fix(security): add rate limiting to WebSocket login endpoint #2568

Merged **tkurki** merged 2 commits into `SignalK:master` from `dirkwa:ws-login-rate-limit` 2 weeks ago

dirkwa commented 2 weeks ago • edited by coderabbitai bot Contributor

Summary

- Add rate limiting to the WebSocket login path, which previously allowed unlimited brute-force attempts
- Replace `express-rate-limit` for login routes with a shared `LoginRateLimiter` so HTTP and WebSocket login share a single per-IP budget (default 100 attempts/10min window)
- Extract `getClientIp()` helper in ws.ts to deduplicate IP resolution logic
- Convert `test/rate-limit.js` to strict TypeScript

Motivation

The HTTP login endpoints were protected by `express-rate-limit`, but the WebSocket login path (`processLoginRequest` in `ws.ts`) called `login()` directly with no throttling. An attacker could bypass HTTP rate limiting entirely by opening a WebSocket connection and attempting unlimited password guesses at ~20 req/s (bcrypt-limited). A 10,000-word dictionary attack would complete in ~8 minutes over a single connection.

Reported via GHSA-vmfm-ch9h-5c7g.

Why a custom rate limiter instead of express-rate-limit?

`express-rate-limit` is Express middleware — its store API is internal and not designed for manual consumption outside the middleware pipeline. Rather than coupling to that internal API (which has broken across major versions before), we introduced a lightweight `LoginRateLimiter` (~45 lines) with a simple `check(ip)` interface that both the Express middleware wrapper and the WebSocket handler call directly. This ensures a single shared budget across both protocols: 50 HTTP + 50 WS = 100 total, not 100 + 100.

`express-rate-limit` remains in use for the other HTTP rate limiters (access requests, request status, login status) in `serverroutes.ts` — those have no WebSocket equivalent and don't need sharing.

Manual verification

Tested against a live server on port 4000 with security enabled:

- Fired 100 HTTP login attempts — budget exhausted, attempt 101 returned 429
- Opened a WebSocket connection and sent a login attempt — returned 429 with the shared rate limit message
- Confirmed the budget is shared: HTTP attempts reduce the WebSocket budget and vice versa

Test changes

- Converted `test/rate-limit.js` to `test/rate-limit.ts`
- Each login rate-limit scenario now uses its own server instance to ensure test isolation (the previous tests shared one server, so later tests were operating with an already-exhausted budget)
- Added: WebSocket login rate limiting (100 WS attempts then 429)
- Added: cross-protocol budget sharing (50 HTTP + 50 WS then both protocols blocked)

Rate Limiting for WebSocket Login Endpoint

Overview

This PR adds rate limiting to the WebSocket login handler and centralizes login throttling across HTTP and WebSocket flows using a shared LoginRateLimiter. The limiter enforces a per-IP budget (default 100 attempts per 10 minutes) to mitigate high-rate brute-force and dictionary attacks.

Key Changes

New Module: Login Rate Limiter

- Adds `src/login-rate-limiter.ts` implementing an in-memory, per-IP limiter with:
  - createLoginRateLimiter(windowMs, max) factory

- check(ip): { allowed, retryAfterMs }
- dispose() cleanup
- periodic cleanup of expired entries and a shared `LOGIN_RATE_LIMIT_MESSAGE`

WebSocket IP handling and Rate Limiting

- Adds `getClientIp(app, spark)` in `src/interfaces/ws.ts` to centralize IP resolution (respects `settings.trustProxy` and uses the first value from X-Forwarded-For when enabled).
- Extends `SecurityStrategy` with optional `loginRateLimiter?: LoginRateLimiter`.
- `processLoginRequest` uses the shared limiter: it calls `rateLimiter.check(getClientIp(...))` and, on denial, responds over the socket with `statusCode: 429`, `state: 'COMPLETED'`, and `message: LOGIN_RATE_LIMIT_MESSAGE` without proceeding to login logic.

HTTP login rate limiting refactor

- Replaces express-rate-limit usage for login routes in `src/tokensecurity.ts` with the shared `LoginRateLimiter`.
- Adds Express middleware that calls `loginRateLimiter.check(req.ip ?? '')`, sets `Retry-After` on denial, and returns HTTP 429 with `{ message: LOGIN_RATE_LIMIT_MESSAGE }`.
- Retains express-rate-limit for other HTTP-only rate limiters.

Tests and Test Cleanup

- Replaces `test/rate-limit.js` with `test/rate-limit.ts` (strict TypeScript).
- Tests create isolated server instances per scenario.
- Adds suites verifying:
  - HTTP login rate limiting
  - WebSocket login rate limiting
  - Cross-protocol budget sharing (HTTP and WebSocket share the same per-IP bucket)
  - HTTP API rate limiting for other endpoints
  - Trust-proxy behavior and per-IP bucketing via X-Forwarded-For
- Fixes trustProxy test to exhaust a forwarded-IP bucket and confirm different forwarded IPs remain allowed.

Security Impact

- Enforces a shared per-IP login budget across HTTP and WebSocket, preventing previously unthrottled high-rate login attempts (addresses GHSA-vmfm-ch9h-5c7g).
- Manual verification confirms cross-protocol enforcement: exhausting HTTP attempts causes subsequent WebSocket login attempts to be rate-limited.

Miscellaneous

- A follow-up issue (**feat(security): add input validation for HTTP_RATE_LIMITS (reject zero/negative values)** #2570) is created to add input validation for HTTP_RATE_LIMITS (ensure windowMs and max are positive integers); input validation is considered out of scope for this security fix.

coderabbitai bot commented 2 weeks ago • edited

> Note
>
> ## Reviews paused
>
> It looks like this branch is under active development. To avoid overwhelming you with review comments due to an influx of new commits, CodeRabbit has automatically paused this review. You can configure this behavior by changing the `reviews.auto_review.auto_pause_after_reviewed_commits` setting.
>
> Use the following commands to manage reviews:
>
> - `@coderabbitai resume` to resume automatic reviews.
> - `@coderabbitai review` to trigger a single review.
>
> Use the checkboxes below for quick actions:
>
> - [ ] ▶️ Resume reviews
> - [ ] 🔍 Trigger review

▶ 📝 Walkthrough

▶ 🚥 Pre-merge checks | ✅ 2 | ❌ 1

▶ ✨ Finishing Touches

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

coderabbitai bot reviewed 2 weeks ago

View reviewed changes

coderabbitai bot left a comment

Actionable comments posted: 2

▶ 🤖 Prompt for all review comments with AI agents

▶ 🪄 Autofix (Beta)

▶ ℹ️ Review info

`src/interfaces/ws.ts` Show resolved

`src/login-rate-limiter.ts` Outdated Show resolved

dirkwa force-pushed the `ws-login-rate-limit` branch from **`bb69b18`** to **`0537967`** Compare
2 weeks ago

coderabbitai bot reviewed 2 weeks ago

View reviewed changes

coderabbitai bot left a comment

Actionable comments posted: 2

▶ ♻️ Duplicate comments (1)

▶ 🤖 Prompt for all review comments with AI agents

▶ 🪄 Autofix (Beta)

▶ ℹ️ Review info

`src/login-rate-limiter.ts` Show resolved

`test/rate-limit.ts` Outdated Show resolved

dirkwa force-pushed the `ws-login-rate-limit` branch from **`0537967`** to **`dcf019f`** 2 weeks ago Compare

coderabbitai bot reviewed 2 weeks ago

View reviewed changes

coderabbitai bot left a comment

Actionable comments posted: 2

▶ ♻️ Duplicate comments (1)

▶ 🤖 Prompt for all review comments with AI agents

▶ 🪄 Autofix (Beta)

▶ ℹ️ Review info

`src/index.ts` Outdated Show resolved

`test/rate-limit.ts` Outdated Show resolved

`fix(security): add rate limiting to WebSocket login endpoint` ··· ✓ `5703ab5`

dirkwa force-pushed the `ws-login-rate-limit` branch from **`dcf019f`** to **`5703ab5`** 2 weeks ago Compare

coderabbitai bot reviewed 2 weeks ago

View reviewed changes

coderabbitai bot left a comment

> **Caution**
>
> Some comments are outside the diff and can’t be posted inline due to platform limitations.

▶ ⚠️ Outside diff range comments (1)

▶ ♻️ Duplicate comments (3)

▶ 🤖 Prompt for all review comments with AI agents

▶ ℹ️ Review info

dirkwa commented 2 weeks ago — Contributor — Author

@coderabbitai

The input validation for `HTTP_RATE_LIMITS` (rejecting zero/negative values) is a valid improvement, but it is pre-existing code outside the scope of this security fix. Please create a follow-up issue for it so we can address it separately.

👀 1

This was referenced 2 weeks ago

feat(security): add input validation for HTTP_RATE_LIMITS (reject zero/negative values) #2569

Open

feat(security): add input validation for HTTP_RATE_LIMITS (reject zero/negative values) #2570

Open

coderabbitai bot commented 2 weeks ago

> Tip
>
> For best results, initiate chat on the files or code changes.

▶ 🧩 Analysis chain

`@dirkwa`: Done — I've opened `#2570` to track input validation for `HTTP_RATE_LIMITS` (rejecting zero/negative values) as a separate follow-up. Note: the integration couldn't set the assignee automatically, so you may want to assign it manually.

▶ ✏️ Learnings added

▶ 🧠 Learnings used

dirkwa commented 2 weeks ago — Contributor — Author

ready for human review

tkurki added the fix label 2 weeks ago

tkurki reviewed 2 weeks ago

View reviewed changes

`test/rate-limit.ts` — Show resolved

`test(rate-limit): fix trustProxy test to verify per-IP buckets` ··· ✓ `74902b0`

coderabbitai bot reviewed 2 weeks ago

View reviewed changes

coderabbitai bot left a comment

Actionable comments posted: 1

▶ ♻️ Duplicate comments (1)

▶ 🤖 Prompt for all review comments with AI agents

▶ 🪄 Autofix (Beta)

▶ ℹ️ Review info

`test/rate-limit.ts` Show resolved

dirkwa commented 2 weeks ago Contributor Author

ready for review, i ignored the nitpicks.

1 similar comment

tkurki merged commit `215d81e` into `SignalK:master` 2 weeks ago View details

4 checks passed

coderabbitai bot mentioned this pull request 2 weeks ago

feat(ws): limit concurrent WebSocket connections per IP #2578

Merged

Sign up for free **to join this conversation on GitHub**. Already have an account? Sign in to comment

Reviewers

tkurki

coderabbitai[bot]

Assignees

No one assigned

Labels

fix

Projects

None yet

Milestone

No milestone

Development

Successfully merging this pull request may close these issues.

None yet

2 participants