Développement Web

2024-2025

Protocole HTTP

Site du cours : `https://defelice.up8.site/devwebbdd.html`
Les exercices marqués d'un @ sont optionnels et à faire dans un second temps.

Quelques commandes :

- Afficher ses adresses ip : `$ip address`
- Pour connaitre l'adresse ip d'un nom de domaine : `host www.machin.fr`
- Lancer netcat prêt à se connecter sur le port 12345 : `$netcat -l -p 12345`
- Lancer netcat pour se connecter sur le port 12345 à l'adresse ip 127.0.0.1 : `netcat 127.0.0.1 12345`
- Indiquer à netcat (option -C) la transformation implicite des caractères fin de ligne LF en CRLF (necessaire pour la plupart des protocoles de communication, http par exemple) : `netcat -C 127.0.0.1 12345`
- Pour lancer un serveur web simple (avec le module python http.server) sur le port 12345 : `python3 -m http.server 12345`
- Le minimum sur les sockets (API de la communication réseau interprocessus) en python
  - Importer le module `socket` qui gère les sockets avec python : `import socket` (`help` de python pour en savoir plus)
  - Créer un socket : `monSock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)` Un tel socket peut servir de socket d'écoute ou de communication.
  - Faire écouter le port 12345 à un socket par l'interface `127.0.0.1` :
    ```
    sockEcoute.bind(("127.0.0.1",12345))
    sockEcoute.listen(5)
    sockCommuni,adresse=sockEcoute.accept()
    ```
    Cette dernière instruction attend jusqu'à ce qu'une tentative de connexion d'un autre socket reussisse. Dans ce cas un nouveau socket de communication `sockCommuni` est créé sur un port libre choisit par le système. `adresse` est simplement l'adresse ip de l'autre socket.
    On peu aussi choisir d'écouter sur toutes les interfaces en utilisant la fausse adresse `0.0.0.0` à l'étape `bind`.
  - Pour établir une communication à un socket qui écoute : `sockCommuni.connect(("127.0.0.1",12345))` Attend une confirmation que la connexion ai été réussie ou refusée. L'adresse à indiquer est l'adresse ip du système sur lequel existe le socket qui écoute le port 12345.
  - Une fois la communication établie entre deux sockets, un socket peut envoyer des octets à l'autre avec : `sockCommuni.send(b"une suite d'octet \ xf3 de toute sorte")`
  - Une fois la communication établie entre deux sockets, un socket peut récupérer des octets (non encore lus) que lui a envoyé l'autre socket avec `chaine=sockCommun.recv(34)`
  - Pour fermer un socket `socket.close()`.

Exercice 1. *Dialogue (fait en tp)*

Retrouver parmi les adresses du tableau les noms des "propriétaires" en dialoguant avec eux grace à la commande netcat.

Exercice 2. *Serveur Test (fait en tp)*

A l'aide par exemple du module `http.server` de python, lancer, sur votre machine, un serveur http possèdant au moins la page "/nom.html" (ne pas remplacer <nom> par votre nom) qui est une page html contenant votre nom (ou un pseudo) sur le port 12345. Vérifiez que votre serveur fonctionne et allez inscrire votre adresse ip au tableau si elle n'y est pas déjà. Aller consulter les autres serveurs web de la classe afin compléter la liste des couples (adresse ip - pseudo).

Exemple : pour consulter une page web à travers le port 12345 : `http://nom.domaine:12345/page.html` ou alors `http://adresse.ip:12345/maPage.html`

Exercice 3. *ENS de lyon*

Écrire dans un fichier requete2.sh une commande qui effectue une requête http pour récupérer une page existante sur un serveur http quelconque (par exemple la page à `http://encorebrussels.com/past/`).

Vous devez utiliser utiliser un serveur http (et pas https).

Vous pouvez utiliser les chainages de commandes avec pipe comme par exemple `$ echo -e "Bonjour\r\n"| netcat ... .`

Exercice 4. *Recherche query*

Écrire dans un fichier requete1.sh une commande qui effectue une recherche (`?q=.....`) sur le mot "recherche" dans la page `www.google.com` en utilisant la requête `GET` du protocole http. La réponse du serveur doit être affichée sur la sortie standard.

Exercice 5. *Votre navigateur*

Donnez un exemple de requête GET envoyée par votre navigateur. Vous pouvez utiliser netcat.

Exercice 6. *Lettre ou Autre*

Écrire un script python `lettre.py` qui écoute sur le port 12345 et qui, après avoir accepté une connexion, a chaque fois qu'il reçoit un octet ayant un code ascii d'une lettre répond par la suite d'octet "LETTRE" Et à chaque fois qu'il récupère un octet ayant une autre valeur répond "AUTRE".

Exercice 7. *Lettre ou Autre 2*

Écrire un script python qui tente de se connecter à l'adresse `127.0.0.1` au port `12345` et y envoie toutes les 5 secondes un octet ayant une valeur aléatoire. Testez-le avec le script `lettre.py`.

Exercice 8. *Serveur*

Écrire en python un programme `monServeur.py` qui, lorsqu'il est lancé se comporte comme un faux-serveur web écoutant sur le port 12345 et ne comprenant que la requête GET du protocole HTTP/1.1. Il peut ne connaître qu'une page (de votre choix) à l'adresse / et doit la renvoyer (vous pouvez aussi l'améliorer en ajoutant d'autre aspect du protocole).