COMMUNITY CONTENT

Home › Submit › 785632

Submit #785632: hcengineering platform v0.7.382 Server-Side Request Forgery

Title	hcengineering platform v0.7.382 Server-Side Request Forgery
Description	https://github.com/hcengineering/platform

Server-Side Request Forgery (SSRF) via Import Endpoints

Description

The `/import` endpoint (both GET and POST) accepts an arbitrary URL from the authenticated user and makes an HTTP request to it from the server. There is no validation on the URL scheme, host, or IP address. An attacker can use this to probe internal network services, access cloud metadata endpoints (e.g., `http://x.x.x.x/`), scan ports, and exfiltrate data from internal services.

The server also logs user-supplied cookie values to stdout, leaking potentially sensitive session tokens.

Vulnerable Code

File: `server/front/src/index.ts` (Lines 679–765, GET handler)

```typescript
const handleImportGet = async (req: Request, res: Response): Promise<void> => {
  // ... auth check ...
  const url = req.query.url as string          // User-controlled URL - NO VALIDATION
  const cookie = req.query.cookie as string | undefined

  console.log('importing from', url)
  console.log('cookie', cookie)                // SENSITIVE DATA LOGGED

  const options =
    cookie !== undefined
      ? { headers: { Cookie: cookie } }        // User-controlled Cookie header forwarded
      : {}

  https.get(url, options, (response) => {      // SSRF: Fetches arbitrary URL
    // ... stores response in workspace storage ...
  })
}
```

File: `server/front/src/index.ts` (Lines 767–845, POST handler — identical pattern)

Community Content

Submissions are made by VulDB community users. VulDB is *not responsible* for their content nor the links to external sources.

Please use the raw information shown and the links listed *with caution*. They might contain malicious and harmful actions, code or data.

The corresponding VulDB entries contain the moderated, verified, and normalized information provided within the raw submission.

Documentation

- Submission Policy
- Data Processing
- CVE Handling

```typescript
const handleImportPost = async (req: Request, res: Response): Promise<void> => {
  // ... auth check ...
  const { url, cookie } = req.body              // User-controlled
  console.log('importing from', url)
  console.log('cookie', cookie)                 // SENSITIVE DATA LOGGED
  // ... same https.get(url, options, ...) pattern ...
}
```

Attack Scenario

1. Attacker authenticates with any valid workspace token.
2. `GET /import?token=<token>&url=http://x.x.x.x/latest/meta-data/iam/security-credentials/` — Reads AWS instance role credentials.
3. `GET /import?token=<token>&url=http://localhost:5432/` — Probes internal PostgreSQL port.
4. `GET /import?token=<token>&url=http://internal-admin-panel:8080/api/secrets` — Accesses internal services not exposed to the internet.
5. The response body is stored in the workspace's blob storage, allowing the attacker to retrieve it afterward.

Impact

Access to internal network services, cloud metadata credentials, and internal APIs. In cloud environments, this frequently leads to full infrastructure compromise via stolen IAM credentials.

Recommendation

1. **Implement a URL allowlist** — Only allow HTTPS and restrict to known external domains.
2. **Block private/reserved IP ranges** (RFC 1918, link-local, loopback, cloud metadata):
```typescript
import { isPrivate } from 'ip';
const parsed = new URL(url);
const resolved = await dns.resolve(parsed.hostname);
if (resolved.some(ip => isPrivate(ip) || ip.startsWith('169.254'))) {
  throw new Error('Forbidden: Internal addresses are not allowed');
}
```
3. **Remove `console.log('cookie', cookie)`** — this leaks sensitive session tokens to logs.
4. **Remove cookie forwarding** — the server should never forward user-supplied Cookie headers.

User Ghufran Khan (UID 95493)

Submission 03/22/2026 04:29 PM (15 days ago)

Moderation 04/05/2026 06:00 PM (14 days later)

Status Accepted

VulDB entry [hcengineering Huly Platform 0.7.382 Import Endpoint index.ts server-side request forgery]

Points 17