New issue

SQL Injection Vulnerability in Database Server in mcp-data-vis #19

Open

wing3e opened 2 weeks ago · edited by wing3e Edits Owner

mcp-data-vis SQL Injection Vulnerability in Database Server

1) CNA / Submission Type

- Submission type: Report a vulnerability (CVE ID request)
- Reporter role: Independent security researcher
- Report date: March 15, 2026

2) Reporter Contact (fill before submit)

- Reporter name: `Winegee`
- Reporter email: `Winegee@zju.edu.cn`
- Permission to share contact with vendor: `Yes`

3) Vendor / Product Identification

- Vendor: AlejandroArciniegas
- Product: mcp-data-vis
- Repository: https://github.com/AlejandroArciniegas/mcp-data-vis
- Reviewed local source path: `datasets_set/001/datasets_001/AlejandroArciniegas_mcp-data-vis`
- Affected component(s):
- `src/servers/database/server.js`

4) Vulnerability Type

- CWE: CWE-89 (Improper Neutralization of Special Elements used in an SQL Command)
- Short title: SQL injection in `create_table`

5) Affected Versions

- Confirmed affected: 1.0.0 (version declared in local dataset)
- Suspected affected range: versions containing the same request-to-sink flow documented below
- Fixed version: Not available at time of report (March 15, 2026)

6) Vulnerability Description

AlejandroArciniegas mcp-data-vis contains an SQL injection vulnerability in `src/servers/database/server.js`. The `create_table` tool constructs a `CREATE TABLE` statement by embedding an attacker-controlled `schema` value directly into SQL text and executes it with `db.exec()` without parameterization or strict validation. An attacker who can invoke the vulnerable MCP handler can execute unintended SQL statements against the application's SQLite database, which may result in unauthorized data access, modification, or deletion.

7) Technical Root Cause

1. `js/sql-injection`
   - Source: `src/servers/database/server.js:211` (`request`)
   - Sink: `src/servers/database/server.js:344`
   - Sink code: `await db.exec(sql);`
2. Relevant construction:
   - `src/servers/database/server.js:343`
   - Code: `const sql = `CREATE TABLE IF NOT EXISTS ${table_name} (${schema})`;`
3. Validation gap:
   - `table_name` is regex-validated, but `schema` is inserted into the SQL statement without sanitization or parameter binding.

```
      case "create_table": {
        const { table_name, schema } = args;

        // Basic validation
        if (!/^[a-zA-Z_][a-zA-Z0-9_]*$/.test(table_name)) {
          throw new Error("Invalid table name");
        }

        const sql = `CREATE TABLE IF NOT EXISTS ${table_name} (${schema})`;
        await db.exec(sql);
```

```
        return {
          content: [
            {
              type: "text",
              text: `Table '${table_name}' created successfully`,
            },
          ],
        };
```

8) Attack Prerequisites

- Ability to invoke the exposed MCP tool or RPC method that reaches `create_table`.
- No intermediary validation layer that constrains or rewrites the attacker-controlled `schema` argument.
- Access to a deployment where the database server component is exposed to untrusted callers.

9) Proof of Concept / Reproduction Guidance

PoC transport: JSON-RPC `tools/call` to the MCP server.

Representative request:

```
{"jsonrpc":"2.0","id":1,"method":"tools/call","params":{"name":"create_table","argument
```

Analysis:

- `CallToolRequestSchema` dispatches attacker-controlled request parameters into the `create_table` handler.
- The handler validates `table_name` but does not validate or parameterize `schema`.
- The resulting SQL text is passed to `db.exec(sql)`, which executes the constructed statement string.
- A malicious `schema` value can therefore alter the intended statement and introduce additional SQL operations.

10) Security Impact

- Confidentiality: High, because injected SQL can expose database contents.
- Integrity: High, because injected SQL can modify schema or stored data.
- Availability: High, because injected SQL can delete tables or otherwise render the database unusable.
- Scope: Unchanged.

11) CVSS v3.1 Suggestion

- Suggested vector for deployments that expose the vulnerable handler to untrusted callers: `CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H`
- Suggested base score: 9.8 (Critical)
- If the MCP interface is restricted to trusted authenticated operators, adjust `PR` accordingly.

12) Workarounds / Mitigations

- Remove support for attacker-supplied raw schema fragments.
- Replace free-form schema input with a strict structured schema definition that is validated server-side.
- Reject metacharacters and statement separators in user-controlled SQL fragments.
- Limit access to the database MCP server to trusted users and trusted transports only.

13) Recommended Fix

- Eliminate direct string interpolation of user-controlled values into SQL statements.
- Replace `schema` with a constrained declarative format that is converted into SQL by trusted server-side logic.
- Add regression tests proving that attacker-controlled input cannot alter SQL statement structure or execute unintended statements.
- Publish a security advisory with an explicit fixed version once a patch is released.

14) References

- Repository: https://github.com/AlejandroArciniegas/mcp-data-vis
- Package name: `mcp-data-vis`
- Reviewed source file: `src/servers/database/server.js`
- CWE-89: https://cwe.mitre.org/data/definitions/89.html

15) Credits

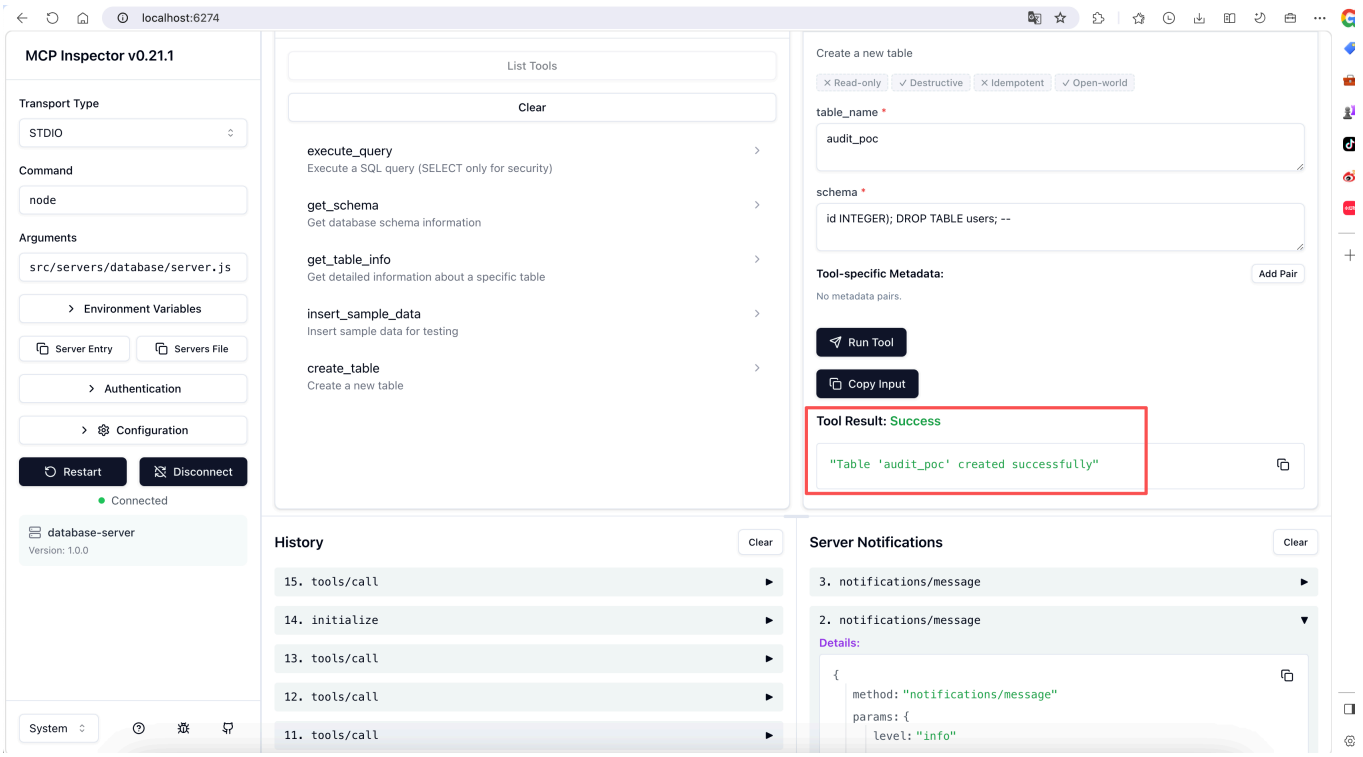
- Discoverer: `Winegee`
- Discovery method: Static analysis plus repository source-code audit

16) Additional Notes for Form Mapping

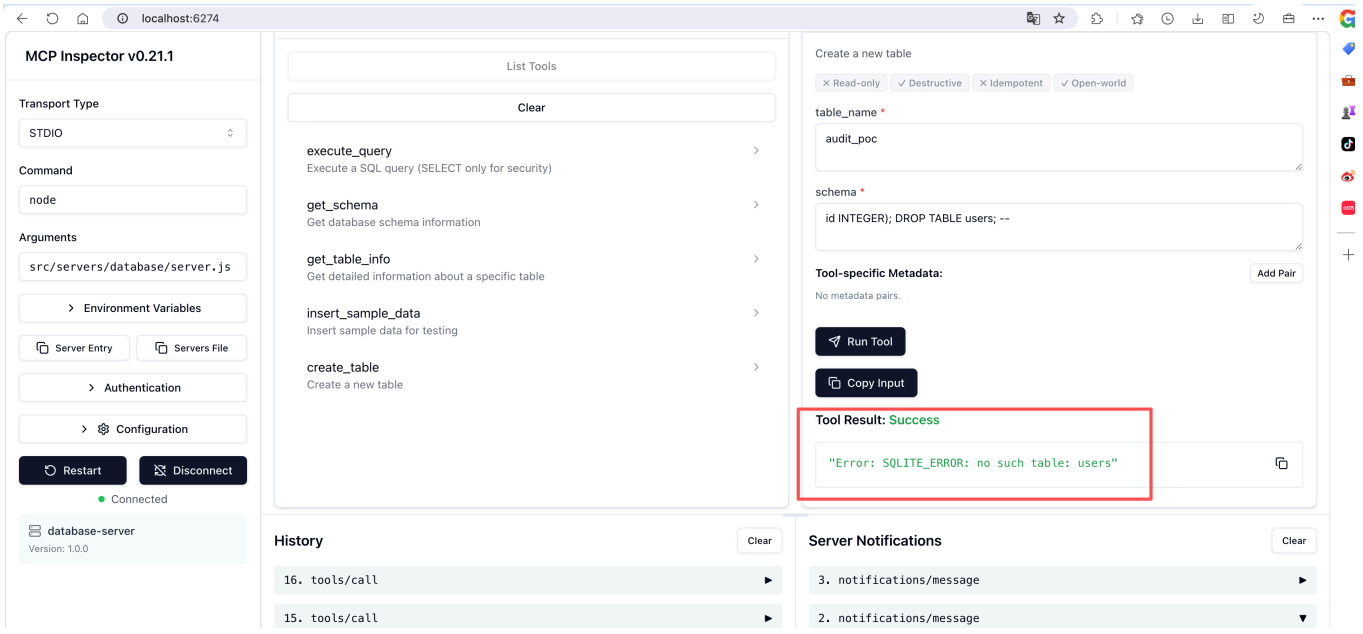
- This issue is distinct from the SSRF issue in the repository's web-scraper component and should be tracked separately.
- Issue status at report time: source-code confirmed in the local dataset.
- Dynamic exploit replay status: not performed for every repository unless separately noted.

- Version-range accuracy should be finalized by the maintainer against release history before public disclosure.

wing3e 2 weeks ago Owner Author



The first execution succeeds



The second execution shows no such table, indicating that the first execution was successful

Metadata

Assignees

No one assigned

Labels

No labels

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

Code with agent mode

No branches or pull requests

Participants