Engineer

SQL-ін'єкція

атака, що робить вхідні дані частиною запиту до бази даних

Що це таке

SQL-ін’єкція, англійською SQL injection, часто скорочено SQLi, — це атака, за якої застосунок складає SQL-команду з тексту, що надійшов від когось іншого, і база даних зрештою читає частину цього тексту як SQL, а не як значення. Українською її ще називають впровадженням SQL-коду. Common Weakness Enumeration веде цю ваду як CWE-89 і описує її як команду, повністю або частково зібрану з вхідних даних ззовні, без попередньої нейтралізації символів, що мають у SQL особливе значення. Шпаргалка OWASP із запобігання формулює те саме простіше: атака спрацьовує, коли дані, яким не можна довіряти, стають частиною синтаксису запиту.

Корінь проблеми в тому, що SQL не проводить чіткої межі між інструкціями команди й даними всередині неї. Одна одинарна лапка завершує рядковий літерал, і все, що зловмисник напише після неї, розбирається як продовження SQL. Два дефіси починають коментар, і так можна відкинути решту команди, яку написав розробник. Крапка з комою, там, де це дозволяють база даних і драйвер, завершує одну команду й починає наступну. Жоден із цих знаків не екзотичний: ті самі символи трапляються в прізвищах і довільному тексті.

Посібник OWASP Web Security Testing Guide поділяє атаки на три класи за тим, як результати потрапляють до зловмисника. Внутрішньоканальні (in-band) атаки отримують відповідь тим самим каналом, яким її викликали, зазвичай просто на сторінці застосунку. Позаканальні (out-of-band) змушують базу даних доставити результати в інше місце, наприклад окремим HTTP-з’єднанням. Інференційні, або сліпі, атаки не отримують даних узагалі й відновлюють їх по одному питанню за раз, спостерігаючи, як поводиться застосунок. Посібник називає п’ять поширених технік у цих класах: оператор

`UNION`, що приєднує до першого `SELECT` ще один, булеві умови, навмисно спровоковані помилки, передавання даних окремим каналом і навмисне сповільнення відповіді.

SQL-ін’єкція належить до ширшої родини. У CWE вона стоїть під загальною вадою ін’єкції, CWE-74, поруч із міжсайтовим скриптингом, а OWASP Top Ten 2025 року відносить обидві до категорії A05 Injection. Спільне в них одне: дані перетинають межу й потрапляють туди, де їх читають як інструкції. Тут інтерпретатором є SQL-парсер бази даних, а ліки полягають у тому, щоб тримати кожне значення повністю поза текстом SQL.

Застосуйте поняття до рішення

Застосуйте «Що таке SQL-ін'єкція?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Облікові записи, паролі й права налаштовані так, щоб доступ відповідав роботі, а звільнені втрачали його того ж дня.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Написала 1 000 правил RBAC для Linux-серверів — 1,000 правил доступу написано, щоб усунути вразливість

Написала 1 000 правил RBAC для Linux-серверів Зв'язатися з нами →

Чому це важливо

Шкода залежить від того, що дозволено обліковому запису бази даних, під яким працює застосунок, а дозволено йому часто багато. CWE-89 перелічує читання даних, яких зловмисник ніколи не мав бачити, обхід автентифікації без знання пароля, зміну даних, що визначають, кому що можна, зміну або видалення записів, а за деяких налаштувань і виконання команд операційної системи. OWASP на сторінці про атаку додає адміністративні операції: вимкнення бази даних, читання вмісту з диска сервера бази даних і захоплення цього сервера з правами адміністратора.

Обидва джерела вважають цю ваду серйозною. CWE оцінює ймовірність експлуатації як високу, а сторінка OWASP про атаку описує ваду як таку, яку легко знайти і легко використати. Вона є в кожному випуску CWE Top 25 Most Dangerous Software Weaknesses з 2019 по 2025 рік. Запис CWE також згадує хвилю 2008 року, коли один і той самий рядок атаки, що спрацьовував проти багатьох різних програм, дав змогу зламати велику кількість вебсерверів і змусити їх роздавати шкідливий код.

Захист натомість добре відомий і дешевий. Шпаргалка OWASP ставить на перше місце підготовлені запити (prepared statements) з прив'язаними параметрами, і CWE теж їх рекомендує: SQL і значення передаються окремо, тож база даних не може сплутати одне з іншим. Усе інше, від мінімальних привілеїв до ретельної обробки помилок, — це додатковий шар, що обмежує шкоду, коли помилка все ж проскочить.

Як це виглядає на практиці

Візьмімо інтернет-магазин, сторінка історії замовлень якого шукає замовлення за адресою електронної пошти, введеною у форму. Розробник складає SQL за допомогою f-рядка в Python:

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

Відвідувач вводить у поле `x' OR '1'='1` . Програма вставляє це як є, і база даних отримує таке:

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

Лапка після `x` закриває рядок, який відкрив розробник, а умова `OR` після неї завжди істинна, тож база даних повертає всі замовлення з таблиці. CWE-89 і посібник із тестування використовують той самий трюк із завжди істинною умовою, який може ще й змусити форму входу впустити відвідувача, що не знає ні імені користувача, ні пароля.

Гірше введення завершує команду й починає іншу. У прикладі із запису CWE як назву товару вводять `name'; DELETE FROM items; --` , тож за першою командою йде друга, а дефіси перетворюють решту на коментар. Чи це виконається, залежить від бази даних і драйвера, і чимало сучасних інтерфейсів таке відхиляють. `PQexecParams` у PostgreSQL приймає щонайбільше одну SQL-команду в рядку, і документація вважає це певним додатковим захистом від SQL-ін'єкцій. Модуль `sqlite3` у Python видає помилку, щойно `execute()` отримує більше однієї команди, а psycopg відмовляється виконувати кілька команд в одному виклику, якщо передано параметри. Ніщо з цього не зупиняє завжди істинну умову вище, якій вистачає однієї команди.

Виправлення полягає в тому, щоб передавати значення окремо. PEP 249, стандарт інтерфейсу баз даних у Python, вимагає, щоб кожен драйвер оголошував свій стиль заповнювачів у константі модуля `paramstyle`, і перелічує п'ять стилів: `qmark` (`?`), `numeric` (`:1`), `named` (`:name`), `format` (`%s`) і `pyformat` (`%(name)s`). Модуль `sqlite3` закріплений за стилем `qmark` і також приймає `named`. Psycopg, драйвер для PostgreSQL, приймає `%s` і `%(name)s`:

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

`%s` схожий на форматування рядків у Python, але нічого не форматується: SQL і кортеж значень доходять до драйвера як два окремі аргументи. Psycopg 3 тримає їх окремо всю дорогу й надсилає команду та її параметри на сервер нарізно, а не зливає їх на клієнті. Шпаргалка OWASP показує це на власному прикладі введення, `tom' or '1'='1`: прив'язане як параметр, воно стає просто дивним іменем користувача, яке шукають буквально, і воно ні з чим не збігається.

PostgreSQL пропонує такий самий поділ і в чистому SQL. `PREPARE` фіксує команду з нумерованими параметрами, а `EXECUTE` згодом передає значення:

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

Вороже введення, записане тут як SQL-літерал, надходить у `$1` як одна дивна адреса електронної пошти й не знаходить жодного замовлення.

На платформі з професійними та організаційними даними вхідні дані валідують ще до того, як щось потрапить у базу даних, а потім серверна частина входить до бази під роллю з мінімальними привілеями, яка може лише виконувати функції застосунку, і все параметризовано. Прив'язування не дає даним стати SQL, а вузька роль обмежує, куди може дістатися помилка.

Де виникають проблеми

Рядок усе одно склеюють. F-рядок, оператор `%`, `+` чи `.format()` вливають значення в SQL ще до того, як його побачить драйвер, і так само діє заповнювач, узятий у лапки, як-от `'%s'`. OWASP зазначає, що склеювання недовірених значень із SQL до підготовки запиту зводить увесь захист нанівець, а документація psycopg каже розробникам ніколи не використовувати `%` чи `+`, щоб вливати значення в запити. Передавайте кожне значення другим аргументом `execute()`, а заповнювач лишайте без лапок.

Екранування вручну. Подвоєння лапок чи додавання зворотних скісних рисок здається рівноцінним, але правила залежать від діалекту SQL і його налаштувань. Приклад OWASP — режим MySQL `NO_BACKSLASH_ESCAPES`, який вимикає екранування зворотною скісною рискою, і шпаргалка рішуче не радить покладатися на екранування. Замініть рядки, екрановані вручну, прив'язаними параметрами.

Валідація або WAF як рішення. Валідація ловить помилки, але довільний текст мусить допускати символи, потрібні зловмисникові: у записі CWE прикладом є прізвище O'Reilly, яке, найімовірніше, пройде валідацію і все одно містить лапку. Списки заборонених слів мають шпарини, а CWE оцінює ефективність брандмауера застосунків лише як помірну. OWASP каже прямо: валідоване значення від цього не стає безпечним для склеювання з SQL. Валідуйте далі заради коректності, а прив'язуйте заради безпеки.

Назви таблиць, стовпців або порядок сортування з введення. Заповнювачі підставляють лише значення, тож розробника, якому треба дати користувачеві вибрати стовпець чи напрям `ORDER BY`, тягне вставити текст напряму. OWASP радить зіставляти кожен вибір із невеликим набором назв, якими володіє сам застосунок, і відхиляти все інше, наприклад перетворювати запитаний напрям на літерал `ASC` або `DESC`. Коли назва справді мусить бути динамічною, модуль `psycopg.sql` уміє складати ідентифікатори, хоча OWASP зауважує, що навіть правильно взята в лапки назва не обмежує, до яких таблиць можна спрямувати застосунок.

Збережені процедури, які вважають безпечними. Виклик процедури з прив'язаними аргументами захищає сам виклик, а не SQL, який процедура будує всередині. CWE зазначає, що збережені процедури запобігають деяким типам SQL-ін'єкцій, але не виключають інших. Усередині бази даних

використовуйте статичний SQL або прив'язуйте значення й у динамічному SQL; у PostgreSQL це `EXECUTE ... USING` для значень і `format('%I', ...)` для ідентифікаторів.

ORM, у який вклеюють рядки. Об'єктно-реляційне відображення (ORM) прив'язує значення, коли ним користуються як задумано, і CWE визнає за шарами збереження на кшталт Hibernate значний захист за правильного використання. OWASP додає, що запити Hibernate все одно можна атакувати, коли розробники склеюють введення з текстом запиту. Користуйтеся параметрами ORM, а не складанням рядків.

Обліковий запис застосунку з правами адміністратора. Якщо обліковий запис може все, то може й зловмисник. OWASP радить облікові записи лише для читання там, де потрібне тільки читання, окремого користувача бази даних для кожного вебзастосунку, представлення (views), що показують лише потрібні стовпці, і жодних прав адміністратора для облікових записів застосунків. Коли застосунок працює лише через збережені процедури, йому слід дозволити виконувати їх і нічого більше. Мінімальні привілеї обмежують шкоду, але самої атаки не відвертають.

Докладні помилки бази даних, показані користувачам. Повідомлення про помилку, що розкриває структуру команди, допомагає зловмисникові підлаштувати наступну спробу, і техніка на основі помилок із посібника з тестування спирається саме на це. CWE радить мінімальні повідомлення про помилки. Показуйте користувачеві загальне повідомлення про збій, а подробиці лишайте на сервері для своїх розробників.

Довіра до даних, які вже лежать у базі. Значення, яке безпечно записали, все одно може стати небезпечним, коли пізніший код вставить його в SQL. OWASP каже, що її заходи діють незалежно від того, чи дані надходять із HTTP-запиту, з іншого сервісу чи були записані раніше, а CWE попереджає, що подальші процеси можуть забути екранувати те, що колись записали. Прив'язуйте значення, прочитані з власних таблиць, так само, як прив'язуєте введення з форми.

Що почитати далі

- CWE-89 in the Common Weakness Enumeration — сам запис про ваду: наслідки, приклади й заходи захисту разом з їхніми обмеженнями.

- OWASP SQL Injection Prevention Cheat Sheet — параметризовані запити, збережені процедури, динамічні ідентифікатори й мінімальні привілеї.
- OWASP Web Security Testing Guide: Testing for SQL Injection — класи атак і техніки такими, якими їх бачить тестувальник.
- PEP 249: Python Database API Specification v2.0 — спільний інтерфейс драйверів Python, разом із п'ятьма стилями заповнювачів.
- Psycopg 3: Passing parameters to SQL queries — правила для заповнювачів і правильний спосіб будувати динамічний SQL.
- PostgreSQL: Command Execution Functions — як libpq передає параметри окремо від рядка команди.

Словник

Обирайте за фактичними умовами

Опишіть шлях розгортання, експлуатаційні ризики та чергову команду. Ми можемо визначити зміну з перевірками, шляхом відновлення та чіткою відповідальністю.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке SQL-ін'єкція?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

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

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/sql-injection/