Lab - Encrypting and Decrypting Data Using OpenSSL (Instructor Version)

**Instructor Note**: Red font color or gray highlights indicate text that appears in the instructor copy only.

# 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

## 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:

### Encrypting a Text File

* + - 1. Log into CyberOPS Workstation VM.
      2. Open a terminal window.
      3. 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]$

* + - 1. 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]$

* + - 1. 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]$

#### Question:

Document the password.

Type your answers here.

Student choice of password

* + - 1. 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**

#### Question:

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

Type your answers here.

No. The file seems broken as just symbols are displayed. The symbols are shown because OpenSSL has generated a binary file.

* + - 1. 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:

* + - 1. 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]$

#### Questions:

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

Type your answers here.

Yes. While message.enc is encrypted, it is now correctly displayed because it has been converted from binary to text and encoded with Base64.

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

Type your answers here.

The encrypted message can now be copied and pasted in an email message, for example.

## Decrypting Messages with OpenSSL

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

* + - 1. 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**

* + - 1. OpenSSL will ask for the password used to encrypt the file. Enter the same password again.
      2. 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**

#### Questions:

Was the letter decrypted correctly?

Type your answers here.

Yes, the letter was decrypted correctly.

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

Type your answers here.

Because message.enc was Base64 encoded after the encryption process took place, message.enc must be Base64 decoded before OpenSSL can decrypt it.

End of document