

---

**Master CRYPTIS**

---

# **Rapport de projet protocole et programmation réseau**

Réalisation d'un proxy pour le Web (protocole HTTP)

---

**Réalisé par:**

ANTONIO Claudio

MISSAOUI Yasmine

**Professeur :**

Pierre-François Bonnefoi

2021-2022

## 1. Le proxy

Le proxy dans cet projet est un programme python qui fonctionne en tant que client et serveur, on utilise deux thread son bon fonctionnement, la thread main s'occupe du fonctionnement du serveur et une autre de la partie client.

Il est configurable depuis une page html. En effet, la configuration permet à l'utilisateur d'insérer les filtre qu'il veut en fonction des paramètres affichés sur la page html.

La partie serveur du proxy est joignable à l'adresse suivant : **127.0.0.1** et le port **8080**

## 2. Les différentes étapes suivies lors du développement de notre travail

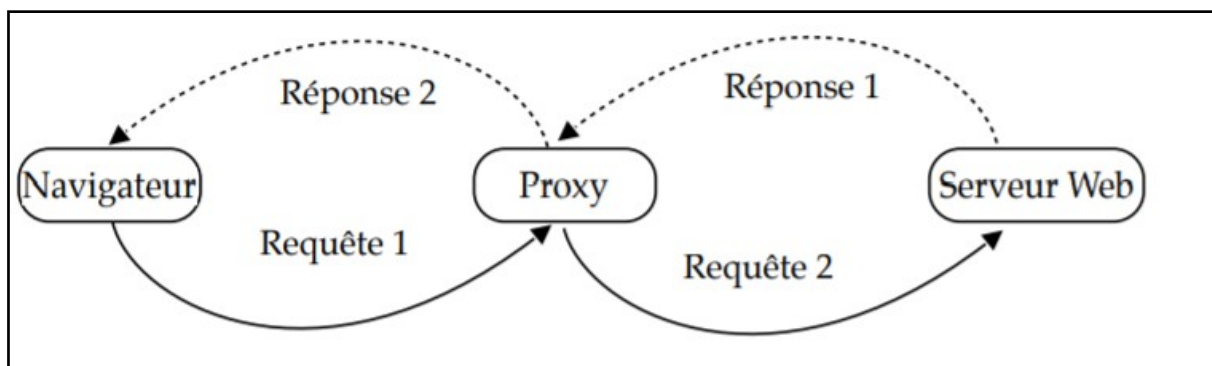


Figure 1:Schéma du proxy

### 2.1. La requête 1

Le navigateur envoie au proxy une requête la thread main du proxy va la récupérer et traiter les données reçues pour avoir le host , le port ( dans le cas où le port n'est pas indiqué, on utilise le port 80 par défaut).

on va supprimer les lignes indésirables qui commencent en :

- Connection : Keep-Alive;
- Proxy-Connection: Keep-Alive;
- Accept-Encoding: gzip ;

Voici un extrait du code qui fait cela :

```
def request(ligne,host):
    liste=ligne.splitlines()
    ## faire la 1ere ligne
    res=""
    if liste[0].split(" ")[0]=="GET":
        res+=liste[0].split(" ")[0]+" "+liste[0].split(" ")[1]+" HTTP/1.0\r\n"
    if liste[0].split(" ")[0]=="CONNECT":
        print(" Le proxy traite que les requêtes HTTP ")
    if liste[0].split(" ")[0]=="POST":
        res+=liste[0].split(" ")[0]+" "+liste[0].split(" ")[1]+" HTTP/1.0\r\n"
    # faire l'en-tete
    for i in range(1,len(liste)):
        if liste[i].split(":")[0] not in ["Connection","Proxy-Connection","Accept-Encoding"]:
            if liste[i]=="":
                res=res.rstrip()
                res+="\r\n\r\n"
            else:
                tmp=liste[i].rstrip()
                res+=tmp+"\r\n"
    return res
```

Figure 2: Extrait du code de la fonction qui prépare la requête

Lors de notre travail , on ne s'intéresse pas au protocole HTTPS. Nous ne traitons que les requête GET et POST du protocole HTTP.

## 2.2. La Requête 2 et réponse 1

Lorsque la requête sera prête, le host et ainsi que le port, la thread main du proxy passe la main à l'autre thread qui va lancer à son tour le client et c'est celui-ci qui va envoyer la requête au serveur web.

Avant l'envoi de la requête, on récupère l'adresse IP du host et pour se faire, on utilise deux méthode :

- **host** : c'est la première méthode , qui utilise l'outil **host** qui permet de récupérer l'adresse ip d'un host.
- **Gethostbyname** : c'est la deuxième méthode, qui utilise la fonction **gethostbyname** du module **socket**.

**Remarque** : Parfois aucune des méthodes n'arrivent pas à récupérer l'adresse ip d'un host donnée.

Voici un extrait du code :

```
cmd_ext = subprocess.Popen('host -t a '+host, stdin=subprocess.PIPE, stdout=subprocess.PIPE, shell=True)
(sortie_standard, sortie_erreur) = cmd_ext.communicate()
if (cmd_ext.returncode != 0): # on teste la valeur de succes de la commande
    try:
        host = "www."+host.split("/")[1]
        adresse_serveur = socket.gethostbyname(host)
    except Exception as e:
        print ("Probleme dans le host ", e.args)
        sys.exit(1)
else :
    adresse_serveur=sortie_standard.split(" ")[-1].split("\n")[0]
```

Figure 3: Extrait du code de la partie qui permet de récupérer l'adresse ip d'un host

quand le serveur web va répondre au client, on va appliquer un filtre sur le contenu html avant d'envoyer la réponse au navigateur.

## 2.3. Réponse 2

- Le client envoie la réponse au navigateur et rend la main au serveur.

## 3. Difficultés rencontrées

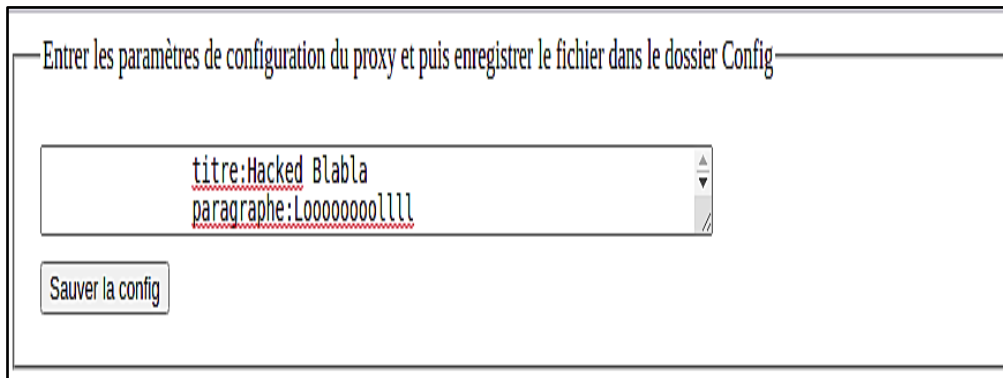
Pour la partie filtre on essaie de changer le contenu de la balise **<p>** mais vu le fait que certains site web ajoute du style css dans cette balise qui fait que les ne soient pas uniforme donc on n'a pas réussi à trouver un standard pour remplacer le contenu sans avoir un impact dans le code html en général.

On a eu un approche mais cela affecte le code html en général c-à-d si il y a deux balises **<p>** une comme celle ici **<p> Bonjour </p>** et une autre

**<p style="color: blue"> Bonjour </p>**, notre algorithme remplace bien la première mais pas la deuxième.

### 3.Scénario d'exécution du proxy

1. Le proxy est configurable depuis une page html, cette page n'est pas lancée par un serveur quelconque. En fait , ce n'est rien d'autre qu'une page html que l'on ouvre avec un navigateur et on utilise un objet **Blob [1]** en JavaScript pour sauvegarder le contenu de forme TEXTAREA en fichier txt. voici un exemple :



Entrez les paramètres de configuration du proxy et puis enregistrer le fichier dans le dossier Config

titre:Hacked Blabla  
paragraphe:Looooooooo1111

Sauver la config

Figure 4: Le form html de configuration

Une fois le fichier de configuration enregistré on doit le placer dans le dossier **Config** du projet

On active le proxy dans le navigateur

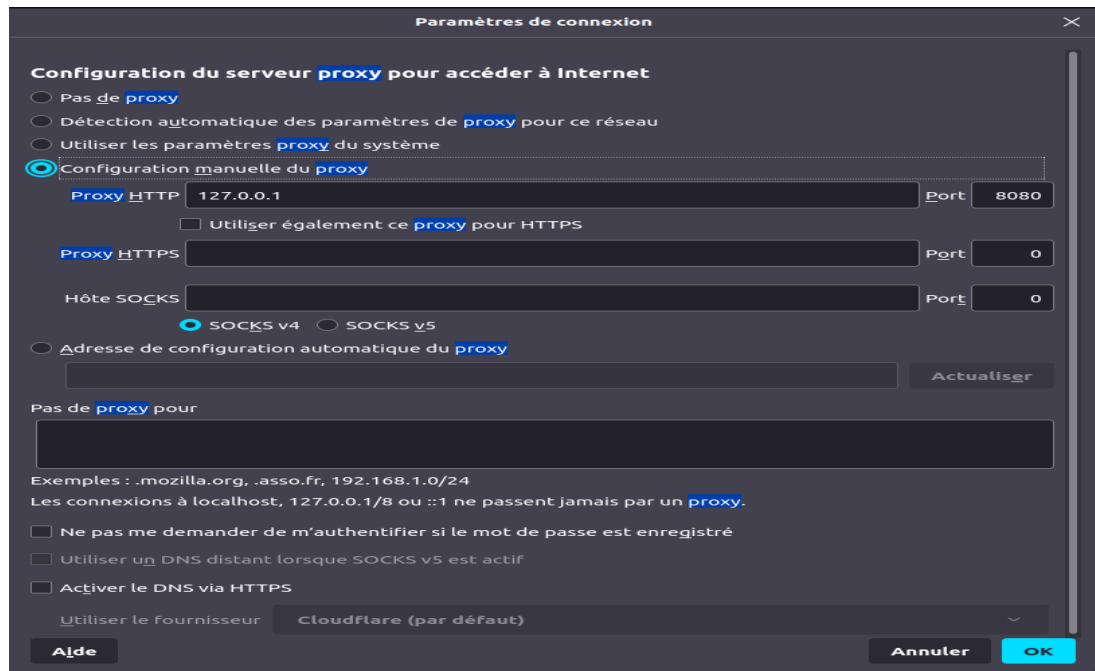


Figure 5: Paramètres pour la connexion au proxy

On lance le programme proxy

```
+ ~/Documents/Proxy-projet main python proxyServer.py  
Server listening .....
```

Figure 6: Lancement du programme

Voici un résultat où on change la balise title de la page html du site <http://cryptis.fr/>.

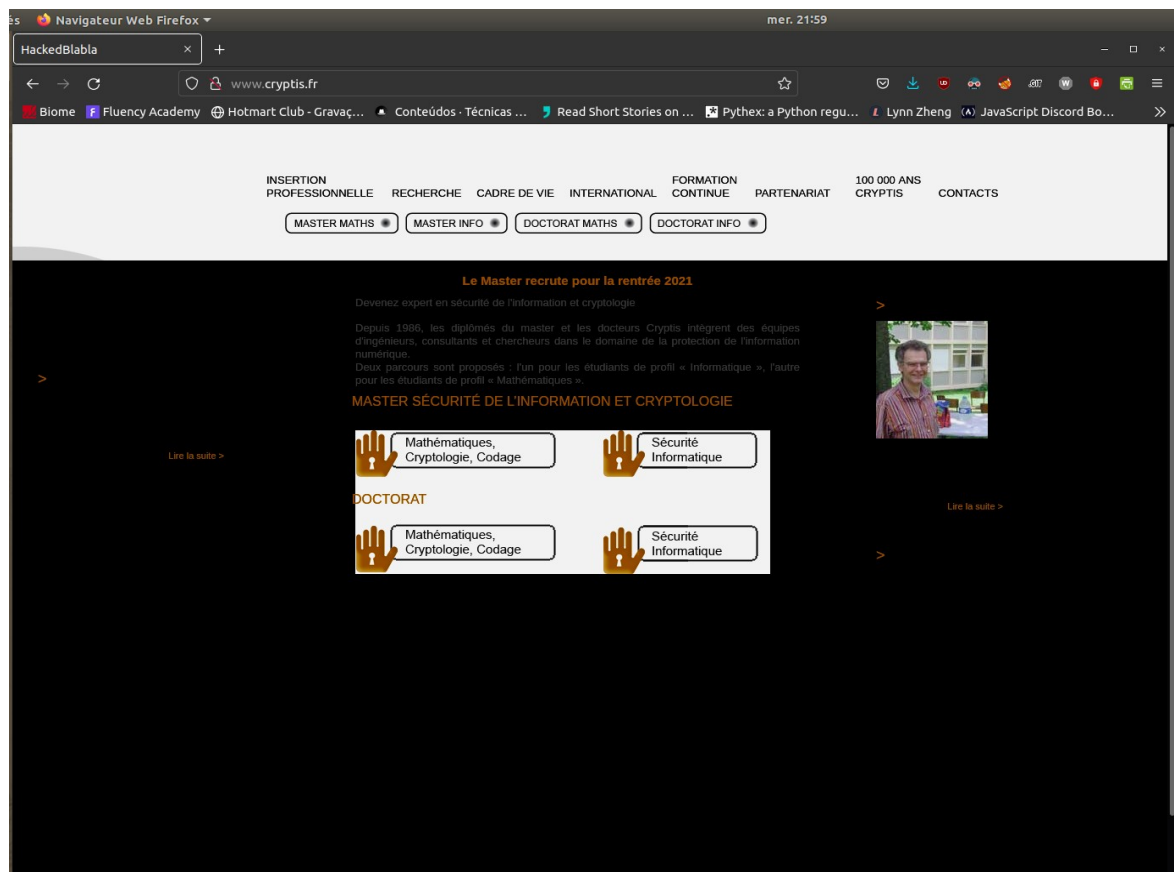


Figure 7: Exemple de la page web qu'on a remplacé son titre

## Références

[1] : <http://purl.eligrey.com/github/FileSaver.js>