Cyber-Sécurité
TD4 - 2023

Scraping/ Python

TD individuel
Attention « bien lire le TD » : 1-Objectif et 2-Réalisation pour le prochain TD

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

1. - Objectif du TD

1.1 L'ensemble des exercices du TD sont faits sans librairie sauf si une librairie est nommée dans l'exercice. Vous devez faire des fonctions Python(def)

1.2 Vous devez faire des copies d'écran de Wireshark si nécessaire pour les intégrer dans NoteBook

- Pour intégrer une image dans en python3 et sur NoteBook :
  - **from IPython.display import Image**
  - **from IPython.display import display**
  - **x = Image(filename='ASCII.png')**
  - **display(x)**

1.3 Ce TD devra obligatoirement être remis à la date et heure indiqué sur DVO

◦ Votre .zip contiendra obligatoirement :

L'ensemble des documents demandé dans le TD seront au format NOTEBOOK « inbpy »
L'ensemble des programmes seront faits en python « .py » sur NOTEBOOK « inbpy » avec commentaires
Toute non remise ou manquement aux consignes sera lourdement pénalisé

1.4 .

2. - Préparation et Réalisation Obligatoire pour le prochain TD

L'évolution d'internet comporte certes de nombreux avantages mais aussi porteuse de risques.

2.1 Pour le prochain TD vous devez obligatoirement faire une **présentation individuelle**

◦ Cette présentation **individuelle** avec au minimum **un Slide par Item** et devra être remise sur Moodle avant la présentation du prochain TD au format powerpoint (.pptx) ou jupiter notebook(.inbpy) :

3. - Description : *Hypertext Transfer Protocol*

3.1 Hypertext Transfer Protocol

L'***Hypertext Transfer Protocol*** (**HTTP**, littéralement « protocole de transfert hypertexte ») est un protocole de communicationclient-serveur développé pour le *World Wide Web*. HTTPS (avec S pour *secured*, soit « sécurisé ») est la variante du HTTP *sécurisée* par l'usage des protocoles SSL ou TLS.

HTTP est un protocole de la couche application. Il peut fonctionner sur n'importe quelle connexion fiable, dans les faits on utilise le protocole TCP comme couche de transport. Un serveur HTTP utilise alors par défaut le **port 80** (**443 pour HTTPS**).

Les clients HTTP : les plus connus sont les navigateurs Web permettant à un utilisateur d'accéder à un serveur contenant les données. Le client envoie un message de requête HTTP au serveur. Le serveur, qui fournit des ressources telles que des fichiers HTML et d'autres contenus, ou exécute d'autres fonctions pour le compte du client, renvoie un message de réponse au client. La réponse contient des informations sur l'état d'achèvement de la demande et peut également contenir le contenu demandé dans le corps du message.

Un navigateur Web est un exemple d'agent d'utilisateur (UA). Les autres types d'agent d'utilisateur incluent les logiciels d'indexation utilisés par les fournisseurs de recherche (robots Web), les navigateurs vocaux, les applications mobiles et les autres logiciels qui accèdent au contenu Web, l'utilise ou le consomme.

HTTP est conçu pour permettre aux éléments de réseau intermédiaires d'améliorer ou d'activer les communications entre les clients et les serveurs. Les sites Web à fort trafic bénéficient souvent de serveurs de cache Web qui transmettent du contenu au nom des serveurs en amont afin d'améliorer le temps de réponse.

Les navigateurs Web mettent en **cache** les ressources Web précédemment utilisées et les réutilisent lorsque cela est possible pour réduire le trafic réseau.

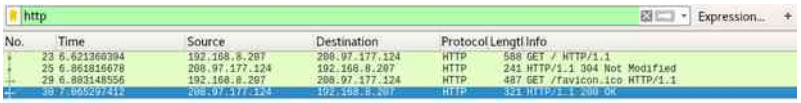
> HTTP est un protocole de couche application conçu dans le cadre de la suite de protocoles Internet. Sa définition suppose un protocole de couche de transport fiable et sous-jacent et le protocole de contrôle de transmission (TCP) est couramment utilisé.

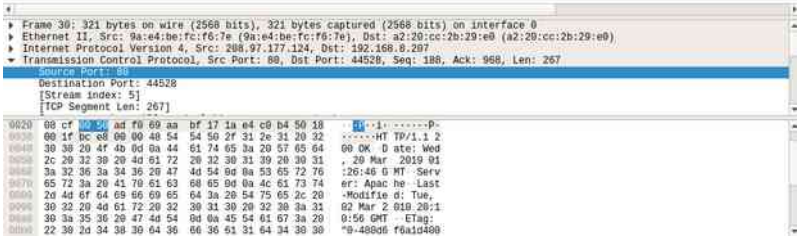
4. - Exercices : *Hypertext Transfer Protocol*

4.1 **Exo1-1** :Démarrez également votre analyseur de paquets Wireshark, en filtrant «http» (uniquement les lettres, pas les guillemets),, afin que seuls les messages HTTP capturés soient affichés ultérieurement dans la fenêtre de liste de paquets. (Nous ne nous intéressons ici qu'au protocole HTTP, et ne voulons pas voir l'encombrement de tous les paquets capturés).

4.2 **Exo1-2** :Écrire une fonction python qui permet de donner le status code d'un site avec la bibliotheque resquests

- ***Lancez votre navigateur Web*** http://www.perdu.com/ pour vous permettre ***de Valider que votre fonction est ok***
- Arrêtez la capture de paquets Wireshark. , Votre fenêtre Wireshark devrait ressembler à la fenêtre ci-dessous:





- L'exemple de la figure ci-dessus montre dans la fenêtre de liste de paquets que deux messages HTTP ont été capturés: le message GET (de votre navigateur au serveur Web) et le message de réponse du serveur à votre navigateur, qui dans ce cas est **une réponse 200 OK avec le contenu HTML**.
- La fenêtre de contenu du paquet affiche les détails du message sélectionné. Rappelez-vous que, puisque le message HTTP était acheminé dans un segment TCP, dans un datagramme IP, acheminé dans une trame Ethernet, Wireshark affiche également les informations relatives aux paquets Trame , Ethernet, IP et TCP.
- essayer avec autres URL en http uniquement
  - : **http://www.hockey-corsaires.com/ pour valider programme python**
  - **: http://www.ecoledulouvre.fr/ pour valider programme python**
- Dans le protocole HTTP, une méthode est une **commande** spécifiant un type de requête, c'est-à-dire qu'elle demande au serveur d'effectuer une action. En général l'action concerne une ressource identifiée par l'URL qui suit le nom de la méthode.

Dans l'illustration ci-contre, une requête GET est envoyée pour récupérer la page d'accueil des sites web http://www.hockey-corsaires.com/ et http://www.ecoledulouvre.fr/

- Il existe de nombreuses autres méthodes, les plus courantes étant GET, HEAD et POST :

- **GET :** C'est la méthode la plus courante pour demander une ressource. Une requête GET est sans effet sur la ressource, il doit être possible de répéter la requête sans effet.
- **HEAD :** Cette méthode ne demande que des informations sur la ressource, sans demander la ressource elle-même.
- **POST :** Cette méthode est utilisée pour transmettre des données en vue d'un traitement à une ressource (le plus souvent depuis un formulaire HTML). L'URI fourni est l'URI d'une ressource à laquelle s'appliqueront les données envoyées. Le résultat peut être la création de nouvelles ressources ou la

modification de ressources existantes. À cause de la mauvaise implémentation des méthodes HTTP (pour Ajax) par certains navigateurs (et la norme HTML qui ne supporte que les méthodes GET et POST pour les formulaires), cette méthode est souvent utilisée en remplacement de la requête PUT, qui devrait être utilisée pour la mise à jour de ressources.

- **OPTIONS :** Cette méthode permet d'obtenir les options de communication d'une ressource ou du serveur en général.
- **CONNECT :** Cette méthode permet d'utiliser un proxy comme un tunnel de communication.
- **TRACE :** Cette méthode demande au serveur de retourner ce qu'il a reçu, dans le but de tester et effectuer un diagnostic sur la connexion.
- **PUT :** Cette méthode permet de remplacer ou d'ajouter une ressource sur le serveur. L'URI fourni est celui de la ressource en question.
- **PATCH :** Cette méthode permet, contrairement à PUT, de faire une modification **partielle** d'une ressource.
- **DELETE :** Cette méthode permet de supprimer une ressource du serveur.

Ces 3 dernières méthodes nécessitent généralement un accès privilégié.

Certains serveurs autorisent d'autres méthodes de gestion de leurs ressources (par exemple WebDAV).

- Il est possible pour réaliser cette exercice que vous soyez obligé de vider le cache du navigateur comme la plupart des navigateurs Web effectuent la mise en cache des objets et effectuent donc un GET conditionnel lors de la récupération d'un objet HTTP.

Avant de suivre les étapes ci-dessous, assurez-vous que le cache de votre navigateur est vide. (Pour ce faire,
sous Firefox, sélectionnez Outils-> Effacer l'historique récent et cochez la case Cache.
Ou pour Internet Explorer, sélectionnez Outils-> Options Internet-> Supprimer le fichier.

Ces actions supprimeront les fichiers en cache du cache de votre navigateur.

Dans Chrome, Si vos outils de développement sont ouverts, maintenez le bouton de rechargement enfoncé pour pouvoir effacer le cache lors du rechargement.

4.3 **Exo1-3 :** En consultant avec wireshark les informations contenues dans les messages HTTP GET et de réponse, répondez aux questions suivantes. : en incluant une capture d'écran indiquant où, dans le résultat, vous obtenez les informations nécessaires

- Votre navigateur utilise-t-il HTTP version 1.0 ou 1.1?
- Quelle est la version de HTTP utilisée par le serveur?
- Quelles langues (le cas échéant) votre navigateur indique-t-il qu'il peut accepter sur le serveur?
- Quelle est l'adresse IP de votre ordinateur?
- Quelle est l'adresse IP du serveur www.perdu.com ?
- Quel est le code d'état renvoyé du serveur à votre navigateur?

- Quand le fichier HTML que vous avez récupéré a-t-il été modifié pour la dernière fois sur le serveur?
- Combien d'octets de contenu sont renvoyés à votre navigateur?

4.4 Il est possible pour réaliser cet exercice que vous soyez obligé de vider le cache du navigateur comme la plupart des navigateurs Web effectuent la mise en cache des objets et effectuent donc un GET conditionnel lors de la récupération d'un objet HTTP.

5. - Python - requests et Beautifulsoup

lien : https://realpython.com/python-requests/
https://www.crummy.com/software/BeautifulSoup/bs4/doc/

5.1 **Exo2-1** :scraping sur : https://pixabay.com/images/search

- Écrire un programme python qui vous permettra récupérer des photos sur un sujet donner par exemple « le ciel » scraping sur https://pixabay.com/images/search/:
- Vérifier avec wireshark le protocole de récupération de l'image et faire une copie d'écran de l'ensemble du protocole de transfert
- rajouter a votre programme la création de répertoire par sujet pour stocker X images
- rajouter a votre programme la possibilité de mettre plusieurs sujets de rechercher avec un répertoire par sujet
- sauvegarder avec la bibliothèque pandas l'ensemble des data techniques **(métadonnées**) de l'image par exemple:
  nom de l'image , appareil :Nikon D800 , objectif :70.0-200.0 mm f/2.8, prise de vue :175.0mm, focale : f/3.5 , vitesse : 1/2s, iso :ISO 800

6. - Python – Scraping requests et Beautifulsoup

6.1 **Exo3-1** : Scraping sur : https://fr.wikipedia.org/wiki/Web_scraping

- Écrire un programme python le texte par le mot clé « Web_scraping»
- Vérifier avec wireshark le protocole de récupération du texte et faire une copie d'écran de l'ensemble du protocole de transfert
- rajouter a votre programme la création de répertoire par sujet pour stocker
- rajouter a votre programme la possibilité de faire des recherches multiples

7. - Python - requests et Beautifulsoup

7.1 **Exo4-1** : Scraping par groupe de 2 avec récupération des data (mail, image, contacts , prix ,autres liens ,...)

- les sites seront donnés par groupe par l'enseignant :
  - fr.linkedin.com
  - Ebay.fr
  - Amazon.fr
  - FDJ
  - Facebook
  - Twitter