Feb 18, 2019

1. Objectifs du TP

- Manipuler des threads dans le but de paralléliser une application.
- Utiliser la classe `Net::HTTP`.
- Comprendre l'importance d'un bon password.
- Apprendre à lire de la doc.

2. Présentation du sujet

Soit un serveur Web qui est protégé par une très classique (et un peu vieillotte) "Basic Authentication" (RFC2617 et suivante). Notre but, aujourd'hui, va être d'écrire un programme capable de trouver trois mots de passe pour le déverrouiller.

Attention: le but de cet exercice n'est évidemment pas de vous inciter à faire des choses illégales mais de vous faire pratiquer des notions utiles en Ruby et de vous faire comprendre les dangers d'un mot de passe trop faible.

3. Les outils fournis

Bonne nouvelle: le serveur vous est fourni. Il s'agit d'une simple instance de WEBrick (un serveur web en Ruby), histoire de faire les choses simplement. Il se lance avec un simple `ruby server_obscured.rb`, et écoute à l'adresse `http://127.0.0.1:7654`. Lancé en local, nous ne gênerons pas le réseau même avec un trafic massif. La page à l'URL `http://localhost:7654/epreuve` est protégée par mot de passe. Le serveur a besoin d'un dictionnaire pour changer les mots de passe.

Mauvaise nouvelle: c'est vous qui écrivez le client qui va bruteforcer les serveurs.

Bonne nouvelle: Ruby a (bien sûr) tout ce qu'il faut déjà écrit pour ça. Il vous suffira

de lire la documentation de Net::http. Attention, si vous travaillez avec du ruby1.9, l'URI fonctionne légèrement différemment (le lien de la doc désigne du ruby 2.2): il vous faudra extraire manuellement le path de l'URI (`req = Net::HTTP::Get.new(uri.path)`).

4. Password par défaut

Lancez un navigateur. Lancez le serveur. Allez sur la page du serveur avec le navigateur. Accédez à la page protégée. Le premier password à trouver est évident. Réfléchissez, trouvez…

5. Mot du dictionnaire

5.1. Version simple

À l'aide de `Net::http`, écrivez un client qui va tester tous les mots du dictionnaire, les uns après les autres.

5.2. Version parallélisée.

- Écrivez une classe `Distributeur` dotée d'une méthode `get(n)` qui renvoie `n` candidats password à tester (pour le moment, des mots du dictionnaire).
- Vérifiez que votre classe est *thread-safe* en ayant plusieurs thread qui font des `get(1)`
- Lancez plusieurs thread, qui vont demander des mots au distributeurs, puis les tester, puis recommencer.

6. BruteForce

Maintenant, dotez votre programme de la capacité d'obtenir un mot composé de lettres minuscules et de chiffres. Le troisième des password du serveur est un mot composé de 6 lettres minuscules ou chiffres.

Conseil: pour gérer l'énumération, regardez la méthode

`Array#repeated_permutation`.

Supports de cours pour l'IUT
francois.delobel@uca.fr

TPs mis à disposition le plus simplement possible pour mes étudiants.