Command Injection in MCP Server mcp-code-review-server

Package

crazyrabbitLTC/mcp-code-review-server (GitHub)

Affected versions

<= 0.1.0

Patched versions

None

Description

The MCP Server at https://github.com/crazyrabbitLTC/mcp-code-review-server/ 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 tools

The MCP Server exposes 2 tools that rely on Node.js child process API exec() which is an unsafe and vulnerable API when concatenated with untrusted user input. The tools affected by this unsafe command execution pattern are as follows:

- ***analyze_repo***
- ***code_review***

Both tools construct Repomix commands by concatenating the *specificFiles* parameter directly into shell commands and executes via exec(). The *repoPath* parameter can also be exploited in the code_review tool.

Take ***analyze_repo*** for example, LLM-exposed user input for *specificFiles* args can be replaced with shell meta-characters like ;, &, |, or others to change the behavior from running the expected command to another command.

analyze_repo:

call chain:

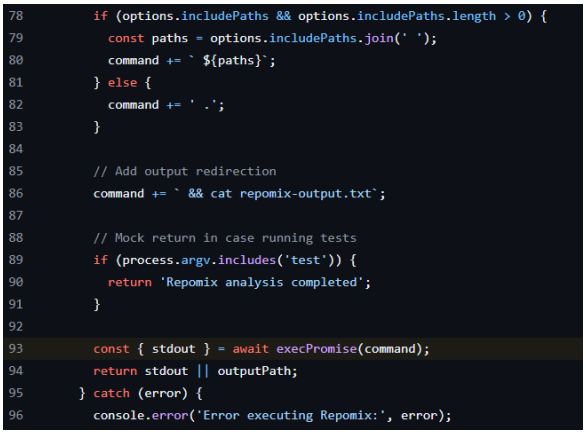
1. MCP client calls ***analyze_repo*** tool with *specificFiles* parameter
2. Tool handler in src/index.ts passes *specificFiles* to executeRepomix() as includePaths
3. executeRepomix() in src/repomix.ts concatenates includePaths directly into shell command
4. Command executed via unsafe exec() call

Vulnerable lines of code:

(*Command construction with user input*): https://github.com/crazyrabbitLTC/mcp-code-review-server/blob/main/src/repomix.ts#L78-L83

(*Parameter definition without validation*): https://github.com/crazyrabbitLTC/mcp-code-review-server/blob/ac4f2817/src/index.ts#L34-L38

(*Unsafe exec() call*): https://github.com/crazyrabbitLTC/mcp-code-review-server/blob/main/src/repomix.ts#L93



```
78        if (options.includePaths && options.includePaths.length > 0) {
79          const paths = options.includePaths.join(' ');
80          command += ` ${paths}`;
81        } else {
82          command += ' .';
83        }
84
85        // Add output redirection
86        command += ` && cat repomix-output.txt`;
87
88        // Mock return in case running tests
89        if (process.argv.includes('test')) {
90          return 'Repomix analysis completed';
91        }
92
93        const { stdout } = await execPromise(command);
94        return stdout || outputPath;
95      } catch (error) {
96        console.error('Error executing Repomix:', error);
```

code_review:

call chain:

1. MCP client calls ***code_review*** tool with *specificFiles* or *repoPath* parameter
2. Tool handler in src/index.ts passes parameters to executeRepomix() as includePaths
3. executeRepomix() in src/repomix.ts concatenates includePaths directly into shell comm and
4. Command executed via unsafe exec() call

Vulnerable lines of code:

(*Command construction with user input*): https://github.com/crazyrabbitLTC/mcp-code-review-server/blob/main/src/repomix.ts#L78-L83

(*Parameter definition without validation*): https://github.com/crazyrabbitLTC/mcp-code-review-server/blob/main/src/index.ts#L58-L64

(*Unsafe exec() call*): https://github.com/crazyrabbitLTC/mcp-code-review-server/blob/main/src/repomix.ts#L93

PoC: Using Inspector

[Note: we only test the ***analyze_repo*** tool, the same procedure applies to ***code_review***]

1. *Start the MCP server:*

- npm install
- npm run build
- select LLM_PROVIDER=OPEN_AI and set OPENAI_API_KEY

2. *Open the MCP Inspector:*

npx @modelcontextprotocol/inspector

3. *In MCP Inspector:*

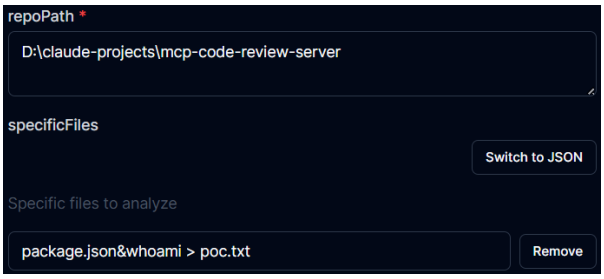
- set transport type: STDIO
- set the command to node
- set the arguments to build/index.js
- click Connect
- go to the Tools tab and click List Tools
- select the ***analyze_repo*** tool

4. *Verify the file poc.txt does not exist in the current working directory:*

- type poc.txt
- The system cannot find the file specified

5. *In the **specificFiles** field, input:*

package.json&whoami > poc.txt
(In the *repoPath* field, input: D:\claude-projects\mcp-code-review-server. And all other parameters use their default values)



6. *Click Run Tool*

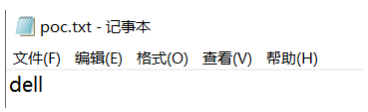
The command executed will be:
repomix --style plain package.json&whoami > poc.txt && cat repomix-output.txt

7. *Observe the request being sent:*

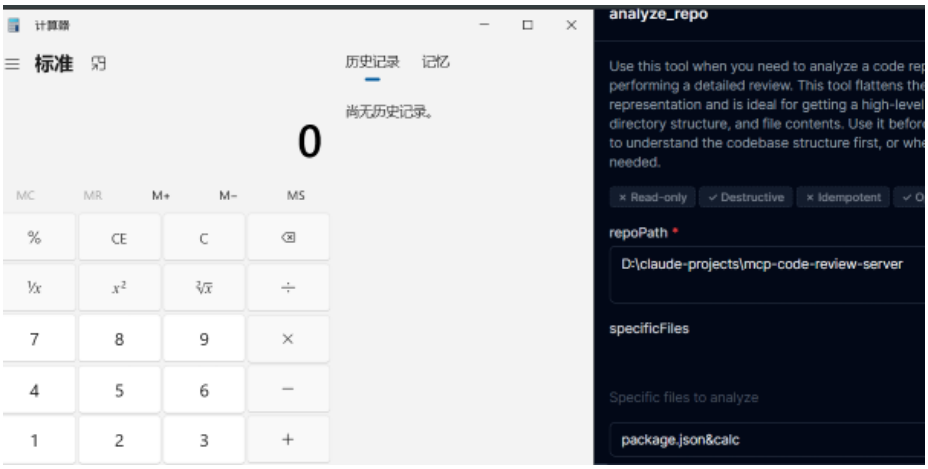
8. Response:

9. *Confirm that the injected command executed:*

- type poc.txt
- dell



*In addition, in the **specificFiles** field, we can also input:*
package.json&calc



After clicking Run Tool, the calculator will pop up in the running interface.

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 *specificFiles* and *repoPath* parameters.
- 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
3. GHSA-xc68-rrqc-qgq3