NAME : Sales Aj Angelo S.

SUBJECT : _____

PROFESSOR : Engr. KRISELYN B. CABADING

DATE/TIME/RM : 09/28/2024 1:30pm-6:00pm 5202

ACTIVITY : Activity 6.2

SEAT NO.: _____ SCORE: _____

(Fill-out all the blanks, excluding the SCORE, please.)

Lab - Encrypting and Decrypting Data Using OpenSSL

Objectives

Part 1: Encrypting Messages with OpenSSL

Part 2: Decrypting Messages with OpenSSL

Background / Scenario

OpenSSL is an open source project that provides a robust, commercial-grade, and full-featured toolkit for the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols. It is also a general-purpose cryptography library. In this lab, you will use OpenSSL to encrypt and decrypt text messages.

Note: While OpenSSL is the de facto cryptography library today, the use presented in this lab is NOT recommended for robust protection. Below are two security problems with this lab:

1) The method described in this lab uses a weak key derivation function. The ONLY security is introduced by a very strong password.
2) The method described in this lab does not guarantee the integrity of the text file.

This lab should be used for instructional purposes only. The methods presented here should NOT be used to secure truly sensitive data.

Required Resources

- CyberOps Workstation virtual machine

Instructions

Part 1: Encrypting Messages with OpenSSL

OpenSSL can be used as a standalone tool for encryption. While many encryption algorithms can be used, this lab focuses on AES. To use AES to encrypt a text file directly from the command line using OpenSSL, follow the steps below:

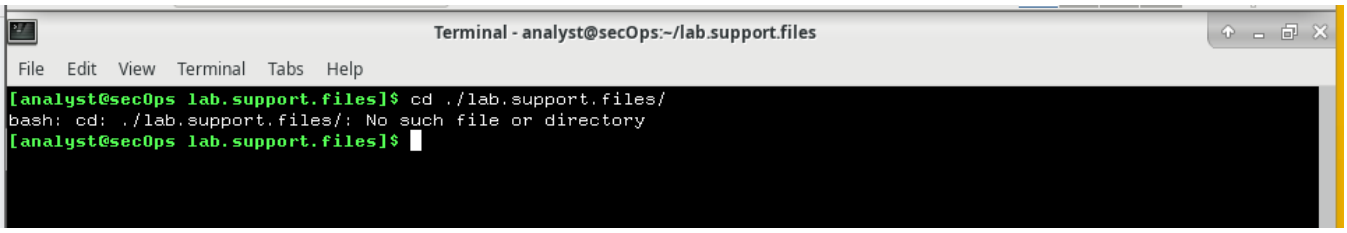
Step 1: Encrypting a Text File

a. Log into CyberOPS Workstation VM.

b. Open a terminal window.

c. Because the text file to be encrypted is in the /home/analyst/lab.support.files/ directory, change to that directory:

```
[analyst@secOps ~]$ cd ./lab.support.files/
[analyst@secOps lab.support.files]$
```



d. Type the command below to list the contents of the encrypted **letter_to_grandma.txt** text file on the screen:

```
[analyst@secOps lab.support.files]$ cat letter_to_grandma.txt
Hi Grandma,
I am writing this letter to thank you for the chocolate chip cookies you sent
me. I got them this morning and I have already eaten half of the box! They are
absolutely delicious!

I wish you all the best. Love,
Your cookie-eater grandchild.
[analyst@secOps lab.support.files]$
```

e. From the same terminal window, issue the command below to encrypt the text file. The command will use AES-256 to encrypt the text file and save the encrypted version as **message.enc**. OpenSSL will ask for a password and for password confirmation. Provide the password as requested and be sure to remember the password.

```
[analyst@secOps lab.support.files]$ openssl aes-256-cbc -in
letter_to_grandma.txt -out message.enc
```

```
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
[analyst@secOps lab.support.files]$
```

Document the password.

```
[analyst@secOps lab.support.files]$ openssl aes-256-cbc -in letter_to_grandma.txt -out message.enc
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
[analyst@secOps lab.support.files]$
```

the pasword was not shown but i have successfully set the password for encrypting the letter to grandma.txt

f. When the process is finished, use the **cat** command again to display the contents of the **message.enc** file.

```
[analyst@secOps lab.support.files]$ cat message.enc
```

Did the contents of the **message.enc** file display correctly? What does it look like? Explain.

```
Verifying - enter aes-256-cbc encryption password:
[analyst@secOps lab.support.files]$ cat message.enc
Salted__
[analyst@secOps lab.support.files]$
```

No it was not displayed correctly unlike the first output but i think it was shown like this because of the encryption i made and it was successfully encrypted as the output was shown

g. To make the file readable, run the OpenSSL command again, but this time add the **-a** option. The **-a** option tells OpenSSL to encode the encrypted message using a different encoding method of Base64 before storing the results in a file.

Note: Base64 is a group of similar binary-to-text encoding schemes used to represent binary data in an ASCII string format.

```
[analyst@secOps lab.support.files]$ openssl aes-256-cbc -a -in
letter_to_grandma.txt -out message.enc
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
```

```
[analyst@secOps lab.support.files]$ openssl aes-256-cbc -a -in letter_to_grandma.txt -out message.enc
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
[analyst@secOps lab.support.files]$
```

h. Once again, use the **cat** command to display the contents of the, now re-generated, **message.enc** file:

Note: The contents of **message.enc** will vary.

```
[analyst@secOps lab.support.files]$ cat message.enc
```

```
U2FsdGVkX19ApWyrn8RD5zNp0RPCuMGZ98wDc26u/vmj1zyDXobGQhm/dDRZasG7
rfnth5Q8NHValEw8vipKGM66dNFyyr9/hJUzCoqhFpRHgNn+Xs5+TOtz/QCPN1bi
08LGTSzOpfkg76XDCk8uPy1hl/+Ng92sM5rgMzLXfEXtaYe5UgwOD42U/U6q73pj
a1ksQrTWsv5mtN7y6mh02Wobo3A1ooHrM7niOwK1a3YKrSp+ZhYzVTrtksWDl6Ci
XMufkv+FOGn+SoEEuh7l4fk0LIPEfGsExVFB4TGdTiZQApRw74rTAZaE/dopaJn0
sJmR3+3C+dmgzZIKEHWsJ2pgLvj2Sme79J/XxwQVNpw=
[analyst@secOps lab.support.files]$
```

Is **message.enc** displayed correctly now? Explain.

```
[analyst@secOps lab.support.files]$ cat message.enc
U2FsdGVkX1/GcmASw0w7Dih20qfiX8oP5kRwh+L1RSve3Rr5kWtcuUinu5Ah+jgk
5cuj5poo72LBvZJt8Vt8gzDX10n7MKTZxb9TSAIKzruOH3enwyYHq76q5gHf6wby
LQEOWNo7Vh892f8UJPsw1Lp7DTU0C92t+Db1aSMiwcR7DxWHpAC+uPTMVaR+Lw+d
X5wKRPe+ZmXNzrSFfXthRs4Y1XKKpcHnqHm17hbjHU7vpTb1VdcaekSvMIpFzNif
SFTD3OQGBmh5+Lk2Td+I4b8fcd7lav/YsbpUMZ9uic4g6HusfUwtzONltbF1APDT
/Oc15HyC8BLI6NySQTPLUCpd+zK5vqahK4j51hUZj/Y=
[analyst@secOps lab.support.files]$
```

yes it was successfully displayed correctly due to it was converted from binary to text encoded with base64

Can you think of a benefit of having **message.enc** Base64-encoded?

it will be much more efficient for the user to copy the code unlike the first output that was not converted to text

Part 2: Decrypting Messages with OpenSSL

With a similar OpenSSL command, it is possible to decrypt **message.enc**.

a. Use the command below to decrypt message.enc:

```
[analyst@secOps lab.support.files]$ openssl aes-256-cbc -a -d -in message.enc
-out decrypted_letter.txt
```

```
Terminal - analyst@secOps:~/lab.support.files
File  Edit  View  Terminal  Tabs  Help
[analyst@secOps ~]$ cd ./lab.support.files/
[analyst@secOps lab.support.files]$ openssl aes-256-cbc -a -d -in message.enc -o
ut decrypted_letter.txt
Extra arguments given.
aes-256-cbc: Use -help for summary.
[analyst@secOps lab.support.files]$ openssl aes-256-cbc -a -d -in message.enc -o
ut decrypted_letter.txt
enter aes-256-cbc decryption password:
[analyst@secOps lab.support.files]$ openssl aes-256-cbc -a -d -in message.enc -o
ut decrypted_letter.txt
enter aes-256-cbc decryption password:
bad decrypt
139855754011072:error:06065064:digital envelope routines:EVP_DecryptFinal_ex:bad
 decrypt:crypto/evp/evp_enc.c:536:
[analyst@secOps lab.support.files]$ openssl aes-256-cbc -a -d -in message.enc -o
ut decrypted_letter.txt
enter aes-256-cbc decryption password:
[analyst@secOps lab.support.files]$
```

b. OpenSSL will ask for the password used to encrypt the file. Enter the same password again.

c. When OpenSSL finishes decrypting the **message.enc** file, it saves the decrypted message in a text file called **decrypted_letter.txt**. Use the **cat** display the contents of **decrypted_letter.txt**:

```
[analyst@secOps lab.support.files]$ cat decrypted_letter.txt
```

Was the letter decrypted correctly?

Yes, the letter was decrypted correctly.

```
[analyst@secOps lab.support.files]$ cat decrypted_letter.txt
Hi Grandpa,
I am writing this letter to thank you for the chocolate chip cookies you sent me
. I got them this morning and I have already eaten half of the box! They are abs
olutely delicious!

I wish you all the best. Love,
Your cookie-eater grandchild.


[analyst@secOps lab.support.files]$
```

The command used to decrypt also contains -a option. Can you explain?

Before OpenSSL can decrypt message.enc, it must first be Base64 decoded since message.enc was Base64 encoded after the encryption process.