Seguridad en Redes Practica 2.1

David Antuña Rodríguez Javier Carrión García

1 OpenSSL

1.1 Cifrado de bloque

Los textos generados no son iguales porque la clave de cifrado del algoritmo es distinta a pesar de que las contraseñas coincidan.

```
usuario_vms@ptol103:~/SeR$
usuario_vms@ptol103:~/SeR$ cmp cipher1.bin cipher2.bin
cipher1.bin cipher2.bin son distintos: byte 9, línea 1
usuario_vms@ptol103:~/SeR$
usuario_vms@ptol103:~/SeR$ cmp cipher1.bin cipher3.bin
cipher1.bin cipher3.bin son distintos: byte 9, línea 1
usuario_vms@ptol103:~/SeR$
usuario_vms@ptol103:~/SeR$
cmp cipher2.bin cipher3.bin
cipher2.bin cipher3.bin son distintos: byte 9, línea 1
usuario_vms@ptol103:~/SeR$
```

Figure 1.1.1 : Comparación de los ficheros generados.

Por defecto OpenSSL utiliza salt al derivar la clave de cifrado para evitar los ataques por diccionario. Al utilizar la opción no salt se elimina el componente aleatorio de la clave, la salt, por lo que la misma clave da lugar al mismo texto cifrado.

```
usuario_vms@ptol103:~/SeR$
usuario_vms@ptol103:~/SeR$ cmp cipherlno.bin cipher2no.bin
usuario_vms@ptol103:~/SeR$ cmp cipherlno.bin cipher3no.bin
usuario_vms@ptol103:~/SeR$ cmp cipher2no.bin cipher3no.bin
usuario_vms@ptol103:~/SeR$
```

Figure 1.1.2 : Comparación de los cifrados sin salt.

Al cifrar o descifrar con OpenSSL se pueden utilizar las siguientes opciones.

• -p: muestra la key e iv usados para cifrar/descifrar.

```
usuario_vms@ptoll03:~/SeR$ openssl enc -aes-128-ecb -p -in textol.txt -out cipherl_p.bin enter aes-128-ecb encryption password:

Verifying - enter aes-128-ecb encryption password:
salt=B6CCBC9A182C9D12
key=3A5A79414042BE53E13D03A75B2155A5
usuario_vms@ptoll03:~/SeR$
```

Figure 1.1.3 : Cifrado utilizando la opción -p.

```
usuario_vms@ptol103:~/SeR$ openssl aes-128-ecb -d -p -in cipherl_p.bin -out textol_p.txt
enter aes-128-ecb decryption password:
salt=B6CCBC9A18ZC9D12
key=3A5A79414042BE53E13D03A75B2155A5
usuario_vms@ptol103:~/SeR$ ■
```

Figure 1.1.4: Descifrado utilizando la opción -p.

- -P: muestra la key e iv generados por no se llega a cifrar/descifrar.
- -pass: si no se quiere que openSSL solicite la clave por consola se puede utilizar esta opcion para indicar el origen de la misma.

```
usuario_vms@ptoll03:-/SeR$ usuario_vms@ptoll03:-/SeR$ openssl enc -aes-128-ecb -pass file:pass.txt -in textol.txt -out cipherl_pass.bin usuario_vms@ptoll03:-/SeR$ openssl aes-128-ecb -d -pass file:pass.txt -in cipherl_pass.bin -out cipherl_pass.txt usuario_vms@ptoll03:-/SeR$
```

Figure 1.1.5: Cifrado/descifrado utilizando la opcion -pass.

• -S: permite especificar un salt, debe ser una cadena de digitos hexadecimales.

Figure 1.1.6: Cifrado/descifrado utilizando la opción -S.

- -K: permite especificar la key a utilizar, debe ser una cadeba de digitos hexadecimales. Obliga a utilizar tambien la opción-iv, porque el iv se genera al generar la key.
- -iv: permite especificar el iv a utilizar, debe ser una cadeba de digitos hexadecimales. Si se especifica se ignora el iv que genere el proceso de creacion de la key, salvo que se hayan utilizado opciones especiales de generacion de la key.

Figure 1.1.7: Cifrado/descifrado utilizando las opciones -K y -iv.

El receptor recibe la salt en el propio mensaje cifrado, figura 1.1.8 , y obtiene el iv al generar la clave con dicha salt y la contraseña privada que ha de conocer.

```
usuario_vms@pto1103:~/SeR$ xxd cipherl_salt.bin
0000000: 5361 6c74 6564 5f5f b6d6 7fb9 22f5 3afc Salted__...".:.
0000010: 6160 h22h 2h24 7ff8 99a6 d970 5ecc 9445 a`++$ p^ F
```

Figure 1.1.8 : Salt en el texto cifrado.

1.2 Modos de bloque

La diferencia es que en ecb podemos discernir los contornos de la figura y en cbc no se ve nada. Esto ocurre porque ecb no oculta los patrones del fichero origen en el fichero cifrado y cbc si.

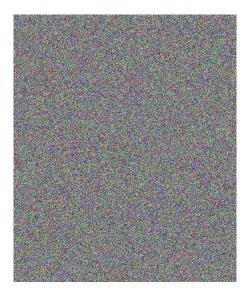


Figure 1.2.1: Cifrado con cbc.

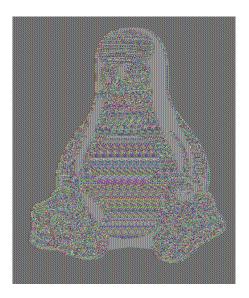


Figure 1.2.2 : Cifrado con ecb.

1.3 Cifrado de flujo

El texto en claro es **Attack at dawn**, la clave es **Secre** para el algoritmo compilado o **5365637265** para OpenSSL. Ambas claves son la misma pero a OpenSSL hay que proporcionarsela en hexadecimal y al compilado se le da la cadena.

Usar:

./rc4 "Attack at dawn" "Secre" echo -ne "Attack at dawn" | openssl enc -rc4-40 -K 5365637265 | xxd –u -p

Figure 1.3.1 : Cifrado con rc4-40.

1.4 Funciones hash y HMAC

Se ha realizado el resumen de /etc/services que tiene un tamaño de TODO. Usar:

ls -l /etc/services

openssl dgst -md5 /etc/services

openssl dgst -sha1 /etc/services

openssl dgst -sha256 /etc/services

Figure 1.4.1 : Hash utilizando md5.

Figure 1.4.2 : Hash utilizando sha1.

Figure 1.4.3: Hash utilizando sha256.

El valor del HMAC depende de la clave utilizada y la función de hash escogidas. Se han obtenido los siguientes HMAC del fichero /etc/services. Usar:

ls -l /etc/services openssl dgst -md5 -hmac seguridad /etc/services openssl dgst -sha1 -hmac seguridad /etc/services openssl dgst -sha256 -hmac seguridad /etc/services

Figure 1.4.4: HMAC utilizando md5.

Figure 1.4.5: HMAC utilizando sha1.

Figure 1.4.6: HMAC utilizando sha256.

2 GnuPG

2.1 Cifrado y descifrado

Se ha cifrado el fichero /etc/services utilizado la clave seguridad con batch mode, modo por bloques.

Usar:

```
cp /etc/services $HOME/services 
gpg2 --version y escoger uno de los algoritmos 
gpg2 --symmetric --cipher-algo <algoritmo escogido> -a $HOME/services
```

2.2 Funciones hash

Se ha calculado el hash de /etc/services con varios algorimos.

Usar·

```
cp /etc/services $HOME/services
gpg2 --version y escoger uno de los algoritmos
gpg2 --symmetric --cipher-algo <algoritmo escogido> -a $HOME/services
```

Figure 2.2.1: Hash utilizando .

Figure 2.2.2: Hash utilizando .

Figure 2.2.3: Hash utilizando .