TP4 - Le Protocole HTTP

Conseil avant de commencer : n'hésitez pas à installer l'extension Firefox "Live HTTP Headers".

1 Une première utilisation de telnet

Telnet est un protocole simple de connexion à distance : il permet de transmettre des caractères entre une machine locale (écran + clavier) et une machine distante [1].

Par défaut, un client telnet se connecte au service TCP *telnet* mais il est possible de préciser le service TCP de la façon suivante : `telnet M S`. Dans ce cas le client telnet se connecte au service S de la machine M.

Quel est l'effet de la commande suivante : `telnet time-c.nist.gov daytime` ?

Décrire le protocole *daytime* d'après le résultat de cette commande.

2 Protocole HTTP

Pour le reste du TP, nous travaillerons sur le protocole HTTP. Quelle est le port utilisé par ce protocole ?

Nous utiliserons `telnet` pour envoyer directement des commandes aux serveurs « à la main ». Vous n'utiliserez donc pas la commande `HEAD` depuis le shell, vous utiliserez d'abord `telnet` pour vous connecter au port www avant d'envoyer du texte.

2.1 Méthode `HEAD`

Pour obtenir l'en-tête d'une page web sans pour autant la télécharger complètement, on peut utiliser simplement la méthode HEAD :

```
HEAD / HTTP/1.0
User-Agent: telnet
Host: www.le.domaine.voulu.com
```

Il faudra taper deux fois sur entrée pour que le serveur sache que vous avez fini votre requête.

Il faudra respecter rigoureusement ce format pour que le serveur comprenne votre requête : il faut respecter le protocole à la lettre, sinon c'est l'incident diplomatique !

Si vous ne tapez pas assez vite et obtenez des erreurs "timed out", copiez/collez depuis un éditeur dans lequel vous aurez préparé le texte de la requête.

Observez les différences entre les en-têtes de la page de garde de www.emi.u-bordeaux.fr, www.labri.fr, www.archlinux.org, www.perdu.com ?

Pour chaque exemple, sur quelle machine est implanté le serveur et quel est le type de ce serveur ? Quelle est la classe de réponse ?

2.2 Classes de réponse

En utilisant la méthode `HEAD`, essayez d'obtenir les différentes classes de réponse.

— Succès. `/* facile */`

— Erreur client. `/* vraiment facile */`

1. À essayer depuis chez vous : `telnet towel.blinkenlights.nl`

— Inchangé. `/* plus dur */` [2]
— Redirection. `/* vous ne pouvez pas l'inventer */`.
Sous Firefox ouvrez l'URL `http://dept-info.labri.fr/~thibault/test-redir` D'après vous, que se passe-t-il ?

2.3 Méthode `GET` simple et entêtes

Utilisez telnet pour obtenir le contenu du document `/~thibault/test-redir` sur le serveur `dept-info.labri.fr`. Obtenez vous la même chose que sous Firefox ?

De manière plus simple, on peut utiliser
`wget --server-response http://dept-info.labri.fr/~thibault/test-redir`

2.4 HTTP + ssl : HTTPS

De nombreux sites web sont passés au HTTPS par défaut. Récupérer une page sur un tel serveur avec telnet est difficile car il faudrait faire à la main l'échange de clé de chiffrement, la vérification du certificat, etc. Cependant il est possible d'utiliser un client ssl pour établir cette connexion chiffrée et ensuite effectuer une requette au serveur http exactement comme nous l'avons fait avec telnet. Par exemple grace à la commande `openssl s_client -connect serveur:https`. Essayez de récuperer une page web en HTTPS.

3 Utilisation d'un proxy

Comme vous utilisez régulièrement Firefox comme navigateur, savez-vous si vous avez configuré ce dernier en connexion directe sur internet, ou bien en passant par un proxy ? Si vous ne le savez pas, essayez de trouver cette option de configuration sous votre navigateur.

Depuis l'université, vous pouvez utiliser le proxy `cache.u-bordeaux.fr` sur le port 3128. Avec *telnet*, récupérez la page d'accueil de `http://www.inria.fr` au travers du proxy. Il faut cette fois donner l'URL en entier à la méthode GET. Expliquer les différences entre une connexion avec et sans proxy. Quels sont les intérêts d'un proxy ?

4 Vos traces

Le site `https://korben.info/ip` vous permet d'avoir une idée du genre d'informations qu'un serveur web obtient simplement de votre adresse IP.

Consultez celui-ci en mode accès direct puis en accès via le proxy de l'université, et comparez le résultat de la première expérience (le nom d'hôte, notamment).

Avec accès direct, votre machine a-t-elle été identifiée ?

Même question que précédemment, mais en utilisant l'accès via votre proxy. comme il s'agit d'une connexion securisée, n'oubliez pas de cocher la case pour activer le proxy pour tous les protocoles avec firefox.

5 Cookie et formulaire

Pour aller à l'essentiel, un cookie est un enregistrement d'informations par le serveur sur l'ordinateur client (le vôtre), informations que ce même serveur peut aller relire et modifier ultérieurement. Plus précisément, un cookie se compose d'un ensemble de variables (ou de champs) que le client et le serveur s'échangent lors de transactions HTTP, lesquelles variables sont tout simplement stockées sur la machine cliente. Un cookie est obligatoirement rattaché à un nom de domaine et un ensemble d'URL de telle sorte que seule une requête provenant du même serveur pourra y accéder. Par exemple, grâce à un programme CGI, le serveur a la possibilité de mettre à jour ou d'effacer un cookie. Mais pour cela, il doit spécifier

2. pour un site montrant un champ `Last-Modified`, après le `HEAD` et l'en-tête `Host`, envoyez un en-tête du type `If-Modified-Since: Sat 05 Nov 2005 23:23:59 GMT` (en remettant la même date qu'annoncée par le serveur)

tous les attributs du cookie, par conséquent seul le serveur qui a créé un cookie peut le modifier ou le supprimer.

Sachez que leur fonctionnement est assez simple sous Firefox : les cookies sont stockés dans un fichier binaire qui est une base de données SQLite. Vous pouvez ouvrir ce fichier en utilisant `sqlitebrowser` depuis la ligne de commande.

Examinez votre fichier de cookies (`$HOME/.mozilla/firefox/*.default/cookies.sqlite`) Les tuples ont la structure suivante :

- **id** : clé primaire
- **name** : le nom laissé par le serveur qui a déposé le cookie
- **value** : ça parait évident !
- **host** : le nom du serveur qui a laissé ce cookie
- **path** : restriction sur le chemin d'accès du serveur
- **expiry** : date d'expiration du cookie en secondes depuis le 1er janvier 1970. On peut utiliser `date -d @1477402427` pour convertir en un format plus humain.
- **lastAccessed** : dernière utilisation en microsecondes depuis le 1er janvier 1970
- **isSecure** : si égal à 1, le cookie ne peut transiter que *via* une connexion SSL
- **isHttpOnly** : si égal à 1, le cookie n'est pas lisible par un script côté client

Le formulaire `http://perso.aquilenet.fr/~samuel/cookies/` traite deux cookies « nom » et « fruit » et permet de les mettre à jour. Après avoir soumis le formulaire, constatez que son effet a bien été pris en compte dans la base de données des cookies de Firefox.

Après combien de temps les cookies expirent-ils ?

6 Vraiment à la main

Écrivons en python un outil qui récupère la page d'accueil d'un domaine, que l'on utilisera ainsi : `./httpget.py www.ledomaine.com`.

On se réfèrera à `https://docs.python.org/3.5/library/socket.html` pour les détails de bibliothèque.

6.1 Allons-y !

On commence bien sûr notre script python par `#!/usr/bin/python3` et l'on met le droit `+x` dessus.

Puisque l'on veut ouvrir une connection TCP sur le port 80, il faut faire les étapes dans l'ordre suivant :

- Importer le module de sockets avec `import socket`
- Récupérer le paramètre : `import sys` et l'on pourra utiliser `sys.argv[1]`
- Utiliser `socket.getaddrinfo(sys.argv[1],"http",0,socket.SOCK_STREAM)` pour pour traduire le nom de domaine en adresse IP avec les bons paramètres pour une connexion de type flux (TCP, donc). La réponse est retournée sous forme d'une liste des différentes adresses possibles. Chaque adresse est formée d'un tuple `(family,type,proto,name,sockaddr)` . Les trois premiers éléments seront utiles pour créer la socket en les passant tels quels, et le dernier pour se connecter.
- Pour simplifier dans un premier temps, on utilise juste le premier élément de la liste. Il faut alors
  - créer un objet socket à l'aide de la fonction `socket.socket` en passant en paramètre simplement les trois premiers éléments du tuple.
  - utiliser la méthode `connect()` de l'objet socket, en lui passant simplement le 5e élément du tuple (l'adresse), pour se connecter réellement.
  - Utiliser la méthode `send()` de l'objet socket, il faut que ce soit un tableau d'octets, on peut utiliser simplement `b"GET /\r\n\r\n"` Pensez bien à écrire deux retours chariot, pour introduire la ligne blanche.
  - Utiliser la méthode `recv()` de l'object socket pour récupérer le résultat (on peut lui passer une taille quelconque pourvu qu'elle soit raisonnable). Il suffit d'utiliser un `print` pour l'afficher.

6.2 Raffinements

Pour récupérer toute la page, il faut mettre une boucle autour des appels `recv()` et `print`.

En cas d'erreur lors des étapes de connexion, il faut passer simplement à l'adresse suivante, plutôt que laisser l'exception terminer le programme ! Si la connexion réussit pour une des adresses, on peut passer à la suite. Sinon il faut imprimer l'exception pour indiquer l'erreur à l'utilisateur. On pourra par exemple utiliser `None` pour savoir si la boucle s'est terminée sans parvenir à se connecter à l'une des adresses.

Traitez correctement les cas d'erreur : essayez par exemple avec `localhost` (sur lequel aucun serveur web ne tourne) et avec `trucbidule` (qui n'existe pas), cela lève une exception.