

Budibase / budibase Public

<> Code Issues 273 Pull requests 16 Discussions Actions Projects

Commit 21bc3f8

 calexiou committed on Nov 19, 2025





feat(worker): rate limiting and lockouts in auth flow

 master (#17428) ·  database-2.1.0 ... 3.23.25

1 parent [8b15ca5](#) commit 21bc3f8 

 2 files changed +134 -2 lines changed

[↑ Top](#) 

- ✓  packages/worker/src
 - ✓  api/controllers/global
 -  auth.ts
 -  environment.ts

 2 files changed +134 -2 lines changed

...s/worker/src/api/controllers/global/auth.ts

```

@@ -5,6 +5,7 @@ import {
5 5     events,
6 6     utils as utilsCore,
7 7     configs,
8 +   cache,
8 9   } from "@budibase/backend-core"
9 10  import {
10 11   ConfigType,
@@ -37,6 +38,33 @@ const { setCookie, getCookie, clearCookie } = utilsCore
37 38

```

```

38 39 // LOGIN / LOGOUT
39 40
41 + const normalizeEmail = (e: string) => (e || "").toLowerCase()
42 + const failKey = (email: string) => `auth:login:fail:${normalizeEmail(email)}`
43 + const lockKey = (email: string) => `auth:login:lock:${normalizeEmail(email)}`
44 + const isLocked = async (email: string) => !(await cache.get(lockKey(email)))
45 + const onFailed = async (email: string) => {
46 +   if (!email) return
47 +   const key = failKey(email)
48 +   const current = Number((await cache.get(key)) || 0) || 0
49 +   const next = current + 1
50 +   await cache.store(key, next, env.LOGIN_LOCKOUT_SECONDS)
51 +   console.log(
52 +     `[auth] failed login email=${normalizeEmail(email)} count=${next}`
53 +   )
54 +   if (next >= env.LOGIN_MAX_FAILED_ATTEMPTS) {
55 +     await cache.store(lockKey(email), 1, env.LOGIN_LOCKOUT_SECONDS)
56 +     await cache.destroy(key)
57 +     console.log(
58 +       `[auth] account locked email=${normalizeEmail(email)} for
59 +         ${env.LOGIN_LOCKOUT_SECONDS}s`
60 +     )
61 +   }
62 +   const clearFailureState = async (email: string) => {
63 +     if (!email) return
64 +     await cache.destroy(failKey(email))
65 +     await cache.destroy(lockKey(email))
66 +   }
67 +
40 68 async function passportCallback(
41 69   ctx: Ctx,
42 70   user: User,
@@ -75,14 +103,52 @@ export const login = async (
75 103 ) => {
76 104   const email = ctx.request.body.username
77 105
78 -   const user = await userSdk.db.getUserByEmail(email)
79 -   if (user && (await userSdk.db.isPreventPasswordActions(user))) {

```

```
106 +   const dbUser = await userSdk.db.getUserByEmail(email)
107 +   if (dbUser && (await userSdk.db.isPreventPasswordActions(dbUser))) {
108 +     console.log(
109 +       `[auth] login prevented due to sso enforcement
110 +       email=${normalizeEmail(email)}`
111 +     )
80 111     ctx.throw(403, "Invalid credentials")
81 112   }
113 +   if (dbUser && (await isLocked(email))) {
114 +     console.log(
115 +       `[auth] login blocked due to lock email=${normalizeEmail(email)}`
116 +     )
117 +     ctx.set("X-Account-Locked", "1")
118 +     ctx.set("Retry-After", String(env.LOGIN_LOCKOUT_SECONDS))
119 +     ctx.throw(403, "Account temporarily locked. Try again later.")
120 +   }
82 121
83 122   return passport.authenticate(
84 123     "local",
85 124     async (err: any, user: User, info: any) => {
125 +     if (err || !user) {
126 +       if (dbUser) {
127 +         await onFailed(email)
128 +       }
129 +       if (await isLocked(email)) {
130 +         ctx.set("X-Account-Locked", "1")
131 +         ctx.set("Retry-After", String(env.LOGIN_LOCKOUT_SECONDS))
132 +         console.log(
133 +           `[auth] login blocked (post-failure) due to lock
134 +           email=${normalizeEmail(
135 +             email
136 +           )}`
137 +         )
138 +         return ctx.throw(403, "Account temporarily locked. Try again later.")
139 +       }
140 +       const reason =
141 +         (info && info.message) || (err && err.message) || "unknown"
142 +       console.log(
143 +         `[auth] password auth failed email=${normalizeEmail(email)}
144 +         reason=${reason}`
```

```

143 +     )
144 +     // delegate to shared passport failure handling to preserve specific
    messages (e.g. expired)
145 +     return passportCallback(ctx, user as any, err, info)
146 +   }
147 +
148 +   await clearFailureState(email)
149 +   console.log(
150 +     `[auth] password auth success email=${normalizeEmail(user.email)}`
151 +   )

```

```

86 152     await passportCallback(ctx, user, err, info)
87 153     await context.identity.doInUserContext(user, ctx, async () => {
88 154         await events.auth.login("local", user.email)

```



```
@@ -133,6 +199,57 @@ export const reset = async (
```

```

133 199   ) => {
134 200     const { email } = ctx.request.body
135 201

```

```

202 +   const lcEmail = (email || "").toLowerCase()
203 +   const ip = (ctx.ip || "").toString()
204 +
205 +   // rate limit keys
206 +   const emailKey = `auth:pwdreset:email:${lcEmail}`
207 +   const ipKey = `auth:pwdreset:ip:${ip}`
208 +
209 +   const incr = async (key: string, windowSeconds: number) => {
210 +     const current = Number((await cache.get(key)) || 0) || 0
211 +     const next = current + 1
212 +     await cache.store(key, next, windowSeconds)
213 +     return next
214 +   }
215 +
216 +   // apply per-email and per-ip rate limits
217 +   const nextEmail = await incr(
218 +     emailKey,
219 +     env.PASSWORD_RESET_RATE_EMAIL_WINDOW_SECONDS
220 +   )
221 +   const nextIp = await incr(ipKey, env.PASSWORD_RESET_RATE_IP_WINDOW_SECONDS)
222 +
223 +   const emailLimited = nextEmail > env.PASSWORD_RESET_RATE_EMAIL_LIMIT

```

```
224 +   const ipLimited = nextIp > env.PASSWORD_RESET_RATE_IP_LIMIT
225 +
226 +   if (emailLimited || ipLimited) {
227 +     // surfaced for ui to display
228 +     ctx.set(
229 +       "X-RateLimit-Email-Limit",
230 +       String(env.PASSWORD_RESET_RATE_EMAIL_LIMIT)
231 +     )
232 +     ctx.set(
233 +       "X-RateLimit-Email-Remaining",
234 +       String(Math.max(env.PASSWORD_RESET_RATE_EMAIL_LIMIT - nextEmail, 0))
235 +     )
236 +     ctx.set("X-RateLimit-IP-Limit", String(env.PASSWORD_RESET_RATE_IP_LIMIT))
237 +     ctx.set(
238 +       "X-RateLimit-IP-Remaining",
239 +       String(Math.max(env.PASSWORD_RESET_RATE_IP_LIMIT - nextIp, 0))
240 +     )
241 +     // best-effort retry window
242 +     const retryAfter = Math.max(
243 +       env.PASSWORD_RESET_RATE_EMAIL_WINDOW_SECONDS,
244 +       env.PASSWORD_RESET_RATE_IP_WINDOW_SECONDS
245 +     )
246 +     ctx.set("Retry-After", String(retryAfter))
247 +     console.log(
248 +       `[auth] password reset rate limited email=${lcEmail} ip=${ip}
249 +       emailCount=${nextEmail} ipCount=${nextIp}`
250 +     )
251 +     return ctx.throw(429, "Too many password reset requests. Try again later.")
252 +   }
```

```
136 253   await authSdk.reset(email)
```

```
137 254
```

```
138 255   ctx.body = {
```



packages/worker/src/environment.ts



```
@@ -87,6 +87,21 @@ const environment = {
```

```
87 87   PASSPORT_OIDCAUTH_FAILURE_REDIRECT:
```

```
88 88     process.env.PASSPORT_OIDCAUTH_FAILURE_REDIRECT || "/error",
```

```
89 89
```

```
90 + LOGIN_MAX_FAILED_ATTEMPTS:
91 +   parseIntSafe(process.env.LOGIN_MAX_FAILED_ATTEMPTS) || 5,
92 + LOGIN_LOCKOUT_SECONDS:
93 +   parseIntSafe(process.env.LOGIN_LOCKOUT_SECONDS) || 900,
94 +
95 + // password reset rate limiting
96 + PASSWORD_RESET_RATE_EMAIL_LIMIT:
97 +   parseIntSafe(process.env.PASSWORD_RESET_RATE_EMAIL_LIMIT) || 3,
98 + PASSWORD_RESET_RATE_EMAIL_WINDOW_SECONDS:
99 +   parseIntSafe(process.env.PASSWORD_RESET_RATE_EMAIL_WINDOW_SECONDS) || 900,
100 + PASSWORD_RESET_RATE_IP_LIMIT:
101 +   parseIntSafe(process.env.PASSWORD_RESET_RATE_IP_LIMIT) || 20,
102 + PASSWORD_RESET_RATE_IP_WINDOW_SECONDS:
103 +   parseIntSafe(process.env.PASSWORD_RESET_RATE_IP_WINDOW_SECONDS) || 900,
104 +
90 105 // Budibase AI
91 106 BUDIBASE_AI_API_KEY: process.env.BUDIBASE_AI_API_KEY,
92 107 BUDIBASE_AI_DEFAULT_MODEL: process.env.BUDIBASE_AI_DEFAULT_MODEL,
```



Comments 0



Please [sign in](#) to comment.