Projet Final PHP – Plateforme de Bug Bounty / CTF

Ce projet constitue votre évaluation finale du module PHP. Vous allez réaliser votre première plateforme de bug bounty / CTF 100% en PHP, en réutilisant les concepts d'un site e-commerce (utilisateurs, articles, panier, factures, administration), mais orientés cybersécurité.

L'objectif : concevoir un site où des utilisateurs peuvent acheter l'accès à des challenges de sécurité, résoudre ces challenges, soumettre des flags, et gérer leur profil de pentester.

---

1. Organisation du Projet

- **Travail en groupe de 2 ou 3 personnes maximum.**
- **Backend obligatoirement en PHP,** framework de **VOTRE CHOIX.**
- Si vous choisissez de travailler avec un framework (ce n'est pas obligatoire mais fortement recommandé), merci de **préciser pourquoi** vous avez utilisé ce framework et pas un autre, dans votre soutenance orale.
- **Base de données :** MySQL/MariaDB via PhpMyAdmin.
- **HTML/CSS/JS autorisés** côté front, mais la priorité est à la logique PHP et à la sécurité.

---

2. Concept Général

Vous devez mettre en place une **plateforme de bug bounty / CTF** qui permet :

- à des utilisateurs de s'inscrire et de se connecter ;
- d'acheter l'accès à des challenges de sécurité (web, pwn, crypto, forensic, etc.) ;
- de voir les détails d'un challenge, d'y accéder et de soumettre un flag ;
- de gérer un panier et de valider des "commandes" de challenges ;
- de générer des "factures" (Invoice) pour les achats réalisés ;
- d'afficher un tableau de bord utilisateur (profil pentester) ;
- de proposer un espace administrateur pour gérer les utilisateurs et les challenges.

L'ensemble doit respecter des **bonnes pratiques de sécurité de base** (hash de mot de passe, protection minimale contre les injections, gestion de session, etc, pour ça je vous fais confiance 😉).

---

3. Pages Obligatoires

3.1 Page d'Inscription – /register

Formulaire de création de compte pentester :

Champs minimum :

- username (unique) ;
- email (unique) ;
- password (confirmé) ;
- éventuellement : bio, skill_level (Junior / Intermédiaire / Senior).

Fonctionnalités :

- Le mot de passe doit être **chiffré** via les fonctions PHP appropriées.
- Une fois le compte créé, l'utilisateur doit être **automatiquement connecté** et redirigé vers la page Home.

3.2 Page de Connexion – /login

Formulaire de connexion :

- Authentification par username ou email + password.

En cas de succès :

- création d'une session PHP ;
- redirection vers la page Home.

En cas d'échec :

- message d'erreur générique (éviter de révéler si l'utilisateur existe ou non).

Bonus recommandé (mais pas obligatoire) :

- compteur de tentatives ratées avec verrouillage temporaire du compte après X tentatives.

3.3 Page d'Accueil – /

Page d'accueil listant les challenges disponibles :

Affiche l'ensemble des challenges avec, au minimum :

- titre ;
- catégorie (web, pwn, crypto, forensic, etc.) ;
- difficulté : Noob / Mid / Ca commence à être ardu / Je vais devenir Fou / Give Me CYBERSECURITY title **(les titres peuvent être modifiés)** ;
- prix (en crédits ou en €, pas de $ ou de roubles) ;
- auteur ;
- date de publication.

Les challenges récents doivent apparaître en premier.

Cette page est accessible aux utilisateurs non connectés.

3.4 Page de Détail d'un Challenge – /detail

Affichage détaillé d'un challenge passé en paramètre via la requête GET (ex. /detail?id=3) :

Informations affichées :

- titre ;
- description ;
- catégorie ;
- difficulté ;
- prix ;
- auteur ;
- statistiques (nombre de résolutions, note moyenne, etc. si vous les implémentez).

Possibilités :

Pour un utilisateur connecté :

- ajouter le challenge à son panier ;
- si le challenge est déjà acheté/actif : afficher un formulaire de soumission de flag.

Pour un utilisateur non connecté :

- message l'invitant à se connecter/inscrire pour acheter ou résoudre le challenge.

Cette page est accessible aux non connectés, mais certaines actions (ajout au panier, soumission de flag) nécessitent d'être connecté.

3.5 Page de Création de Challenge – /sell

Page permettant de créer un nouveau challenge (réservée aux utilisateurs connectés, rôle "creator" ou standard selon vos choix) :

Formulaire avec au minimum :

- titre ;
- description ;
- catégorie (liste déroulante) ;
- difficulté ;
- prix (en crédits) ;
- lien d'accès au lab/challenge (URL, IP/port, etc.) ;
- image ou logo du challenge (URL ou upload, en respectant les contraintes de taille) ;
- éventuellement : nombre maximal d'instances / tentatives (stock).

À l'enregistrement :

- insertion en base dans la table des challenges ;
- association avec l'auteur (user connecté).

3.6 Page de Panier – /cart

Page affichant le panier de l'utilisateur :

Liste des challenges actuellement dans le panier (pour l'utilisateur connecté) avec :

- titre ;
- prix ;
- quantité (dans ce contexte, ça peut représenter le nombre de "slots" d'accès ou rester à 1).

Fonctionnalités :

- possibilité de supprimer un challenge du panier ;
- possibilité de modifier la quantité si vous gérez un stock (ex. labs à places limitées) ;
- affichage du total ;
- bouton "Valider le panier" si le solde de l'utilisateur est suffisant.

Un utilisateur non connecté doit être redirigé vers /login.

3.7 Validation du Panier – /cart/validate

Page de confirmation et de traitement de la commande :

Vérifie que le solde de l'utilisateur est suffisant pour acheter tous les challenges du panier.

Si oui :

- décrémente son solde ;
- met à jour les stocks (si gérés) ;
- insère une entrée dans la table Invoice ;
- vide le panier ;
- génère l'accès aux challenges (logique à définir : simple marquage comme "acquis" ou gestion d'instances).

L'utilisateur doit pouvoir saisir des informations de facturation :

- adresse ;
- ville ;
- code postal.

Affiche un récapitulatif de la commande validée.

3.8 Page de Modification de Challenge – /edit

Page accessible principalement depuis la page détail d'un challenge via une méthode POST :

Permet de :

- modifier les informations d'un challenge (titre, description, prix, difficulté...) ;
- supprimer un challenge.

Sécurité :

- seul l'auteur du challenge peut le modifier/supprimer ;
- un administrateur peut modifier/supprimer n'importe quel challenge.

3.9 Page de Compte – /account

Page permettant d'afficher les informations d'un compte utilisateur.

Deux comportements possibles :

1. Sans paramètre GET (ou ?id correspondant à l'utilisateur connecté)

Afficher les informations du compte actuel :

- pseudo, email, bio, niveau, photo de profil, solde ;

Afficher :

- les challenges créés par cet utilisateur ;
- les challenges achetés/résolus ;
- les factures (invoice) associées ;
- son score total (points cumulés).

Possibilité de :

- modifier ses informations (email, bio, mot de passe, photo, etc.) ;
- ajouter des crédits à son solde (simulation).

2. Avec paramètre GET (ex. /account?id=5)

Afficher le profil public d'un autre utilisateur :

- pseudo, avatar, niveau, nombre de challenges créés/résolus ;

Afficher uniquement les informations publiques (pas d'email, pas d'options de modification).

3.10 Espace Administrateur – /admin

Espace dédié aux administrateurs uniquement :

Accessible uniquement si l'utilisateur connecté a le rôle admin.

Doit permettre de :

- **lister tous les challenges :**
    - possibilité de les modifier/supprimer ;
    - possibilité de les désactiver temporairement (ex. challenge compromis).
- **lister tous les utilisateurs :**
    - possibilité de changer leur rôle (user/creator/admin) ;
    - possibilité de bannir/suspendre un compte ;
    - possibilité de remettre à zéro un score/solde si besoin.

Vous pouvez structurer cette partie en plusieurs sous-pages (ex. /admin/users, /admin/challenges) pour une meilleure lisibilité.

---

4. Schéma de Base de Données Minimal

Votre base doit contenir au minimum les tables suivantes. Vous pouvez en ajouter si nécessaire.

4.1 Table User

Contient les informations sur les utilisateurs de la plateforme.

Champs minimum recommandés :

- id ;
- username ;
- email ;
- password ;
- balance ;
- role ;
- profile_picture ;
- bio ;
- skill_level ;
- created_at.

4.2 Table Challenge

Contient les informations sur chaque challenge proposé.

Champs minimum recommandés :

- id ;
- title ;
- description ;
- category ;
- difficulty ;
- price ;
- author_id ;
- image_url ;
- access_url ;
- flag_hash ;
- created_at ;
- is_active.

4.3 Table Cart

Relie un utilisateur à des challenges présents dans son panier.

Champs minimum :

- id ;
- user_id ;

- challenge_id ;
- quantity.

4.4 Table ChallengeInstance

Permet de gérer une notion de stock ou d'instances disponibles pour un challenge.

Champs minimum :

- id ;
- challenge_id ;
- available_instances.

4.5 Table Invoice

Représente une facture générée lors d'un achat de challenge(s).

Champs minimum :

- id ;
- user_id ;
- date ;
- amount ;
- billing_address ;
- billing_city ;
- billing_zip.

4.6 Table Submission

Permet de suivre les soumissions de flag.

Champs recommandés :

- id ;
- user_id ;
- challenge_id ;
- flag_submitted ;
- is_valid ;
- submitted_at.

---

5. Fonctionnalités Supplémentaires (Bonus)

Pour obtenir des points supplémentaires, vous pouvez implémenter certaines des fonctionnalités suivantes.

5.1 Système de Favoris

- Permettre à un utilisateur de **marquer un challenge en favori**.
- Ajouter une section "Challenges favoris" sur la page /account.

5.2 Barre de Recherche Avancée

Ajouter une barre de recherche sur la page Home :

- recherche par titre ;
- filtres par catégorie, difficulté, prix, type de vulnérabilité (XSS, SQLi, XXE, etc.).

5.3 Classement (Scoreboard)

- Calculer un score par utilisateur basé sur :
  - le nombre de challenges résolus ;
  - la difficulté des challenges ;
  - éventuellement le temps de résolution.
- Afficher un leaderboard des meilleurs pentesters.

5.4 Système de Notation et de Commentaires

Permettre aux utilisateurs ayant résolu un challenge de :

- lui attribuer une note (1 à 5 étoiles) ;
- laisser un commentaire/retour d'expérience.

Précaution : penser à protéger contre les attaques XSS (échappement des caractères HTML).

5.5 Envoi d'E-mails

Envoi d'e-mails pour :

- confirmation d'inscription ;
- mot de passe oublié ;
- confirmation d'achat de challenge ;
- notification d'acceptation ou d'invalidation d'un flag (si vous implémentez une validation manuelle).

5.6 Sécurité Applicative Renforcée

- Utilisation de **requêtes préparées** pour toutes les interactions avec la base (protection SQLi).

- Validation et nettoyage des entrées utilisateur.
- Gestion correcte des sessions (regénération d'ID de session après login, destruction à la déconnexion).
- Gestion des rôles (user, creator, admin) côté backend (ne jamais se fier uniquement au front).

---

6. Guide de Mise en Place

6.1 Environnement de Travail

Vous devez installer un environnement de type LAMP/MAMP/XAMPP selon votre OS.

Windows : XAMPP (PHP 8).

macOS : MAMP (configuré en PHP 8 si possible).

Linux :

- Ubuntu/Debian : `sudo apt-get install lamp-server^`
- Arch/Manjaro : `sudo pacman -S httpd php php-apache mysql phpmyadmin`
- Après installation :
  - `sudo systemctl enable --now httpd`
  - `sudo systemctl enable --now mysql`

Le dossier racine de votre serveur sera typiquement :

- **XAMPP :** htdocs (via le bouton "Explorer") ;
- **MAMP :** /Applications/MAMP/htdocs ;
- **LAMP :** /var/www (ou /var/www/html selon la config).

Créez un dossier `php_exam` dans ce répertoire et placez-y votre fichier `index.php`.

6.2 Test Rapide de PHP

Dans `php_exam/index.php` :

Puis rendez-vous sur :

- `http://localhost/php_exam`
- ou `http://localhost:8888/php_exam` (MAMP par défaut).

Tout le reste du projet devra être développé dans ce dossier.

6.3 Base de Données et PhpMyAdmin

Rendez-vous sur `http://localhost/phpmyadmin` :

1. Créez une nouvelle base de données, par exemple `php_exam_db`.
2. Créez les tables décrites dans la section "Schéma de base de données minimal".
3. Identifiants de connexion classiques (en local, à adapter selon votre installation) :
    - user : root
    - mot de passe : root ou vide

Exemple de connexion PHP :

connect_error) { die("Erreur de connexion : " . $mysqli->connect_error); } $result = $mysqli->query("SELECT * FROM User"); ?>

Organisez votre code afin d'**éviter la duplication** de ce bloc de connexion dans tous vos fichiers (par exemple via un fichier de configuration inclus).

---

7. Rendu Attendu

7.1 README

Votre README doit contenir au minimum :

- une **description du projet** ;
- les **prérequis** (version de PHP, MySQL, extensions éventuelles) ;
- les **instructions d'installation** :
    - création de la base ;
    - import du fichier SQL ;
    - configuration des identifiants de connexion ;
- les **identifiants de test** (un compte utilisateur simple, un compte admin) ;
- un **bref résumé des fonctionnalités implémentées** (obligatoires + bonus).

7.2 Fichier SQL

Vous devez **impérativement** fournir un fichier SQL permettant de générer la base de données :

Dans PhpMyAdmin :

- sélectionnez votre base `php_exam_db` ;
- onglet Exporter ;
- cliquez sur Exécuter ;
- récupérez le fichier `php_exam_db.sql`.

Ajoutez ce fichier à la racine de votre repo.

Un malus de 7 points sera appliqué si ce fichier n'est pas présent dans le repo.

7.3 Contraintes Supplémentaires

Le rendu se fait sous la forme :

- d'un **repository git** contenant :
    - tout le code source PHP ;
    - les fichiers statiques (HTML, CSS, JS, images…) ;
    - le fichier d'export SQL de la base de données ;
    - un fichier README clair et détaillé.

Le dossier racine du repo doit correspondre au contenu de votre dossier `php_exam`.

Attention à la taille des fichiers (photos de profil, images…) si vous utilisez un Gitea ou autre forge avec limites (par exemple 1 Mo par fichier).

Aucune technologie backend autre que PHP n'est autorisée.

---

8. Consignes de Notation

La notation portera principalement sur :

- la **fonctionnalité globale du site** (respect des pages et scénarios demandés) ;
- la **qualité du code PHP** (organisation, lisibilité, réutilisation) ;
- la **gestion de la base de données** (schéma, relations, cohérence) ;
- la **prise en compte de la sécurité** :
    - hash des mots de passe ;
    - protection basique contre les injections SQL ;
    - gestion des sessions ;
    - gestion des rôles et des accès ;
- les **fonctionnalités bonus implémentées** (scoreboard, favoris, commentaires, recherche avancée, envoi d'e-mails, etc.) ;
- la **qualité de la documentation** (README).

Le design graphique du site est secondaire : un site simple mais fonctionnel, propre et relativement sécurisé sera mieux noté qu'un site très joli mais fragile ou incomplet.

---

9. Date Limite

Le projet est à rendre avant le **Jeudi 19 FÉVRIER à 23h59**. Votre soutenance aura lieu le Vendredi 20 Février avec Sara.

Seul le dernier commit avant cette date sera pris en compte. Tout autre commit au-delà de cette date sera considéré comme non rendu (et je sais qu'on peut modifier la date des commit, mais je suis admin sur Gitea, donc je peux voir si vous avez modifié la date, essayez pas de m'entuber parce que je chercherai pas à comprendre, ça sera 0 si je vous crame)

Le lien vers votre repository devra être transmis sur **votre excel de groupe** (Les groupes n'ayant pas de lien se verront attribué **un malus de 3 points**)

Bon courage à tous, j'ai 100% confiance en vous vous pouvez le faire (je l'ai fait en 1 semaine et demi pour tester)

Si vous avez des questions, vous pouvez m'envoyer un message Teams ou Discord, je répondrai dès que je serai disponible dans des heures de travail normal (pas de message à 2h du mat svp ça réveille ma copine)

Maxime ISIDORE

Mentor Cybersécurité / Informatique

Bordeaux YNOV Campus