Command Injection in MCP Server WireMCP

Title

Command Injection in multiple WireMCP tools due to unsafe use of child_process.exec

Summary

A command injection vulnerability exists in WireMCP due to unsafe use of child_process.exec when constructing Tshark CLI commands with user-controlled input. Successful exploitation allows attackers to execute arbitrary shell commands with the privileges of the MCP server process.

Details

The following MCP tools construct command strings using user-supplied parameters and execute them via child_process.exec:

- ***capture_packets***: interpolates *interface* and *duration*
- ***get_summary_stats***: interpolates *interface* and *duration*
- ***get_conversations***: interpolates *interface* and *duration*
- ***check_threats***: interpolates *interface* and *duration*
- ***analyze_pcap***: interpolates *pcapPath*
- ***extract_credentials***: interpolates *pcapPath*

Because exec 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 could supply a malicious value in *interface* to inject arbitrary shell commands, which would then be executed with the privileges of the MCP server process.

The vulnerability results from shell-based command execution combined with direct interpolation of untrusted 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

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 index.js
- click Connect
- go to the Tools tab and click List Tools
- select the ***capture_packets*** tool

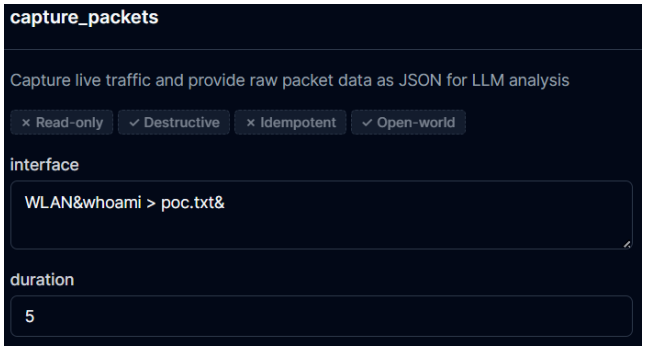
4. *Verify the file poc.txt does not exist in the current working directory:*

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

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

WLAN&whoami > poc.txt&

[*duration* is set to the default value of 5]

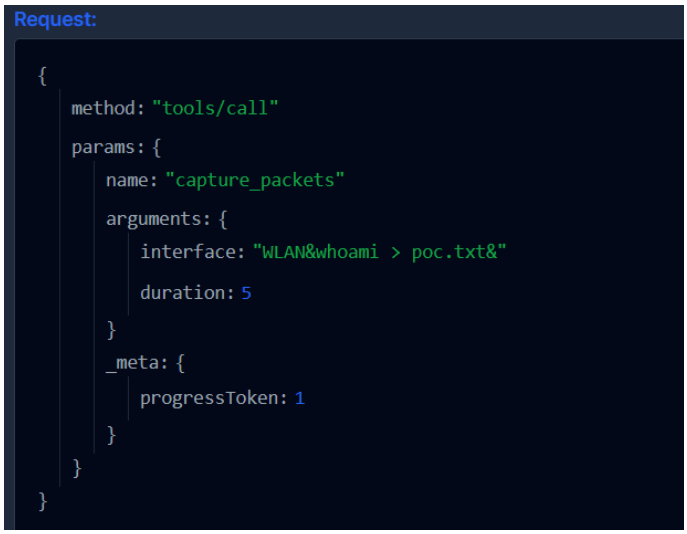


6. *Click Run Tool*

The command executed will be:

tshark -i WLAN&whoami > poc.txt& -w temp_capture.pcap -a duration:5

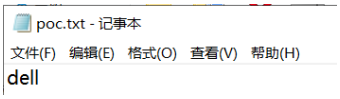
7. *Observe the request being sent:*



8. Response:

9. *Confirm that the injected command executed:*

- type 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 *interface*, *pcapPath*, *and duration* parameters.
- Use parameter separation with proper escaping to prevent shell command injection.

References

https://github.com/0xKoda/WireMCP/issues/12
https://github.com/0xKoda/WireMCP/pull/13

Affected products

Ecosystem

npm

Package name

wiremcp

Affected versions

<= 1.0.0

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.