Missing Consent Verification in OAuth Proxy Callback Facilitates Confused Deputy Vulnerabilities

High **jlowin** published **GHSA-rww4-4w9c-7733** 2 weeks ago

Package

fastmcp (pip)

Affected versions	Patched versions
<3.2.0	3.2.0

Description

Summary

While testing the *GitHubProvider* OAuth integration, which allows authentication to a FastMCP MCP server via a FastMCP OAuthProxy using GitHub OAuth, it was discovered that the FastMCP OAuthProxy does not properly validate the user's consent upon receiving the authorization code from GitHub. In combination with GitHub's behavior of skipping the consent page for previously authorized clients, this introduces a Confused Deputy vulnerability.

Technical Details

An adversary can initiate an authentication flow by connecting their malicious MCP client to a benign MCP server using the *GitHubProvider* OAuth integration. During this flow, the attacker consents to connect their client to the MCP server and, at that point, can capture the GitHub authorization URL they are redirected to after granting consent. The attacker can then lure a victim, who is already logged into GitHub and has previously connected an MCP client to the benign MCP server, to open this captured URL. As a result, the victim's browser is immediately redirected to the OAuthProxy's callback endpoint, which does not correctly enforce that this browser has just given consent. The OAuthProxy then redirects the victim's browser to the malicious MCP client's callback URL with a valid authorization code. The attacker can exchange this code for an access token to the benign MCP server associated with the victim's GitHub account, potentially gaining unauthorized access to resources tied to that account.

Although this issue was verified in practice only for the *GitHubProvider*, a review of the source code, specifically the `OAuthProxy._handle_idp_callback` function, shows that the IdP callback handler does not verify whether the browser sending the `state` and `code` has previously consented to connecting the client to the server. As long as a valid `state` and `code` pair is provided, the OAuthProxy requests an access token from the IdP and then redirects the user-agent to the client's callback URL with a new `code` and the corresponding `state`, allowing the client to retrieve the access token from the proxy. This pattern causes all OAuth integrations whose IdP allows skipping the consent page to be vulnerable to this attack.

Skipping the consent page is not, by itself, a vulnerability on the IdP side. Many providers legitimately skip consent for first-party or previously authorized clients with the same scopes. In this case, the core problem lies in the OAuthProxy callback handler not correctly verifying that the browser issuing the callback request is the same one that has just given the required consent.

Steps to reproduce

1. Set up an MCP server using the *GitHubProvider* integration.
2. Connect a benign MCP client to this MCP server.
3. Configure your default browser to route all traffic through an interception proxy such as Burp Suite.
4. In a private browsing window or a second browser, log into the GitHub account used in step 2.
5. As the attacker, connect a new (malicious) MCP client to the MCP server from step 1.
6. When the browser opens for the attacker's client, enable interception in your proxy.
7. In the browser, confirm the consent prompt.
8. In the proxy, forward all requests up to the authorization request to the GitHub authorization server.
9. Copy the authorization URL and drop the intercepted request.
10. Simulate luring the victim onto the URL by opening this URL in the browser window opened in step 4.
11. Observe that the malicious client receives a valid authorization code and gains access to the benign MCP server using the victim's GitHub account.

In a more realistic scenario, the malicious client could be a public MCP client or a simple web server that logs the received authorization code or token, which the attacker then uses to obtain the access token and connect to the MCP server as the victim.

Recommendation

To mitigate this issue, the OAuthProxy should verify that the browser sending the authorization code has actually given consent for the corresponding client. This can be achieved by setting and validating a consent cookie or similar browser-bound state, as described in the mitigations section for this vulnerability in the MCP specification.

Severity

High

CVE ID

CVE-2026-27124

Weaknesses

▶ CWE-441

Credits

an7y Reporter