
```

Ahora enviamos el archivo encriptado a Pepe y lo desciframos. Como podemos ver, el archivo se desencripta sin problemas y nos muestra el contenido.

```
pepe@nazaretroque: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
pepe@nazaretroque:~$ gpg --decrypt archivo_c.txt.asc  
gpg: cifrado con clave de 1024 bits RSA, ID D38BC7DFCC7525A0, creada el 2019-11-06  
"Pepe Pepito <pepito@gmail.com>"  
Archivo cifrado para Pepe  
pepe@nazaretroque:~$
```

## Ejercicio 2.

Utiliza la herramienta `openssl` para cifrar y descifrar un archivo con un algoritmo y clave de tu elección.

Primero creamos un archivo como venimos haciendo hasta ahora, llamado `archivo_openssl.txt`. Creamos las claves privada y pública respectivamente con los comandos que se ven en la imagen siguiente:

```
nazaretroque@nazaretroque: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
nazaretroque@nazaretroque:~$ openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:2048 -pkeyopt rsa_keygen_pubexp:3 -out privkey-nazaret.pem  
.....+++  
.....+++  
nazaretroque@nazaretroque:~$ openssl pkey -in privkey-nazaret.pem -out pubkey-nazaret.pem -pubout  
nazaretroque@nazaretroque:~$
```

Una vez tenemos las llaves, ciframos el archivo que habíamos creado con el comando que se ve en la imagen. Tras esto mostramos el contenido del archivo.

```
nazaretroque@nazaretroque: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
nazaretroque@nazaretroque:~$ openssl pkeyuti -encrypt -in archivo_openssl.txt -p  
ubin -inkey pubkey-nazaret.pem -out archivo_openssl_cifrado.bin  
nazaretroque@nazaretroque:~$ cat archivo_openssl_cifrado.bin  
tg000((0&q.0N0000V000 /韮[L0000000D0(00t0f0000RN000Y00000)03$000i]00060  
/$00000zX0000(0j000000m000)0%0=00P000000Y0]?00G00000KR0bFA7E0v00000A100X0  
nazaretroque@nazaretroque:~$ K000uhh00800R[g0000000/0000qdAqF_io<00'000A  
nazaretroque@nazaretroque:~$
```

Ahora descriptaremos el archivo para poder ver el mensaje que tenemos, tal y como se puede observar en la imagen de abajo.

```
nazaretroque@nazaretroque: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
nazaretroque@nazaretroque:~$ openssl pkeyutil -decrypt -in archivo_openssl_cifrado.bin -inkey privkey-nazaret.pem -out texto_descifrado.txt  
nazaretroque@nazaretroque:~$ cat texto_descifrado.txt  
Vamos a cifrar el mensaje con openssl  
nazaretroque@nazaretroque:~$
```



### Ejercicio 3.

Busca información sobre la vulnerabilidad del Heartbleed para contestar a las siguientes cuestiones:

- a) ¿En qué consiste la misma?
  - b) ¿Cómo saber si nuestro sistema la sufre?
  - c) ¿Cómo podemos subsanarla?
- a) Heartbleed es una vulnerabilidad de openssl con la que se puede leer la memoria de todos los sistemas protegidos con versiones de openssl que sufren dicho bug. Esto compromete las claves secretas de los usuarios, por lo que los archivos encriptados, las contraseñas y los usuarios quedan expuestos.

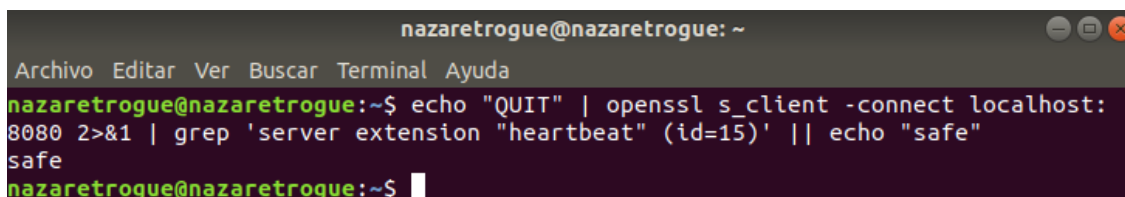
Esta vulnerabilidad está presente en la versión 1.0.1 y 1.0.2-beta de la biblioteca de openssl. Los atacantes utilizaban esta vulnerabilidad desde 5 meses antes de que fuera detectada. Fue descubierta en abril de 2014 y parcheada en menos de una semana, pero los daños causados ya eran considerables pues llevaba presente desde 2012 aproximadamente. No obstante, hace poco salieron un par de artículos donde se mostraba que a día de hoy, 5 años después de que se parcheara la vulnerabilidad, sigue habiendo sistemas que la sufren.

- b) Para saber si nuestro sistema sufre la vulnerabilidad hay diferentes formas de hacerlo. Existen diversas herramientas, por ejemplo [SSL Labs](#) de Qualys; no obstante, existe una forma de comprobarlo utilizando el propio SSL. Para ello, se ejecuta el comando:

```
echo "QUIT"|openssl s_client -connect localhost:8080 2>&1|grep 'server  
extension "heartbeat" (id=15)' || echo safe
```

Este comando significa que nos conectamos a la dirección que deseamos comprobar con la opción -connect, en este caso localhost en el puerto 8080. Si encuentra que la salida del comando contiene la extensión “heartbeat” (la que tiene la vulnerabilidad heartbleed), lo mostrará en pantalla; en otro caso, mostrará “safe”.

Ejecutando este comando en mi máquina el resultado es el siguiente:



```
nazaretroque@nazaretroque: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
nazaretroque@nazaretroque:~$ echo "QUIT" | openssl s_client -connect localhost:  
8080 2>&1 | grep 'server extension "heartbeat" (id=15)' || echo "safe"  
safe  
nazaretroque@nazaretroque:~$
```

Por lo que hemos comprobado que el sistema está a salvo.

- c) Para evitar esta vulnerabilidad es imprescindible tener las bibliotecas de openssl actualizadas para evitar tener una versión que contenga la vulnerabilidad y no esté parcheada. Teniendo cualquier otra versión distinta de la 1.0.1 o la 1.0.2-beta.

d) Bibliografía utilizada para buscar información sobre la vulnerabilidad:

- <http://heartbleed.com/>
- <https://www.openssl.org/news/secadv/20140407.txt>
- <https://en.wikipedia.org/wiki/Heartbleed>
- <https://geekflare.com/how-to-test-heart-bleed-ssl-vulnerabilities-cve-2014-0160/>
- <https://blog.malwarebytes.com/exploits-and-vulnerabilities/2019/09>