Fix username case-collision vulnerability across write and read paths #1098

Merged f merged 3 commits into `f:main` from `mdisec:case-insensitive-exploit-f…`

3 weeks ago

mdisec commented 3 weeks ago • edited by coderabbitai bot

Contributor

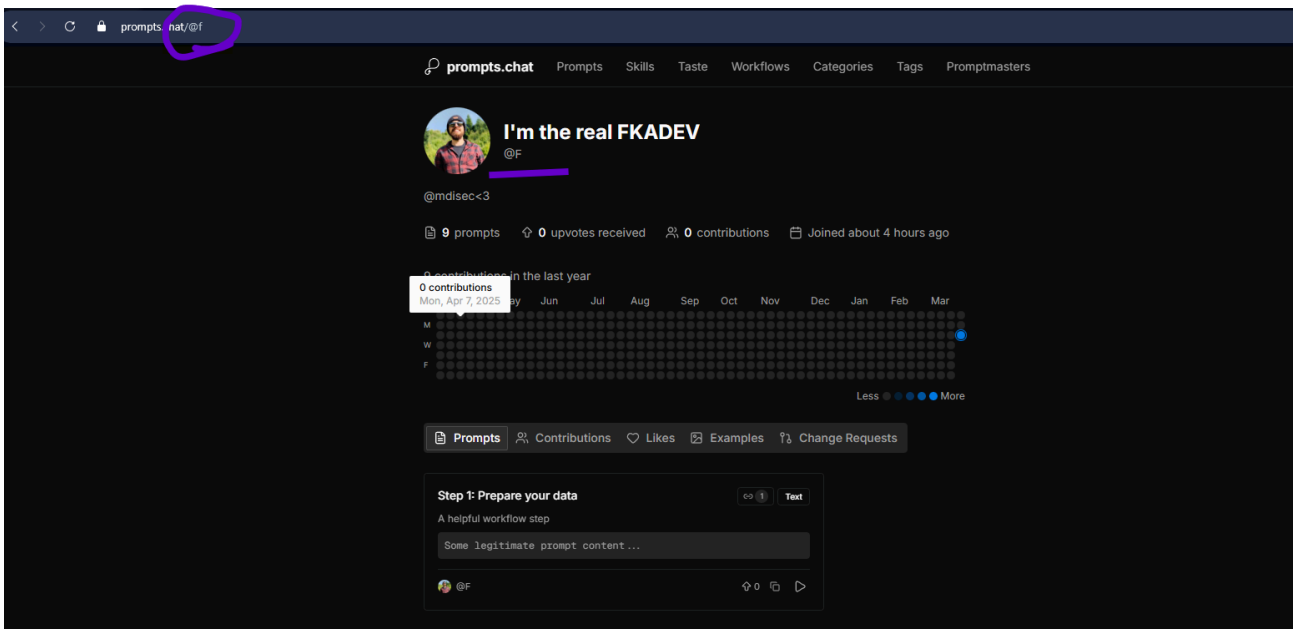
Username Case-Collision Vulnerability Report

Severity: High
Type: Business Logic / Identity Confusion
Date: 2026-03-24
Status: Open

Executive Summary

A case-sensitivity mismatch between username uniqueness checks in the registration and profile update endpoints allows an attacker to create a case-variant of any existing username (e.g., `Admin` vs `admin`). PostgreSQL's default collation treats these as distinct values, so both coexist in the database. Because the platform resolves usernames case-insensitively across 22+ display surfaces using `findFirst` without deterministic ordering, the attacker's profile can intermittently replace the victim's on their own canonical URL and across all UI surfaces.

For instance: Visit the https://prompts.chat/@f which is not pointing to the original account.



Root Cause

Registration: Case-Insensitive (Correct)

File: `src/app/api/auth/register/route.ts:50-51`

```
const existingUsername = await db.user.findFirst({
  where: { username: { equals: username, mode: "insensitive" } },
});
```

Registration correctly blocks case-variants. Attempting to register `Admin` when `admin` exists returns `username_taken`.

Profile Update: Case-Sensitive (Vulnerable)

File: `src/app/api/user/profile/route.ts:48-59`

```
if (username !== session.user.username) {
  const existingUser = await db.user.findUnique({
    where: { username },          // case-sensitive by default
    select: { id: true },
  });

  if (existingUser && existingUser.id !== session.user.id) {
    return NextResponse.json(
      { error: "username_taken", message: "This username is already taken" },
      { status: 400 }
    );
```

```
  }
}
```

`findUnique` uses PostgreSQL's default case-sensitive collation. Changing your username to `Admin` when only `admin` exists returns no match, and the update succeeds.

OAuth Registration: Case-Sensitive (Vulnerable)

File: `src/lib/auth/index.ts:84`

```
while (await db.user.findUnique({ where: { username: finalUsername } })) {
  finalUsername = `${baseUsername}${counter}`;
  counter++;
}
```

The OAuth adapter lowercases the GitHub login at line 53 but checks uniqueness with case-sensitive `findUnique` at line 84. A GitHub user `johndoe` will pass the check even if `JohnDoe` already exists in the database.

CSV Import: Case-Sensitive (Vulnerable)

File: `src/app/api/admin/import-prompts/route.ts:150-153`

```
let user = await db.user.findFirst({
  where: {
    OR: [
      { username: normalizedUsername },   // case-sensitive, no mode specified
      { email: pseudoEmail },
    ],
  },
});
```

The import function normalizes to lowercase but the query is case-sensitive. If `JohnDoe` exists in the DB, a search for `johndoe` will miss it and create a duplicate unclaimed account.

Database Constraint

File: `prisma/schema.prisma:14`

```
username    String    @unique
```

PostgreSQL's `UNIQUE` constraint uses the column's collation, which is case-sensitive by default. The constraint permits both `admin` and `Admin` as separate rows.

Exploitation

Prerequisites

- Attacker has a registered account on the platform
- Target user exists with a known username

Steps to Reproduce

```
1. Attacker registers with any username (e.g., "attacker123")

2. Attacker sends:
   PATCH /api/user/profile
   Content-Type: application/json
   Cookie: <session>

   { "name": "Real Admin", "username": "Admin" }

3. Server checks: findUnique({ where: { username: "Admin" } })
   → Returns null (only "admin" exists, case-sensitive match)
   → Check passes

4. Database now contains:
   | id   | username | role  |
   |------|----------|-------|
   | AAA  | admin    | ADMIN |
   | BBB  | Admin    | USER  |

5. Visiting /@admin triggers:
   findFirst({ where: { username: { equals: "admin", mode: "insensitive" } } })
   → Matches BOTH rows
   → Returns whichever PostgreSQL finds first in heap (non-deterministic)
```

Abuse Cases

1. Profile Page Hijacking (High)

File: `src/app/[username]/page.tsx:73`

The profile page resolves usernames with `findFirst` + `mode: "insensitive"` and no `orderBy`. When two users share a case-insensitive username, PostgreSQL returns whichever row it encounters first during a sequential scan. After `VACUUM` or `autovacuum` reorganizes heap pages, the ordering can silently flip.

The attacker's bio, avatar, prompts, and custom links replace the victim's on their canonical URL.

2. OpenGraph Image Spoofing (High)

File: `src/app/[username]/opengraph-image.tsx:51`

Same `findFirst` pattern. When the victim's profile is shared on Slack, Twitter, or Discord, the preview card (avatar, name, prompt count) may render the attacker's data. Social platforms cache these cards aggressively, so the spoofed preview persists even after the vulnerability is patched.

3. Profile Link Phishing (High)

File: `src/app/[username]/page.tsx:85` (customLinks display)

The attacker populates their profile with up to 5 custom links (website, GitHub, Twitter, sponsor, etc.) pointing to phishing or payment pages. Users visiting `/@admin` see attacker-controlled links under a URL they trust. The "sponsor" link type is particularly dangerous as it implies a financial action.

4. Unclaimed Account Attribution Theft (High)

Files: `src/app/api/admin/import-prompts/route.ts:150` + `src/app/[username]/page.tsx:73`

When prompts are bulk-imported from CSV, unclaimed accounts are created for community contributors (e.g., `johndoe` with email `johndoe@unclaimed.prompts.chat`). If an attacker takes `JohnDoe` via profile update, the profile page's case-insensitive `findFirst` may resolve to the attacker. The attacker inherits the public visibility and prompt count of the real contributor.

5. SEO / Metadata Poisoning (Medium)

File: `src/app/[username]/page.tsx:39` (`generateMetadata`)

```
title: `${user.name || user.username} (@${user.username})`
```

The `generateMetadata` function uses the same `findFirst` pattern. Search engines index the attacker's name in the `<title>` tag for the victim's URL. The victim's search presence is overwritten.

6. Webhook Impersonation (Medium)

File: `src/lib/webhook.ts:237`

```
[WEBHOOK_PLACEHOLDERS.AUTHOR_USERNAME]: prompt.author.username
```

When the attacker creates a prompt, configured webhooks fire with `{{AUTHOR_USERNAME}}` set to `Admin`. The Slack Block Kit preset renders this as **"Created by @Admin"** with a link to `/@Admin`. Team members in private channels trust the content because the author appears to be the admin.

7. Notification Actor Spoofing (Medium)

File: `src/components/layout/notification-bell.tsx:116`

```
<span className="font-medium">@{notification.actor?.username}</span>
```

The attacker comments on the victim's prompt. The victim's notification bell shows **"@Admin commented on your prompt"**. The victim trusts the comment content, which could contain malicious instructions or phishing links.

8. Contributor Search Confusion (Medium)

File: `src/app/api/users/search/route.ts:26`

```
{ username: { contains: query, mode: "insensitive" } }
```

When a prompt author searches for a contributor to credit, both `admin` and `Admin` appear. Selecting the wrong entry grants the attacker visible contributor attribution on the victim's prompt (avatar on prompt cards, name on detail pages).

9. OAuth Registration Collision (Medium)

File: `src/lib/auth/index.ts:53,84`

No profile update exploit needed. When a new GitHub user whose login is a case-variant of an existing username signs up via OAuth:

1. Adapter lowercases: `username = profile.login.toLowerCase()`
2. `findUnique({ where: { username } })` is case-sensitive and misses the existing variant
3. A second user with a visually identical username is created automatically

This widens the attack surface: the collision can be created passively by any GitHub signup.

10. Admin Panel Wrong-Target Moderation (Medium)

File: `src/app/api/admin/users/route.ts:75`

Admin user search is case-insensitive, so both `admin` and `Admin` appear for the same query. An administrator may flag, ban, or modify the wrong account. The attacker could deliberately provoke moderation to get the real user actioned instead.

11. Leaderboard Reputation Splitting (Low)

File: `src/app/api/leaderboard/route.ts:58-88`

Both users appear as separate leaderboard entries with visually identical usernames. While votes are not stolen (they are ID-based), the attacker's entry links to `/@Admin` which resolves ambiguously. Users confused about which entry is real may interact with the attacker's prompts.

12. Content Similarity Author Enumeration (Low)

File: `src/app/api/prompts/route.ts:160`

```
existingPromptAuthor: similarPrompt.author.username
```

The similarity check returns the existing author's username in the 409 response. An attacker can submit content similar to known prompts to discover their authors, then target those usernames for case-variant takeover.

Affected Locations

Vulnerable Write Paths (source of collision)

File	Line	Query	Issue
`src/app/api/user/profile/route.ts`	49	`findUnique({ where: { username } })`	Case-sensitive uniqueness check
`src/lib/auth/index.ts`	29	`findUnique({ where: { username } })`	Case-sensitive in username generator
`src/lib/auth/index.ts`	84	`findUnique({ where: { username: finalUsername } })`	Case-sensitive in OAuth adapter
`src/app/api/admin/import-prompts/route.ts`	153	`findFirst({ where: { username: normalizedUsername } })`	Case-sensitive contributor lookup

Affected Read Paths (exploitable after collision)

File	Line	Surface
`src/app/[username]/page.tsx`	39, 73	Profile page + metadata
`src/app/[username]/opengraph-image.tsx`	51	Social sharing preview
`src/components/prompts/prompt-card.tsx`	338-367	Prompt cards (author + contributors)
`src/app/prompts/[id]/page.tsx`	291, 376-420	Prompt detail (structured data, author links)
`src/components/comments/comment-item.tsx`	228-236	Comment author display
`src/components/layout/notification-bell.tsx`	101, 116, 133	Notification actor
`src/components/layout/header.tsx`	586-601	Profile menu link
`src/components/prompts/related-prompts.tsx`	74-78	Related prompt author
`src/components/promptmasters/promptmasters-content.tsx`	112-128	Leaderboard entries
`src/components/prompts/contributor-search.tsx`	104, 165	Contributor picker
`src/components/admin/users-table.tsx`	293-391	Admin user management
`src/lib/webhook.ts`	237	Webhook payloads
`src/app/api/leaderboard/route.ts`	83	Public leaderboard API
`src/app/api/prompts/route.ts`	160	Similarity check response
`src/app/api/users/search/route.ts`	26	User search API

D. Data Cleanup

Before deploying the fix, check for existing collisions:

```
SELECT LOWER(username) AS normalized, COUNT(*), ARRAY_AGG(username || ' (id=' || i
FROM "User"
GROUP BY LOWER(username)
HAVING COUNT(*) > 1;
```

Resolve any existing duplicates before adding the database constraint from section C.

Summary by CodeRabbit

- **Bug Fixes**
  - Trim whitespace and normalize emails/usernames to lowercase on register and profile updates.
  - Duplicate username/email errors now return HTTP 409 with clear error codes.
  - Username validation tightened: lowercase letters, digits, underscores only (max 30).
- **Behavior Changes**
  - When multiple accounts match a username, profile and social preview now deterministically pick the earliest-created account.
- **Database**
  - Case-insensitive unique indexes added for username and email.

❤️ 1

 `Fix username case-collision vulnerability across write and read paths` ✕ `1464475`

coderabbitai bot commented 3 weeks ago • edited

No actionable comments were generated in the recent review. 🎉

▶ ℹ️ Recent review info

▶ 📝 Walkthrough

▶ 📟 Pre-merge checks | ✅ 2 | ❌ 1

▶ ✨ Finishing Touches

Thanks for using CodeRabbit! It's free for OSS, and your support helps us grow. If you like it, consider giving us a shout-out.

▶ ❤️ Share

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

coderabbitai bot reviewed 3 weeks ago

View reviewed changes

coderabbitai bot left a comment

Actionable comments posted: 1

> Caution
>
> Some comments are outside the diff and can't be posted inline due to platform limitations.
>
> ▶ ⚠️ Outside diff range comments (1)

▶ 🧹 Nitpick comments (1)

▶ 🤖 Prompt for all review comments with AI agents

▶ ℹ️ Review info

`src/lib/auth/index.ts` Outdated

Comment on lines 29 to 30

```
     29 | +   while (await db.user.findFirst({ where: { username: { equals: username
30   30 |       username = `${baseUsername}${counter}`;
```

coderabbitai bot 3 weeks ago

⚠️ *Potential issue* | 🔴 *Critical*

▶ 🧩 Analysis chain

App-side case-insensitive checks lack both DB-level constraint and transaction protection, leaving TOCTOU races on multiple write paths.

The code checks `username` case-insensitively at query time but the database enforces only case-sensitive uniqueness (`@unique` in schema, `CREATE UNIQUE INDEX ... ("username")`). Two concurrent requests can both find no match and commit `admin` and `Admin`. This affects three paths:

1. `generateUsername()` (lines 26–32): check-then-return username for creation
2. `createUser()` (lines 84–87, 91–100): check-then-create without transaction
3. Profile route (lines 5–16, 19–27): check-then-update without transaction

Fix requires either:

- Add DB-level case-insensitive constraint and wrap all check-then-write logic in `db.$transaction()`, or
- Use upsert or database-enforced uniqueness on `LOWER(username)` with proper violation handling

Per coding guidelines: *"Use transactions for multi-step database operations in Prisma."*

▶ 🤖 Prompt for AI Agents

 `drop case sensitive approach to the username and email across the app` ✕ `30784e0`

mdisec commented 3 weeks ago — Contributor — Author

I personally believe usernames and email must be stored lower-case before doing anything business related decision due to many different reason:

1. TOCTOU races are unfixable at the app layer — mode: "insensitive" checks let two concurrent requests both see "not taken" and commit admin and Admin.
2. DB-level LOWER() unique indexes are the only reliable safety net, and they work best when all stored values are already normalized.
3. Unicode/whitespace tricks (e.g. admin\u200B@x.com vs admin@x.com) can bypass uniqueness checks if
   inputs aren't trimmed and lowercased before validation for (MySQL with different charset configs)
4. Simpler queries everywhere — once storage is lowercase, every read path can use exact matches instead of case-insensitive lookups, which are faster and index-friendly.

This is the standard approach used by GitHub, GitLab, and most identity systems.

coderabbitai bot reviewed 3 weeks ago

View reviewed changes

 **coderabbitai** bot left a comment

Actionable comments posted: 1

▶ ♻️ Duplicate comments (1)

▶ 🧹 Nitpick comments (1)

▶ 🤖 Prompt for all review comments with AI agents

▶ ℹ️ Review info

src/lib/auth/index.ts

Comment on lines +43 to 67

```
+       // If a username was provided, try to claim an unclaimed account t
+       if (providedUsername) {
+         const username = providedUsername;
56      const unclaimedEmail = `${username}@unclaimed.prompts.chat`;
57      const unclaimedUser = await db.user.findUnique({
58        where: { email: unclaimedEmail },
59      });
-
+
61      if (unclaimedUser) {
-         // Claim this account - update with real user info
63        const claimedUser = await db.user.update({
64          where: { id: unclaimedUser.id },
65          data: {
66            name: data.name,
-           email: data.email,
+           email: normalizedEmail,
68          avatar: data.image,
69          emailVerified: data.emailVerified,
-         githubUsername: githubUsername || undefined, // Store immu
+         githubUsername: githubUsername || undefined,
71        },
72      });
-
+
74      return {
75        ...claimedUser,
76        image: claimedUser.avatar,
77      } as AdapterUser;
78    }
```

coderabbitai bot 3 weeks ago

⚠️ *Potential issue* | 🔴 *Critical*

▶ 🧩 Analysis chain

Claiming the placeholder account is racy and violates transaction guideline.

This read-then-write flow has a TOCTOU window: `findUnique` (line 47–49) followed by `update` (line 52–61). Two concurrent requests can both find the same `unclaimedUser`, and the second update will overwrite the first claim because the write filters only on `id`, not on the original unclaimed email.

Use a conditional update that filters on both `id` and `email`, then verify the update succeeded (exactly one row claimed). Alternatively, wrap both operations in `db.$transaction()` for atomicity.

This violates the coding guideline: "Use transactions for multi-step database operations in Prisma."

▶ 🤖 Prompt for AI Agents

`adding migration files to lower index username and email` ✕ `cb0be62`

f merged commit **`61abe5d`** into `f:main` 3 weeks ago

2 of 3 checks passed

View details

Reviewers

coderabbitai[bot]

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Successfully merging this pull request may close these issues.

None yet

3 participants