

f / prompts.chat Public

<> Code Issues 10 Pull requests 35 Actions Projects Models S

Commit 7b81836



mdisec committed 3 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



```

...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 1  import { NextRequest, NextResponse } from "next/server";
2 2  import { auth } from "@/lib/auth";
3 3  import { db } from "@/lib/db";
4 4  + import { checkPromptAccess } from "@/lib/prompt-access";
4 5
5 6  // POST - Upvote a prompt
6 7  export async function POST(
@@ -21,14 +22,11 @@ export async function POST(
21 22  // Check if prompt exists
22 23  const prompt = await db.prompt.findUnique({
23 24  where: { id: promptId },
25 25  + select: { isPrivate: true, authorId: true },
24 26  });
25 27
26 26  - if (!prompt) {
27 27  -   return NextResponse.json(
28 28  -     { error: "not_found", message: "Prompt not found" },
29 29  -     { status: 404 }
30 30  -   );
31 31  -   }
28 28  + const denied = await checkPromptAccess(prompt);
29 29  + if (denied) return denied;
32 30
33 31  // Check if already voted
34 32  const existing = await db.promptVote.findUnique({
@@ -86,6 +84,14 @@ export async function DELETE(
86 84
87 85  const { id: promptId } = await params;
88 86
87 87  + const prompt = await db.prompt.findUnique({
88 88  +   where: { id: promptId },
89 89  +   select: { isPrivate: true, authorId: true },
90 90  +   });
91 91  +
92 92  + const denied = await checkPromptAccess(prompt);
93 93  + 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 @@
1 + import { NextResponse } from "next/server";
2 + import { auth } from "@lib/auth";
3 + import type { Session } from "next-auth";
4 +
5 + /**
6 +  * Check if a user can view a prompt that may be private.
7 +  * Returns true for public prompts, and for private prompts only if the user is
8 +  * the owner or an admin.
9 +  */
10 + export function canViewPrompt(
11 +   prompt: { isPrivate: boolean; authorId: string } | null,
12 +   session: Session | null
13 + ): boolean {
14 +   if (!prompt) return false;
15 +   if (!prompt.isPrivate) return true;
16 +   return prompt.authorId === session?.user?.id || session?.user?.role ===
17 +     "ADMIN";
18 + }
19 +
20 + /**
21 +  * API route guard for prompt privacy. Returns a 404 NextResponse if access is
22 +  * denied, or null if allowed.
23 +  * Returning 403 is also leak existence of the prompt itself !
24 +  * Calls auth() lazily – only when the prompt is private.
25 +  *
26 +  * Usage:
27 +  *   const denied = await checkPromptAccess(prompt);
28 +  *   if (denied) return denied;
29 +  */
30 + export async function checkPromptAccess(
31 +   prompt: { isPrivate: boolean; authorId: string } | null
32 + ): Promise<NextResponse | null> {
33 +   if (!prompt) {
34 +     return NextResponse.json(
35 +       { error: "not_found", message: "Prompt not found" },
36 +       { status: 404 }
37 +     );
38 +   }
39 +   if (!prompt.isPrivate) return null;
40 +   const user = await auth({ promptId: prompt.authorId });
41 +   if (!user) return NextResponse.json(
42 +     { error: "not_found", message: "Prompt not found" },
43 +     { status: 404 }
44 +   );
45 +   if (user.id === prompt.authorId || user.role === "ADMIN") return null;
46 +   return NextResponse.json(
47 +     { error: "not_found", message: "Prompt not found" },
48 +     { status: 404 }
49 +   );
50 + }
```

```
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.