Engineer

SQLインジェクション

入力をSQLの構文の一部に変えてしまう攻撃

どのようなものか

SQLインジェクションは、SQLiとも略される攻撃です。アプリケーションが外部から受け取った文字列を組み込んでSQLの命令を作り、その一部をデータベースが値ではなくSQLとして解釈してしまいます。Common Weakness EnumerationはこれをCWE-89として分類し、SQLで特別な意味を持つ文字を無害化せずに、外部入力の全部または一部から命令を組み立てる弱点だと説明しています。OWASPの対策チートシートは、同じ仕組みをより平易に説明しています。信頼できないデータがクエリの構文の一部になると、攻撃が成立するのです。

根本的な問題は、SQLの命令文と、その中のデータの間に明確な境界がないことです。シングルクォートは文字列リテラルを終わらせるため、その後に攻撃者が入力した文字列は、続きのSQLとして解析されます。ハイフンを二つ続けるとコメントが始まり、開発者が書いた文の残りを無視させることができます。データベースとドライバーが許す場合、セミコロンは一つの文を終え、次の文を始めます。どれも特殊な文字ではなく、人の姓や自由記述にも現れる記号です。

OWASPのWeb Security Testing Guideは、結果が攻撃者に届く方法によって攻撃を三つに分類しています。インバンド攻撃は、送信に使ったのと同じ経路、通常はアプリケーション自身のページで結果を読み取ります。アウトオブバンド攻撃では、データベースに別の場所へ結果を送らせます。例えば、別のHTTP接続を使います。推論型、またはブラインド攻撃は、データそのものを受け取らず、アプリケーションの振る舞いを手掛かりに、一問ずつ問いかけてデータを復元します。同ガイドは、これらの分類にまたがる五つの一般的な手法を挙げています。`UNION` 演算子で最初のクエリに二つ目の `SELECT` をつなぐ手法、真偽条件、意図的に発生させるエラー、アウトオブバンドのリクエスト、意図的な時間遅延です。

SQLインジェクションは、より大きな弱点群の一つです。CWEではクロスサイトスクリプティングとともに、一般的なインジェクションの弱点であるCWE-74の下に位置付けられています。2025年版OWASP Top Tenも、両方をA05 Injectionに分類しています。共通するのは、データが境界を越え、命令として解釈される場所に入り込むことです。ここで解釈を行うのはデータベースのSQLパーサーであり、対策は、すべての値をSQLの文字列から完全に切り離すことです。

概念を判断に生かす

「SQLインジェクションとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 アカウント、パスワード、権限を整理し、業務に必要なアクセスだけを付与し、退職者の権限は当日に解除します。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 Linuxサーバーを守る1,000件のRBACルールを作成 — 1,000 セキュリティ上の穴を塞ぐアクセスルール

Linuxサーバーを守る1,000件のRBACルールを作成 お問い合わせ →

なぜ重要なのか

被害の範囲は、アプリケーションのデータベースアカウントに何が許されているかで決まります。そのアカウントには、広い権限が与えられていることも珍しくありません。CWE-89は、攻撃者に見せてはいけないデータの読み取り、パスワードを知らないままの認証回避、誰に何を許すかを決めるデータの変更、レコードの改ざんや削除、構成によってはOSのコマンドの実行を挙げています。OWASPの攻撃解説は、データベースの停止、データベースサーバー上のファイルの読み取り、管理者としてのデータベースサーバーの乗っ取りといった管理操作も挙げています。

どちらの資料も、これを重大な問題と評価しています。CWEは悪用される可能性を「高い」とし、OWASPの攻撃解説は、弱点を見つけることも悪用することも容易だと説明しています。この弱点は、2019年から2025年までのCWE Top 25 Most Dangerous Software Weaknessesのすべての版に登場しています。CWEの項目には、2008年に一つのSQLインジェクション用文字列が多くの異なるプログラムに通用し、多数のWebサーバーが侵害され、悪意あるコードを配信させられた攻撃の波も記録されています。

一方、防御方法はよく分かっており、低コストです。OWASPのチートシートは、パラメーターをバインドするプリペアドステートメントを第一の対策とし、CWEも推奨しています。SQLと値を別々に渡せば、データベースは両者を取り違えません。最小権限から慎重なエラー処理まで、それ以外の対策は、ミスが入り込んだときに被害を抑える第二の防御層です。

実際の活用例

店舗の注文履歴ページが、フォームに入力されたメールアドレスから注文を検索するとします。開発者はPythonのf文字列でSQLを組み立てています。

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

訪問者が入力欄に `x' OR '1'='1` と入力すると、プログラムがそのまま埋め込み、データベースには次の文が届きます。

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

`x` の後の引用符が、開発者の開いた文字列を閉じ、その後の `OR` 節が常に真になるため、データベースはテーブル内のすべての注文を返します。CWE-89とテストガイドにも、条件を常に真にする同じ手口が載っています。これを使うと、ユーザー名もパスワードも知らない訪問者をログインフォームが受け入れてしまうこともあります。

さらに悪い入力は、一つの文を終わらせて別の文を始めます。CWEの例では、商品名として `name'; DELETE FROM items; --` を入力し、最初の文に二つ目の文を続け、ハイフンで残りをコメントアウトしています。実行されるかどうかはデータベースとドライバーによって異なり、現在のインターフェースの中には拒否するものもあります。PostgreSQLの `PQexecParams` は、一つの文字列につきSQLの命令を一つまでしか受け付けません。ドキュメントも、これがSQLインジェクションに対して一定の助けになるとしています。Pythonの `sqlite3` モジュールは、`execute()` が複数の文を受け取ると必ずエラーを発生させ、psycopgも、パラメーターを渡す場合は一度の呼び出しで複数の文を実行することを拒否します。ただし、どの制限も、上に示した常に真になる条件は防げません。この攻撃に必要なのは一つの文だけだからです。

修正方法は、値を別に渡すことです。Pythonのデータベースインターフェース標準であるPEP 249は、各ドライバーが `paramstyle` というモジュール定数でプレースホルダーの形式を宣言するよう定め、五つの形式を挙げています。`qmark`（`?`）、`numeric`（`:1`）、`named`（`:name`）、`format`（`%s`）、`pyformat`（`%(name)s`）です。`sqlite3` モジュールの定数は `qmark` に固定されていますが、`named` も受け付けます。PostgreSQLのドライバーであるPsycopgは、`%s` と `%(name)s` を受け付けます。

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

`%s` はPythonの文字列書式指定に見えますが、ここでは文字列への埋め込みを行いません。SQLと値のタプルは、別々の二つの引数としてドライバーに届きます。Psycopg 3は最後まで両者を分離し、クライアント側で結合せず、命令とパラメーターを別々にサーバーへ送ります。OWASPのチートシートも、独自の入力例 `tom' or '1'='1` でこの点を説明しています。パラメーターとしてバインドすれば、変わったユーザー名を文字どおりに検索するだけになり、誰にも一致しません。

PostgreSQLは、通常のSQLでも同じ分離を提供しています。 `PREPARE` で番号付きのパラメーターを持つ文を確定し、後から `EXECUTE` で値を渡します。

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

ここではSQLリテラルとして記した攻撃用の入力が、 `$1` には風変わりなメールアドレス一つとして届くため、注文は見つかりません。

職業上のデータや組織のデータを扱うプラットフォームでは、データベースへアクセスする前に入力を検証し、その上でAPIが、アプリケーションの関数の実行だけを許可された最小権限のロールで接続し、すべてにパラメーターを使っています。バインドはデータがSQLに変わることを防ぎ、権限を絞ったロールは、ミスによる影響が届く範囲を制限します。

問題が起きるところ

結局、文字列を組み立ててしまう。 f文字列、 `%` 演算子、 `+` 、 `.format()` は、ドライバーに届く前に値をSQLに埋め込みます。 `'%s'` のように引用符で囲んだプレースホルダーも同様です。OWASPは、文を準備する前に信頼できない値を連結すると、防御全体が無効になると指摘しています。psycopgのドキュメントも、値をクエリに埋め込むために `%` や `+` を決して使わないよう求めています。プレースホルダーは引用符で囲まず、すべての値を `execute()` の第二引数として渡します。

手作業でエスケープする。 引用符を二重にしたり、バックスラッシュを加えたりすれば同じに見えますが、規則はSQLの方言とその設定に依存します。OWASPは、バックスラッシュによるエスケープを無効にするMySQLの `NO_BACKSLASH_ESCAPES` モードを例に挙げ、チートシートではエスケープを強く非推奨としています。手作業でエスケープした文字列を、バインドしたパラメーターに置き換えます。

検証やWAFを解決策と考える。 検証は入力ミスを見つけますが、自由記述には攻撃者が必要とする文字も許す必要があります。CWEの例はO'Reillyという姓です。検証には通りそうですが、引用符が含まれています。危険な単語を並べた拒否リストには抜け道があり、CWEはアプリケーションファイアウォールの有効性を中程度としか評価していません。OWASPの説明を訳すと、「値を検証しても、SQLに連結して安全になるわけではありません」。正しさを確かめるための検証は続け、安全性のためには値をバインドします。

テーブル名、列名、並び順を入力から受け取る。 プレースホルダーが表すのは値だけです。そのため、利用者に列や `ORDER BY` の方向を選ばせたい開発者は、文字列をそのまま埋め込みたくなります。OWASPの対策は、各選択肢をアプリケーション側で管理する少数の名前に対応付け、それ以外は拒否することです。例えば、指定された方向をリテラルの `ASC` または `DESC` に変換します。名前を本当に動的にする必要がある場合は、psycopgの `psycopg.sql` モジュールで識別子を組み立てられます。ただしOWASPは、名前を引用符で囲むだけでは、アプリケーションがアクセスするテーブルを制限できないと指摘しています。

ストアドプロシージャなら安全だと考える。 引数をバインドしてプロシージャを呼び出すと、呼び出し自体は守られますが、プロシージャ内部で組み立てるSQLは守られません。CWEは、ストアドプロシージャで排除できるのは一部のインジェクション経路だけで、ほかの経路は残ると指摘しています。データベース内部では静的なSQLを使うか、動的なSQLでも値をバインドします。PostgreSQLでは、値には `EXECUTE ... USING` を、識別子には `format('%I', ...)` を使います。

ORMの中に文字列を継ぎ足す。 オブジェクトリレーショナルマッパーは、設計どおりに使えば値をバインドします。CWEも、Hibernateなどの永続化層を正しく使うと高い防御効果があると認めています。一方でOWASPは、開発者が入力をクエリの文字列に連結すると、Hibernateのクエリでもインジェクションが起こるとしています。文字列を組み立てず、マッパーのパラメーターを使います。

アプリケーションのアカウントに管理者権限を持たせる。 アカウントが何でもできれば、攻撃者も同じことができます。OWASPは、読み取りだけが必要な場合の読み取り専用アカウント、Webアプリケーションごとの別々のデータベース

ユーザー、必要な列だけを公開するビュー、アプリケーションのアカウントに管理者権限を与えないことを求めています。ストアドプロシージャだけで動くアプリケーションには、その実行だけを許可し、ほかは許可しません。最小権限は被害を抑えますが、インジェクション自体は防ぎません。

詳しいデータベースエラーを利用者に見せる。 文の構造を明らかにするエラーメッセージは、攻撃者が次の試行を調整する手掛かりになります。テストガイドのエラーを利用する手法は、まさにそれを頼りにしています。CWEは、エラーメッセージを最小限にするよう推奨しています。利用者には一般的な失敗の表示だけを返し、詳しい情報はチームが確認できるようサーバー側に残します。

すでにデータベースに入っているデータを信頼する。 安全に保存した値でも、後のコードがSQLに埋め込めば危険になります。OWASPは、HTTPリクエスト、別のサービス、以前の保存処理のどこから来たデータであっても、対策を適用するとしています。CWEも、後続の処理では、保存したデータのエスケープを忘れる可能性があると警告しています。自分のテーブルから読んだ値も、フォームの入力と同じようにバインドします。

参考資料

- Common Weakness EnumerationのCWE-89 — 影響、例、対策とその限界をまとめた弱点の項目です。
- OWASP：SQLインジェクション対策チートシート — パラメーター化クエリ、ストアドプロシージャ、動的な識別子、最小権限を説明しています。
- OWASP Web Security Testing Guide：SQLインジェクションのテスト — テスト担当者の視点から、攻撃の分類と手法を説明しています。
- PEP 249：Python Database API Specification v2.0 — 五つのプレースホルダー形式を含む、Pythonのドライバーに共通するインターフェースです。
- Psycopg 3：SQLクエリへのパラメーターの渡し方 — プレースホルダーの規則と、動的なSQLの正しい組み立て方を説明しています。
- PostgreSQL：コマンド実行関数 — libpqが、命令の文字列とパラメーターを分けて送る仕組みを説明しています。



用語集

実際の制約に合わせて選ぶ

デプロイの流れ、運用上のリスク、緊急対応の担当者をお知らせください。検査、復旧手段、明確な責任の所在を伴う変更の範囲を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「SQLインジェクションとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

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

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/sql-injection/