NANTES ynov CAMPUS

Sommaire

1. Découverte de l'outil OWASP Juice Shop sous Debian

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://IP-VM: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 »**

Afin de pouvoir utiliser le **script dirsearch** il est nécessaire d'avoir les commandes **« python3 »** et **« wget »** d'installées sur la machine :

```
sudo apt install -y python3 && sudo apt install -y wget
```

Nous pouvons par la suite installer la commande dirsearch car elle n'est pas utilisable par défaut : http://ftp.fr.debian.org/debian/pool/main/d/dirsearch/dirsearch_0.4.2+ds-3_all.deb

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

```
sudo wget
http://ftp.fr.debian.org/debian/pool/main/d/dirsearch/dirsearch_0.4.2+ds-
3_all.deb

sudo dpkg -i dirsearch_0.4.2+ds-3_all.deb
```

Maintenant il est possible de lancer un scan de toutes les URI disponibles sur OWASP :

```
root@debian:/home/sam/Téléchargements# dirsearch -u http://192.168.1.59:3000


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

Output File: /root/.dirsearch/reports/192.168.1.59-3000/_23-05-01_10-41-
07.txt

Error Log: /root/.dirsearch/logs/errors-23-05-01_10-41-07.log

Target: http://192.168.1.59:3000/


[10:41:07] Starting:

[10:41:18] 200 - 403B - /.well-known/security.txt

[10:41:44] 301 - 183B - /api-docs -> /api-docs/

[10:41:44] 500 - 3KB - /api/2/explore/

[10:41:45] 301 - 179B - /assets -> /assets/

[10:42:06] 200 - 11KB - /ftp

[10:42:27] 200 - 390KB - /main.js

[10:42:49] 200 - 403B - /security.txt

[10:42:50] 400 - 3KB - /servlet/%C0%AE%C0%AE%C0%AF

[10:43:58] 200 - 10MB - /video

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


Task Completed
```

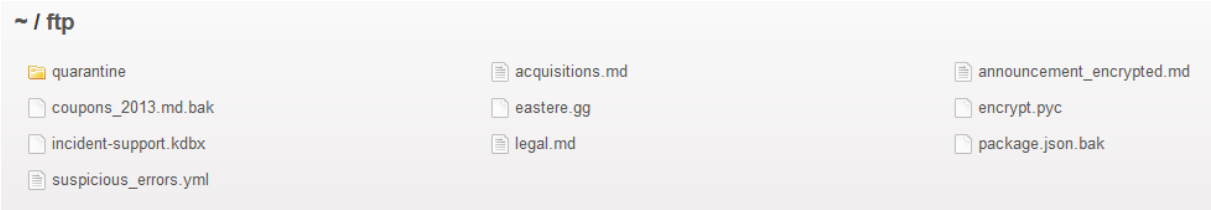
Par exemple ici nous savons que notre site OWASP Juice shop contient une extension **/ftp** et ne contient pas l'extension **/api/2/explore/.**

Toutes les recherches de l'outil sont stockées dans un fichier **/root/.dirsearch/reports/192.168.1.59-3000/_23-05-01_10-41-07.txt**

3. Exploiter une faille pour obtenir un code promo de 80% :

L'objectif ici va t'être d'exploiter une faille du site afin d'obtenir un code promo d'au moins 80%.

Pour commencer nous avons pu voir précédemment le site contient l'extension **/ftp** donc la 1ère chose à faire est d'aller voir ce qu'il contient :



Ici nous pouvons voir qu'il y a un fichier de backup s'appelant **« coupons_2013.md.bak »**

3.1. Attaque par Poison Null Byte

Null Byte Injection est une attaque sérieuse qui consiste à polluer l'entrée d'un site Web (souvent via URL) par une fin de chaîne de caractères mal placée, pour détourner le fonctionnement de celui-ci.

Comme pour l'attaque injection du code arbitraire, la vulnérabilité au Null Byte Injection survient quand les entrées ne sont pas filtrées. Le pirate abuse alors de cette confiance et aspire à avoir des informations confidentielles dont il n'a pas l'autorisation d'approcher.

L'attaque est forgée sur la base du caractère spécial qui représente une fin de chaîne de caractères. Quand ce caractère est encodé en URL il devient **%00**. Sa présence dans la chaîne met fin aussitôt à celle-ci à l'endroit de sa déclaration. Par exemple Si une chaîne de caractères reçue via l'URL a la valeur **Bonj%00our**, alors la chaîne effective qui sera traitée au moment de l'exécution est **Bonj**. La présence de %00 met fin à la chaîne de caractère d'une façon prématurée.

Quand le pirate réussit cette attaque alors il peut avoir accès à des informations confidentielles, comme les fichiers de mots de passe, les codes sources, et parfois même des fichiers système présents en dehors du dossier d'hébergement.

Ici notre objectif est de récupérer le fichier **« package.json.bak »** au format **.md** afin que celui-ci soit lisible et exploitable.

Si nous essayons d'aller sur http://192.168.1.59:3000/ftp/package.json.bak nous avons une erreur nous disant que seuls les fichiers .md et .pdf sont autorisés :

Nous devons donc utiliser l'attaque **« Poison Null Byte »** afin de modifier l'URL et obtenir un fichier au format **.md**

Ce qui donne l'URL suivante : http://192.168.1.59:3000/ftp/package.json.bak%2500.md

Cette URL nous permet d'exploiter une faille du site et de télécharger le fichier de package au format **.md**

Maintenant nous pouvons regarder le contenu du fichier et voir la partie dépendance (celle qui nous intéresse) :

```
"dependencies": {
    "body-parser": "~1.18",
    "colors": "~1.1",
    "config": "~1.28",
    "cookie-parser": "~1.4",
    "cors": "~2.8",
    "dottie": "~2.0",
    "grunt-contrib-concat": "~1.0",
    "grunt-contrib-uglify": "~3.2",
    "hashids": "~1.1",
    "helmet": "~3.9",
    "html-entities": "~1.2",
    "jasmine": "^2.8.0",
    "js-yaml": "3.10",
    "jsonwebtoken": "~8",
    ... ... ... ... ...
    "pdfkit": "~0.8",
    "replace": "~0.3",
    "request": "~2",
    "sanitize-html": "1.4.2",
    "sequelize": "~4",
    "serve-favicon": "~2.4",
    "serve-index": "~1.9",
    "socket.io": "~2.0",
    "sqlite3": "~3.1.13",
    "z85": "~0.0"
```

Nous voyons dans les dépendances utilisées par le site qu'il existe 2 algorithmes de chiffrement (hashids, jsonwebtoken et z85)

Le Twitter d'OWASP est le suivant : https://twitter.com/owasp_juiceshop?lang=fr

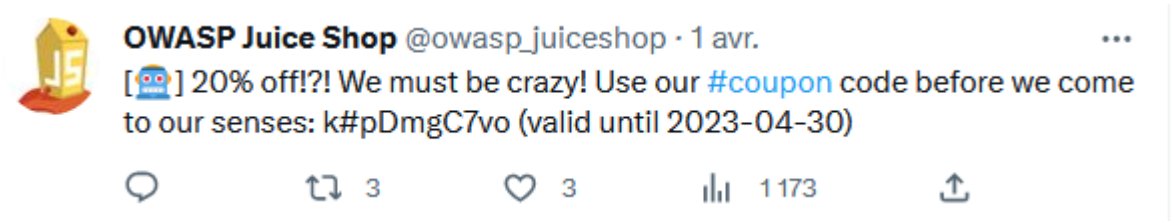
Sur leur Twitter nous pouvons voir que tous les mois il y a un coupon de réduction qui est offert afin de bénéficier d'une réduction.

Nous pourrions donc essayer de voir si les coupons sont chiffrés dans l'un des 3 algorithmes de chiffrement et potentiellement exploiter une faille pour obtenir une meilleure réduction.

3.2. Attaque par Reverse Engineering

Maintenant que nous avons les informations sur l'algorithme de chiffrement des coupons nous pouvons essayer de réaliser du reverse engineering dessus.

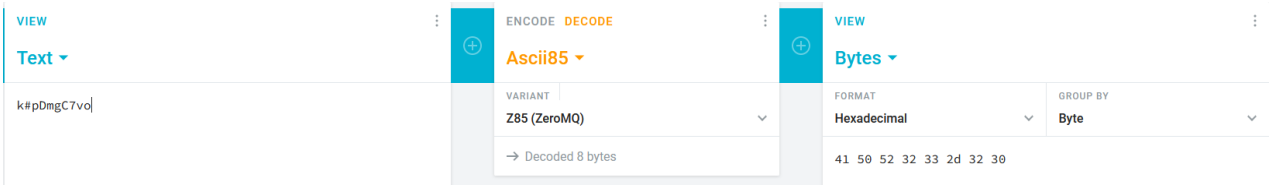
Pour cela il faut tout d'abord récupérer un coupon de réduction sur leur Twitter officiel :



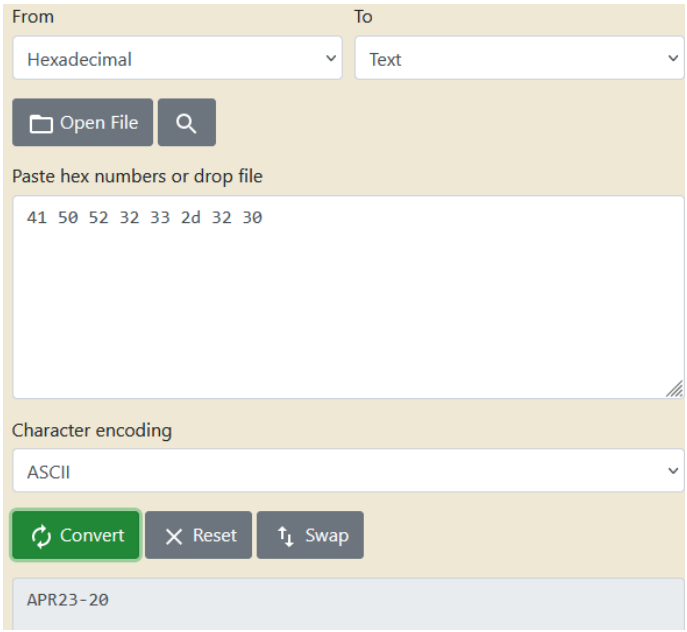
Coupon de réduction de 20% : k#pDmgC7vo

A partir de là il est possible de le déchiffrer en utilisant l'un des 2 algorithmes vu précédemment (ici c'est le z85 qui est utilisé pour les coupons).

Site pour décoder le coupon : https://cryptii.com/pipes/z85-encoder



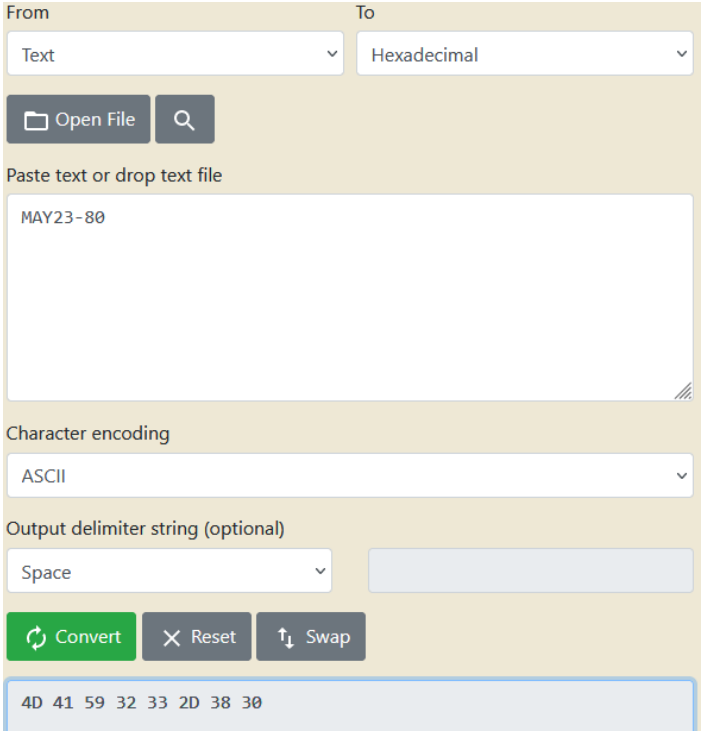
Ici nous obtenons le coupon déchiffré en hexadécimal. Il nous faut donc maintenant le convertir en ascii : https://www.rapidtables.com/convert/number/hex-to-ascii.html



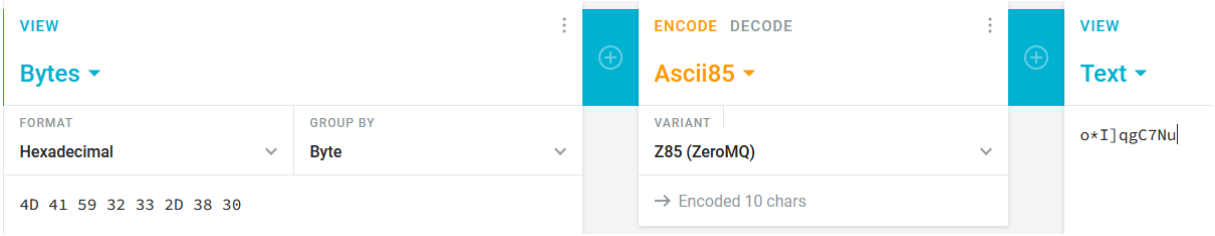
Nous obtenons donc le code de réduction « APR23-20 ». Nous pouvons supposer que le 20 correspond aux 20% de réduction offert par ce coupon et donc nous pouvons tenter de le modifier et de mettre un coupon de 80% ou plus et ensuite le rechiffrer en z85.

/ ! \ Attention : Ce coupon était valide jusqu'à la fin du mois d'Avril. Vu la forme du coupon de réduction nous pouvons en déduire que le coupon de réduction de 80% pour le mois de Mai sera de la forme suivante : « MAY23-80 »

Conversion du coupon vers l'hexadécimal :



Chiffrement du coupon en z85 :



Test du coupon lors d'un paiement en ligne :