cvd_rallly_20260311.md

Coordinated Disclosure Report: DOM-Based XSS in Rallly Reset Password Function

> Note
>
> This report was originally disclosed to rallly maintainers privately over GitHub security advisory. However, they have chosen not the publish the advisory for "low exploitability" reasons. This gist is intended to publicly disclose the silent patch.

Project: rallly (https://github.com/lukevella/rallly)
Vulnerability Type: DOM-Based Cross-Site Scripting (XSS)
Severity: Medium (CVSS 3.1 Base Score: 4.1)
Report Date: March 11, 2026

1. Summary

A DOM-based Cross-Site Scripting (XSS) vulnerability has been discovered in Rallly's reset password functionality. The application improperly trusts a URL parameter (`redirectTo`). An attacker can craft a malicious link that, when opened and interacted with by a user, performs a client-side redirect and executes arbitrary JavaScript in the context of their browser. This could lead to credential theft or internal network pivoting.

> Warning

> It should be emphasised that the main risk of this particular XSS isn't the usual case of stealing a victim's account or data. Rather, the focus is on the impact that an attacker may run arbitrary code on a victim's browser through social engineering. For example, an attacker may entice victims with: "Free Rallly PRO Giveaway!", leading victims eagerly rushing to claim the attacker's account, thus triggering the XSS. You may refer to §3 for the reduced CVSS scoring and §4 for some details on impact.

2. Vulnerability Details

2.1. Affected Component and Code

- **File:** `apps/web/src/app/[locale]/(auth)/reset-password/components/reset-password-form.tsx`
- **Specific Lines:**

```
<Form {...form}>
  <form
    onSubmit={handleSubmit(async ({ password }) => {
      const res = await authClient.resetPassword({
        newPassword: password,
        token,
      });

      if (res.error) {
        form.setError("root", {
          message: res.error.message,
        });
        return;
      }

      // Check if this came from password setup (redirectTo contains settings
      const redirectTo = searchParams?.get("redirectTo");
      if (redirectTo?.includes("/settings/security")) {
        // For password setup, redirect back to settings
        router.push(redirectTo); // <-- sink
      } else {
        // For password reset, redirect to login
        router.push("/login");
      }
```

As shown, the `redirectTo` parameter is taken from the browser's URL query and later passed to `router.push` without validating against the `javascript:` protocol, leading to XSS. This is because `router.push` passes its parameter to `window.location`, which is known to parse and execute JavaScript in `javascript:` URIs. (Note: the NextJS maintainers have declined applying a fix in `router.push` and `router.replace`, so the responsibility falls on library users. See this NextJS GitHub issue.)

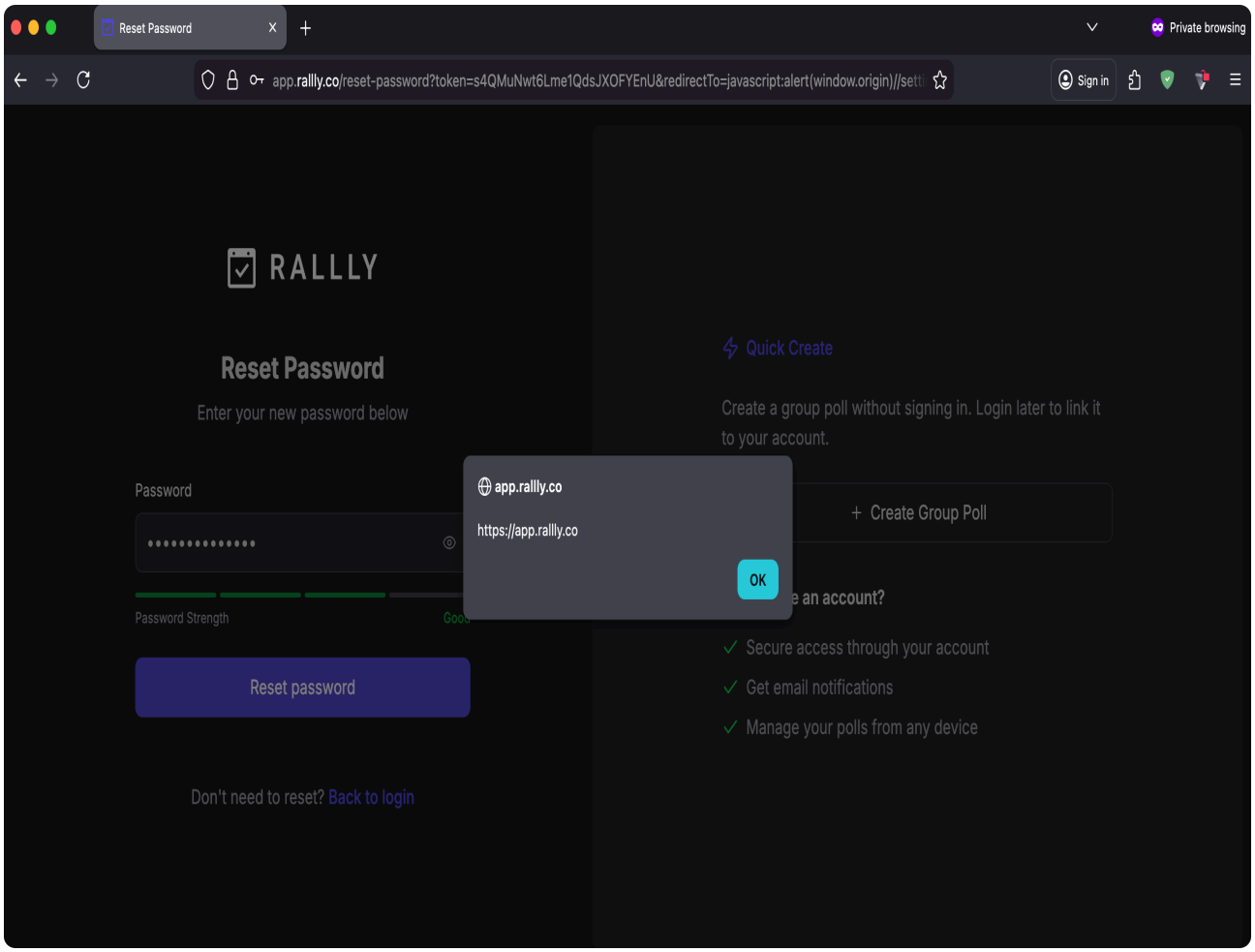
2.2. Steps to Reproduce and Proof of Concept (PoC)

0. Attacker has an account (e.g. by registering).
1. Attacker initiates password reset flow and obtains a reset token.
2. **Attacker crafts a malicious URL:**

   ```
   https://HOST/reset-password?
   token=RESET_TOKEN&redirectTo=javascript:alert(window.origin)//settings/se
   ```

   A sophisticated payload could hide the JavaScript by inserting extra query parameters or using URL encoding to obfuscate the attack, making the URL appear less suspicious to victims.
3. **Victim Action:** A victim clicks the link and enters a password to reset the account. The `router.push` call is triggered and the XSS payload is run.

2.3. Screenshots



3. CVSS v3.1 Score Justification

Base Score: 4.1 (Medium)
Vector: `CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:C/C:N/I:L/A:N`

- **Attack Vector (AV): Network (N)** – The vulnerability is exploitable remotely over the network via a crafted URL.
- **Attack Complexity (AC): Low (L)** – The attack does not require complex conditions.
- **Privileges Required (PR): Low (L)** – Attacker needs to have an account first, then trigger the password reset flow to obtain a reset token.
- **User Interaction (UI): Required (R)** – The victim must click on the attacker's malicious link and enter a password to reset the account.
- **Scope (S): Changed (C)** – The vulnerable component is the client-side code, but the impact (executing arbitrary script) affects the user's browser session and the data accessible within the application's security context.
- **Confidentiality (C): None (N)** – No impact on confidentiality, as the account being reset is not the victim's

- **Integrity (I): Low (L)** – Attacker may impact the browser's integrity, performing actions on behalf of the user
- **Availability (A): None (N)** – No impact on availability

4. Impact

Successful exploitation of this vulnerability has a **Low/Medium** impact:

- **Credential Theft:** A script could present a fake login prompt within the context of the legitimate site to capture the user's credentials.
- **Internal Network Pivoting:** If the victim is within an internal network, the XSS could be used as a foothold to perform attacks against other internal systems from the victim's browser.

5. Remediation Recommendations

To fix this vulnerability, use your custom `validateRedirectUrl` function to validate the `redirectTo` parameter before using it for redirection.

6. Timeline and Disclosure Process

- **2026-03-11:** Vulnerability discovered and this report prepared.
- **2026-03-11:** Report sent to Rallly security contact / maintainers.
- **2026-03-11:** Maintainer acknowledges receipt of the report and merges fix.
- **2026-03-25:** Vulnerability submission to VulDB.
- **2026-??-??:** Public disclosure of security advisory.