f / **prompts.chat** Public

Code | Issues 9 | Pull requests 33 | Actions | Projects | Models | Se

Commit 7b81836

mdisec committed 2 weeks ago · ✕ 2 / 9

```
-fix(security): enforce isPrivate checks on prompt sub-resource endpoints
```

main (#1104)

1 parent 30a8f04 commit 7b81836

7 files changed +98 -12 lines changed

Top

Filter files…

- src
  - app
    - api/prompts/[id]
      - changes
        - [changeId]
          - route.ts
        - route.ts
      - examples
        - route.ts
      - versions
        - route.ts
      - vote
        - route.ts
    - prompts/[id]
      - page.tsx
  - lib

prompt-access.ts

7 files changed +98 -12 lines changed

Search within code

...pi/prompts/[id]/changes/[changeId]/route.ts

```
@@ -2,6 +2,7 @@ import { NextRequest, NextResponse } from "next/server";
2  2   import { z } from "zod";
3  3   import { auth } from "@/lib/auth";
4  4   import { db } from "@/lib/db";
   5 + import { checkPromptAccess } from "@/lib/prompt-access";
5  6
6  7   const updateChangeRequestSchema = z.object({
7  8     status: z.enum(["APPROVED", "REJECTED", "PENDING"]),
@@ -196,7 +197,8 @@ export async function GET(
196 197             select: {
197 198               id: true,
198 199               title: true,
199     -             content: true,
    200 +             isPrivate: true,
    201 +             authorId: true,
200 202             },
201 203           },
202 204         },
@@ -209,6 +211,9 @@ export async function GET(
209 211         );
210 212       }
211 213
    214 +     const denied = await checkPromptAccess(changeRequest.prompt);
    215 +     if (denied) return denied;
    216 +
212 217       return NextResponse.json(changeRequest);
213 218     } catch (error) {
214 219       console.error("Get change request error:", error);
```

src/app/api/prompts/[id]/changes/route.ts

```
@@ -2,6 +2,7 @@ import { NextRequest, NextResponse } from "next/server";
```

```
2  2    import { z } from "zod";
3  3    import { auth } from "@/lib/auth";
4  4    import { db } from "@/lib/db";
   5  + import { checkPromptAccess } from "@/lib/prompt-access";
5  6
6  7    const createChangeRequestSchema = z.object({
7  8      proposedContent: z.string().min(1),
       @@ -94,6 +95,14 @@ export async function GET(
94 95      try {
95 96        const { id: promptId } = await params;
96 97
   98 +     const prompt = await db.prompt.findUnique({
   99 +       where: { id: promptId },
  100 +       select: { isPrivate: true, authorId: true },
  101 +     });
  102 +
  103 +     const denied = await checkPromptAccess(prompt);
  104 +     if (denied) return denied;
  105 +
97 106       const changeRequests = await db.changeRequest.findMany({
98 107         where: { promptId },
99 108         orderBy: { createdAt: "desc" },
```

src/app/api/prompts/[id]/examples/route.ts

```
       @@ -2,6 +2,7 @@ import { NextRequest, NextResponse } from "next/server";
2  2    import { auth } from "@/lib/auth";
3  3    import { db } from "@/lib/db";
4  4    import { z } from "zod";
   5  + import { checkPromptAccess } from "@/lib/prompt-access";
5  6
6  7    const addExampleSchema = z.object({
7  8      mediaUrl: z.string().url(),
       @@ -16,12 +17,11 @@ export async function GET(
16 17
17 18      const prompt = await db.prompt.findUnique({
18 19        where: { id: promptId },
19    -     select: { id: true, type: true },
   20  +     select: { id: true, type: true, isPrivate: true, authorId: true },
```

```
20  21      });
21  22
22      -   if (!prompt) {
23      -     return NextResponse.json({ error: "Prompt not found" }, { status: 404 });
24      -   }
    23  +   const denied = await checkPromptAccess(prompt);
    24  +   if (denied || !prompt) return denied!;
25  25
26  26      // Only allow examples for IMAGE and VIDEO prompts
27  27      if (prompt.type !== "IMAGE" && prompt.type !== "VIDEO") {
```

src/app/api/prompts/[id]/versions/route.ts

```
    @@ -2,6 +2,7 @@ import { NextRequest, NextResponse } from "next/server";
2   2   import { z } from "zod";
3   3   import { auth } from "@/lib/auth";
4   4   import { db } from "@/lib/db";
    5   + import { checkPromptAccess } from "@/lib/prompt-access";
5   6
6   7   const createVersionSchema = z.object({
7   8     content: z.string().min(1, "Content is required"),
    @@ -116,6 +117,14 @@ export async function GET(
116 117     try {
117 118       const { id: promptId } = await params;
118 119
    120 +     const prompt = await db.prompt.findUnique({
    121 +       where: { id: promptId },
    122 +       select: { isPrivate: true, authorId: true },
    123 +     });
    124 +
    125 +     const denied = await checkPromptAccess(prompt);
    126 +     if (denied) return denied;
    127 +
119 128       const versions = await db.promptVersion.findMany({
120 129         where: { promptId },
121 130         orderBy: { version: "desc" },
```

src/app/api/prompts/[id]/vote/route.ts

```
@@ -1,6 +1,7 @@
import { NextRequest, NextResponse } from "next/server";
import { auth } from "@/lib/auth";
import { db } from "@/lib/db";
+ import { checkPromptAccess } from "@/lib/prompt-access";

// POST - Upvote a prompt
export async function POST(
@@ -21,14 +22,11 @@ export async function POST(
    // Check if prompt exists
    const prompt = await db.prompt.findUnique({
      where: { id: promptId },
+       select: { isPrivate: true, authorId: true },
    });

-     if (!prompt) {
-       return NextResponse.json(
-         { error: "not_found", message: "Prompt not found" },
-         { status: 404 }
-       );
-     }
+     const denied = await checkPromptAccess(prompt);
+     if (denied) return denied;

    // Check if already voted
    const existing = await db.promptVote.findUnique({
@@ -86,6 +84,14 @@ export async function DELETE(

    const { id: promptId } = await params;

+     const prompt = await db.prompt.findUnique({
+       where: { id: promptId },
+       select: { isPrivate: true, authorId: true },
+     });
+
+     const denied = await checkPromptAccess(prompt);
+     if (denied) return denied;
```

```
   94 +
89 95        // Delete vote
90 96        await db.promptVote.deleteMany({
91 97          where: {
```

src/app/prompts/[id]/page.tsx

```
@@ -8,6 +8,7 @@ import { AnimatedDate } from "@/components/ui/animated-date";
8  8   import { ShareDropdown } from "@/components/prompts/share-dropdown";
9  9   import { auth } from "@/lib/auth";
10 10  import { db } from "@/lib/db";
   11 + import { canViewPrompt } from "@/lib/prompt-access";
11 12  import { Button } from "@/components/ui/button";
12 13  import { Badge } from "@/components/ui/badge";
13 14  import { Avatar, AvatarFallback, AvatarImage } from "@/components/ui/avatar";
@@ -62,13 +63,20 @@ export async function generateMetadata({ params }:
PromptPageProps): Promise<Met
62 63    const id = extractPromptId(idParam);
63 64    const prompt = await db.prompt.findUnique({
64 65      where: { id },
65    -   select: { title: true, description: true },
   66 +   select: { title: true, description: true, isPrivate: true, authorId: true },
66 67    });
67 68
68 69    if (!prompt) {
69 70      return { title: "Prompt Not Found" };
70 71    }
71 72
   73 +  if (prompt.isPrivate) {
   74 +    const session = await auth();
   75 +    if (!canViewPrompt(prompt, session)) {
   76 +      return { title: "Prompt Not Found" };
   77 +    }
   78 +  }
   79 +
72 80    return {
73 81      title: prompt.title,
74 82      description: prompt.description || `View the prompt: ${prompt.title}`,
```

src/lib/prompt-access.ts

```
@@ -0,0 +1,49 @@
+ import { NextResponse } from "next/server";
+ import { auth } from "@/lib/auth";
+ import type { Session } from "next-auth";
+
+ /**
+  * Check if a user can view a prompt that may be private.
+  * Returns true for public prompts, and for private prompts only if the user is the owner or an admin.
+  */
+ export function canViewPrompt(
+   prompt: { isPrivate: boolean; authorId: string } | null,
+   session: Session | null
+ ): boolean {
+   if (!prompt) return false;
+   if (!prompt.isPrivate) return true;
+   return prompt.authorId === session?.user?.id || session?.user?.role === "ADMIN";
+ }
+
+ /**
+  * API route guard for prompt privacy. Returns a 404 NextResponse if access is denied, or null if allowed.
+  * Returning 403 is also leak existence of the prompt itself !
+  * Calls auth() lazily — only when the prompt is private.
+  *
+  *
+  * Usage:
+  *   const denied = await checkPromptAccess(prompt);
+  *   if (denied) return denied;
+  */
+ export async function checkPromptAccess(
+   prompt: { isPrivate: boolean; authorId: string } | null
+ ): Promise<NextResponse | null> {
+   if (!prompt) {
+     return NextResponse.json(
+       { error: "not_found", message: "Prompt not found" },
+       { status: 404 }
```

```
35 +     );
36 +   }
37 +
38 +   if (!prompt.isPrivate) return null;
39 +
40 +   const session = await auth();
41 +   if (!canViewPrompt(prompt, session)) {
42 +     return NextResponse.json(
43 +       { error: "not_found", message: "Prompt not found" },
44 +       { status: 404 }
45 +     );
46 +   }
47 +
48 +   return null;
49 + }
```

Comments 0

Please sign in to comment.