Memory amplification in SPDY frame parsing leads to denial of service

High **thaJeztah** published **GHSA-pc3f-x583-g7j2** 2 days ago

Package

github.com/moby/spdystream (Go)

Affected versions	Patched versions
<= v0.5.0	v0.5.1

Description

The SPDY/3 frame parser in spdystream does not validate attacker-controlled counts and lengths before allocating memory. A remote peer that can send SPDY frames to a service using spdystream can cause the process to allocate gigabytes of memory with a small number of malformed control frames, leading to an out-of-memory crash.

Three allocation paths in the receive side are affected:

1. **SETTINGS entry count** -- The SETTINGS frame reader reads a 32-bit `numSettings` from the payload and allocates a slice of that size without checking it against the declared frame length. An attacker can set `numSettings` to a value far exceeding the actual payload, triggering a large allocation before any setting data is read.

2. **Header count** -- `parseHeaderValueBlock` reads a 32-bit `numHeaders` from the decompressed header block and allocates an `http.Header` map of that size with no upper bound.

3. **Header field size** -- Individual header name and value lengths are read as 32-bit integers and used directly as allocation sizes with no validation.

   Because SPDY header blocks are zlib-compressed, a small on-the-wire

payload can decompress into attacker-controlled bytes that the parser interprets as 32-bit counts and lengths. A single crafted frame is enough to exhaust process memory.

Impact

Any program that accepts SPDY connections using spdystream -- directly or through a dependent library -- is affected. A remote peer that can send SPDY frames to the service can crash the process with a single crafted SPDY control frame, causing denial of service.

Affected versions

`github.com/moby/spdystream` <= v0.5.0

Fix

v0.5.1 addresses the receive-side allocation bugs and adds related hardening:

Core fixes:

- **SETTINGS entry-count validation** -- The SETTINGS frame reader now checks that `numSettings` is consistent with the declared frame length (`numSettings <= (length-4)/8`) before allocating.
- **Header count limit** -- `parseHeaderValueBlock` enforces a maximum number of headers per frame (default: 1000).
- **Header field size limit** -- Individual header name and value lengths are checked against a per-field size limit (default: 1 MiB) before allocation.
- **Connection closure on protocol error** -- The connection read loop now closes the underlying `net.Conn` when it encounters an `InvalidControlFrame` error, preventing further exploitation on the same connection.

Additional hardening:

- **Write-side bounds checks** -- All frame write methods now verify that payloads fit within the 24-bit length field, preventing the library from producing invalid frames.

Configurable limits:

- Callers can adjust the defaults using `NewConnectionWithOptions` or the lower-level `spdy.NewFramerWithOptions` with functional options: `WithMaxControlFramePayloadSize`, `WithMaxHeaderFieldSize`, and `WithMaxHeaderCount`.

Severity

High

CVE ID

CVE-2026-35469

Weaknesses

No CWEs