Command Injection in MCP Server mcp_code_executor

Package

bazinga012/mcp_code_executor (GitHub)

Affected versions

<= 0.3.0

Patched versions

None

Description

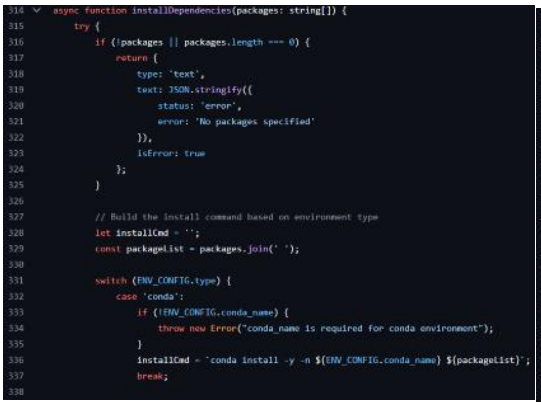
The MCP Server at https://github.com/bazinga012/mcp_code_executor is written in a way that is vulnerable to command injection vulnerability attacks as part of some of its MCP Server tool definition and implementation.

Vulnerable tool

The MCP Server exposes the tool ***install_dependencies*** which relies on Node.js child process API exec which is an unsafe and vulnerable API if concatenated with untrusted user input.

LLM-exposed user input for *packages* arg can be replaced with shell meta-characters like ; or && or others to change the behavior from running the expected command to another command.

Vulnerable line of code: https://github.com/bazinga012/mcp_code_executor/blob/master/src/index.ts#L314-L359



```
async function installDependencies(packages: string[]) {
    try {
        if (!packages || packages.length === 0) {
            return {
                type: 'text',
                text: JSON.stringify({
                    status: 'error',
                    error: 'No packages specified'
                }),
                isError: true
            };
        }

        // Build the install command based on environment type
        let installCmd = '';
        const packageList = packages.join(' ');

        switch (ENV_CONFIG.type) {
            case 'conda':
                if (!ENV_CONFIG.conda_name) {
                    throw new Error("conda_name is required for conda environment");
                }
                installCmd = `conda install -y -n ${ENV_CONFIG.conda_name} ${packageList}`;
                break;
```

```
            case 'venv':
                installCmd = `pip install ${packageList}`;
                break;

            case 'venv-uv':
                installCmd = `uv pip install ${packageList}`;
                break;

            default:
                throw new Error(`Unsupported environment type: ${ENV_CONFIG.type}`);
        }

        // Get platform-specific command
        const { command, options } = getPlatformSpecificCommand(installCmd);

        // Execute installation with unbuffered Python
        const { stdout, stderr } = await execAsync(command, {
            cwd: CODE_STORAGE_DIR,
            env: { ...process.env, PYTHONUNBUFFERED: '1' },
            ...options
        });
```

PoC: Using Inspector

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:*

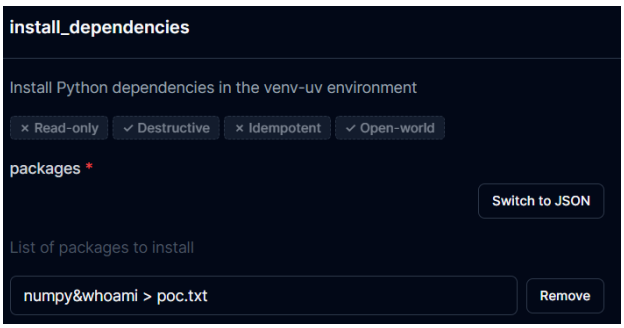
- set transport type: STDIO
- set the command to node
- set the arguments to build/index.js
- click Connect
- go to the Tools tab and click List Tools
- select the ***install_dependencies*** tool

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



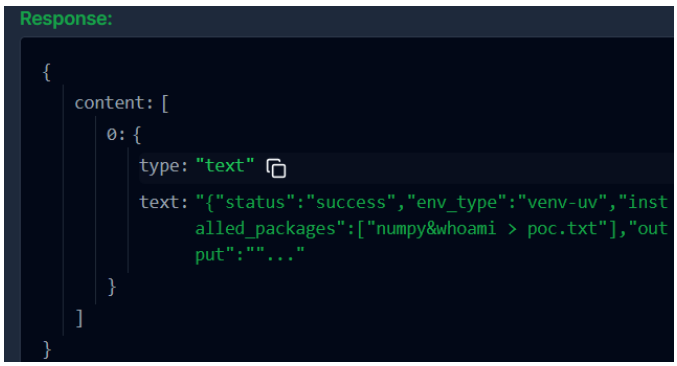
6. *Click Run Tool*

The command executed will be:

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

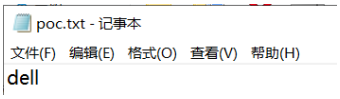
7. *Observe the request being sent:*

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 exec. Use execFile 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 the *packages* parameter.
- Use parameter separation with proper escaping to prevent shell command injection.

References and Prior work

1. Exploiting MCP Servers Vulnerable to Command Injection
2. GHSA-h4w9-g9c5-vfwq