TP évalué, programmation Go

Hugo Blanc - Université Lyon 1
LP ESSIR 2024-2025

December 05, 2024

Le TP est à rendre obligatoirement avant le **dimanche 2 février 2025 à 23h59** (Europe/Paris).
Attention au respect des consignes de rendu (voir Chapitre 3).

Table des matières

1. Énoncé

Le but du TP est d´écrire un programme en Go similaire à `gobuster`. GoBuster est un outil permettant d'identifier les fichiers et répertoires cachés sur un serveur web, et est très utilisé lors de la phase de reconnaissance d'une attaque.

Il fonctionne en effectuant des requêtes HTTP sur un serveur web, en testant différentes URL et en analysant les codes de réponse pour déterminer si les fichiers ou répertoires existent (HTTP 200, 301, 404 etc.).

Les flags à absolument implémenter sont:

- `-d` : chemin vers le dictionnaire qui contient les mots à essayer sur la cible.
- `-q` : mode *quiet*, n'affiche que les résultats qui ont fonctionné (les HTTP 200, pas de headers ni formatage).
- `-t` : permet de définir la *target*, la cible à scanner. Attention, on doit pouvoir scanner une cible sur n'importe quel port, pas seulement les ports 80 et 443.
- `-w` : le nombre de *workers* qui vont scanner en parallèle. La valeur par défaut sera de 1.

2. Usage et sortie du programme

Voici un exemple de ce qui est attendu, avec seulement quelques commandes:

```
$ go run main.go -h
Usage of mygb:
  -d string
      Path to dictionnary file
  -q  When set to true, only show HTTP 200
  -t string
      Target to enumerate
  -w int
      Number of workers to run (default 1)

$ go run main.go -w 10 -d /usr/share/dicts/sec.list -t localhost:8080
Starting MyGB

---
Target: http://localhost:8080
List: /usr/share/dicts/sec.list
Workers: 10
---

Starting scan...
/admin              200
/control-plane      404
/cgi                404
/secret             200

Scan done in 4.879576s
```

3. Rendu

- Une archive **au format zip** est attendue, contenant votre programme et tout autre fichier additionnel qui fera l'objet d'une notation (documentation au format Markdown, dépôt Git, ...).
- Le nom de l'archive devra **impérativement** respecter la convention de nommage suivante: `NOM_PRENOM_GO25.zip`.
- Cette archive devra m'être envoyée par mail. Le sujet du mail devra suivre la convention de nommage suivante: `[ESSIR] Rendu Go 2025 NOM PRENOM`.
- La date de rendu maximale est précisée en début de document.

> ⚠ **Avertissement**
>
> - Veillez à respecter la casse des conventions de nommage.
> - Tout rendu ne respectant pas **tous** ces points **ne sera pas évalué**.
> - En cas de retard, un malus de −1 point par heure de retard sera appliqué à la note finale (heure de réception du mail faisant foi).

4. Critères de notation

La notation se fera selon le barème suivant:

Critère	Détails	Barème
Pas d'erreurs lors du lancement du script	Le programme se lance sans erreurs avec les commandes `go run main.go` et `go build main.go && ./main`.	2
Respect des consignes	Les fonctionnalités demandées sont implémentées.	5
Programmation idiomatique	Le programme est codé de manière idiomatique. Il est rendu avec un ou plusieurs packages, des fichiers go.mod et go.sum, etc.	4
Commentaires et lisibilité	Le programme est lisible et commenté.	4
Documentation	Le programme est rendu avec un fichier `README.md` contenant la documentation du projet. Ce fichier est écrit en Markdown.	2
L'étudiant.e a fait preuve d'initiatives	Couleurs, nouvelles fonctions ou commandes, formats de sortie… Mais attention à ne pas trop en faire: KISS[1]	3
Bonus: dépôt Git	Le TP est rendu avec un dépôt Git correctement utilisé.	2
	TOTAL	**22**

> ✅ **Bonus**
>
> Deux points bonus peuvent être ajouté si le TP est rendu avec un dépôt Git correctement utilisé (commits atomiques, *conventional commits messages*, …). En cas de note maximale, les points seront ajoutés à une autre évaluation.

[1] https://fr.wikipedia.org/wiki/Principe_KISS

5. Licence