SignalK / **signalk-server** Public

Code Issues 196 Pull requests 58 Actions Projects Wiki Sec

Commit 215d81e

dirkwa authored 2 weeks ago · 14 / 14 · Verified

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

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

The WebSocket login path called login() directly with no rate limiting
while the HTTP path was protected by express-rate-limit. This allowed
unlimited brute-force attempts over WebSocket at ~20 req/s.

Replace express-rate-limit for login routes with a shared LoginRateLimiter
that both HTTP and WebSocket login paths consume. A single per-IP budget
(default 100 attempts/10min) is enforced across both protocols.

Convert test/rate-limit.js to TypeScript and add dedicated tests for
WebSocket rate limiting and cross-protocol budget sharing.

* test(rate-limit): fix trustProxy test to verify per-IP buckets

The existing test claimed to verify X-Forwarded-For handling but used a
single static IP for all requests, so it would pass even if the server
ignored the header entirely. Replace with a test that exhausts the
budget for one forwarded IP and then verifies a different forwarded IP
is still allowed through — proving the rate limiter actually keys on the
forwarded address.
```

master (#2568) · v2.25.0

1 parent 9cd2ebe commit 215d81e

6 files changed +475 -229 lines changed

Top

Filter files…

- src
  - interfaces
    - ws.ts
  - login-rate-limiter.ts

- security.ts
- tokensecurity.ts
- test
  - rate-limit.js
  - rate-limit.ts

6 files changed +475 -229 lines changed

Search within code

src/interfaces/ws.ts

```
@@ -24,6 +24,10 @@ import {
24 24   InvalidTokenError,
25 25   WithSecurityStrategy
26 26 } from '../security'
   27 + import {
   28 +   LoginRateLimiter,
   29 +   LOGIN_RATE_LIMIT_MESSAGE
   30 + } from '../login-rate-limiter'
27 31 import { WithConfig } from '../app'
28 32 import {
29 33   findRequest,
@@ -135,6 +139,7 @@ interface SecurityStrategy {
135 139     timeToLive?: number | null
136 140   }>
137 141   isDummy: () => boolean
   142 +   loginRateLimiter?: LoginRateLimiter
138 143 }
139 144
140 145 interface SubscriptionManager {
@@ -668,6 +673,22 @@ function wsInterface(app: WsApp): WsApi {
668 673     })
669 674   }
670 675
   676 +   function getClientIp(app: WsApp, spark: Spark): string {
   677 +     if (
   678 +       app.config.settings.trustProxy &&
   679 +       app.config.settings.trustProxy !== 'false'
```

```
                680 +     ) {
                681 +       const forwardedFor = spark.request.headers['x-forwarded-for']
                682 +       if (typeof forwardedFor === 'string') {
                683 +         const firstIp = forwardedFor.split(',')[0].trim()
                684 +         if (firstIp) {
                685 +           return firstIp
                686 +         }
                687 +       }
                688 +     }
                689 +     return spark.request.connection.remoteAddress
                690 +   }
                691 +
671 692       function processAccessRequest(
672 693         app: WsApp,
673 694         spark: Spark,
                  @@ -681,13 +702,7 @@ function wsInterface(app: WsApp): WsApi {
681 702             message: 'A request has already been submitted'
682 703           })
683 704         } else {
684       -       const forwardedFor = spark.request.headers['x-forwarded-for']
685       -       const clientIp =
686       -         (app.config.settings.trustProxy &&
687       -           app.config.settings.trustProxy !== 'false' &&
688       -           typeof forwardedFor === 'string' &&
689       -           forwardedFor) ||
690       -         spark.request.connection.remoteAddress
                705 +       const clientIp = getClientIp(app, spark)
691 706           requestAccess(
692 707             app as unknown as WithSecurityStrategy & WithConfig,
693 708             msg,
                  @@ -725,6 +740,20 @@ function wsInterface(app: WsApp): WsApi {
725 740       }
726 741
727 742       function processLoginRequest(app: WsApp, spark: Spark, msg: WsMessage): void
                  {
                743 +     const rateLimiter = app.securityStrategy.loginRateLimiter
                744 +     if (rateLimiter) {
                745 +       const { allowed } = rateLimiter.check(getClientIp(app, spark))
                746 +       if (!allowed) {
```

```
747 +         spark.write({
748 +           requestId: msg.requestId,
749 +           state: 'COMPLETED',
750 +           statusCode: 429,
751 +           message: LOGIN_RATE_LIMIT_MESSAGE
752 +         })
753 +         return
754 +       }
755 +     }
756 +
728 757       app.securityStrategy
729 758         .login(msg.login!.username, msg.login!.password)
730 759         .then((reply) => {
```

src/login-rate-limiter.ts

```
    @@ -0,0 +1,53 @@
1  + export const LOGIN_RATE_LIMIT_MESSAGE =
2  +   'Too many login attempts from this IP, please try again later'
3  +
4  + export interface LoginRateLimiter {
5  +   check(ip: string): { allowed: boolean; retryAfterMs: number }
6  +   dispose(): void
7  + }
8  +
9  + interface Entry {
10 +   count: number
11 +   resetTime: number
12 + }
13 +
14 + export function createLoginRateLimiter(
15 +   windowMs: number,
16 +   max: number
17 + ): LoginRateLimiter {
18 +   const entries = new Map<string, Entry>()
19 +
20 +   const cleanup = setInterval(() => {
21 +     const now = Date.now()
22 +     for (const [ip, entry] of entries) {
23 +       if (now >= entry.resetTime) {
```

```
+        entries.delete(ip)
+      }
+    }
+  }, windowMs)
+  cleanup.unref()
+
+  return {
+    check(ip: string): { allowed: boolean; retryAfterMs: number } {
+      const now = Date.now()
+      let entry = entries.get(ip)
+
+      if (!entry || now >= entry.resetTime) {
+        entry = { count: 0, resetTime: now + windowMs }
+        entries.set(ip, entry)
+      }
+
+      entry.count++
+
+      if (entry.count > max) {
+        return { allowed: false, retryAfterMs: entry.resetTime - now }
+      }
+
+      return { allowed: true, retryAfterMs: 0 }
+    },
+    dispose(): void {
+      clearInterval(cleanup)
+      entries.clear()
+    }
+  }
+}
```

src/security.ts

```
@@ -32,6 +32,7 @@ import { generate } from 'selfsigned'
import { Mode } from 'stat-mode'
import { WithConfig } from './app'
import { createDebug } from './debug'
+ import { LoginRateLimiter } from './login-rate-limiter'
import dummysecurity from './dummysecurity'
import { ICallback } from './types'
const debug = createDebug('signalk-server:security')
```

```
@@ -241,6 +242,9 @@ export interface SecurityStrategy {
241 242     username: string,
242 243     password: string
243 244   ) => Promise<{ statusCode: number }>
    245 +
    246 +   /** Shared login rate limiter (optional - only available when token security is active) */
    247 +   loginRateLimiter?: LoginRateLimiter
244 248   }
245 249
246 250   export class InvalidTokenError extends Error {
```

src/tokensecurity.ts

```
@@ -28,12 +28,15 @@ import {
28 28     Path
29 29   } from '@signalk/server-api'
30 30   import ms, { StringValue } from 'ms'
31    - import rateLimit from 'express-rate-limit'
32 31   import bodyParser from 'body-parser'
33 32   import cookieParser from 'cookie-parser'
34 33   import { createHash, randomBytes } from 'crypto'
35 34
36 35   import { createDebug } from './debug'
    36 + import {
    37 +   createLoginRateLimiter,
    38 +   LOGIN_RATE_LIMIT_MESSAGE
    39 + } from './login-rate-limiter'
37 40   import {
38 41     InvalidTokenError,
39 42     SecurityConfig,
@@ -45,7 +48,6 @@ import {
45 48     LoginStatusResponse,
46 49     saveSecurityConfig,
47 50     RequestStatusData,
48    -   getRateLimitValidationOptions,
49 51     ACL,
50 52     SecurityStrategy,
51 53     isOIDCUserIdentifier
```

```
@@ -606,15 +608,18 @@ function tokenSecurityFactory(
606 608         }
607 609       }
608 610
609     -     const loginLimiter = rateLimit({
610     -       windowMs: loginWindowMs,
611     -       max: loginMax,
612     -       message: {
613     -         message:
614     -           'Too many login attempts from this IP, please try again after 10 minutes'
615     -       },
616     -       validate: getRateLimitValidationOptions(app)
617     -     })
    611 +     const loginRateLimiter = createLoginRateLimiter(loginWindowMs, loginMax)
    612 +     strategy.loginRateLimiter = loginRateLimiter
    613 +
    614 +     const loginLimiter = (req: Request, res: Response, next: NextFunction) => {
    615 +       const { allowed, retryAfterMs } = loginRateLimiter.check(req.ip ?? '')
    616 +       if (!allowed) {
    617 +         res.set('Retry-After', String(Math.ceil(retryAfterMs / 1000)))
    618 +         res.status(429).json({ message: LOGIN_RATE_LIMIT_MESSAGE })
    619 +         return
    620 +       }
    621 +       next()
    622 +     }
618 623
619 624       app.use(bodyParser.urlencoded({ extended: true }))
620 625
```

Comments 0

Please sign in to comment.