Travaux pratiques – Chiffrer et déchiffrer des données avec OpenSSL

Objectifs

Partie 1 : Messages de chiffrement avec OpenSSL

Partie 2 : Messages de déchiffrement avec OpenSSL

Contexte/scénario

OpenSSL est un projet open source qui fournit une boîte à outils robuste, de qualité commerciale et complète pour les protocoles TLS (Transport Layer Security) et SSL (Secure Sockets Layer). C'est aussi une bibliothèque de cryptographie à usage général. Au cours de ces travaux pratiques, vous allez utiliser OpenSSL pour chiffrer et déchiffrer des messages texte.

Remarque : Bien qu'OpenSSL soit la bibliothèque de cryptographie de facto aujourd'hui, l'utilisation présentée dans ce laboratoire n'est PAS recommandée pour une protection robuste. Ces travaux pratiques présentent deux problèmes de sécurité :

1) La méthode décrite dans ce laboratoire utilise une fonction de dérivation de clé faible. La sécurité N'est assurée QUE par un mot de passe très fort.
2) La méthode décrite dans ce laboratoire ne garantit pas l'intégrité du fichier texte.

Ces travaux pratiques doivent uniquement être utilisés à des fins de formation. Les méthodes présentées ici ne doivent pas être utilisées pour sécuriser des données sensibles réelles.

Ressources requises

- Machine virtuelle du poste de travail sécurisé téléchargeable depuis : https://www.netacad.com/resources/lab-downloads?courseLang=en-US

Instructions

Partie 1 : Chiffrer des messages avec OpenSSL

OpenSSL peut être utilisé comme outil autonome pour le chiffrement. Même si de nombreux algorithmes de chiffrement peuvent être utilisés, nous utiliserons ici principalement AES. Pour utiliser AES afin de chiffrer un fichier texte directement à partir de la ligne de commande avec OpenSSL, procédez comme suit :

Étape 1: Chiffrement d'un fichier texte

a. Connectez-vous à la VM Security Workstation.

b. Ouvrez une fenêtre de terminal.

c. Comme le fichier texte à chiffrer se trouve dans le répertoire /home/analyst/lab.support.files/, passez dans ce répertoire :

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

d. Tapez la commande ci-dessous pour afficher à l'écran le contenu du fichier texte chiffré **letter_to_grandma.txt** :

```
[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. Dans la même fenêtre de terminal, exécutez la commande ci-dessous pour crypter le fichier texte. La commande utilisera AES-256 pour chiffrer le fichier texte et sauvegarder la version cryptée sous le nom de **message.enc**. OpenSSL vous demande de définir un mot de passe et de confirmer ce mot de passe. Définissez un mot de passe demandé et mémorisez-le.

```
[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]$
```

Documenter le mot de passe.

_____

_____

_____

f. Lorsque le processus est terminé, utilisez à nouveau la commande **cat** pour afficher le contenu du fichier **message.enc**.

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

Le contenu du fichier **message.enc** s'est-il affiché correctement ? À quoi ressemble-t-il ? Expliquez votre réponse.

_____

_____

_____

g. Pour rendre le fichier lisible, exécutez à nouveau la commande OpenSSL, mais ajoutez cette fois l'option **-a**. L'option **-a** indique à OpenSSL de coder le message chiffré en utilisant une méthode de codage différente de Base64 avant de stocker les résultats dans un fichier.

Remarque : Base64 est un groupe de schémas de codage binaire-texte similaires utilisés pour représenter des données binaires dans un format de chaîne ASCII.

```
[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:
```

h. Une fois de plus, utilisez la commande **cat** pour afficher le contenu du fichier **message.enc**, maintenant re-généré :

Remarque : le contenu du fichier **message.enc** peut varier.

```
[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]$
```

Est-ce que **message.enc** s'affiche correctement maintenant ? Expliquez votre réponse.

_____

_____

_____

Pouvez-vous imaginer un avantage à ce que **message.enc** soit codé en Base64 ?

_____

_____

_____

Partie 2 : Messages de déchiffrement avec OpenSSL

Avec une commande OpenSSL similaire, il est possible de déchiffrer le **message.enc**.

a. Utilisez la commande ci-dessous pour décrypter le message.enc:

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

b. OpenSSL demandera le mot de passe utilisé pour chiffrer le fichier. Saisissez de nouveau le même mot de passe.

c. Lorsque OpenSSL termine le décryptage du fichier **message.enc**, il enregistre le message décrypté dans un fichier texte appelé **decrypted_letter.txt**. Utilisez la commande **cat** pour afficher le contenu de **decrypted_letter.txt** :

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

La lettre a-t-elle été correctement déchiffrée ?

_____

_____

_____

La commande utilisée pour déchiffrer contient également une option -a. Pouvez-vous expliquer pourquoi ?

_____

_____

_____