Expired API Keys Are Not Rejected

Moderate **CorentinTh** published **GHSA-866c-mc22-wvv5** 5 days ago

Package

ghcr.io/papra-hq/papra (Docker (ghcr.io))

Affected versions	Patched versions
<= 26.3.0	26.4.0

Description

> Maintainer comment (**@CorentinTh**):
>
> - The only way to set an expiration date to set an expiration date of an api is by directly calling the API with sessions cookies.
> - Normal creation of API keys through the UI are not affected by this issue, as the expiresAt field is not exposed in the UI and is not set at all.

Summary

API keys with an `expiresAt` date are never validated against the current time during authentication. Any API key — regardless of its expiration date — is accepted indefinitely, allowing a user whose key has expired to continue accessing all protected endpoints as if the key were still valid.

Details

The server stores an optional `expires_at` column in the `api_keys` table, it's possible to set an expiresAt value by tinkering with the api. However, the expiry is never enforced at authentication time.

Root cause — `api-keys.repository.ts` (`getApiKeyByHash`):
The database query that looks up a key by its hash applies no time filter:

```
// apps/papra-server/src/modules/api-keys/api-keys.repository.ts
const [apiKey] = await db
  .select()
  .from(apiKeysTable)
  .where(
```

```
    eq(apiKeysTable.keyHash, keyHash), // NO expiresAt check
  );
```

Root cause — `api-keys.middlewares.ts`:
The middleware unconditionally trusts whatever the repository returns:

```
const { apiKey } = await getApiKey({ token, apiKeyRepository });

if (apiKey) {
  // expiresAt is NEVER inspected here either
  context.set('apiKey', apiKey);
  context.set('userId', apiKey.userId);
  context.set('authType', 'api-key');
}
```

Notably, the same codebase correctly enforces expiry for organization invitations (`organizations.repository.ts` uses `gte(organizationInvitationsTable.expiresAt, now)`), confirming this is an oversight rather than an intentional design choice.

Suggested fix — add expiry filter in `getApiKeyByHash`:

```
import { and, eq, gt, isNull, or } from 'drizzle-orm';

const now = new Date();
const [apiKey] = await db
  .select()
  .from(apiKeysTable)
  .where(
    and(
      eq(apiKeysTable.keyHash, keyHash),
      or(
        isNull(apiKeysTable.expiresAt),      // key has no expiry set
        gt(apiKeysTable.expiresAt, now),     // or has not yet expired
      ),
    ),
  );
```

PoC

Environment: Papra `26.3.0` (`ghcr.io/papra-hq/papra:latest`), self-hosted, default configuration.

Step 1 — Register, sign in, and create an API key:

```
# Register
curl -s -c cookies.txt -X POST http://localhost:1221/api/auth/sign-up/email \
  -H "Content-Type: application/json" \
  -d '{"email": "user@example.com", "password": "Password123!", "name": "Test User"}'

# Sign in (saves session cookie)
```

```
curl -s -c cookies.txt -X POST http://localhost:1221/api/auth/sign-in/email \
  -H "Content-Type: application/json" \
  -d '{"email": "user@example.com", "password": "Password123!"}'

# Create API key — save the returned token as API_KEY
curl -s -b cookies.txt -X POST http://localhost:1221/api/api-keys \
  -H "Content-Type: application/json" \
  -d '{
    "name": "test-key",
    "permissions": ["documents:read"],
    "allOrganizations": true
  }'
```

Step 2 — Confirm the key works (baseline):

```
curl -s -w "\nHTTP STATUS: %{http_code}\n" \
  http://localhost:1221/api/api-keys/current \
  -H "Authorization: Bearer $API_KEY"
# → HTTP STATUS: 200
```

Step 3 — Force-expire the key in the database:

```
# Copy DB out of container (sqlite3 not available inside)
docker cp papra:/app/app-data/db/db.sqlite ./papra-live.db

# Set expiry to 2020-01-01T00:00:00Z (timestamp_ms)
sqlite3 ./papra-live.db \
  "UPDATE api_keys SET expires_at = 1577836800000 WHERE name = 'test-key';"

# Confirm
sqlite3 ./papra-live.db \
  "SELECT id, name, datetime(expires_at/1000, 'unixepoch') as expired_at FROM api_keys;"
# → ak_b1c2x9fd2tom79moidoao6rf|test-key|2020-01-01 00:00:00

# Copy back and restart
docker cp ./papra-live.db papra:/app/app-data/db/db.sqlite
docker restart papra && sleep 8
```

Step 4 — Use the expired key:

```
curl -s -w "\nHTTP STATUS: %{http_code}\n" \
  http://localhost:1221/api/api-keys/current \
  -H "Authorization: Bearer $API_KEY"
```

Expected: `401 Unauthorized`

Actual:

```
{"apiKey":{"id":"ak_b1c2x9fd2tom79moidoao6rf","name":"test-key","permissions":["do t
HTTP STATUS: 200
```


Impact

Any user or integration that was granted a time-limited API key retains full access to all endpoints that key was scoped to, indefinitely after expiration. This affects:

- **Contractors / third-party integrations** granted temporary access — their keys never actually expire
- **Compromised keys** that an admin attempted to "expire" as a mitigation — expiry provides no protection
- **Compliance requirements** — any policy requiring time-bounded credential rotation is silently violated

The vulnerability requires the attacker to have legitimately obtained a valid API key at some point (PR:L), but no further interaction or privilege is needed to exploit it after that.

Severity

Moderate 4.3 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	Low
Privileges required	Low
User interaction	None
Scope	Unchanged
Confidentiality	Low
Integrity	None
Availability	None

Learn more about base metrics

CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N

CVE ID

CVE-2026-35462

Weaknesses

▶ CWE-613

Credits

 **Toothless5143** Reporter