Engineer

injection SQL

Attaque transformant une entrée en syntaxe SQL

Ce que c’est

L’injection SQL, souvent abrégée en SQLi, est une attaque dans laquelle une application construit une commande SQL à partir d’un texte fourni par un tiers, et la base finit par lire une partie de ce texte comme du SQL plutôt que comme une valeur. La Common Weakness Enumeration la répertorie sous CWE-89 et décrit la faiblesse comme une commande construite entièrement ou partiellement à partir d’entrées externes, sans neutraliser au préalable les caractères qui ont un sens particulier en SQL. La fiche de prévention d’OWASP formule la même idée plus simplement : l’attaque fonctionne lorsque des données non fiables deviennent une partie de la syntaxe d’une requête.

Le problème vient de ce que SQL ne sépare pas fermement les instructions d’une commande des données qu’elle contient. Une apostrophe termine une chaîne littérale, et ce que l’attaquant saisit ensuite est analysé comme du SQL supplémentaire. Deux tirets commencent un commentaire, ce qui permet d’écarter le reste de l’instruction du développeur. Un point-virgule, lorsque la base et le pilote l’autorisent, termine une instruction et en commence une autre. Aucun de ces signes n’est exotique : ils apparaissent aussi dans les noms de famille et les textes libres.

Le Web Security Testing Guide d’OWASP répartit les attaques en trois classes selon la manière dont les résultats parviennent à l’attaquant. Les attaques dans la même bande, ou in-band, lisent la réponse par le canal utilisé pour envoyer la requête, généralement sur la page de l’application. Les attaques hors bande, ou out-of-band, font livrer les résultats ailleurs par la base, par exemple au moyen d’une connexion HTTP distincte. Les attaques par inférence, ou à l’aveugle, ne reçoivent aucune donnée et la reconstituent une question à la fois à partir du comportement de l’application. Le guide cite cinq techniques courantes dans ces

classes : un opérateur `UNION` qui ajoute un second `SELECT` au premier, des conditions booléennes, des erreurs provoquées, des requêtes hors bande et des délais introduits délibérément.

L'injection SQL appartient à une famille plus large. Dans CWE, elle relève de la faiblesse générale d'injection, CWE-74, à côté du cross-site scripting, et l'OWASP Top Ten de 2025 les place tous deux dans sa catégorie A05 Injection. Le point commun est une donnée qui franchit une frontière pour être lue comme une instruction. Ici, l'interpréteur est le parseur SQL de la base, et le remède consiste à tenir chaque valeur entièrement hors du texte SQL.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce qu'une injection SQL ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Des comptes, mots de passe et droits attribués selon les besoins exacts de chaque poste, puis retirés le jour du départ.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. 1 000 règles RBAC pour sécuriser Linux — 1,000 règles d'accès écrites pour combler une faille de sécurité

1 000 règles RBAC pour sécuriser Linux Contactez-nous →

Pourquoi c'est important

Les dégâts dépendent des droits du compte de base de données de l'application, souvent très étendus. CWE-89 cite la lecture de données que l'attaquant ne devrait jamais voir, le contournement de l'authentification sans connaître de mot de passe, la modification des données qui déterminent les autorisations, la modification ou la suppression d'enregistrements et, dans certaines configurations, l'exécution de commandes du système d'exploitation. La page d'OWASP consacrée à l'attaque ajoute des opérations d'administration comme l'arrêt de la base, la lecture de fichiers du serveur de base de données et la prise de contrôle de ce serveur en tant qu'administrateur.

Les deux sources jugent la gravité élevée. CWE classe la probabilité d'exploitation comme High, élevée, et la page d'OWASP décrit la faille comme facile à trouver et à exploiter. Cette faiblesse figure dans chaque édition du CWE Top 25 Most Dangerous Software Weaknesses de 2019 à 2025. L'entrée CWE rapporte aussi une vague de 2008 durant laquelle une même chaîne d'injection SQL, fonctionnant dans de nombreux programmes différents, a compromis un grand nombre de serveurs web et leur a fait diffuser du code malveillant.

La défense, en revanche, est bien connue et peu coûteuse. La fiche d'OWASP place en premier les instructions préparées avec paramètres liés, et CWE les recommande aussi : le SQL et les valeurs sont transmis séparément, ce qui empêche la base de les confondre. Le reste, du moindre privilège à une gestion prudente des erreurs, constitue une seconde couche qui limite les dégâts lorsqu'une erreur échappe aux contrôles.

À quoi cela ressemble en pratique

Prenons une boutique dont la page d'historique recherche les commandes à partir de l'adresse électronique saisie dans un formulaire. Le développeur construit le SQL avec une f-string Python :

```
cur.execute(f"SELECT id, total FROM orders WHERE email = '{email}'")
```

Un visiteur saisit `x' OR '1'='1` dans le champ. Le programme l'insère dans le texte, et la base reçoit ceci :

```
SELECT id, total FROM orders WHERE email = 'x' OR '1'='1'
```

L'apostrophe après `x` ferme la chaîne ouverte par le développeur, et la clause `OR` qui suit est toujours vraie : la base renvoie donc toutes les commandes de la table. CWE-89 et le guide de test emploient le même procédé de condition toujours vraie, qui peut aussi faire accepter par un formulaire de connexion une personne ne connaissant ni le nom d'utilisateur ni le mot de passe.

Une saisie plus dangereuse termine l'instruction et en commence une autre. L'exemple de l'entrée CWE saisit `name'; DELETE FROM items; --` comme nom d'article : une seconde instruction suit alors la première, et les tirets mettent le reste en commentaire. Son exécution dépend de la base et du pilote, et plusieurs interfaces actuelles la refusent. `PQexecParams` de PostgreSQL accepte au plus une commande SQL par chaîne, ce que sa documentation juge d'une certaine utilité contre l'injection SQL. Le module Python `sqlite3` lève une erreur dès que `execute()` reçoit plusieurs instructions, et psycopg refuse plusieurs instructions dans un même appel lorsque des paramètres sont transmis. Rien de cela n'arrête la condition toujours vraie ci-dessus, qui ne nécessite qu'une seule instruction.

La correction consiste à transmettre la valeur séparément. PEP 249, la norme d'interface Python pour les bases de données, demande à chaque pilote de déclarer son style de marqueurs de paramètres dans une constante de module appelée `paramstyle`. Elle en énumère cinq : `qmark` (`?`), `numeric` (`:1`), `named` (`:name`), `format` (`%s`) et `pyformat` (`%(name)s`). Le module `sqlite3` est fixé à `qmark` et accepte aussi `named`. Psycopg, le pilote PostgreSQL, accepte `%s` et `%(name)s` :

```
cur.execute(
    "SELECT id, total FROM orders WHERE email = %s",
    (email,),
)
```

Le `%s` ressemble au formatage de chaînes Python, mais rien n'est formaté : le SQL et le tuple de valeurs parviennent au pilote comme deux arguments distincts. Psycopg 3 les garde séparés jusqu'au bout, en envoyant au serveur la commande et ses paramètres séparément plutôt qu'en les fusionnant côté client. La fiche d'OWASP illustre le principe avec sa propre saisie, `tom' or '1'='1` : liée comme paramètre, elle est recherchée littéralement comme un nom d'utilisateur étrange et ne correspond à personne.

PostgreSQL propose la même séparation directement en SQL. `PREPARE` fixe l'instruction avec des paramètres numérotés, et `EXECUTE` fournit ensuite les valeurs :

```
PREPARE order_lookup (text) AS
    SELECT id, total FROM orders WHERE email = $1;
EXECUTE order_lookup('x'' OR ''1''=''1');
```

La saisie hostile, écrite ici comme une chaîne littérale SQL, arrive dans `$1` sous la forme d'une seule adresse électronique étrange et ne trouve aucune commande.

Sur une plateforme contenant des données professionnelles et organisationnelles, les entrées sont validées avant tout accès à la base, puis l'API se connecte avec un rôle au moindre privilège autorisé uniquement à exécuter

les fonctions de l'application, et tout est paramétré. La liaison des paramètres empêche les données de devenir du SQL, et le rôle restreint limite ce qu'une erreur pourrait atteindre.

Où les problèmes surviennent

Construire quand même la chaîne. Une f-string, l'opérateur `%`, `+` ou `.format()` fusionne la valeur avec le SQL avant que le pilote ne le voie. Il en va de même d'un marqueur de paramètre placé entre apostrophes, comme `'%s'`. OWASP souligne que concaténer des valeurs non fiables avant de préparer une instruction annule toute la protection, et la documentation psycopg demande de ne jamais utiliser `%` ou `+` pour fusionner des valeurs avec des requêtes. Transmettez chaque valeur dans le second argument de `execute()`, sans entourer le marqueur de paramètre d'apostrophes.

Échapper à la main. Doubler les apostrophes ou ajouter des barres obliques inverses semble équivalent, mais les règles dépendent du dialecte SQL et de ses réglages. L'exemple d'OWASP est le mode `NO_BACKSLASH_ESCAPES` de MySQL, qui désactive l'échappement par barre oblique inverse, et la fiche déconseille fortement l'échappement manuel. Remplacez les chaînes échappées à la main par des paramètres liés.

Prendre la validation ou un WAF pour la correction. La validation détecte des erreurs, mais un texte libre doit autoriser les caractères nécessaires à l'attaquant : l'exemple de CWE est le nom O'Reilly, qui passerait probablement la validation tout en contenant une apostrophe. Les listes de mots interdits présentent des failles, et CWE juge l'efficacité d'un pare-feu applicatif seulement modérée. Selon OWASP, « valider une valeur ne rend pas sa concaténation au SQL sûre ». Continuez à valider pour vérifier la justesse des données, et liez les paramètres pour assurer la sécurité.

Prendre les noms de tables, de colonnes ou l'ordre de tri dans la saisie. Les marqueurs de paramètres ne représentent que des valeurs. Un développeur qui doit laisser choisir une colonne ou le sens de `ORDER BY` est donc tenté de coller le texte dans la requête. OWASP recommande d'associer chaque choix à un petit ensemble de noms maîtrisés par l'application et de refuser tout le reste, par exemple en transformant le sens demandé en la valeur littérale `ASC` ou `DESC`. Lorsqu'un nom doit réellement être dynamique, le module `psycopg.sql` de

psycopg compose les identifiants. OWASP précise toutefois que mettre un nom entre délimiteurs ne limite pas les tables vers lesquelles on peut orienter l'application.

Supposer les procédures stockées sûres. Appeler une procédure avec des arguments liés protège l'appel, pas le SQL qu'elle construit en interne. CWE précise qu'une procédure stockée n'écarte que certaines voies d'injection et en laisse d'autres possibles. Dans la base, utilisez du SQL statique ou liez aussi les valeurs du SQL dynamique ; dans PostgreSQL, cela signifie `EXECUTE ... USING` pour les valeurs et `format('%I', ...)` pour les identifiants.

Insérer des chaînes dans un ORM. Un outil de mapping objet-relationnel lie les valeurs lorsqu'il est utilisé comme prévu, et CWE attribue aux couches de persistance comme Hibernate une protection importante quand elles sont correctement employées. OWASP ajoute que les requêtes Hibernate restent exposées à l'injection si les développeurs concatènent des entrées au texte de requête. Utilisez les paramètres de l'outil, pas la construction de chaînes.

Donner des droits d'administrateur au compte applicatif. Si ce compte peut tout faire, l'attaquant aussi. OWASP demande des comptes en lecture seule lorsque seules des lectures sont nécessaires, un utilisateur de base distinct par application web, des vues n'exposant que les colonnes utiles au compte et aucun droit d'administrateur pour les comptes applicatifs. Lorsqu'une application ne fonctionne qu'au moyen de procédures stockées, elle ne devrait pouvoir exécuter que celles-ci. Le moindre privilège limite les dégâts ; il n'empêche pas l'injection.

Montrer aux utilisateurs les erreurs détaillées de la base. Un message qui révèle la structure de l'instruction aide l'attaquant à adapter sa prochaine tentative. La technique fondée sur les erreurs du guide de test repose précisément sur cela. CWE recommande des messages d'erreur minimaux. Montrez un échec générique à l'utilisateur et conservez les détails côté serveur pour l'équipe.

Faire confiance aux données déjà dans la base. Une valeur stockée de façon sûre peut rester dangereuse lorsqu'un code ultérieur la colle dans du SQL. OWASP indique que ses contrôles s'appliquent quelle que soit l'origine des données : requête HTTP, autre service ou écriture antérieure dans le stockage.

CWE avertit aussi que des traitements ultérieurs peuvent oublier d'échapper ce qui a été enregistré. Liez les valeurs lues dans vos propres tables comme celles d'un formulaire.

Pour aller plus loin

- CWE-89 dans la Common Weakness Enumeration — la fiche de faiblesse : conséquences, exemples et mesures de protection avec leurs limites.
- Fiche OWASP de prévention de l'injection SQL — requêtes paramétrées, procédures stockées, identifiants dynamiques et moindre privilège.
- OWASP Web Security Testing Guide : tester l'injection SQL — les classes et techniques d'attaque telles que les rencontre une personne chargée des tests.
- PEP 249 : spécification v2.0 de l'API Python pour les bases de données — l'interface commune aux pilotes Python, dont les cinq styles de marqueurs de paramètres.
- Psycopg 3 : transmettre des paramètres aux requêtes SQL — les règles relatives aux marqueurs de paramètres et la bonne manière de construire du SQL dynamique.
- PostgreSQL : fonctions d'exécution des commandes — comment libpq transmet les paramètres séparément du texte de commande.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le parcours de déploiement, les risques opérationnels et les personnes d'astreinte. Nous pouvons définir un changement avec des vérifications, une procédure de reprise et une responsabilité claire.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce qu'une injection SQL ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/sql-injection/