




Étude du protocole HTTP

Première partie

1 Introduction

- Ce TP se déroulera sous l’environnement *XUbuntu*. Nous travaillerons essentiellement dans un terminal de commandes système (Menu(→Accessoires)→Terminal Emulator, raccourci : **Meta+t**, *gnome-terminal*).
- Il est recommandé de consigner vos résultats dans un compte-rendu. Vous pouvez utiliser un éditeur de texte brute (**ALT+F2**, puis **gedit** ou **mousepad**) pour cela ou pour préparer vos fichiers de tests.
- Ce TP porte sur l’étude du protocole HTTP. Les RFC de l’IETF représentent la meilleure source d’information sur le sujet. Vous pouvez cependant utiliser les références rapides suivantes, plus digestes :
 - HTTP headers 
 - HTTP request methods 
 - HTTP response status codes 
- Nous présentons maintenant les outils nécessaires à la réalisation de ce TP.

1.1 La commande netcat

La commande **netcat**, à l’instar de la commande **telnet**¹ ouvre une connexion réseau et y connecte directement, c’est-à-dire sans aucun traitement, son entrée standard (par défaut le clavier) et sa sortie standard (par défaut la sortie du terminal où elle est exécutée). **netcat** permet de simuler « manuellement » les deux coté d’un protocole : coté client, et coté serveur.

Coté client : on utilise **netcat** pour se connecter sur le port d’écoute d’un service réseau. Dans ce cas la syntaxe est :

```
1 netcat localhost port [-q -l]
```

En mode interactif, c’est-à-dire sans redirection des entrées/sorties standards (voir plus bas), tout caractère saisi est envoyé via la connexion réseau. On peut alors mettre fin à la connexion avec la séquence **ctrl+d**, qui symbolise le caractère de fin de fichier.

Note importante : l’option **-q -l** sera nécessaire si vous redirigez l’entrée standard de **netcat**. En effet, le processus de **netcat** se terminent dès la fermeture de l’entrée standard (due à la redirection), ne permettant pas ainsi d’attendre ni d’afficher la réponse.

Coté serveur : **netcat** peut également fonctionner pour simuler un serveur en ouvrant un port de connexion sur l’hôte puis en se plaçant en attente sur celui-ci (*listen*) qu’un client s’y connecte. Le fonctionnement reste le même : les messages reçus sont envoyés sur sa sortie standard et ce qui est reçu sur son entrée standard est envoyé à travers la connexion réseau. Dans ce cas la syntaxe est :

```
1 netcat -l [localhost] port
```

Faites un premier test : pour observer le fonctionnement de **netcat**, effectuez les opérations suivantes :

- ouvrez deux terminaux de commandes et placez les côte à côte ;
- dans le terminal de droite, tapez la commande **netcat -l 4000** ;

1. **telnet** est cependant destinée à se connecter à un interpréteur distant et supporte moins de fonctionnalités.

- dans le terminal de gauche, tapez la commande `netcat localhost 4000` ;
- tapez du texte (peu importe quoi) dans le terminal de gauche et appuyez sur **entrée**, le texte apparaît instantanément dans le terminal de droite ; celui-ci a été transmis par la connexion réseau sur le port 4000 ;
- tapez maintenant du texte dans le terminal de droite et appuyez sur **entrée**, le texte apparaît instantanément cette fois dans le terminal de gauche ;
- pressez la combinaison de touche `ctrl+d` dans l'un des deux terminaux (peu importe lequel) ; la connexion est aussitôt fermée et les deux commandes `netcat` se terminent et rendent la main.

Note : vous ne serez pas en mesure d'ouvrir des ports en écoute (serveur) avec des valeurs inférieures à 1024, des privilèges systèmes étant nécessaires pour cela.

Pour plus d'information sur la commande `netcat`, tapez `man netcat` dans un terminal.

1.2 Gestion avancée des entrées/sorties standards

Les exercices qui suivent demanderont de tester de manière répétée des commandes systèmes qui attendent des saisies. Il peut être très laborieux de les faire en effectuant les saisies à chaque fois, en particulier en cas de faute de frappe. Nous allons à la place préparer nos envois et les envoyer de manière automatique dans l'entrée standard de `netcat` grâce aux redirections des entrées et des sorties.

La première méthode consiste à utiliser la commande `echo`. Celle-ci recopie simplement sur sa sortie standard le texte qu'elle a reçu en paramètre. Nous l'utiliserons avec la combinaison de commutateurs `-en`. Le commutateur `-e` indique à `echo` d'interpréter les caractères spéciaux (comme `\t` qui représente une tabulation) et `-n` qui indique de ne pas rajouter de retour de ligne à la fin de la sortie. Essayez cet exemple :

```
1 echo -en "\thello\r\nWorld!"
```

On rappelle que les deux caractères spéciaux `\r\n` représentent un seul retour de ligne.

L'interpréteur du terminal permet de lancer deux commandes en connectant la sortie standard de la première commande sur l'entrée standard de la seconde en séparant les deux commandes par une barre verticale `|`, appelée *pipe* (*tube*). Nous pourrions l'utiliser pour préparer nos requêtes avec `echo` et les envoyer dans `netcat` :

```
1 echo -en "ma_belle_requete_http..." | netcat ...
```

La seconde méthode consiste à préparer nos requêtes dans des fichiers et à envoyer leur contenu dans l'entrée standard de `netcat`. Nous utiliserons pour cela la commande `cat` qui a pour effet d'ouvrir les fichiers dont les noms lui sont donnés en paramètres et d'envoyer leur contenus respectifs sur sa sortie standard. `cat` est utilisée pour afficher des fichiers mais son rôle réel est de concaténer leur contenu. Placez-vous dans le repertoire `intro` des fichiers fournis avec ce sujet et tapez la commande :

```
1 cat fichier1 fichier2
```

Vous constatez que les contenus des deux fichiers sont envoyés dans le terminal (affichés) comme s'il s'agissait d'un seul fichier. Comme pour avec `echo`, nous utiliserons un tube pour rediriger cette sortie dans l'entrée de `netcat`.

Enfin, il peut arriver que la réponse du serveur soit trop volumineuse pour pouvoir être lue complètement, ou bien que vous souhaitiez plutôt l'envoyer dans un fichier pour la consigner dans votre compte-rendu de TP. La solution dans ce cas consiste à faire l'opération opposée : rediriger la sortie standard de `netcat`. L'opérateur système de redirection est le caractère `>` et s'utilise comme ceci :

```
1 netcat ... > fichier_de_sortie
```

Toutefois, cette approche a l'inconvénient que vous ne pouvez pas voir la sortie de `netcat` en temps réel. Dans ce cas, vous pourriez préférer utiliser la commande `tee` qui enregistre son entrée standard (et donc avec une redirection, la sortie standard d'une autre commande) dans un fichier en plus de l'envoyer sur sa sortie standard :

```
1 netcat ... | tee fichier_de_sortie
```

Bien sûr, il est possible de rediriger en même temps l'entrée et la sortie standard :

```
1 cat mon_test | netcat ... | tee la_reponse
```

Enfin, `less` est une commande bien utile pour visionner un flux, se déplacer dedans et même y rechercher du texte :

```
1 cat mon_test | netcat ... | less
```

Notez bien que dans le cas, le flux est seulement visionné mais pas enregistré. Vous pouvez également l'utiliser sur un fichier :

```
1 less mon_fichier_de_sortie
```

Vous en savez maintenant assez pour expérimenter avec le protocole HTTP.

1.3 Votre serveur Web

Vous disposez d'un serveur Web personnel sur chacun des postes. Celui-ci, dès l'ouverture de votre session, fait correspondre le répertoire `~/TPWeb/` (où `~` représente votre répertoire personnel dans le shell) avec l'URL :

`http://localhost/~votrelogin/`

C'est dans ce répertoire que vous devrez placer vos fichiers lorsque vous aurez besoin d'y accéder via le protocole HTTP.

2 Exercices – Le point de vue du client

Exercice 1 – Différents comportements : premières simulations avec netcat

Question 1.1 : Rappelez le numéro de port utilisé par défaut par le protocole HTTP. Pour les questions suivantes, nous utiliserons l'URL

`http://dpt-info.sciences.univ-rouen.fr/`

(Attention au point entre `dpt-info` et `sciences`, il s'agit de notre ancien site web).

Question 1.2 : En utilisant la commande `netcat`, effectuez une simulation d'une requête HTTP avec la méthode `GET` en « version 0.9 » sur cette URL. Qu'obtient-on en guise de réponse ?

Question 1.3 : Recommencez en formulant une requête minimale cette fois en version 1.0 du protocole.

- Quelle différence observez-vous dans la réponse du serveur ?
- Quelle observation peut-on faire sur la version de la réponse ?
- Quelle partie de la réponse est affichée par le navigateur ?
- Rappelez le code statut d'une requête qui s'effectue avec succès.
- Le contenu de la réponse correspond-il à ce que vous pouvez voir depuis la navigateur ? (chargez l'URL dans firefox) Expliquer pourquoi le résultat diffère. Si vous ne trouvez pas la réponse, la question suivante devrait vous aider.

Question 1.4 : Refaites cette requête en utilisant cette fois la version 1.1 du protocole HTTP. Que faut-il ajouter pour que la requête soit valide ?

Pour les questions suivantes, nous utiliserons l'URL

`http://www.univ-rouen.fr/`

Question 1.5 : Effectuez une requête en 1.1 et analysez l'en-tête de la réponse en rappelant le rôle de chaque ligne. Où commence la ressource ?

Question 1.6 : Selon vous, à quoi correspond le nombre hexadécimal transmis entre l'en-tête et le contenu de la ressource ? Y a-t-il d'autres occurrences ? (Note : vous pouvez utiliser la calculatrice en mode *programmation* pour effectuer la conversion en décimal).

Question 1.7 : Téléchargez la page d'accueil à l'aide de la commande `wget` pour vérifier votre théorie (`wget url -O fichier`).

Exercice 2 – Qui suis-je ? (Schizophrénie avérée ?)

Question 2.1 : Dans votre navigateur, testez les adresses suivantes :

- `heberville.univ-rouen.fr`,
- `mastergeii. formations.univ-rouen.fr` et
- `10.0.128.53`.

Question 2.2 : Résolvez les adresses des deux noms de domaine précédents en utilisant la commande `nslookup` (attention, les premières lignes concernent le serveur DNS lui-même). Que peut-on déduire du résultat ?

Question 2.3 : Expliquez comment il est possible d'obtenir des sites web différents en se connectant sur la même machine et le même port. Le protocole est-il responsable de cela ? Comment ?

Question 2.4 : En utilisant uniquement l'adresse IP pour ouvrir la connexion, simulez une requête `GET` pour ces deux sites avec `netcat`.

Exercice 3 – Requêtes avancées

Copiez le contenu du dossier `reqs` contenu dans l'archive de ce TP dans le répertoire correspondant à votre serveur web personnel. Vous adapterez les chemins URL à utiliser en fonction de l'emplacement que vous aurez choisi (si vous avez choisi de faire des sous-dossiers par exemple).

Pour cet exercice, vous utiliserez des requêtes préparées dans des fichiers et vous conserverez chacun d'eux pour accompagner les réponses à vos questions.

Question 3.1 : Donnez l'URL correspondant au fichier nommé `tux.png`. Testez-la dans un navigateur.

Question 3.2 : Écrivez une requête permettant, sans le télécharger, de récupérer le type, la taille ainsi que la date de dernière modification de la ressource associée à cette URL. Vous indiquerez où trouver chacune de ces informations.

Question 3.3 : Cette fois sur la ressource nommée `long.html`, effectuez tout d'abord une requête `GET` classique, puis recommencez en activant la compression `gzip` pour la réponse. Quel gain obtient-on ?

Question 3.4 : Nous allons simuler l'utilisation d'un cache en effectuant une requête conditionnelle sur la ressource `tux.png`. Écrivez une requête `GET` conditionnelle permettant de télécharger la ressource seulement si celle-ci a pas été modifiée après une certaine date. Testez en utilisant une date située entre la date de dernière modification (voir question 1) et la date actuelle (n'utilisez surtout pas une date dans le futur)².

Question 3.5 : Écrivez un fichier permettant d'effectuer deux requêtes `GET` successives sur une seule connexion (un seul `netcat`), la première sur la ressource `long.html`, la seconde sur `tux.png`. Vous vous assurerez donc que la connexion reste ouverte entre les deux requêtes et vous demanderez au serveur de la fermer après le traitement de la seconde requête.

Nous allons maintenant nous intéresser à l'envoi de données au serveur. Pour cela, nous utiliserons le script `echo.php`. Déterminez son URL.

Question 3.6 : À l'aide de la méthode `GET`, simulez l'envoi d'un formulaire contenant deux champs, `nom` et `prenom` avec des valeurs arbitraires. Le script confirme-t-il la réception de ces données ?

Question 3.7 : Recommencez en utilisant cette fois la méthode `POST`. Pensez à donner autant d'information au serveur que lorsqu'il vous envoie des données !

Question 3.8 : Est-il possible d'envoyer des données par ces deux méthodes simultanément ?

Exercice 4 – Questions bonus

2. Alternativement, vous pouvez appliquer la commande `touch -d "01/01/2000" tux.png` pour vous donner plus de marge de manœuvre sur les dates.

Question 4.1 : En récupérant votre cookie de session dans firefox, récupérez la page d'accueil de votre ENT (`ent.univ-rouen.fr`) en vous assurant que vous êtes loggé.

Question 4.2 : Écrire un script dans le langage de votre choix, en utilisant netcat seulement, qui prend en paramètre une URL et télécharge le contenu de la ressource dans le répertoire courant. On tiendra compte des erreurs possibles (ressource non existante). On pourra également, si un fichier existe déjà, ne l'écraser que si la date de modification a changé.

Question 4.3 : Selon le même principe, écrire un script qui télécharge une page ainsi que les ressources incluses (images, fichiers CSS, javascript, etc).

À suivre, partie 2 :
Le point de vue du serveur.