UQAC
Université du Québec
à Chicoutimi

TRAVAIL PRATIQUE #1

Automne 2024

Sécurité des Réseaux et du Web
(8INF138)

Groupe 01

Local : P1-4270

Département d'Informatique et de Mathématiques (DIM)

Florentin Thullier

Courriel : **fthullie@uqac.ca**

Bureau : **P3-5040**

Téléphone : **(418) 545-5011 poste 2355**

Introduction

Au cours de ce début de session, nous avons vu plusieurs notions de réseaux qu'il est important pour vous de maîtriser. Ainsi, ce travail pratique a pour objectif de vous permettre de mettre en pratique certains éléments que nous avons vus en classe.

Énoncé

Pour ce travail, vous allez devoir réaliser **en équipe de deux MAXIMUM**, une application console (CLI) qui permet de scanner les ports TCP ouverts sur une machine hôte donnée.

Pour ce faire, je vous laisse le choix du langage de programmation que vous pouvez utiliser parmi les options suivantes :

- Python (version >= 3.8)
- JavaScript/TypeScript (avec le runtime que vous voulez, *p.ex.* node, bun, *etc.*)
- Golang
- Rust
- C++ (version >= 11)
- Java (version >= 1.8)
- C#

Attention : Votre code ne doit pas contenir de dépendances autres que ce qui est inclus dans la librairie standard, et ce, pour chaque langage (ou runtime) qui permettrait de résoudre le problème énoncé plus facilement. Cependant vous pouvez utiliser des librairies externes pour tout ce qui n'est pas en lien avec le sujet du travail, *p.ex.* un linter, un formatter, *etc.*

Vous allez devoir utiliser un **socket TCP**[1] et parcourir l'ensemble des ports réservés pour déterminer lesquels sont ouverts.

Votre programme doit prendre un seul paramètre en entrée qui peut être soit l'adresse IP de la machine hôte cible, soit un nom de domaine. Par exemple, si vous décidez de réaliser le projet en python, voici deux exemples qui doivent permettre de lancer votre programme correctement :

```
$> python3 scanner.py 192.168.0.1

$> python3 scanner.py google.com
```

[1] https://www.ibm.com/docs/en/i/7.2?topic=programming-how-sockets-work

De plus, vous devez gérer les différents cas d'erreur possible :

- Lorsque vous essayez d'exécuter votre programme sans passer de paramètre, celui-ci doit renvoyer une indication sur l'utilisation de votre programme console, tel que :

```
$> python3 scanner.py

Usage: python scanner.py <target ip or hostname>
```

- Lorsque vous entrez un nom de domaine invalide, vous devez renvoyer l'erreur et arrêter proprement l'exécution du programme:

```
$> python3 scanner.py wrongdomain.name

Hostname could not be resolved: [Errno 11001] getaddrinfo failed
```

- Lorsque vous interrompez votre programme en cours d'exécution (avec un `CTRL+C` par exemple), vous devez afficher un message d'information et arrêter proprement l'exécution du programme:

```
$> python3 scanner.py 192.168.0.1

Scanning 192.168.0.1...
Scan is stopping...
Scan completed successfully.
```

Enfin, vous devez afficher un message d'indication pour chaque port ouvert trouvé, mais pas lorsque le port est fermé, comme dans l'exemple ci-dessous:

```
$> python3 scanner.py google.com

Scanning google.com...
--> Port 80/TCP is open.
--> Port 443/TCP is open.
Scan completed successfully.
```

Remise

Vous devrez remettre ce premier travail dans **une archive « .zip »** que vous nommerez **« scanner.zip »** dans l'espace de dépôt prévu à cet effet sur Moodle, et ce, avant le cours du lundi 21 octobre 2024, soit **au maximum à 23h59 le dimanche 20 octobre 2024**.

L'organisation de votre dossier de rendu doit respecter la structure de fichiers comme montré ci-dessous :

```
scanner/
│
├── src/
│   └── **/*.py || **/*.js || **/*.go || **/*.ts || **/*.java ...
│
└── README.md
```

- **Le dossier `src`** doit contenir toute l'arborescence des fichiers et dossiers que vous allez mettre en place pour votre code.

- **Le fichier `README.md`** doit contenir toutes les informations importantes quant à votre projet. Je vous donne le minimum à inclure, mais sentez-vous libre d'ajouter toutes les informations supplémentaires que vous jugerez pertinentes. Vous pouvez vous référer à https://www.markdownguide.org/cheat-sheet/ pour vous aider à formater convenablement ce fichier.

Le contenu minimal de votre fichier **`README.md`** doit être le suivant :

```
# Travail Pratique #1: TCP Port Scanner

## Auteurs
 - Code permanent, nom, prénom
 - Code permanent, nom, prénom

## Compatibilité

Langage – version <!-- p.ex. Python - 3.10 -->

## Utilisation

> incluez dans cette section toutes les instructions nécessaires pour
faire fonctionner votre projet. Quels sont les préalables : est-ce
que je dois installer un environnement d'exécution, si oui, précisez
sa version, donnez le lien de download ; est-ce que j'ai des
dépendances à installer, si oui quelle est la commande pour ce faire
; etc. (supprimez cette consigne pour votre rendu!)
```

Remarque : Toute autre méthode de remise pour ce travail ne sera pas acceptée. Votre travail sera alors considéré comme non remis. De plus, tout non-respect des consignes de remise entrainera une perte de point.

Bonus

2 points bonus seront donnés si vous faites une version multi-threadée du scanner (ou s'en rapprochant, considérant les potentielles limites du langage que vous allez choisir).

Pénalité pour retard

Tout travail remis en retard sans motif valable sera évalué sur 50%. Une pénalité de 10% additionnels par jour supplémentaire sera appliquée après le premier jour de retard (*p. ex.* 3 jours de retard, le travail est évalué sur 30%).