Cyber-Sécurité
TD7 - 2023

Forensics des e-mail (2) en Python

TD individuel
« La connaissance s'acquiert par l'expérience,
tout le reste n'est que de l'information.»
Albert Einstein

- **Modalité de réalisation du TD**

- **Documents et liens pour la réalisation du TD :**
  - **Python :**
    - Document Python officiel :https://docs.python.org/fr/3/
    - Document Python Fonction officiel : https://docs.python.org/3/library/functions.html
    - Index des bibliothèques officielles Python : https://docs.python.org/3/py-modindex.html
    - Communauté officielle Python : https://www.python.org/community/
      - **Jupiter-Lab :**
        - **Ensemble des exercices de ce TD devra être réaliser avec Jupiter-lab**
  - Installation de Jupyter-Lab : https://jupyter.org/install
  - Documentation de Jupiter-lab : https://docs.jupyter.org/en/latest/
  - Utilisation de Jupiterlab pour le champ **Markdown** :
    - https://jupyterlab.readthedocs.io/en/stable/user/file_formats.html
      - **Modalité de Réalisation du TD :**
        - **Document d'audit avec explication de vos recherches**
        - **fonctions Python de l'ensemble des exercices**
        - **Vous devez préparer une présentation pour expliquer l'ensemble des solutions que vous proposer pour réaliser ces exercices**
      - **Modalité de remise du TD :**
        - Votre .zip contiendra obligatoirement :
          - L'ensemble des fonctions Python dans **jupyter-lab au format « ipynb »**
            - Votre ipynb devra contenir obligatoirement:
              - Votre nom et prénom, N° du TD, N°de l'exercice
              - Question et réponse à l'exercice
              - Code Python, une fonction par cellule avec commentaires
              - ...
        - **Rappel des commandes pour Jupiter-lab**
          - le « **!** » permet de lancer une commande
            - par exemple : !ping 8.8.8.8 (dns de google)
            - *le « # » pour les commentaires*

1 Exercices – Gestion de Mail en Python

1.1 Exo1.1 : Écrire une fonction qui sauvegarde au fomat .eml et qui envoie un e-mail en utilisant le module "smtplib" en Python. La fonction doit accepter les arguments suivants : adresse e-mail du destinataire, objet de l'e-mail, contenu de l'e-mail et les informations d'authentification pour le serveur SMTP

1.2 Exo1.2 : Écrire un fonction qui sauvegarde au fomat .eml et qui vérifie si une adresse e-mail est valide en utilisant une expression régulière.

1.3 Exo1.3 : Écrire un fonction qui envoie un e-mail au fomat .eml à une liste de destinataires. Les adresses e-mail des destinataires doivent être stockées dans un fichier texte.

1.4 Exo1.4 : Écrire un programme Python qui lit un fichier EML contenant un message encodé en base64, le décode et affiche le message en clair. Le programme doit utiliser la bibliothèque email de Python pour extraire le corps du message, vous devez utiliser votre code base64 pour décoder le contenu du message. Le programme doit afficher le message en clair dans la console.

1.5 **Exo1.5 :** Écrire une fonction Python qui prend en entrée un objet message de la bibliothèque email, extrait tous les liens hypertexte du contenu du message et les enregistre dans un fichier portant le nom de l'expéditeur ainsi que la date et l'heure de la réception du message. La fonction doit utiliser les bibliothèques email.utils et datetime pour obtenir les informations nécessaires pour le nommage du fichier et la gestion de la date et de l'heure. Le nom du fichier doit être construit en utilisant le format suivant : "expediteur_date_heure.txt".

- Faire un jeu d'essai avec des mail contenant des liens hypertexte.

1.6 Exo1.6 : Écrire un fonction qui vérifie si un nom de domaine est valide en utilisant une expression régulière.

1.7 Exo1.7 : Écrire un fonction qui envoie un e-mail qui envoie un e-mail au fomat .eml avec une pièce jointe. La pièce jointe doit être un fichier PDF.

1.8 Exo1.8 : Écrire un fonction qui envoie un e-mail avec une pièce jointe. La pièce jointe doit être une image.

1.9 Exo1.9 : Écrire un fonction envoie un e-mail en utilisant un modèle HTML pour le corps de l'e-mail.

1.10 Exo1.10 : Écrire un fonction qui sauvegarde au fomat .eml et qui envoie un e-mail en utilisant un modèle HTML pour le corps de l'e-mail et une pièce jointe.

1.11 Exo1.11 : Écrire un fonction qui récupère les e-mails non lus d'une boîte de réception Gmail.

1.12 Exo1.12 : Écrire un fonction qui compte le nombre d'e-mails non lus d'une boîte de réception Gmail.

1.13 Exo1.13 : Écrire un fonction qui envoie un e-mail à plusieurs destinataires avec une pièce jointe différente pour chaque destinataire. Les adresses e-mail des destinataires et les fichiers de pièces jointes doivent être stockés dans un fichier CSV.

1.14 Exo1.14 : Écrire un fonction qui récupère les e-mails de plusieurs boîtes de réception différentes et les fusionne en un seul fichier CSV. Le fichier CSV doit contenir les détails suivants pour chaque e-mail : expéditeur, destinataire, objet, date et heure d'envoi.

1.15 Exo1_15 : Écrire un fonction qui vérifie si un e-mail est une réponse à un e-mail précédent en analysant le contenu du message. Le fonction doit être capable de suivre les fils de discussion et de déterminer si une réponse est une réponse directe ou une réponse indirecte.

1.16 Exo1_16 : Écrire un fonction qui utilise un mail au format .eml et qui envoie des e-mails programmés en utilisant le module "schedule" de Python. Le fonction doit permettre à l'utilisateur de planifier des envois d'e-mails à des heures et des dates spécifiques.

1.17

2 Exercices – Gestion de Mail en Python

2.1 **Exo2_1 :** Écrire un fonction qui extrait les pièces jointes d'un fichier EML et les stocke dans un répertoire spécifié. Le fonction doit utiliser les bibliothèques "email" et "mailbox" de Python pour traiter le fichier EML. Le fonction doit vérifier si le fichier de la pièce jointe existe déjà et éviter de télécharger à nouveau les pièces jointes déjà téléchargées.

2.2 **Exo2_2 :** Écrire un fonction qui convertit un fichier EML en un fichier PDF. Le fonction doit utiliser les bibliothèques "email", "mailbox" et "reportlab" de Python pour traiter le fichier EML et générer le fichier PDF. Le fonction doit inclure les en-têtes du message, le corps du message et toute pièce jointe.

2.3 **Exo2_3 :** Écrire un fonction qui extrait les URL d'un fichier EML et vérifie si elles sont malveillantes en utilisant une API de vérification d'URL. Le fonction doit utiliser les bibliothèques "email" et "mailbox" de Python pour traiter le fichier EML et l'API de vérification d'URL pour vérifier les URL. Le fonction doit afficher les résultats de la vérification pour chaque URL.

2.4 **Exo2.4 :** Écrire un fonction qui utilise les informations d'un fichier EML pour envoyer une réponse automatique au destinataire. Le fonction doit utiliser les bibliothèques "email" et "smtplib" de Python pour traiter le fichier EML et envoyer la réponse automatique. Le fonction doit inclure le texte de la réponse automatique, qui peut être personnalisé en fonction du contenu du message d'origine.

2.5

3 Exercices – Dump d'un Mail en Python

3.1 **Exo3_1 : Écrire un fonction en Python qui permet de lire un fichier EML et d'afficher son contenu sous forme de texte ASCII et sous forme de code hexadécimal(dump).**

- Le fonction doit utiliser la bibliothèque email pour analyser le fichier EML et extraire les en-têtes et le corps du message.
- Le fonction doit également utiliser la bibliothèque binascii pour afficher le contenu du message sous forme de code hexadécimal.
- Le fonction doit afficher les en-têtes du message, le contenu du message sous forme de texte ASCII et sous forme de code hexadécimal.

- Le contenu hexadécimal doit être affiché en groupes de 32 octets par ligne.
- Le chemin du fichier EML à lire doit être spécifié en tant que paramètre d'entrée du fonction.
- Le fonction doit être bien commenté et facilement compréhensible.

3.2 **Exemple de Sortie Attendue**

- nom de fichier: "test.eml"
- Octet de début : 0
- Octet de fin : 25
- Octets par ligne : 10
- 00000000 : 42 4f 4e 4a 4f 55 52 0a 41 55 : BONJOUR.AU
- 00000010 : 20 52 45 56 4f 49 52 0a 41 20 : REVOIR.A
- 00000020 : 44 45 4D 41 49 4E : DEMAIN

- les caracteres non imprimable seront remplacés par des « . »

4 Challenge - 5

4.1 **Challenge 5** :

4.1.1 **Challenge 5a :** Écrire un fonction qui utilise le module "imaplib" de Python pour se connecter à un serveur IMAP et récupérer les e-mails d'une boîte de réception. Le fonction doit stocker les e-mails dans une base de données SQLite et utiliser l'API Natural Language Processing de Python pour extraire les entités nommées et les sujets principaux de chaque e-mail. Le fonction doit également inclure une interface utilisateur qui permet à l'utilisateur de rechercher des e-mails en fonction des entités nommées ou des sujets principaux.

4.1.2 **Challenge 5b:** Écrire un programme qui utilise l'API Gmail pour créer un système de messagerie en temps réel. Le fonction doit utiliser les WebSockets pour recevoir des messages en temps réel et stocker les messages dans une base de données MongoDB. Le fonction doit également inclure une interface utilisateur qui permet aux utilisateurs de s'inscrire et de se connecter, d'envoyer des messages, de recevoir des messages en temps réel et de rechercher des messages en fonction du contenu. Le fonction doit également implémenter une fonctionnalité de chiffrement de bout en bout pour les messages.