New issue

OS Command Injection Vulnerability in build_project and run_tests of xcode-mcp-server #4

Open

BruceJqs opened 2 weeks ago

OS Command Injection Vulnerability in build_project and run_tests of xcode-mcp-server

1) CNA / Submission Type

- Submission type: Report a vulnerability (CVE ID request)
- Reporter role: Independent security researcher
- Report date: Apr 14, 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: PolarVista
- Product: xcode-mcp-server
- Repository: https://github.com/PolarVista/Xcode-mcp-server
- Affected component(s):
- `src/index.ts`

4) Vulnerability Type

- CWE: CWE-78 (Improper Neutralization of Special Elements used in an OS Command)
- Short title: OS command injection in Xcode build and test MCP tools

5) Affected Versions

- Confirmed affected: `1.0.0`
- Suspected affected range: revisions containing the same request-to-command-execution flows listed below
- Fixed version: Not available at time of report

6) Vulnerability Description

An OS command injection vulnerability (CWE-78) has been identified in xcode-mcp-server version 1.0.0, specifically within the build_project and run_tests MCP tools in src/index.ts. The tools accept user-supplied arguments such as projectPath, scheme, configuration, and destination, interpolate them unsafely into shell command strings, and execute the resulting command with child_process.exec without neutralizing shell metacharacters. An attacker with network access to the MCP interface can inject arbitrary operating system commands that execute 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. `js/command-injection-from-request`
    - Source: `src/index.ts:328` (`request`)
    - Source: `src/index.ts:334` (`const { projectPath, scheme, configuration = "Debug", destination, includeWarnings = false } = request.params.arguments;`)
    - Propagation: `src/index.ts:335` (`this.buildProject(projectPath, scheme, configuration, destination, includeWarnings)`)
    - Propagation: `src/index.ts:393` (`const command = \` which xcodebuild && xcodebuild -project "${projectPath}" ...``)
    - Sink: `src/index.ts:403`
    - Sink code: `const { stdout, stderr } = await execAsync(command, { maxBuffer: 100 * 1024 * 1024 });`
2. `js/command-injection-from-request`
    - Source: `src/index.ts:328` (`request`)
    - Source: `src/index.ts:351` (`request.params.arguments.projectPath`)
    - Source: `src/index.ts:352` (`request.params.arguments.scheme`)
    - Source: `src/index.ts:353` (`request.params.arguments.configuration`)

- Source: `src/index.ts:356` (`request.params.arguments.destination`)
- Propagation: `src/index.ts:350` (`this.runTests(...)`)
- Propagation: `src/index.ts:179` (`const command = \` which xcodebuild && xcodebuild -project "${projectPath}" ...``)
- Sink: `src/index.ts:190`
- Sink code: `const { stdout, stderr } = await execAsync(command, { maxBuffer: 100 * 1024 * 1024 });`

8) Attack Prerequisites

- Attacker can invoke MCP tools exposed by the affected xcode-mcp-server instance.
- The `build_project` or `run_tests` tool is reachable by the attacker.
- The server runs on a system where shell commands can be executed by the Node.js process.
- No effective authentication, authorization, allowlist, sandbox, or runtime policy blocks attacker-controlled shell metacharacters before command execution.
- A valid Xcode project is not required for the demonstrated command-injection primitive, because the injected command executes as part of shell evaluation.

9) Proof of Concept / Reproduction Guidance

This proof of concept provides a concise, CVE-style reproduction example for the reported issue.

1. Start the affected server with MCP Inspector

```
cd Xcode-mcp-server
npx @modelcontextprotocol/inspector node build/index.js /tmp/xcode-mcp-base
```

2. Reproduction request
   Call the `run_tests` tool in MCP Inspector with the following arguments:

```
{
  "projectPath": "/tmp/Fake.xcodeproj",
  "scheme": "\"; id >&2; : \"",
  "configuration": "Debug"
}
```

3. Validation

- Confirm that the MCP tool response or stderr-visible output contains the output of `id`, for example `uid=... gid=... groups=...`.
- The payload closes the quoted `-scheme` argument, executes `id >&2` as an independent shell command, and then uses `: "` to absorb the remaining command text.

10) Security Impact

- Confidentiality: High (attacker-controlled commands can read sensitive files and environment data accessible to the server process).
- Integrity: High (attacker-controlled commands can modify files and application state accessible to the server process).
- Availability: High (attacker-controlled commands can terminate processes, consume resources, or disrupt the host).
- Scope: Unchanged.

11) CVSS v3.1 Suggestion

- Suggested vector: `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)
- Adjust `AV` and `PR` if the vulnerable MCP server is strictly local-only or available only to authenticated trusted users.

12) Workarounds / Mitigations

- Disable or remove the `build_project` and `run_tests` tools when the MCP server is reachable by untrusted callers.
- Do not execute user-provided strings through a shell.
- Restrict allowed project paths, scheme names, configurations, and destinations to validated allowlists where feasible.
- Add authentication, authorization, logging, and rate limiting for build and test execution tools.
- Run the MCP server with least-privilege permissions and an OS-level sandbox where possible.

13) Recommended Fix

- Eliminate the request-to-command-execution data flows documented above.
- Replace `exec(command)` with `execFile` or `spawn` using a fixed executable path and an argument array with `shell: false`.
- Avoid constructing command strings with user-controlled values.
- Validate `projectPath` as a path under an approved workspace, and validate `scheme`, `configuration`, `destination`, `testIdentifier`, and `skipTests` against strict allowlists or safe character sets.
- Quote-free argument arrays should be used for `xcodebuild`, `xcrun xcresulttool`, and `xcrun xccov`.
- Add regression tests proving payloads such as `"; id >&2; : "` cannot execute shell commands through `build_project` or `run_tests`.
- Publish a maintainer security advisory once a patch is released.

14) References

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

15) Credits

- Discoverer: `BruceJin`
- Discovery method: Static analysis (CodeQL) plus repository source-code audit and dynamic reproduction with MCP Inspector

16) Additional Notes for Form Mapping

- Audit verdict: Exploitable: attacker-controlled MCP request arguments can reach shell command execution sinks.
- Dynamic exploit replay status: completed successfully with MCP Inspector using `id >&2`.
- Maintainer should validate release mapping before coordinated disclosure.

For furthermore information, please refer to BruceJqs/public_exp#19

Metadata

Assignees

No one assigned

Labels

No labels

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

No branches or pull requests

Participants