

AES , OpenSSL y ECB

José Manuel Evangelista Tiburcio

9 de Septiembre 2024

Descripción Breve

Este reporte aborda el estudio y la implementación de técnicas de cifrado mediante el uso de la herramienta OpenSSL. A lo largo de los ejercicios, se han explorado distintos modos de operación de cifrado simétrico, como ECB y CBC, así como el uso de contraseñas para derivar claves criptográficas seguras mediante el esquema PBKDF2. Se ha trabajado con la configuración de padding para asegurar que los datos se ajusten al tamaño de bloque requerido por el algoritmo AES.

Preguntas

A continuación se presentan algunas preguntas con viñetas personalizadas en forma de candado:

🔒 ¿De cuántos bits es la llave `6e4b27a9a3a32165eb29eff45c204b0a` que se usó tanto para cifrar como para descifrar los archivos? Justifique su respuesta.

En el comando de OpenSSL que estamos utilizando , con el algoritmo de cifrado de AES-128, como es estandarizado para ese caso en particular, se espera una clave de cifrado de 128 bits para poder funcionar, y en grl. la clave que pasamos contiene 32 caracteres hexadecimales y cada caracter contiene 4 bits, obteniendo asi los 128.

🔒 Ejecute el siguiente comando:

```
openssl enc -v -aes-128-ecb -K 64f3ad6017h65eb2f5f4e0b82f5ceb31 -in mensaje.txt >
```

Tome en cuenta que la llave cambió. ¿Se cifró el archivo? ¿Qué error indica OpenSSL? ¿A qué le atribuye la causa de ese error?

No, realmente el archivo no se cifro, sino que OpenSSL, intento realizar el proceso, por eso se genero el archivo .enc pero no fue cifrado.

Se indica el error:

```
non-hex digit  
invalid hex key value
```

que indica que la clave para el cifrado tiene un formato incorrecto es decir "h" dentro de la clave no es un valor hexadecimal, recordando que en hexadecimal, solo se permiten los caracteres 0-9 y A-F (o a-f), pero h está fuera de este rango. En general, el error se debe a que la clave contiene un carácter no válido (h), ya que solo se permiten dígitos hexadecimales (0-9, A-F). Para corregir el error, la clave debe estar completamente en formato hexadecimal.

🔍 ¿Cuál es el riesgo de introducir la llave en el mismo comando?

El principal riesgo reside en que si algún atacante logra obtener el registro de los comandos que hemos ejecutado puede vulnerar nuestra clave de cifrado, y así, apesar de usar AES-128 que es un buen algoritmo de cifrado, la información sea vulnerada.

🔍 ¿Qué opciones de OpenSSL necesitó indicar para evitar la advertencia sobre la derivación de llave y para desplegar la llave derivada de la contraseña que usted ingresó? ¿Qué otra información se despliega?

Para evitar la advertencia sobre la función de derivación de llave, debemos usar la opción `-pbkdf2` en el comando. Esto indica que OpenSSL debe usar PBKDF2 para derivar la clave a partir de la contraseña, lo que es más seguro que el método de derivación más antiguo.

Ademas se despliega el salt que es el valor aleatorio utilizado en la derivación de la clave, y la clave derivada de la contraseña. Este es el valor que se utiliza efectivamente para el cifrado de los datos.

🔍 ¿Cuál es la opción en el comando de OpenSSL que se usa para indicar que se descifrá el archivo?

La opción `-d` es la que le indica a OpenSSL que se debe descifrar el archivo.

🔍 Investigue las diferentes técnicas de padding para el cifrado de archivos.

1. PKCS#7 (o PKCS#5) Padding

Descripción: Es la técnica de padding más ampliamente utilizada en cifrados por bloques. Se añade una cantidad de bytes de padding igual al número de bytes que faltan para completar el tamaño del bloque.

Cómo funciona: Si el último bloque tiene 6 bytes, entonces se añaden 10 bytes con el valor `0x0A` (10 en decimal), indicando que 10 bytes han sido añadidos.

Ejemplo: Si el tamaño del bloque es 16 bytes y el último bloque tiene 13 bytes de datos, se añadirán 3 bytes de padding con el valor `0x03`.

Ventaja: Es fácil de implementar y eliminar, ya que el valor del padding mismo indica cuántos bytes fueron añadidos.

Desventaja: No puede usarse si el tamaño del bloque es mayor que 255 bytes.

2. ANSI X.923

Descripción: En esta técnica, todos los bytes de padding, excepto el último, se llenan con ceros (`0x00`). El último byte indica cuántos bytes se añadieron como padding.

Cómo funciona: Si se necesitan 4 bytes de padding, los primeros 3 bytes serán `0x00`, y el último será `0x04`.

Ejemplo: Si el tamaño del bloque es 16 bytes y el último bloque tiene 14 bytes de datos, se añadirán 2 bytes: `0x00 0x02`.

Ventaja: Los valores ceros en los bytes de padding pueden simplificar la detección de padding, pero el último byte asegura que se puede descartar correctamente.

3. ISO/IEC 7816-4

Descripción: Esta técnica de padding se utiliza principalmente en el contexto de tarjetas inteligentes y otros sistemas de seguridad.

Cómo funciona: Se añade un byte `0x80` (en hexadecimal) seguido de ceros (`0x00`) hasta completar el bloque.

Ejemplo: Si el tamaño del bloque es 16 bytes y el último bloque tiene 12 bytes de datos, se añadirán los bytes: 0x80 0x00 0x00 0x00.

Ventaja: Utiliza un valor específico (0x80) que no puede aparecer en el padding de otras técnicas, lo que lo hace fácil de identificar.

4. Zero Padding

Descripción: Se añade padding con valores de ceros (0x00) hasta completar el tamaño del bloque.

Cómo funciona: Simplemente se llenan los bytes faltantes con 0x00.

Ejemplo: Si el tamaño del bloque es 16 bytes y el último bloque tiene 14 bytes de datos, se añaden 2 bytes de padding: 0x00 0x00.

Ventaja: Es simple.

Desventaja: No es recomendable si los datos originales podrían terminar en ceros, ya que sería difícil distinguir entre los ceros originales y el padding.

5. Bit Padding

Descripción: En esta técnica, se añade un único 1 bit seguido de ceros hasta completar el tamaño del bloque.

Cómo funciona: Se añade un 1 seguido de ceros hasta llenar el bloque.

Ejemplo: Si el último bloque tiene 7 bytes y el tamaño del bloque es de 8 bytes, se añadirán los bits: 10000000.

Ventaja: Simple y fácil de detectar.

Desventaja: Solo se puede utilizar con cifrados que operan a nivel de bits y no es común para cifrados por bloques tradicionales.

6. ISO 10126

Descripción: Similar a ANSI X.923, pero en lugar de usar ceros para los bytes de padding, utiliza valores aleatorios. El último byte siempre indica cuántos bytes se añadieron.

Cómo funciona: Se rellenan los bytes faltantes con valores aleatorios, y el último byte indica la cantidad de bytes añadidos.

Ejemplo: Si se necesitan 3 bytes de padding, los dos primeros serán aleatorios, y el último será 0x03.

Ventaja: Los valores aleatorios dificultan que los atacantes predigan el padding.

Desventaja: Es más difícil de implementar correctamente debido a la necesidad de aleatoriedad.

7. No Padding (-nopad)

Descripción: Se puede especificar que no se utilice padding, pero solo es adecuado cuando el tamaño de los datos ya es un múltiplo exacto del tamaño del bloque.

Ventaja: No se agrega nada a los datos originales.

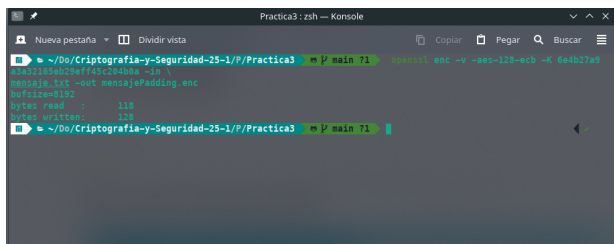
Desventaja: Si los datos no son múltiplos del tamaño de bloque, el cifrado fallará o el bloque final estará incompleto.

🔍 ¿En qué modo de operación sí se puede obtener (parcialmente) la imagen, y por qué?

En ECB, debido a como opera, lo cual vimos a través de la práctica, por otro lado CBC, no permite vislumbrar siquiera parcialmente la imagen.

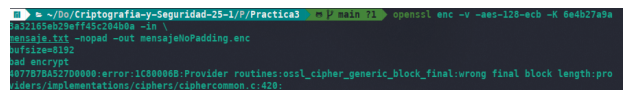
Imágenes

A continuación se presentan las imágenes agrupadas en bloques de dos por línea:



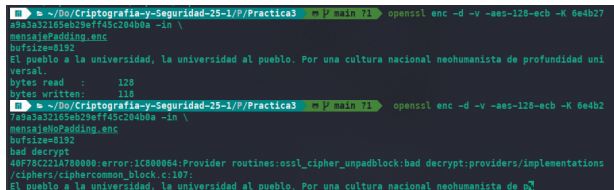
```
Practica3: zsh — Konsole
Nueva pestaña  Dividir vista
~/Do/Criptografia-y-Seguridad-25-1/P/Practica3 m P main 71
7a9a3a32165eb29eff45c204b0a -in \
mensaje.txt -out mensajePadding.enc
bufsize=8192
bytes read : 118
bytes written: 144
~/Do/Criptografia-y-Seguridad-25-1/P/Practica3 m P main 71
```

Figure 1: Imagen 1



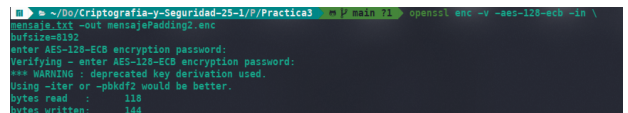
```
~/Do/Criptografia-y-Seguridad-25-1/P/Practica3 m P main 71 openssl enc -v -aes-128-ecb -K 6e4b27a9
7a9a3a32165eb29eff45c204b0a -in \
mensaje.txt -topad -out mensajeNoPadding.enc
bufsize=8192
bad encrypt
107b78a527d0000:error:1C80006B:Provider routines:ossl_cipher_generic_block_final:wrong final block length:pro
viders/Implementations/ciphers/ciphercommon.c:420:
```

Figure 2: Imagen 2



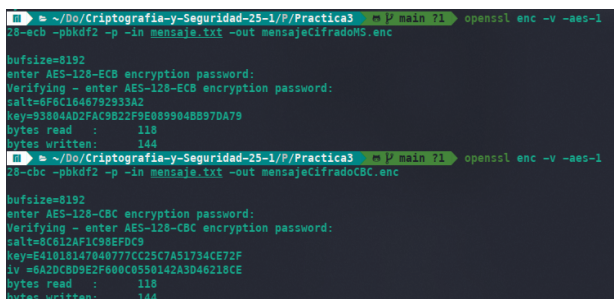
```
~/Do/Criptografia-y-Seguridad-25-1/P/Practica3 m P main 71 openssl enc -d -v -aes-128-ecb -K 6e4b27
a9a3a32165eb29eff45c204b0a -in \
mensajePadding.enc
bufsize=8192
El pueblo a la universidad, la universidad al pueblo. Por una cultura nacional neohumanista de profundidad uni
versal.
bytes read : 118
bytes written: 144
~/Do/Criptografia-y-Seguridad-25-1/P/Practica3 m P main 71 openssl enc -d -v -aes-128-ecb -K 6e4b2
7a9a3a32165eb29eff45c204b0a -in \
mensajeNoPadding.enc
bufsize=8192
bad decrypt
40f76c221a780800:error:1C80006B:Provider routines:ossl_cipher_unpadBlock:bad decrypt:providers/Implementations
/ciphers/ciphercommon_block.c:107:
El pueblo a la universidad, la universidad al pueblo. Por una cultura nacional neohumanista de p
```

Figure 3: Imagen 3



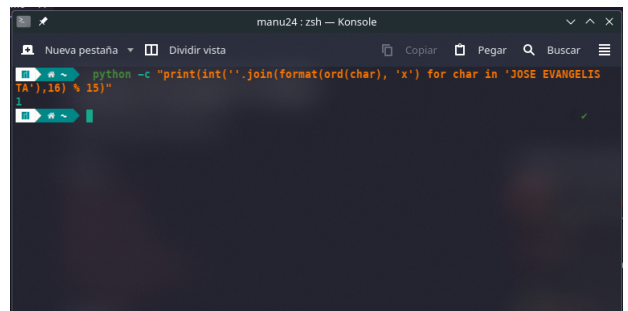
```
~/Do/Criptografia-y-Seguridad-25-1/P/Practica3 m P main 71 openssl enc -v -aes-128-ecb -in \
mensaje.txt -out mensajePadding.enc
bufsize=8192
enter AES-128-ECB encryption password:
Verifying - enter AES-128-ECB encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
bytes read : 118
bytes written: 144
```

Figure 4: Imagen 4



```
~/Do/Criptografia-y-Seguridad-25-1/P/Practica3 m P main 71 openssl enc -v -aes-1
28-ecb -pbkdf2 -p -in mensaje.txt -out mensajeCifradoMS.enc
bufsize=8192
enter AES-128-ECB encryption password:
Verifying - enter AES-128-ECB encryption password:
salt=6F6C1646792933A2
key=93004AD2FAC9822F9E0899048B97DA79
bytes read : 118
bytes written: 144
~/Do/Criptografia-y-Seguridad-25-1/P/Practica3 m P main 71 openssl enc -v -aes-1
28-cbc -pbkdf2 -p -in mensaje.txt -out mensajeCifradoCBC.enc
bufsize=8192
enter AES-128-CBC encryption password:
Verifying - enter AES-128-CBC encryption password:
salt=8C612AF1C98EFC9
key=E41018147040777CC25C7A51734CE72F
iv =6A2DCB09E2F600C0550142A3D46218CE
bytes read : 118
bytes written: 144
```

Figure 5: Imagen 5



```
manu24: zsh — Konsole
Nueva pestaña  Dividir vista
~/Do/Criptografia-y-Seguridad-25-1/P/Practica3 m P main 71 python -c "print(int(''.join(format(ord(char), 'x') for char in 'JOSE EVANGELIS
TA'),16) \n 15)"
1
```

Figure 6: Imagen 6