Arbitrary File Write Vulnerability in mcp-mt4-server of 8nite/metatrader-4-mcp #1

Open

BruceJqs opened 2 weeks ago

Arbitrary File Write Vulnerability in mcp-mt4-server of 8nite/metatrader-4-mcp

1) CNA / Submission Type

- Submission type: Report a vulnerability (CVE ID request)
- Reporter role: Independent security researcher
- Report date: Apr 16, 2026

2) Reporter Contact

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

3) Vendor / Product Identification

- Vendor: 8nite
- Product: metatrader-4-mcp / mcp-mt4-server
- Repository: https://github.com/8nite/metatrader-4-mcp
- Affected component(s):
- `src/index.ts`
- `windows-server/server.js`

4) Vulnerability Type

- CWE: CWE-73 (External Control of File Name or Path)
- Short title: Arbitrary file write via MCP-controlled EA name

5) Affected Versions

- Confirmed affected: 1.0.0
- Confirmed affected commit: `454eccc99d6cdac1bda65b99b2c06f893b3c48f2`
- Suspected affected range: revisions containing the same MCP request-to-filesystem flows listed below
- Fixed version: Not available at time of report

6) Vulnerability Description

An arbitrary file write vulnerability (CWE-73) has been identified in mcp-mt4-server version 1.0.0 (commit `454eccc`), specifically within the sync_ea MCP tool in src/index.ts. The tool accepts a user-supplied ea_name argument and concatenates it into filesystem paths without validating that the resolved path remains inside the intended ea-strategies/active or ea-strategies/logs directories. An attacker with network access to the MCP interface can supply parent-directory traversal sequences (e.g., ../..) to create or overwrite files at arbitrary writable locations on the server. Additionally, the sync_ea_from_file tool can read arbitrary local files via a user-controlled file_path argument and write their contents through the same vulnerable path. No fixed version is available at the time of reporting.

7) Technical Root Cause

1. Primary arbitrary file write in `sync_ea`
    - Source: `src/index.ts:295` (`CallToolRequestSchema` handler receives MCP tool arguments)
    - Argument extraction: `src/index.ts:296`
    - Tool dispatch: `src/index.ts:320`
    - Path construction: `src/index.ts:681`
    - Sink: `src/index.ts:682`
    - Sink code: `fs.writeFileSync(eaPath, args.ea_content, 'utf8');`
    - Additional log path construction: `src/index.ts:685`
    - Additional sink: `src/index.ts:687`
    - Sink code: `fs.writeFileSync(syncLogPath, logEntry, 'utf8');`
2. Additional writes using the same attacker-controlled `ea_name`
    - Source: `src/index.ts:295` (`CallToolRequestSchema` handler receives MCP tool arguments)
    - Sink: `src/index.ts:700`
    - Sink code: `fs.writeFileSync(syncLogPath, successLog, 'utf8');`

- Sink: `src/index.ts:713`
- Sink code: `fs.writeFileSync(syncLogPath, manualLog, 'utf8');`
- Sink: `src/index.ts:770`
- Sink code: `fs.writeFileSync(logPath, logContent, 'utf8');`
- Sink: `src/index.ts:775`
- Sink code: `fs.copyFileSync(activePath, compiledPath);`
- Sink: `src/index.ts:792`
- Sink code: `fs.writeFileSync(logPath, manualLog, 'utf8');`

3. Related arbitrary file read surface in `sync_ea_from_file`

- Source: `src/index.ts:295` (`CallToolRequestSchema` handler receives MCP tool arguments)
- Tool dispatch: `src/index.ts:326`
- Existence check: `src/index.ts:882`
- Read sink: `src/index.ts:894`
- Sink code: `const eaContent = fs.readFileSync(args.file_path, 'utf8');`
- Write continuation: `src/index.ts:901`
- Continuation code: `return await this.syncEA({ ea_name: eaName, ea_content: eaContent });`

4. Related HTTP bridge path traversal surface

- Source: `windows-server/server.js:451` (`POST /api/ea/upload` receives HTTP request body)
- Path construction: `windows-server/server.js:464`
- Sink: `windows-server/server.js:467`
- Sink code: `await fs.writeFile(eaFilePath, ea_content, "utf-8");`
- Compile path construction: `windows-server/server.js:505`
- Access sink: `windows-server/server.js:509`
- Sink code: `await fs.access(eaFilePath);`

8) Attack Prerequisites

- Attacker can invoke the MCP server tools, for example through a configured MCP client, MCP Inspector, or another client connected to the server process.
- The MCP server process has filesystem write permissions to the attacker-selected target path.
- No external sandbox, container policy, or operating-system-level access control prevents writes outside the intended EA workspace.
- Exploitation of the primary local write path does not require a live MetaTrader 4 instance, because the file is saved locally before the server attempts any MT4 bridge API call.

9) Proof of Concept / Reproduction Guidance

This proof of concept demonstrates attacker-controlled file creation outside the intended `ea-strategies/active` directory using the exposed `sync_ea` MCP tool.

1. Invoke `sync_ea` with a traversal payload in `ea_name`.

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "tools/call",
  "params": {
    "name": "sync_ea",
    "arguments": {
      "ea_name": "../../mt4-mcp-poc",
      "ea_content": "// path traversal poc\n"
    }
  }
}
```

2. Validation

- Confirm that `mt4-mcp-poc.mq4` is created in the server working directory instead of under `ea-strategies/active/`.
- Confirm that `mt4-mcp-poc_sync.log` is created in the server working directory instead of under `ea-strategies/logs/`.
- Repeat with another path writable by the server process to confirm that `ea_name` is not constrained to a safe EA directory.

3. Related read-and-write demonstration

```
{
  "jsonrpc": "2.0",
  "id": 2,
  "method": "tools/call",
  "params": {
    "name": "sync_ea_from_file",
    "arguments": {
      "file_path": "/etc/hosts",
      "ea_name": "../../copied-hosts"
    }
  }
}
```

- If `/etc/hosts` is readable by the server process, its contents are read and written through the same `sync_ea` path to a caller-selected destination.

10) Security Impact

- Confidentiality: Low to Medium. The related `sync_ea_from_file` path can read arbitrary text files accessible to the server process and copy their contents to attacker-selected EA output files.
- Integrity: High. An attacker can create or overwrite arbitrary files writable by the MCP server process.
- Availability: High. Overwriting application files, logs, configuration, or other writable files can disrupt the MCP server or dependent workflows.
- Scope: Unchanged.

11) CVSS v3.1 Suggestion

- Suggested vector: `CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:H/A:H`
- Suggested base score: 8.1 (High)
- If the MCP server is exposed without authentication or reachable by untrusted clients, consider `PR:N`, which increases severity.

12) Workarounds / Mitigations

- Restrict MCP server access to trusted local users and trusted MCP clients only.
- Run the MCP server under a dedicated low-privilege account with a minimal writable directory.
- Use OS sandboxing, containers, or filesystem permissions to prevent writes outside a designated workspace.
- Avoid using `sync_ea`, `compile_ea`, or `sync_ea_from_file` with untrusted input until a patch is available.

13) Recommended Fix

- Treat `ea_name` as an identifier, not a path. Reject path separators, parent-directory traversal, absolute paths, Windows drive prefixes, and special device names.
- Canonicalize constructed paths with `path.resolve` and enforce that every write target remains under the intended base directory, such as `ea-strategies/active`, `ea-strategies/logs`, or `ea-strategies/compiled`.
- Apply the same canonicalization and containment checks to `sync_ea_from_file` input paths if arbitrary reads are not explicitly intended.
- Apply equivalent validation in the Windows HTTP bridge for `ea_name` and any file names accepted from HTTP request bodies.
- Add regression tests proving that MCP-controlled `ea_name` and `file_path` values cannot escape configured workspace directories.

14) References

- Repository: https://github.com/8nite/metatrader-4-mcp
- Reviewed source file: `src/index.ts`
- Reviewed source file: `windows-server/server.js`
- CWE-73: https://cwe.mitre.org/data/definitions/73.html
- CWE-22: https://cwe.mitre.org/data/definitions/22.html

15) Credits

- Discoverer: `BruceJin`
- Discovery method: Static analysis (CodeQL), source-code audit, and dynamic reproduction

16) Additional Notes for Form Mapping

- Audit verdict: Exploitable: attacker-controlled MCP tool arguments can reach filesystem write sinks.
- Dynamic exploit replay status: Completed successfully.
- Primary vulnerable tool: `sync_ea`
- Additional affected tools: `compile_ea`, `sync_ea_from_file`
- Related affected HTTP endpoint: `POST /api/ea/upload`
- Maintainer should validate release mapping before coordinated disclosure.

For furthermore information, please refer to BruceJqs/public_exp#30

Metadata

Assignees

No one assigned

Labels

No labels

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

No branches or pull requests

Participants