OpenSSL Básico

En lo que sigue estudiaremos con detalle los servicios más notables que puede ofrecer el paquete de software que conocemos como OpenSSL.

Conozca la Versión con que Trabaja

Para conocer la versión de `OpenSSL` que está usando necesita ejecutar en su línea de órdenes (`$`) lo siguiente:

```
openssl version
```

Obtendrá un diálogo como éste:

```
Equipo:dir user$ openssl version
OpenSSL 1.0.2n  7 Dec 2017
```

La línea de Órdenes

Se puede operar con `OpenSSL` desde la terminal del sistema según la siguiente sintaxis:

```
openssl command [ command_opts ] [ command_args ]
```

o desde su propia línea de órdenes; para entrar en ella ejecutamos la siguiente orden:

```
 openssl
```

y entonces encontraremos el siguiente símbolo de línea de órdenes (prompt):

```
OpenSSL>
```

en cuya línea podremos ejecutar órdenes de `OpenSSL`. Para salir de esta línea basta ejecutar al menos una de las siguientes órdenes:

- `quit`
- `exit`

El siguiente es un ejemplo típico de este diálogo:

```
Equipo:dir user$ openssl
OpenSSL> quit
Equipo:dir user$
```

Las *órdenes básicas* (ejercicio) son éstas:

- `openssl engine` proporciona información sobre el motor de la versión.
- `openssl ciphers` proporciona un listado de cifrados `SSL/TLS` soportados.
- `openssl speed` devuelve información del rendimiento del conjunto de herramientas en cálculos de funciones criptográficas.
- `openssl prime` devuelve información sobre si el argumento, un número natural, es primo o no.

- `openssl prime -generate -bits 100` genera un número primo de 100 bits; en lugar de 100 podemos poner otro número `n`.
-
- En general `openssl help` da un listado de las órdenes que acepta.

La Orden `prime`

Su estructura básica es la siguiente:

```
openssl prime [-bits n] [-checks n] [-generate] [-hex] [-safe] p
```

La orden `prime` es usada para generar números primos o para verificar si los números son primos. Los resultados son probabilísticos: tienen una probabilidad extremadamente alta de ser correctos, pero no están garantizados.

Las opciones son las siguientes:

- `-bits n`

  Especifique el número de bits para el número primo generado. Debe usarse junto con `-generate`.

- `-checks n`

  Realiza una prueba de primalidad probabilística de Miller-Rabin con `n` iteraciones. El valor predeterminado es 20.

- `-generate`

  Genere un número primo pseudoaleatorio. Debe usarse junto con -bits.

- `-hex`

  Salida en formato hexadecimal.

- `-safe`

  Genera solo números primos "seguros" (es decir, un p primo de modo que (p-1)/2 también sea primo).

- `p`

  Prueba si el número `p` es primo.

El Cifrado

La base de las órdenes de cifrado es:

```
openssl enc
```

Para concer los detalles puede ejecutar

```
openssl enc --help
```

Con la siguiente orden podríamos cifrar el fichero `secreto.txt` usando `aes256` (cfr. OpenSSL warning):

```
openssl enc -e -aes256 -base64 -md sha256 -pbkdf2 -iter 100000 -salt \
            -in secreto.txt -out secreto.txt.enc
```

El diálogo que se establece es el siguiente:

```
Equipo:dir user$ echo el secreto se protege a sí mismo >> secreto.txt
Equipo:dir user$ openssl enc -aes256 -base64 -md sha256 -pbkdf2 -iter 100000 -salt
-in secreto.txt -out secreto.txt.enc
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
Equipo:dir user$ ls
secreto.txt secreto.txt.enc
Equipo:dir user$
```

mediante lo cual el fichero `secreto.txt` resulta cifrado por medio de `AES` en modo `CBC`. `OpenSSL` ha pedido una contraseña y la confimación; esta contraseña será usada en este caso para:

- derivar una clave de cifrado
- derivar el vector de inicialización

Considérese lo siguiente:

1. El fichero resultante del cifrado es codificado en base 64 y escrito en `secreto.txt.enc`.
2. En la orden anterior puede ser suprimido `-e` pues es la opción por defecto.
3. En la orden anterior puede ser sustituido `-base64` por `-a`, pues la opción `-a` produce igualmente una codificación base 64 tras el cifrado.
4. Si en lugar de base 64 se quiere usar hexadecimal, basta para ello con suprimir la opción `-base64`, pues hexadecimal es la codificación por defecto.

   ```
   openssl enc -e -aes256 -md sha256 -pbkdf2 -iter 100000 -salt \
               -in secreto.txt -out secreto.txt.hex.enc
   ```

   y para cerciorarse de ello basta con abrir el fichero cifrado `secreto.txt.hex.enc` con `xxd`:

   ```
   Equipo:dir user$ xxd secreto.txt.hex.enc
   00000000: 5361 6c74 6564 5f5f a844 c11a e2e9 f1a0  Salted__.D......
   00000010: 8036 d7b3 73cd 9dfa 305d 86b5 2417 61e6  .6..s...0]..$.a.
   00000020: ae40 d3a2 5ff9 73a8 cac7 a10d 8225 e6e3  .@.._.s......%..
   00000030: 03f0 4575 cb9a ee42 b9ce 2c10 81c7 0818  ..Eu...B..,.....
   ```

5. En la mayoría de los casos es necesario aportar un vector de inicialización. El vector es aportado con la opción `-iv`, así pues el retazo: `-iv ivec` especifica como opción de cifrado el vector de inicialización elegido `ivec` que estará expresado en hexadecimal.
6. Si el método de cifrado lo requiere y no es aportado vector de inicialización alguno, el sistema lo genera a partir de la contraseña junto a la clave de cifrado.
7. Si no deseamos que se nos pida la contraseña, tenemos tres opciones:

    1. Incluir la opción `-k`, en cuyo caso la contraseña será tomada igual al siguiente argumento.
    2. Incluir la opción `-K`, que opera como `-k` salvo que el argumento estará en hexadecimal.
    3. Incluir la opción `-kfile`, en cuyo caso la contraseña será la primera línea del fichero que será el argumento.
8. La opción `-md` servirá para designar según su argumento el sistema de resumen que debe usar `OpenSSL` para crear la clave en función de la contraseña aportada. Su argumento puede ser: `md2`, `md5`, `sha`, `sha1`, `sha256`, etc.
9. La opción `-nopad` deshabilita el relleno de bloques estándar.
10. La opción `-engine e` usa el motor `e`, posiblemente un dispositivo hardware.
11. En algunas versiones de openssl la opción `-z` habilitaba la compresión `zlib` del resultado. Después de que un fichero fuera cifrado (y puede que codificado base 64) era comprimido vía `zlib` y viceversa, al descifrar `zlib` era aplicado previamente.

Como ejemplo de lo anterior, tenga el siguiente:

```
Equipo:dir user$ echo el secreto se protege a sí mismo > secreto.txt
Equipo:dir user$ openssl enc -e -aes256 -kfile secreto.txt -base64  \
    -md sha256 -pbkdf2 -iter 100000 -salt \
    -in quijote_cap_01.txt -out quijote_cap_01.txt.enc
Equipo:dir user$ ls
quijote_cap_01.txt quijote_cap_01.txt.enc secreto.txt
```

El descifrado

El descifrado se lleva a cabo con el retazo básico:

```
openssl enc -d
```

y resto en consecuencia con la forma de cifrado. Para el ejemplo de la sección anterior, el descifrado sería como sigue:

```
openssl enc -d -aes256 -base64 -md sha256 -pbkdf2 -iter 100000 -salt \
            -in secreto.txt.enc -out secreto_new.txt
```

según la cual:

- es tomado el fichero cifrado, en nuestro caso `secreto.txt.enc`,
- es descodificado desde la base 64 en virtud de la presencia de `-d`,
- es descifrado, también en virtud de la opción `-d`, mediante AES
- genera el fichero `secreto_new.txt`, que conformaba el texto llano.

El diálogo es el siguiente:

```
Equipo:dir user$ openssl enc -d -aes256 -kfile secreto.txt -base64  \
    -md sha256 -pbkdf2 -iter 100000 -salt \
    -in quijote_cap_01.txt.enc -out quijote_cap_01_new.txt
Equipo:dir user$ cat quijote_cap_01_new.txt
Capítulo Primero
```

```
Que trata de la condición y ejercicio del famoso hidalgo D. Quijote de
la Mancha ...
```

Calcular la Huella Hash con `sha256`

Para calcular la huella hash de un fichero haremos lo siguiente:

```
openssl dgst -sha256 path/to/myfile
```

y si deseamos calcular la huella de un string, p.e. "simple text", podemos hacer lo siguiente:

```
echo -n "simple text" | openssl dgst -sha256
```

el diálogo que se produce en este caso es el siguiente:

```
$ echo -n "simple text" | openssl dgst -sha256
2609c7c28788898a337c063ff1c3b92275832bddeda014a790d109fad3ba85e2
```

La Simple Codificación en base 64

Puede resultar de utilidad la simple codificación de un fichero en base 64. Esto es posible desde `OpenSSL` a través de la orden:

```
openssl enc -base64 -in text.plain -out text.base64
```

cifrado que puede ser revertido mediante

```
openssl enc -d -base64 -in text.base64 -out text.plain
```

Generar una clave en la línea de órdenes

Para muchos efectos es sobradamente suficiente, por su aproximación a la aleatoriedad, con los password que puede generar `OpenSSL`. La forma de hacerlo es con esta orden:

```
openssl rand -base64 n
```

donde `n` es el número de caracteres que queremos para el password. Por ejemplo:

```
openssl rand -base64 32
```

generaría un password de 32 caracteres escogidos entre: letras, mayúsculas y minúsculas, y dígitos del 0 al 9. El ejemplo anterior ofrecería el siguiente diálogo:

```
Equipo:dir user$ openssl rand -base64 32
6bGlB9j8sCLqoCl6PQ/yXn8OvXNw7M9F3JlcXFufeCI=
```

Es posible dirigir la salida a un fichero y no presentarlo por pantalla, con la siguiente orden:

```
openssl rand -base64 -out outfile.txt n
```

por ejemplo:

```
openssl rand -base64 -out mipassword.txt 15
```

Como ejemplo, el siguiente diálogo:

```
Equipo:dir user$ openssl rand -base64 -out mipassword.txt 15
Equipo:dir user$ cat mipassword.txt
Z8cxtLXR9XSTPGG/QUNzFnsPRxrmTshyYlaUJEAF0pA=
Equipo:dir user$
```

Si queremos añadir al fichero `mipassword.txt` una nueva clave pero en una línea nueva, podemos usar la siguiente orden:

```
openssl rand -base64 n >> mipassword.txt
```

claro está que sustituyendo `n` por un número, por ejemplo 15 si queremos una clave de 20 caracteres.

Herramienta Auxiliar

En los sistemas `Linux` y `Mac OS` el volcado de una sesión de consola a un fichero puede ser llevado a cabo con la orden `script` y para concluirlo ejecutaremos `exit` desde la línea de órdenes de la consola. Este sería un ejemplo típico:

```
Equipo:dir user$ script prueba_volcado.txt
Script started, output file is prueba_volcado.txt
bash-3.2$ ls -al
total 32
drwx------  4 fmgo  staff    136  8 dic 17:21 .
drwxr-xr-x  8 fmgo  staff    272  8 dic 15:38 ..
-rw-r--r--  1 fmgo  staff     43  8 dic 17:21 prueba_volcado.txt
-rw-------  1 fmgo  staff  10711  5 dic 14:12 quijote_cap_01.txt
bash-3.2$ exit
exit

Script done, output file is prueba_volcado.txt
Equipo:dir user$
```

Podemos ver el efecto de lo hecho como sigue:

```
Equipo:dir user$ cat prueba_volcado.txt
Script started on Fri Dec  8 17:21:20 2017
bash-3.2$ ls -al
total 32
drwx------  4 fmgo  staff    136  8 dic 17:21 .
drwxr-xr-x  8 fmgo  staff    272  8 dic 15:38 ..
-rw-r--r--  1 fmgo  staff     43  8 dic 17:21 prueba_volcado.txt
-rw-------  1 fmgo  staff  10711  5 dic 14:12 quijote_cap_01.txt
bash-3.2$ exit
exit
```

```
Script done on Fri Dec  8 17:21:32 2017
Equipo:dir user$
```

El fichero `prueba_volcado.txt` contendría lo ejecutado en la terminal entre la orden `script prueba_volcado.txt` y la orden `exit`.

Para añadir contenido a un fichero existente usaríamos la opción `-a`:

```
script -a prueba_volcado.txt
```

Si se quiere hacer un volcado de la sesión para reproducir a modo de vídeo usaríamos al efecto `script` con la opción `--timing=file.tm`:

```
script --timing=file.tm script.out
```

esto produce la pareja de ficheros `file.tm` y `script.out` cuando cerremos el volcado con `exit`. Ambos ficheros servirán para reproducir la sesión previamente volcada, lo cual será posible mediante la orden:

```
scriptreplay --timing file.tm --typescript script.out
```

Existe la posibilidad de acelerar la reproducción del vídeo con la opción `--divisor n`, donde `n` es un número natural no nulo; en caso de poner como `n` un número natural `0` o un entero negativo, entonces la reproducción es inmediato como haría `cat`.

```
scriptreplay --timing file.tm --typescript script.out --divisor 9
```

Ejercicios

1. Lleve a cabo el cifrado de un fichero extenso en formato `.pdf` en las siguientes condiciones:
    - convierta el fichero `.pdf` a base 64
    - cifre con `AES`
    - use una clave escrita en un fichero que habrá generado previamente con `OpenSSL` y que tenga, digamos, 20 caractéres que pueden ser escogidos entre: letras, mayúsculas y minúsculas, y dígitos del 0 al 9.
    - use como método de resumen `sha256`

      seguidamente descífrelo y devuélvalo de base 64 comprobando que el fichero resultado es idéntido al que fue cifrado por usted.

2. Haga lo anterior haciendo un volcado con `script` de la sesión de terminal y hágalo de forma que luego pueda ser reproducido a modo de vídeo.

Referencias:

- OpenBSD manual page server
- Manpage
- OpenSSL en Manpage