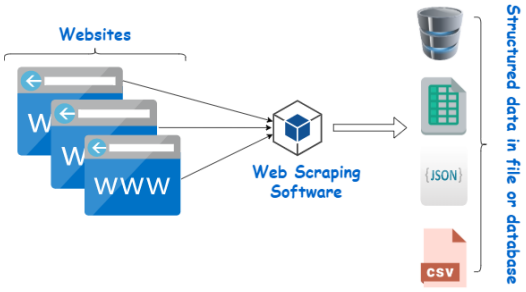
## Sécurité Informatique

# TP6 : Scraping avancé et protection des applications web

**Objectifs :**

- **Maîtriser** les techniques de scraping avancées (bypass CAPTCHA, clics automatisés, scrolling).
- **Construire** des scrapers capables de gérer les sessions et les interactions complexes.
- **Protéger** vos applications web contre le scraping abusif en Python et en PHP (avec et sans framework).
- **Réaliser** un projet final comparant automatiquement les prix d'un produit sur des sites tunisiens.

## A. Introduction au Web Scraping :




Le ***Web Scraping*** est une technique puissante permettant d'extraire automatiquement des données à partir de sites web. Cependant, les sites web mettent en place des mécanismes pour limiter ou bloquer ces pratiques (CAPTCHA, limites de requêtes, etc.).

Ce TP vous guidera dans la mise en œuvre de techniques avancées pour surmonter ces obstacles, comme la gestion de sessions, le traitement des ***CAPTCHA***, ou l'automatisation d'interactions utilisateur (scroll, clics, enchères).

**1.** Installez les bibliothèques nécessaires :

```
pip install requests beautifulsoup4
```

**2.** Testez ce code pour récupérer le contenu de la page de web de votre école ***ESSTHS*** :

```
import requests

url = "https://www.essths.rnu.tn"
response = requests.get(url)

print("Statut HTTP :", response.status_code)
print("Contenu de la page :")
print(response.text)
```

**3.** Testez ce code pour extraire des titres et liens :

```
from bs4 import BeautifulSoup

soup = BeautifulSoup(response.text, "html.parser")
titles = soup.find_all("h1")
links = soup.find_all("a")

print("Titres trouvés :")
for title in titles:
    print(title.text)

print("\nLiens trouvés :")
for link in links:
    if "href" in link.attrs:
        print(link["href"])
```

## B. Techniques avancées de scraping

Les sites web intègrent des mécanismes ***anti-scraping*** que nous allons apprendre à contourner.

### Partie 1 : Gestion des sessions et authentification

Certains sites nécessitent une connexion pour accéder aux données.

**1.** Simulez une connexion avec des cookies :

```
session = requests.Session()

# Connectez-vous
login_url = "https://www.esssths.rnu.tn/login"
payload = {"username": "user", "password": "pass"}
session.post(login_url, data=payload)

# Accédez à une page protégée
protected_url = " https://www.esssths.rnu.tn/home"
response = session.get(protected_url)
print(response.text)
```

**2.** Utilisez ***session.cookies*** pour vérifier les cookies de connexion.

### Partie 2 : Automatisation d'un défilement de page (Scrolling)

Le défilement automatique est utile sur les pages chargées dynamiquement (par ex. : réseaux sociaux, enchères).

**3.** Installez ***Selenium*** :

```
pip install selenium
```

4. Automatiser le scrolling :

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time

driver = webdriver.Chrome()

driver.get("https://www.esstths.rnu.tn")
body = driver.find_element("tag name", "body")

for _ in range(10):  # Scrollez 10 fois
    body.send_keys(Keys.PAGE_DOWN)
    time.sleep(1)  # Pause entre les scrolls
```

### Partie 3 : Automatisation des clics et enchères

5. Localisez un bouton d'enchère et automatisez les clics :

```
bid_button = driver.find_element("id", "bid_button")

while True:  # Automatisation infinie
    bid_button.click()
    time.sleep(10)  # Attendre avant le prochain clic
```

### Partie 4 : Contourner les CAPTCHAs

Les ***CAPTCHA*** peuvent être résolus à l'aide de services tiers comme ***2Captcha***.

6. Soumettre un CAPTCHA :

```
import requests

captcha_api_key = "VOTRE_API_KEY"
captcha_image_path = "captcha.png"

# Soumettre l'image du CAPTCHA
with open(captcha_image_path, "rb") as file:
    response = requests.post(
        "https://2captcha.com/in.php",
        data={"key": captcha_api_key, "method": "post"},
        files={"file": file},
    )
    captcha_id = response.text.split("|")[1]

# Obtenir la solution
result = requests.get(

f"https://2captcha.com/res.php?key={captcha_api_key}&action=get&id={captcha_id}"
)
print("Solution CAPTCHA :", result.text)
```

## C. Sécuriser les applications web

Un script ou programme peut être configuré pour s'exécuter automatiquement à chaque démarrage. Voici trois méthodes courantes :

- **Sécurisation en Python (Flask)**

**1.** Limitez les requêtes avec ***Flask-Limiter*** :

```
from flask import Flask
from flask_limiter import Limiter

app = Flask(__name__)
limiter = Limiter(app, default_limits=["5 per minute"])

@app.route("/")
@limiter.limit("2 per minute")
def home():
    return "Bienvenue !"

if __name__ == "__main__":
    app.run()
```

- **Sécurisation en PHP (sans framework)**

**2.** Implémentez une limite de requêtes basée sur l'IP :

```
<?php
session_start();

if (!isset($_SESSION['last_request_time'])) {
    $_SESSION['last_request_time'] = time();
}

$current_time = time();
$time_diff = $current_time - $_SESSION['last_request_time'];

if ($time_diff < 5) {
    http_response_code(429);
    echo "Trop de requêtes. Réessayez plus tard.";
    exit();
}

$_SESSION['last_request_time'] = $current_time;
echo "Bienvenue !";
?>
```

- **Méthode 3 : Sécurisation en PHP (avec Laravel)**

**3.** Ajoutez un middleware :

```
pip install selenium
```

4. Implémentez le middleware :

```
public function handle($request, Closure $next)
{
    if ($request->session()->has('last_request_time')) {
        $lastRequestTime = $request->session()->get('last_request_time');
        if (time() - $lastRequestTime < 5) {
            return response('Trop de requêtes', 429);
        }
    }
    $request->session()->put('last_request_time', time());
    return $next($request);
}
```

## D. Mini TP Projet : Scraper avancé de comparaison de prix

### 1. Objectif :

Créez un scraper qui :

- Résout les CAPTCHAs.
- Recherche un produit sur **3 sites tunisiens** connus (ex. : Mytek, Jumia, Tdiscount).
- Compare les prix et affiche le meilleur.

### 2. Étapes suggérées :

- **Étape 1 :** Implémentez un formulaire demandant le nom du produit.
- **Étape 2 :** Automatiser la recherche sur chaque site.
- **Étape 3 :** Extraire les prix et les comparer.
- **Étape 4 :** Retournez le résultat sous forme de tableau.

### 3. Livrable attendu :

Un script complet (***Python*** ou ***PHP***) capable de fournir une comparaison en temps réel.