Unbounded memory allocation in readRequestBody allows denial of service via HTTP transport

High **QuantGeekDev** published **GHSA-353c-v8x9-v7c3** 9 hours ago

Package

mcp-framework (npm)

Affected versions

<= 0.2.21

Patched versions

None

Description

Summary

The `readRequestBody()` function in `src/transports/http/server.ts` (lines 224-240) concatenates HTTP request body chunks into a string with no size limit, allowing a remote unauthenticated attacker to crash the server via memory exhaustion with a single large HTTP POST request.

Details

File: `src/transports/http/server.ts`, lines 224-240

```
private async readRequestBody(req: IncomingMessage): Promise<any> {
    return new Promise((resolve, reject) => {
      let body = '';
      req.on('data', (chunk) => {
        body += chunk.toString();   // No size limit
      });
      req.on('end', () => {
        try {
          const parsed = body ? JSON.parse(body) : null;
          resolve(parsed);
        } catch (error) {
          reject(error);
        }
      });
```

```
      req.on('error', reject);
    });
  }
```

A `maxMessageSize` configuration value exists in `DEFAULT_HTTP_STREAM_CONFIG` (4MB, defined in `src/transports/http/types.ts` line 124) but is never enforced in `readRequestBody()`. This creates a false sense of security.

PoC

Local testing with 50MB POST payloads against the vulnerable `readRequestBody()` function:

Trial	Payload	RSS growth	Time	Result
1	50MB	+197MB	42ms	Vulnerable
2	50MB	+183MB	46ms	Vulnerable
3	50MB	+15MB	43ms	Vulnerable
4	50MB	+14MB	32ms	Vulnerable
5	50MB	+65MB	38ms	Vulnerable

Reproducibility: 5/5 (100%)

Impact

- **Denial of Service:** Any mcp-framework HTTP server can be crashed by a single large POST request to /mcp
- **No authentication required:** readRequestBody() executes before any auth checks (auth is opt-in, default is no auth)
- **Dead config:** maxMessageSize exists but is never enforced, giving a false sense of security
- **Affected:** All applications using mcp-framework HttpStreamTransport (60,000 weekly npm downloads)

CWE-770: Allocation of Resources Without Limits or Throttling
Suggested CVSS 3.1: 7.5 (AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H)

Suggested Fix

Enforce `maxMessageSize` in `readRequestBody()`:

```
private async readRequestBody(req: IncomingMessage): Promise<any> {
    const maxSize = this._config.maxMessageSize || 4 * 1024 * 1024;
    return new Promise((resolve, reject) => {
      let body = '';
      let size = 0;
```

```
      req.on('data', (chunk) => {
        size += chunk.length;
        if (size > maxSize) {
          req.destroy();
          reject(new Error('Request body too large'));
          return;
        }
        body += chunk.toString();
      });
      // ...
    });
  }
```

Disclosure Timeline

This report follows coordinated disclosure. I request a 90-day window before public disclosure.

Reporter: Raza Sharif, CyberSecAI Ltd (contact@agentsign.dev)

Severity

High

CVE ID

CVE-2026-39313

Weaknesses

▶ CWE-770

Credits

razashariff Reporter