Command Injection in MCP Server mcp_code_executor

Title

Command Injection in ***install_dependencies*** tool due to unsafe use of child_process.execAsync

Summary

A command injection vulnerability exists in *mcp_code_executor* due to unsafe use of child_process.execAsync when constructing Python execution and package installation commands with user-controlled input. Successful exploitation allows attackers to execute arbitrary shell commands with the privileges of the MCP server process.

Details

The MCP tool ***install_dependencies*** constructs a command string using user-supplied parameter *packages*, and executes it via child_process.execAsync. Because execAsync invokes commands through a system shell, specially crafted input containing shell metacharacters (such as `;`, `&`, or `|`) may be interpreted as additional commands rather than treated as data.

For example, an attacker may supply a malicious value in *packages* to inject arbitrary shell commands, which are then executed with the privileges of the MCP server process.

The vulnerability results from shell-based command execution combined with direct interpolation of untrusted user input. In MCP environments, LLM-generated tool parameters influenced by external content may trigger execution of injected commands without direct local user interaction.

PoC

1. *Start the MCP server:*

- npm install
- npm run build
- Set environment variables (ENV_TYPE=venv-uv, UV_VENV_PATH, CODE_STORAGE_DIR)

2. *Open the MCP Inspector:*

npx @modelcontextprotocol/inspector

3. *In MCP Inspector:*

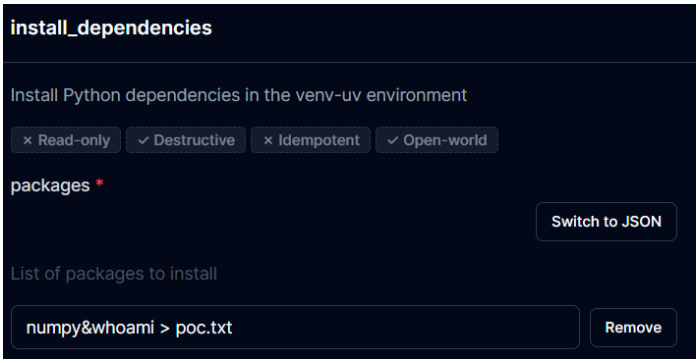
- set transport type: STDIO
- set the command to node
- set the arguments to build/index.js
- click Connect
- go to the Tools tab and click List Tools
- select the ***install_dependencies*** tool

4. *Verify the file poc.txt does not exist in the CODE_STORAGE_DIR:*

- type D:\claude-projects\mcp_code_executor\generated_code\poc.txt
- The system cannot find the file specified

5. *In the packages field, input:*

numpy&whoami > poc.txt



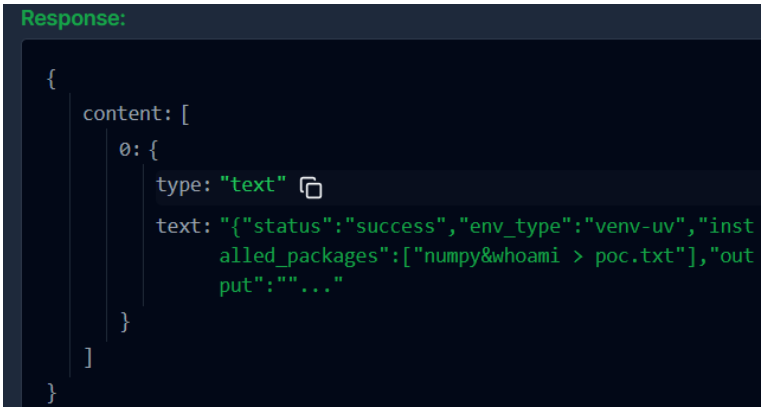
6. *Click Run Tool*

The command executed will be:

<UV_VENV_PATH>\Scripts\activate && uv pip install numpy&whoami > poc.txt

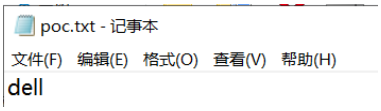
7. *Observe the request being sent:*

8. Response:



9. *Confirm that the injected command executed:*

- type D:\claude-projects\mcp_code_executor\generated_code\poc.txt
- dell



Impact

Successful exploitation allows attackers to execute arbitrary commands on the server hosting the MCP service. This may allow attackers to execute commands, access sensitive data, or modify the host environment depending on the privileges of the MCP server.

Recommendation

- Don't use execAsync. Use execFileAsync instead, which pins the command and provides the arguments as array elements.
- Apply strict input validation to all tool parameters exposed to MCP clients, especially *packages* parameter.
- Use parameter separation with proper escaping to prevent shell command injection.

Reference

https://github.com/bazinga012/mcp_code_executor/issues/17
https://github.com/bazinga012/mcp_code_executor/pull/18 (patch)

Affected products

Ecosystem

npm

Package name

mcp_code_executor

Affected versions

<= 0.3.0

Patched versions

None

Severity

CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:H/I:H/A:H

Weaknesses (CWE)

CWE-78

Credit

Discovered and reported by Yinci Chen.