🖥️ Rapport de Test – Challenge Web : “Where is Mona Lisa?”

Auteur : Morad Halmi **Établissement :** École2600 **Date :** Samedi 24 mai

🎯 Objectif

Identifier où se trouve la "Mona Lisa" sur un site présentant différentes œuvres de Léonard de Vinci. L’objectif est de localiser un flag, potentiellement caché via un défaut d’implémentation.

🔍 Démarche et investigation

1. Reconnaissance initiale

Lors de la navigation sur la page principale, j’ai observé :

- L’affichage d’un tableau différent à chaque rafraîchissement (probablement aléatoire ou basé sur une base de données côté serveur).
- Un bouton nommé **"Admin Page"** menant vers `/login`.

J’ai inspecté :

- Le code HTML
- Les cookies de session
- Les images affichées dynamiquement

2. Tentatives d'accès non autorisé

En analysant les chemins accessibles, j’ai manuellement testé :

```
/admin
/monalisa
/flag
/hidden
/view?painting=monalisa
/static/images/Mona-Lisa.jpg
```

Aucun de ces chemins ne m’a mené au flag.

🧪 Techniques explorées

Test de stéganographie

- Téléchargement de l'image `Last-Supper.jpg`
- Exécution de plusieurs outils :
  - `strings` pour extraire du texte brut
  - `steghide` pour tenter une extraction dissimulée (aucun résultat, pas de mot de passe évident)
  - `binwalk` (non disponible sur mon environnement au moment du test)

Tentatives d'accès administrateur

Pensant que le flag pouvait se trouver derrière une interface d'administration, j'ai inspecté `/login` :

- Formulaire HTML présent
- Réponse serveur : `"Welcome! We're still building this page..."`

Ce comportement m'a fait penser à un backend inachevé mais potentiellement vulnérable.

💉 Injection SQL

J'ai tenté une **injection SQL classique** dans le champ **username** :

```
admin' OR 1=1 --
```

Le site m'a redirigé vers la même page, mais par réflexe, j'ai vérifié le **code source HTML** de la réponse. À l'intérieur, j'ai trouvé un commentaire contenant le flag :

```
<!-- DVCTF{e0isCuyMeFXoT8aCBBiy} -->
```

✅ Résultat

Le flag obtenu est :

```
DVCTF{e0isCuyMeFXoT8aCBBiy}
```

🔐 Vulnérabilité identifiée

- **Injection SQL** permettant un contournement d'authentification.

- **Fuite d’information sensible via un commentaire HTML** dans la page `/login`.

🛡️ Recommandations

1. Utiliser des requêtes SQL préparées (`prepared statements`) pour toutes les interactions avec la base de données.
2. Ne jamais inclure de données sensibles ou de flags dans le code source côté client.
3. Implémenter des contrôles d’accès stricts, même sur des pages "en cours de développement".
4. Ajouter un monitoring/logging côté serveur pour détecter des comportements anormaux comme des injections SQL.

🧩 Conclusion

Ce challenge m’a permis de mettre en œuvre une approche progressive typique d’un testeur junior :

- Phase de reconnaissance
- Exploration manuelle
- Tests d’injection
- Recherche stéganographique

La solution finale a été obtenue par l’exploitation d’une **faille SQL classique**, démontrant l’importance de sécuriser chaque couche d’un site web, y compris les pages encore en développement.

Morad Halmi École2600 Samedi 24 mai