SSRF Protection Bypass (TOCTOU & Default Insecure)

High **igor-magun-wd** published **GHSA-2x8m-83vc-6wv4** last week

Package

flowise (npm)

Affected versions	Patched versions
<= 3.0.13	3.1.0

flowise-components (npm)

<= 3.0.13	3.1.0

Description

Summary

The core security wrappers (secureAxiosRequest and secureFetch) intended to prevent Server-Side Request Forgery (SSRF) contain multiple logic flaws. These flaws allow attackers to bypass the allow/deny lists via DNS Rebinding (Time-of-Check Time-of-Use) or by exploiting the default configuration which fails to enforce any deny list.

Details

The flaws exist in packages/components/src/httpSecurity.ts.

Default Insecure: If process.env.HTTP_DENY_LIST is undefined, checkDenyList returns immediately, allowing all requests (including localhost).

DNS Rebinding (TOCTOU): The function performs a DNS lookup (dns.lookup) to validate the IP, and then the HTTP client performs a new lookup to connect. An attacker can serve a valid IP first, then switch to an internal IP (e.g., 127.0.0.1) for the second lookup.

PoC

nsure HTTP_DENY_LIST is unset (default behavior).

Use any node utilizing secureFetch to access http://127.0.0.1.

Result: Request succeeds.

Scenario 2: DNS Rebinding

Attacker controls domain attacker.com and a custom DNS server.

Configure DNS to return 1.1.1.1 (Safe IP) with TTL=0 for the first query.

Configure DNS to return 127.0.0.1 (Blocked IP) for subsequent queries.

Flowise validates attacker.com -> 1.1.1.1 (Allowed).

Flowise fetches attacker.com -> 127.0.0.1 (Bypass).

Run the following for manual verification

"// PoC for httpSecurity.ts Bypasses
import * as dns from 'dns/promises';

// Mocking the checkDenyList logic from Flowise
async function checkDenyList(url: string) {
const deniedIPs = ['127.0.0.1', '0.0.0.0']; // Simplified deny list logic

```
if (!process.env.HTTP_DENY_LIST) {
    console.log("⚠️  HTTP_DENY_LIST not set. Returning allowed.");
    return; // Vulnerability 1: Default Insecure
}

const { hostname } = new URL(url);
const { address } = await dns.lookup(hostname);

if (deniedIPs.includes(address)) {
    throw new Error(`IP ${address} is denied`);
}
console.log(`✅ IP ${address} allowed check.`);
```

}

async function runPoC() {
console.log("--- Test 1: Default Configuration (Unset HTTP_DENY_LIST) ---");
// Ensure env var is unset
delete process.env.HTTP_DENY_LIST;
try {
await checkDenyList('http://127.0.0.1');
console.log("[PASS] Default config allowed localhost access.");
} catch (e) {
console.log("[FAIL] Blocked:", e.message);
}

```
console.log("\n--- Test 2: 'private' Keyword Bypass (Logic Flaw) ---");
process.env.HTTP_DENY_LIST = 'private'; // User expects this to block localhost
try {
    await checkDenyList('http://127.0.0.1');
    // In real Flowise code, 'private' is not expanded to IPs, so it only blocks the
string "private"
    console.log("[PASS] 'private' keyword failed to block localhost (Mock
simulation).");
} catch (e) {
    console.log("[FAIL] Blocked:", e.message);
}
```

}

runPoC();"

Impact

Confidentiality: High (Access to internal services if protection is bypassed).

Integrity: Low/Medium (If internal services allow state changes via GET).

Availability: Low.

Severity

High 7.1 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	High
Privileges required	Low
User interaction	None
Scope	Unchanged
Confidentiality	High
Integrity	High
Availability	Low

Learn more about base metrics

CVSS:3.0/AV:N/AC:H/PR:L/UI:N/S:U/C:H/I:H/A:L

CVE ID

CVE-2026-41272

Weaknesses

No CWEs

Credits

ESPanda666 Reporter