SEGURIDAD EN SISTEMAS OPERATIVOS

4º Grado en Informática

Curso 2017-18

Práctica 1.- Administración de la seguridad en Linux

Sesión 5.- Cifrado de archivos

1.- Introducción

Vamos a introducir algunos principios básicos de criptografía para comprender mejor los conceptos que vamos a manejar en esta práctica.

Entendemos por *cifrado* la conversión de datos de una forma legible a otra no legible. Estas técnicas ayudan a proteger la privacidad de nuestros datos. Para tener de nuevo acceso a los datos debemos realizar el proceso inverso, el *descifrado*. Este proceso necesita cierta información adicional que denominamos *clave*.

Existen tres tipos de técnicas criptográficas:

- *Cifrado simétrico* o de *clave secreta*: se utilizar la misma clave para el cifrado y el descifrado (Figura 1).
- *Cifrado asimétrico* o de *clave pública*: para cada paso se utiliza una clave. Se utiliza especialmente cuando el canal de comunicación es inseguro (Figura 2). Cada parte tiene una clave pública y otra privada. La privada es un secreto, mientras que la pública es compartida con todos aquellos con los que nos queremos comunicar de forma secreta.
- *Funciones hash*: estas funciones no involucran una clave, si no que en su lugar usan un valor hash de longitud fija que se calcula en base a un mensaje de texto plano. Se utilizan para comprobar la integridad de un mensaje/archivo y asegurarse de que no ha sido alterado, comprometido o infectado, durante la transmisión. Algunos algoritmos son MD-5, SHA-1, SHA-256, etc.

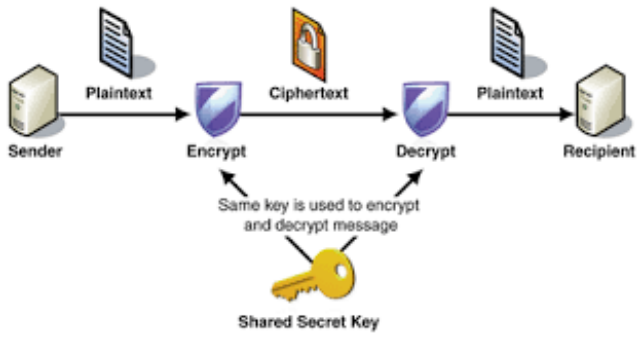

Figura 1.- Cifrado simétrico.

Figura 2.- Cifrado asimétrico

En esta sesión, veremos como cifrar archivos/mensajes, en la siguiente utilizaremos una herramienta para el cifrado de sistemas de archivos completos.

2.- GnuPG

GnuPG (gpg) es una implementación abierta del estándar OpenPGP (Prety Good Privacity) que sirve para cifrar mensajes, documentos, etc. Utiliza un sistema de claves públicas, lo que quiere decir que cada usuario tiene dos claves: una pública y otra privada. La clave privada es la que usamos para descifrar aquellos mensajes que nos has enviado cifrados con nuestra clave pública. La clave privada solo debe conocerla el propietario. La clave pública es da a conocer a aquellas personas para que nos puedan enviar mensajes cifrados.

- Creación de claves públicas y privadas

Una vez instalado el paquete gnupg, lo primero que debemos de hacer es generar las claves pública y privada ejecutando la orden:

```
$ gpg -gen-key
```

Esta orden nos pedirá que indiquemos algunos elementos:

1) Elegir el algoritmo a utilizar de entre las posibilidades que nos ofrece:

```
$ gpg --gen-key
gpg (GnuPG) 2.0.24; Copyright (C) 2013 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.

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:
```

Podemos elegir RSA[1].

2) Seleccionar el tamaño de la clave:

```
Su elección: 1
las claves RSA pueden tener entre 1024 y 4096 bits de longitud.
¿De qué tamaño quiere la clave? (2048)
```

Sen general, un mayor tamaño de clave es más seguro. En el ejemplo se elige 1024 por que no es necesaria tanta seguridad.

3) Seleccionar el tiempo durante el cual será válida la clave:

```
El tamaño requerido es de 1024 bits
Por favor, especifique el período de validez de la clave.
```

1 https://es.wikipedia.org/wiki/RSA.

```
         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)? 1
La clave caduca sáb 04 nov 2017 11:52:58 CET
¿Es correcto? (s/n) s
```

4) **Crearemos un identificador de usuario y una clave. Cada clave se mapea con un id de usuario. El programa nos preguntará por nuestro nombre, dirección de correo electrónico y una frase de paso.**

```
GnuPG debe construir un ID de usuario para identificar su clave.

Nombre y apellidos: jose gomez
Dirección de correo electrónico: jagomez@ugr.es
Comentario:
Ha seleccionado este ID de usuario:
   "jose gomez <jagomez@ugr.es>"

¿Cambia (N)ombre, (C)omentario, (D)irección o (V)ale/(S)alir?
```

5) **Para generar la clave, el sistema usa datos aleatorios del sistema que podemos generar con varias acciones en el sistema: acceder a disco o la red, movimientos de ratón, etc.**

```
Necesita una frase contraseña para proteger su clave secreta.

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: clave 522093E3 marcada como de confianza absoluta
claves pública y secreta creadas y firmadas.

gpg: comprobando base de datos de confianza
gpg: 3 dudosa(s) necesarias, 1 completa(s) necesarias,
modelo de confianza 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: 2017-11-04
pub  1024R/522093E3 2017-11-03 [caduca: 2017-11-04]
     Huella de clave = 99B8 F751 40D8 8441 4634 6CA3 29DB FD00 5220 93E3
uid  [ absoluta ] jose gomez <jagomez@ugr.es>
sub  1024R/7CFB092A 2017-11-03 [caduca: 2017-11-04]
```

La cadena marcada en negro en la salida anterior es la clave de identificación, en el ejemplo, `522093E3`.

6) **Podemos ver las claves generadas con las opciones** `--list-keys` **y** `--list-secret-key`:

```
$ gpg --list-keys
/home/jagomez/.gnupg/pubring.gpg
--------------------------------
pub   1024R/522093E3 2017-11-03 [caduca: 2017-11-04]
```

```
uid    [  absoluta ] jose gomez <jagomez@ugr.es>
sub   1024R/7CFB092A 2017-11-03 [caduca: 2017-11-04]
$ gpg --list-secret-keys
. . .
```

7) **Podemos exportar el archivo de clave pública con las órdenes:**

```
$ gpg –armor –export –output jagomez-pubkey.gpg jagomez
$  gpg –armor –export –output jagomez-pubkey.gpg admin@dominio.com
```

En el primer caso, el archivo jagomez_pubkey.gpg tendrá la clave pública, en el segundo utilizamos nuestra dirección de correo o el identificador visto en el paso 5. Para exportar la clave privada podemos usar la orden:

```
gpg –armor –output archivo_sal –export-secret-key claveID.
```

- **Borrar la clave de los anillos**

Se llama anillo a los archivos que guardan las claves pública y privada. Generalmente, las claves públicas se guardan en el archivo *.pubting.gpg* y las privadas en el *.secring.gpg*. Si se desea borrar las claves, primero debemos borrar la clave privada y luego la pública.

Para el borrado, usamos la orden con la opción `--delete-keys`**. Para la pública,** `--delete-key ClaveID` **y para la privada** `--delete-secret.key claveID`**.**

- **Huella de la clave**

Las claves están identificadas por lo que se denomina huella, que son una serie de números que se usan para verificar si una clave pertenece realmente al propietario. Si recibimos una clave, podemos ver cual es su huella y posteriormente pedirle al propietario que nos indique su huella. Si ambas huellas coinciden, la clave es correcta (no ha sido manipulada o falseada):

```
$ gpg --fingerprint admin@dominio.com
```

- **Cifrar/descifrar mensajes**

Para cifrar un mensaje usaremos la orden de la forma siguiente:

```
$ gpg –armor –recipient admi@dominio.com –encrypt a.txt
```

También, podemos cifrar un archivo con la opción `-output archivo`**. Si en las órdenes anteriores no se usar la opción** `–armor`**, lo que se cifra se deja en un archivo de tipo binario (no ASCII).**

Para descifrar el mensaje previamente cifrado, usamos la opción `decrypt`**, como en el ejemplo:**

```
$ gpg –decrypt text-txt.asc
```

En este caso se nos pedirá nuestra clave.

- **Firmar mensajes**

Firmar un mensaje sirve para que el receptor del mismo pueda verificar que lo que ha recibido es de quién dice ser, es decir, comprobamos que la firma es buena.

Podemos firmar un archivo de texto con la opción `-clearsig`:

```
$ gpg -clearsign texto.txt
```

El sistema nos pide la clave y la añade al contenido del archivo. Si firmamos con la opción `-sign`, el archivo de salida será binario y tendrá una extensión *.gpg*. Para validar la firma y descifrarlo, tenemos que usar la opción `-decrypt`.

Verificamos en mensaje firmado con la opción `-verify`:

```
$ gpg -verify text.txt.asc
```

Si la firma no es correcta nos dará un error del tipo "`gpg: CRC error; ..`"

- **Trabajar con claves en servidores**

Se pueden subir las claves públicas a *servidores de claves* para no tener que enviarlas mediante otros canales que puede ser más engorroso. Por ejemplo, podemos subir nuestra clave a un servidor denominado keyserver con la orden:

```
$ gpg -send-keys -keyserver keyserver.ubuntu.com claveID
```

Existen servidores de claves públicos que podemos usar sin necesidad de instalar uno propio. Algunos de ellos son: http://pgp.mit.edu, http://keyserver.ubuntu.com, http://zimmermann.mayfirst.org, o en España http://www.rediris.es/keyserver. También, existe un directorio global de servidores de claves https://keyserver.pgp.com/vkd/GetWelcomeScreen.event desarrollado por PGP Corp.

Si por ejemplo, hemos tenido que formatear el equipo y deseamos recuperar las claves, podemos usar la orden `gpg -import claveID`. Primero, recuperamos la clave pública y luego la privada:

```
$ gpg -import jagomez_pubkey.gpg
$ gpg -import [otro_usuario].gpg
```

Podemos buscar las claves públicas de personas a las que deseamos enviar un mensaje cifrado en los servidores de claves. Para ello, hacemos la búsqueda con la opción `-keyserver servidor -search-keys claveID`. El servidor muestras las claves que coincida con `claveID`.

Para importar una clave concreta, usamos los parámetros `-keyserver servidor -recv-keys claveID`.

- **Subshell gpg**

GnuPG nos permite un shell con multitud de opciones para trabajar con claves: firmar, cambiar

contraseña, cambiar fecha de expiración, etc. Accedemos al shell con la opción: `--edit-key claveID`.

- Clave de revocación

La clave de revocación es una clave que hace que cuando importamos nuestro anillo de claves se invalide esa clave. Para generarla utilizamos la opción `-gen-revoke`. Esta se puede crear cuando creamos las claves (por si se olvida la contraseña, en cuyo caso no podemos crear la clave de revocación) o bien cuando estas se hayan comprometido. Esta clave se ha de guardar en un lugar seguro. Por ejemplo:

```
$ gpg -output nombre2revoke.asc -gen-revoke nombre2r
```

Si queremos revocar las claves solo tenemos que importar el archivo que contiene la clave de revocación. Una vez revocadas, no podemos cifrar/firmar mensajes aunque sí descifrarlos (con el consiguiente mensaje de revocación).

```
$ gpg -import nombre2revoke.asc
```

Otras cosas que nos permite hacer la herramienta es crear un *anillo de confianza*, que consiste en tener claves de personas firmadas por otras que las firma y que con su firma garantizan que esa clave no se ha alterado y que es quien dice ser.

- Programa que soportan GnuPG

Es posible usar GnuPG de forma gráfica, no solo desde la línea de órdenes. En entornos KDE podemos usar KPGP[2] y en Gnome, SeaHorse[3].

Por otra parte, algunos clientes de correo como Mozilla Thunderbird, a través del plugin **`Enigmail`**, soportan `gpg`. En dispositivos Android, podemos usar *APG*[4] para tareas de cifrado, y si deseamos llevar encima siempre GnuPG en una llave USB podemos usar http://gpg4usb.cpunk.de.

Ejercicio 1.-

a) Utiliza la herramienta `gpg` para crear unas claves personales.
b) Una vez creadas, utilizalas para cifrar y descifrar un archivo.
c) Agrupate con un compañero e intercambiar un archivo cifrado por cada uno que el otro debe descifrar.

3.- OpenSSL

2 https://utils.kde.org/projects/kgpg/.
3 https://wiki.gnome.org/action/show/Apps/Seahorse?action=show&redirect=Seahorse.
4 https://play.google.com/store/apps/details?id=org.thialfihar.android.apg&hl=es.

`OpenSSL` es una biblioteca que implementa las operaciones criptográficas básicas: cifrado simétrico y de clave pública, firma digital, funciones hash, etc. También, implementa el conocido protocolo SSL (Secure Socket Layer). Podemos ver las opciones que admite ejecutando:

```
$ openssl –list-standar-commands
```

Algunas de ellas son:

`ca`	Crea autoridades de certificación
`dgst`	Calcula la función hash
`enc`	Cifra/descifra utilizando un algoritmo de clave privada (la clave se puede generar o estar almacenada en un archivo).
`genrsa`	Permite generar claves públicas/privadas con el algoritmo RSA.
`password`	Genera la función hash de clave (por ejemplo, para almacenar la clave en una base de datos o archivo – esto no hace más seguro nuestro sistema, sino que nos ayuda a confinar los daños en caso de que se produzca una brecha).
`pkcs12`	Herramientas para gestionar información de acuerdo con el estándar PKCS#12[5] (estándar criptográfico de clave pública).
`rand`	Genera cadenas de bits seudo-aleatorias.
`rsa`	Gestión de datos RSA.
`rsautl`	Cifra/descifra o firma/verifica firmas con RSA.
`x509`	Gestión de datos para X509
`verify`	Comprobaciones para X509
...	

OpenSSL utiliza una amplio número de algoritmos de clave secreta. Podemos verlos con la orden:

```
$ openssl list-cipher-commads
```

Por ejemplo, en la lista aparece el algoritmo `base64` que es una forma de codificar información binaria en caracteres alfanuméricos (no es realmente un algoritmo de clave privada):

```
$ echo "35358925728578" > numero.txt
$ openssl enc -base64 -in numero.txt
MzUzNTg5MjU3Mjg1NzgK
```

Podemos cifrar un texto con el algoritmo AES utilizando el modo CBC[6] y una clave de 256 bits:

```
$ echo "I love OpensSSl!">texto.txt
$ openssl enc -aes-256-cbc -in texto.txt -out cifrado.bin
      enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
$ cat cifrado.bin
Salted__[illegible]
```

Donde la clave secreta se calcula a partir de la clave dada. El archivo de salida es binario como puede observarse en el ejemplo anterior. Podemos descifrarlo:

5 https://en.wikipedia.org/wiki/PKCS_12.
6 https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation.

```
$ openssl enc -aes-256-cbc -d -in cifrado.bin -pass: xxxxx
```

Podemos obtener más información sobre `openssl` en Ubuntu en https://help.ubuntu.com/community/OpenSSL. También sobre programación segura con esta herramienta en https://www.ibm.com/developerworks/opensource/library/l-openssl/index.html y http://www.linuxjournal.com/article/4822.

Ejercicio 2.- Utiliza la herramienta `openssl` para cifrar y descifrar un archivo con un algoritmo y clave de tu elección.

- **Heartbleed**

En abril de 2004 se detectó una importante vulnerabilidad en la biblioteca criptográfica openssl conocida con el nombre "heartbleed" que afectó a más de la mitad de los servidores mundiales. Puede tener más detalles de la misma en http://heartbleed.com, y en multitud de artículos publicados en Internet dada su relevancia.

Ejercicio 3.- Busca información sobre esta vulnerabilidad para contestar a las siguientes cuestiones:

a) ¿En qué consiste la misma?
b) ¿Cómo saber si nuestro sistema la sufre?
c) ¿Cómo podemos subsanarla?

4.- Otras herramientas

Otras herramientas que podemos utilizar en Linux para cifrar son:

`crypt`	La herramienta original de sistema Unix. Suministra un nivel de seguridad bajo.
`Bcrypt`	Utiliza el algoritmo *Blowfish*[7] y mejora a `crypt`.
`Ccrypt`	Utiliza el algoritmo *Rijndael*[8], usado por AES. También mejora a `crypt`.
`7-Zip`	Herramienta de compresión que incorpora cifrado AES.
`TrueCrypt`	Herramienta de cifrado que ha quedado en desuso por rumores de falta de seguridad que finalmente se mostró que no eran ciertos. Se ha sustituido por `mcrypt` (biblioteca *libmcrypt*).

7 https://es.wikipedia.org/wiki/Blowfish.
8 http://searchsecurity.techtarget.com/definition/Rijndael.