SPÉCIMEN · RAPPORT D'EXEMPLE

Rapport de **test d'intrusion**

Application web & API REST :
évaluation en boîte grise authentifiée

CLIENT

OWASP Juice Shop

Boutique e-commerce & API REST · juice-shop.example

RÉFÉRENCE HX16-2026-0142	DATE DU RAPPORT 8 juillet 2026
TYPE DE TEST Boîte grise, authentifié	MÉTHODOLOGIE OWASP WSTG · API Top 10

Saïd Mimouni
Pentester — hex16

hex16.fr
contact@hex16.fr

01 · SYNTHÈSE EXÉCUTIVE

Six vulnérabilités, dont **une critique** exploitable sans privilège.

Un test manuel de 7 jours en boîte grise a mis en évidence des défauts de contrôle d'accès et de logique métier permettant à un utilisateur authentifié d'accéder aux données d'autres clients et de déclencher des remboursements non autorisés.

1	2	2	1
CRITIQUE	ÉLEVÉ	MOYEN	FAIBLE

VERDICT

La plateforme présente une **isolation entre comptes clients insuffisante**. La faille critique **F-01** permet à n'importe quel compte de lire les factures d'autres clients en modifiant un identifiant dans l'URL, un scénario directement bloquant pour une revue SOC 2 ou un contrôle de sécurité client.

RECOMMANDATION PRIORITAIRE

Corriger en priorité **F-01, F-02 et F-03** : elles partagent une cause racine : l'autorisation est vérifiée au niveau de la route mais **jamais au niveau de l'objet**. Un contrôle d'accès centralisé côté serveur ferme les trois d'un coup.

Contexte du test. L'évaluation a couvert l'application web et son API REST, en accès authentifié avec deux comptes clients distincts fournis par l'équipe. L'objectif : mesurer ce qu'un utilisateur légitime (ou un client malveillant) peut atteindre au-delà de son propre périmètre. Les tests ont été menés à la main selon l'OWASP WSTG et l'API Security Top 10, sans action destructive, sur l'environnement de recette convenu.

02 · PÉRIMÈTRE & MÉTHODOLOGIE

Ce qui a été testé, et comment.

CIBLE APPLICATIVE	CIBLE API
app.juice-shop.example	api.juice-shop.example/v1
TYPE DE TEST	**FENÊTRE DE TEST**
Boîte grise, authentifié · 2 comptes	7 jours ouvrés · juin 2026

03 · REGISTRE DES VULNÉRABILITÉS

RÉF.	VULNÉRABILITÉ	SÉVÉRITÉ	CVSS	STATUT
F-01	Accès aux factures d'autres clients (BOLA) GET /api/v1/invoices/{id}	● CRITIQUE	9.1	Ouvert
F-02	Rejeu d'une requête de remboursement POST /api/v1/refund	● ÉLEVÉ	8.2	Ouvert
F-03	Élévation de privilège par mass assignment PATCH /api/v1/users/me	● ÉLEVÉ	8.1	Ouvert
F-04	XSS stocké dans le nom d'un widget Dashboard · champ « title »	● MOYEN	6.4	Ouvert
F-05	Jetons JWT sans révocation, expiration 30 j Authentification · session	● MOYEN	5.9	Ouvert
F-06	Absence de limitation de débit sur la connexion POST /api/v1/login	● FAIBLE	3.7	Ouvert

La sévérité combine l'impact technique (CVSS v3.1) et le contexte métier réel observé pendant le test. Les fiches détaillées des trois vulnérabilités les plus graves suivent ; F-04 à F-06 sont décrites en annexe technique du rapport complet.

04 · CONSTATS DÉTAILLÉS

F-01

Accès aux factures d'autres clients (BOLA / IDOR)

9.1
CRITIQUE · CVSS 3.1

ENDPOINT	CATÉGORIE
`GET /api/v1/invoices/{id}`	API1:2023 · BOLA

DESCRIPTION

L'endpoint autorise l'accès sur la **seule base de la session authentifiée**, sans vérifier que la facture appartient au compte de l'appelant. L'identifiant `{id}` étant un entier séquentiel, il suffit de l'incrémenter pour lire les factures de **tous les autres clients** : coordonnées, montants, adresses, articles commandés.

PREUVE D'EXPLOITATION

```
# Client Bob (compte standard). Sa propre facture : id 3041.
$ curl -s .../v1/invoices/3041 -H "Authorization: Bearer <bob>"
200 OK  {"id":3041,"user":"bob","total":"1 290.00" ...}
# Même jeton, on décrémente l'id → facture d'un autre client.
$ curl -s .../v1/invoices/3040 -H "Authorization: Bearer <bob>"
200 OK  {"id":3040,"user":"alice@northwind.io","total":"14 500.00" ...}
# → fuite inter-comptes confirmée · 3 000+ factures énumérables.
```

IMPACT

Un seul compte client suffit à extraire **toute la base de factures** : coordonnées clients, montants, historique d'achat. Fuite de données personnelles de tous les clients, exposition RGPD, rupture de confiance. Portée **modifiée (S:C)** : la faille franchit l'isolation entre comptes.

RECOMMANDATION

Vérifier la propriété de l'objet à chaque requête, côté serveur, avant de renvoyer la ressource.

- Filtrer systématiquement par le compte de la session : **`WHERE invoice.user_id = session.user_id`** ; ne jamais faire confiance à l'`id` seul.
- Remplacer les identifiants séquentiels exposés par des **UUID non devinables** (défense en profondeur, pas un correctif à lui seul).
- Centraliser l'autorisation objet dans un middleware unique, plutôt que dans chaque contrôleur.

Références : OWASP API1:2023 (BOLA) · WSTG-ATHZ-04 · CWE-639

04 · CONSTATS DÉTAILLÉS (SUITE)

F-02

Rejeu d'une requête de remboursement

8.2
ÉLEVÉ · CVSS 3.1

ENDPOINT	CATÉGORIE
POST /api/v1/refund	Logique métier · WSTG-BUSL-03

DESCRIPTION

La requête de remboursement n'est protégée par **aucun jeton d'idempotence ni contrôle d'unicité**. La même requête signée peut être renvoyée un nombre illimité de fois : chaque rejeu crédite à nouveau le compte. Le serveur ne détecte pas qu'un remboursement pour cette transaction a déjà été émis.

PREUVE D'EXPLOITATION

```
# Un remboursement légitime de 50,00 € est capturé, puis rejoué 3×.
$ for i in 1 2 3; do
  curl -s -X POST https://api.juice-shop.example/v1/refund \
    -H "Authorization: Bearer <jeton>" \
    -d '{"tx":"TX-88213","amount":"50.00"}'; done
200 {"refund":"RF-1","status":"credited"}
200 {"refund":"RF-2","status":"credited"}  # doublon accepté
200 {"refund":"RF-3","status":"credited"}  # → 150 € crédités pour 50 € dus
```

IMPACT

Fraude financière directe : un client crédite son propre compte de façon répétée à partir d'une seule transaction réelle. À l'échelle et en automatisé, la perte est **non bornée** et n'apparaît pas immédiatement dans les journaux applicatifs, chaque rejeu ressemblant à une opération valide.

RECOMMANDATION

Rendre l'opération idempotente et unique par transaction.

- Exiger une **clé d'idempotence** (en-tête `Idempotency-Key`) et rejeter tout rejeu de la même clé.
- Poser une **contrainte d'unicité** en base : au plus un remboursement par `tx`, dans une transaction atomique.
- Vérifier que le montant remboursé ne dépasse jamais le montant réglé de la transaction d'origine.

Références : OWASP WSTG-BUSL-03 · API6:2023 · CWE-837

04 · CONSTATS DÉTAILLÉS (SUITE)

F-03

Élévation de privilège par mass assignment

8.1
ÉLEVÉ · CVSS 3.1

ENDPOINT	CATÉGORIE
PATCH /api/v1/users/me	API3:2023 · BOPLA

DESCRIPTION

L'endpoint de mise à jour du profil lie directement le corps JSON reçu au modèle utilisateur. Le champ **`role`**, censé être en lecture seule, est accepté s'il est présent dans la requête. Un utilisateur standard se promeut ainsi **administrateur de la boutique** en une seule requête.

PREUVE D'EXPLOITATION

```
# Bob est "member". Il ajoute un champ role non prévu par l'UI.
$ curl -s -X PATCH https://api.juice-shop.example/v1/users/me \
    -H "Authorization: Bearer <jeton_bob>" \
    -d '{"name":"Bob","role":"admin"}'
200 OK  {"id":812,"name":"Bob","role":"admin"}
# → accès à la gestion des membres, facturation et export org.
```

IMPACT

Prise de contrôle complète du back-office de la boutique par un compte standard : gestion des utilisateurs, paramètres de paiement, export des données. Combinée à **F-01**, la portée dépasse le compte de l'attaquant.

RECOMMANDATION

N'accepter en entrée que les champs explicitement autorisés.

- Utiliser une **liste blanche** de propriétés modifiables (`name`, `email`…) ; ignorer tout autre champ.
- Traiter `role` et les attributs sensibles via un flux d'administration dédié, contrôlé côté serveur.
- Journaliser et alerter sur toute tentative de modification d'un champ protégé.

Références : OWASP API3:2023 (BOPLA) · WSTG-BUSL-01 · CWE-915

05 · DÉMARCHE SUIVIE

Quatre étapes, du cadrage au contre-test.

01

Cadrage

Périmètre, comptes de test et règles d'engagement écrites, prix fixe validé avant tout test.

02

Test manuel

7 jours d'exploration à la main, canal direct ouvert. Le critique signalé le jour de sa découverte.

03

Rapport

Chaque faille : criticité, reproduction, impact réel, correctif concret, plus cette synthèse.

04

Contre-test

Après correction : re-vérification de chaque faille et attestation à jour. Inclus, sous 30 jours.

06 · PROCHAINE ÉTAPE & ATTESTATION

Contre-test & attestation de correction

Une fois les vulnérabilités corrigées, chaque constat est re-testé pour confirmer sa fermeture. Une **attestation datée et signée** (listant le périmètre, la méthodologie et l'état final de chaque faille) est alors délivrée. C'est ce document que vous transmettez tel quel à vos clients, à un auditeur SOC 2 ou dans un dossier ISO 27001.

Ce rapport est un spécimen : OWASP Juice Shop est une application volontairement vulnérable, utilisée ici comme cible de démonstration. La structure, la profondeur et le format correspondent exactement à un livrable réel hex16.

hex16

Test d'intrusion & audit de sécurité · web & API

Un test de cette nature pour votre application ? · **hex16.fr** · contact@hex16.fr · réponse sous 24 h