

Budibase / budibase Public

<> Code Issues 273 Pull requests 16 Discussions Actions Projects

# Commit 5b0fe83



melohagan authored 3 weeks ago · 29 / 29 · Verified

Merge pull request #18236 from Budibase/fix/ssrf-rest

fix: block internal REST targets by default

master (#18236) · database-2.1.0 ... 3.33.4

2 parents [34aa529](#) + [473bc2c](#) commit 5b0fe83

4 files changed +298 -69 lines changed

Top



- ✓ packages
  - ✓ backend-core/src/blacklist
    - blacklist.ts
  - ✓ tests
    - blacklist.spec.ts
  - ✓ server/src
    - ✓ api/routes/tests/queries
      - rest.spec.ts
    - ✓ automations/tests
      - oauth2.spec.ts

4 files changed +298 -69 lines changed



✓ ...ges/backend-core/src/blacklist/blacklist.ts

@@ -3,52 +3,140 @@ import net from "net"

3 3 import env from "../environment"

```
4 4      import { promisify } from "util"
5 5
6 - let blacklistArray: string[] | undefined
7 + const DEFAULT_BLACKLIST = [
8 +   "127.0.0.0/8",
9 +   "10.0.0.0/8",
10 +  "172.16.0.0/12",
11 +  "192.168.0.0/16",
12 +  "169.254.0.0/16",
13 +  "0.0.0.0/8",
14 +  "::1/128",
15 +  "fc00::/7",
16 +  "fe80::/10",
17 + ] as const
18 + let blacklist: net.BlockList | undefined
19
7 19   const performLookup = promisify(dns.lookup)
8 20
9 - async function lookup(address: string): Promise<string[]> {
10 -   if (!net.isIP(address)) {
11 -     // need this for URL parsing simply
12 -     if (!address.startsWith("http")) {
13 -       address = `https://${address}`
14 -     }
15 -     address = new URL(address).hostname
21 + function getVersion(address: string): "ipv4" | "ipv6" {
22 +   return net.isIP(address) === 6 ? "ipv6" : "ipv4"
23 + }
24 +
25 + function getMaxPrefixLength(address: string): number | null {
26 +   const ipVersion = net.isIP(address)
27 +   if (ipVersion === 4) {
28 +     return 32
16 29   }
30 +   if (ipVersion === 6) {
31 +     return 128
32 +   }
33 +   return null
34 + }
35 +
```

```
36 + function getValidSubnetPrefix(address: string, prefix: string): number | null {
37 +   if (!/^\\d+$/ .test(prefix)) {
38 +     return null
39 +   }
40 +
41 +   const parsedPrefix = Number.parseInt(prefix, 10)
42 +   const maxPrefixLength = getMaxPrefixLength(address)
43 +   if (maxPrefixLength == null || parsedPrefix > maxPrefixLength) {
44 +     return null
45 +   }
46 +
47 +   return parsedPrefix
48 + }
49 +
50 + function parseAddress(address: string) {
51 +   if (net.isIP(address)) {
52 +     return address
53 +   }
54 +   if (!address.startsWith("http")) {
55 +     address = `https://${address}`
56 +   }
57 +   return new URL(address).hostname.replace(/^[|\\]$ /g, "")
58 + }
59 +
60 + async function lookup(address: string): Promise<string[]> {
61 +   address = parseAddress(address)
17 62     const addresses = await performLookup(address, {
18 63       all: true,
19 64     })
20 65     return addresses.map(addr => addr.address)
21 66   }
22 67
68 + function addEntryToBlacklist(blockList: net.BlockList, entry: string) {
69 +   const trimmed = entry.trim()
70 +   if (!trimmed) {
71 +     return
72 +   }
73 +
74 +   const segments = trimmed.split("/")
75 +   if (segments.length > 2) {
```

```
76 + console.log(`Ignoring invalid blacklist entry: ${trimmed}`)
77 + return
78 + }
79 +
80 + const [ip, prefix] = segments
81 + const parsedIp = net.isIP(ip)
82 + if (segments.length === 2) {
83 +   const parsedPrefix = getValidSubnetPrefix(ip, prefix)
84 +   if (parsedIp && parsedPrefix !== null) {
85 +     blacklist.addSubnet(ip, parsedPrefix, getIpVersion(ip))
86 +     return
87 +   }
88 +
89 +   console.log(`Ignoring invalid blacklist entry: ${trimmed}`)
90 +   return
91 + }
92 +
93 + if (parsedIp) {
94 +   blacklist.addAddress(ip, getIpVersion(ip))
95 + }
96 + }
97 +
```

```
23 98 export async function refreshBlacklist() {
24 -   const blacklist = env.BLACKLIST_IPS
25 -   const list = blacklist?.split(",") || []
26 -   let final: string[] = []
27 -   for (let addr of list) {
28 -     const trimmed = addr.trim()
29 -     if (!net.isIP(trimmed)) {
30 -       const addresses = await lookup(trimmed)
31 -       final = final.concat(addresses)
32 -     } else {
33 -       final.push(trimmed)
```

```
99 + const next = new net.BlockList()
100 + for (const entry of DEFAULT_BLACKLIST) {
101 +   addEntryToBlacklist(next, entry)
102 + }
103 +
104 + const configuredBlacklist = env.BLACKLIST_IPS?.split(",") || []
105 + for (const entry of configuredBlacklist) {
```

```
106 +     const trimmed = entry.trim()
107 +     if (!trimmed) {
108 +         continue
109 +     }
110 +
111 +     const [ip] = trimmed.split("/")
112 +     if (net.isIP(ip)) {
113 +         addEntryToBlacklist(next, trimmed)
114 +         continue
115 +     }
116 +
117 +     const addresses = await lookup(trimmed)
118 +     for (const address of addresses) {
119 +         addEntryToBlacklist(next, address)
34 120     }
35 121     }
36 -     blacklistArray = final
122 +
123 +     blacklist = next
37 124     }
38 125
39 126     export async function isBlacklisted(address: string): Promise<boolean> {
40 -     if (!blacklistArray) {
127 +     if (!blacklist) {
41 128         await refreshBlacklist()
42 129     }
43 -     if (blacklistArray?.length === 0) {
44 -         return false
45 -     }
46 -     // no need for DNS
130 +
47 131     let ips: string[]
48 132     if (!net.isIP(address)) {
49 -     ips = await lookup(address)
133 +     try {
134 +         ips = await lookup(address)
135 +     } catch {
136 +         return true
137 +     }
50 138     } else {
```

```

51 139     ips = [address]
52 140     }
53 -   return !!blackListArray?.find(addr => ips.includes(addr))
141 +   return ips.some(ip => blackList!.check(ip, getIpVersion(ip)))
54 142     }

```

▼ ...-core/src/blacklist/tests/blacklist.spec.ts ...

```

... @@ -1,46 +1,155 @@
1 1     import { refreshBlacklist, isBlacklisted } from ".."
2 -   import env from "../../environment"
2 +   import { setEnv } from "../../environment"
3 3
4 4     describe("blacklist", () => {
5 -     beforeAll(async () => {
6 -       env._set(
7 -         "BLACKLIST_IPS",
8 -         "www.google.com,192.168.1.1, 1.1.1.1,2.2.2.2/something"
9 -       )
10 -      await refreshBlacklist()
11 -    })
5 +     describe("default ranges", () => {
6 +       let restoreEnv: (() => void) | undefined
12 7
13 -     it("should blacklist 192.168.1.1", async () => {
14 -       expect(await isBlacklisted("192.168.1.1")).toBe(true)
15 -     })
8 +     beforeAll(async () => {
9 +       restoreEnv = setEnv({ BLACKLIST_IPS: undefined })
10 +       await refreshBlacklist()
11 +     })
16 12
17 -     it("should allow 192.168.1.2", async () => {
18 -       expect(await isBlacklisted("192.168.1.2")).toBe(false)
19 -     })
13 +     afterAll(async () => {
14 +       restoreEnv?.()
15 +       await refreshBlacklist()
16 +     })
20 17
21 -     it("should blacklist www.google.com", async () => {

```

```
22 -     expect(await isBlacklisted("www.google.com")).toBe(true)
23 -   })
18 +   it("should blacklist localhost", async () => {
19 +     expect(await isBlacklisted("127.0.0.1")).toBe(true)
20 +   })
24 21
25 -   it("should handle a complex domain", async () => {
26 -     expect(
27 -       await isBlacklisted("https://www.google.com/derp/?something=1")
28 -     ).toBe(true)
29 -   })
22 +   it("should blacklist RFC1918 addresses", async () => {
23 +     expect(await isBlacklisted("192.168.1.1")).toBe(true)
24 +     expect(await isBlacklisted("10.0.0.1")).toBe(true)
25 +     expect(await isBlacklisted("172.16.0.1")).toBe(true)
26 +   })
30 27
31 -   it("should allow www.microsoft.com", async () => {
32 -     expect(await isBlacklisted("www.microsoft.com")).toBe(false)
28 +   it("should blacklist link-local addresses", async () => {
29 +     expect(await isBlacklisted("169.254.169.254")).toBe(true)
30 +   })
31 +
32 +   it("should allow public IPs by default", async () => {
33 +     expect(await isBlacklisted("8.8.8.8")).toBe(false)
34 +   })
35 +
36 +   it("should block addresses that fail lookup or parsing", async () => {
37 +     expect(await isBlacklisted("http://[")).toBe(true)
38 +   })
39 +
40 +   it("should block addresses when DNS lookup fails", async () => {
41 +     expect(await isBlacklisted("https://budibase-ssrf.invalid")).toBe(true)
42 +   })
33 43   })
34 44
35 -   it("should blacklist an IP that needed trimming", async () => {
36 -     expect(await isBlacklisted("1.1.1.1")).toBe(true)
45 +   describe("configured entries", () => {
46 +     let restoreEnv: (() => void) | undefined
```

```
47 +
48 +   beforeAll(async () => {
49 +     restoreEnv = setEnv({
50 +       BLACKLIST_IPS: "www.google.com,192.168.1.1,1.1.1.1",
51 +     })
52 +     await refreshBlacklist()
53 +   })
54 +
55 +   afterAll(async () => {
56 +     restoreEnv?.()
57 +     await refreshBlacklist()
58 +   })
59 +
60 +   it("should blacklist 192.168.1.1", async () => {
61 +     expect(await isBlacklisted("192.168.1.1")).toBe(true)
62 +   })
63 +
64 +   it("should allow public IPs that are not configured", async () => {
65 +     expect(await isBlacklisted("8.8.8.8")).toBe(false)
66 +   })
67 +
68 +   it("should blacklist www.google.com", async () => {
69 +     expect(await isBlacklisted("www.google.com")).toBe(true)
70 +   })
71 +
72 +   it("should handle a complex domain", async () => {
73 +     expect(
74 +       await isBlacklisted("https://www.google.com/derp/?something=1")
75 +     ).toBe(true)
76 +   })
77 +
78 +   it("should allow www.microsoft.com", async () => {
79 +     expect(await isBlacklisted("www.microsoft.com")).toBe(false)
80 +   })
81 +
82 +   it("should blacklist an IP that needed trimming", async () => {
83 +     expect(await isBlacklisted("1.1.1.1")).toBe(true)
84 +   })
85 +
86 +   it("should blacklist 1.1.1.1/something", async () => {
```

```
87 +     expect(await isBlacklisted("1.1.1.1/something")).toBe(true)
88 +   })
37 89   })
38 90
39 -   it("should blacklist 1.1.1.1/something", async () => {
40 -     expect(await isBlacklisted("1.1.1.1/something")).toBe(true)
91 +   describe("malformed CIDR entries", () => {
92 +     let restoreEnv: (() => void) | undefined
93 +
94 +     afterEach(async () => {
95 +       restoreEnv?.()
96 +       restoreEnv = undefined
97 +       await refreshBlacklist()
98 +     })
99 +
100 +     it("should ignore out of range ipv4 prefixes", async () => {
101 +       restoreEnv = setEnv({ BLACKLIST_IPS: "1.1.1.1/33" })
102 +
103 +       await refreshBlacklist()
104 +
105 +       expect(await isBlacklisted("1.1.1.1")).toBe(false)
106 +       expect(await isBlacklisted("1.1.1.2")).toBe(false)
107 +     })
108 +
109 +     it("should ignore partially numeric prefixes", async () => {
110 +       restoreEnv = setEnv({ BLACKLIST_IPS: "2.2.2.2/1foo" })
111 +
112 +       await refreshBlacklist()
113 +
114 +       expect(await isBlacklisted("2.2.2.2")).toBe(false)
115 +       expect(await isBlacklisted("64.0.0.1")).toBe(false)
116 +     })
117 +
118 +     it("should ignore empty prefixes", async () => {
119 +       restoreEnv = setEnv({ BLACKLIST_IPS: "3.3.3.3/" })
120 +
121 +       await refreshBlacklist()
122 +
123 +       expect(await isBlacklisted("3.3.3.3")).toBe(false)
124 +     })
```

```

125 +
126 +     it("should ignore entries with multiple slashes", async () => {
127 +         restoreEnv = setEnv({ BLACKLIST_IPS: "4.4.4.4/24/extra" })
128 +
129 +         await refreshBlacklist()
130 +
131 +         expect(await isBlacklisted("4.4.4.4")).toBe(false)
132 +         expect(await isBlacklisted("4.4.4.5")).toBe(false)
133 +     })
41 134     })
42 135
43 -     it("should blacklist 2.2.2.2", async () => {
44 -         expect(await isBlacklisted("2.2.2.2")).toBe(true)
136 + describe("valid CIDR entries", () => {
137 +     let restoreEnv: (() => void) | undefined
138 +
139 +     afterEach(async () => {
140 +         restoreEnv?.()
141 +         restoreEnv = undefined
142 +         await refreshBlacklist()
143 +     })
144 +
145 +     it("should blacklist the full configured ipv4 subnet", async () => {
146 +         restoreEnv = setEnv({ BLACKLIST_IPS: "5.5.5.0/24" })
147 +
148 +         await refreshBlacklist()
149 +
150 +         expect(await isBlacklisted("5.5.5.1")).toBe(true)
151 +         expect(await isBlacklisted("5.5.5.200")).toBe(true)
152 +         expect(await isBlacklisted("5.5.6.1")).toBe(false)
153 +     })
45 154     })
46 155 })

```

▼ ...r/src/api/routes/tests/queries/rest.spec.ts ...

```

↑
@@ -2,9 +2,10 @@ import * as setup from "../utilities"
2 2     import TestConfiguration from "../../../tests/utilities/TestConfiguration"
3 3     import { BodyType, Datasource, SourceName } from "@budibase/types"
4 4     import { getCachedVariable } from "../../../threads/utils"
5 + import { blacklist, setEnv as setCoreEnv } from "@budibase/backend-core"

```

```

5     6     import { generator } from "@budibase/backend-core/tests"
6     7     import type { MockAgent } from "undici"
7     - import { setEnv } from "../../../../../environment"
8     + import { setEnv as setServerEnv } from "../../../../../environment"
8     9     import { installHttpMocking, resetHttpMocking } from
    ..../.../tests/jestEnv"
9     10
10    11    describe("rest", () => {
@@ -120,7 +121,7 @@ describe("rest", () => {
120   121    }
121   122
122   123    beforeAll(async () => {
123   -   restoreEnv = setEnv({ REST_REJECT_UNAUTHORIZED: false })
124   +   restoreEnv = setServerEnv({ REST_REJECT_UNAUTHORIZED: false })
124   125    config = setup.getConfig()
125   126    await config.init()
126   127    datasource = await config.api.datasource.create({
@@ -150,7 +151,7 @@ describe("rest", () => {
150   151
151   152    it("should automatically retry on fail with cached dynamics", async () => {
152   153    const basedOnQuery = await createQuery({
153   -   path: "one.example.com",
154   +   path: "example.com",
154   155    })
155   156
156   157    let cached = await getCacheVariable(basedOnQuery._id!, "foo")
@@ -176,10 +177,10 @@ describe("rest", () => {
176   177    const body1 = [{ name: "one" }]
177   178    const body2 = [{ name: "two" }]
178   179    mockAgent!
179   -   .get("http://one.example.com")
180   +   .get("http://example.com")
180   181    .intercept({ path: "/", method: "GET" })
181   182    .reply(200, body1, { headers: jsonHeaders })
182   -   const twoExample = mockAgent!.get("http://two.example.com")
183   +   const twoExample = mockAgent!.get("http://example.org")
183   184    twoExample
184   185    .intercept({ path: "/", method: "GET", query: { test: "one" } })

```

```
185 186 .reply(500, { message: "fail" }, { headers: jsonHeaders })
@@ -196,7 +197,7 @@ describe("rest", () => {
196 197     schema: {},
197 198     readable: true,
198 199     fields: {
199 -     path: "two.example.com",
200 +     path: "example.org",
200 201     queryString: "test={{ foo }}",
201 202     },
202 203     })
@@ -209,6 +210,37 @@ describe("rest", () => {
209 210     expect(cached.rows[0].name).toEqual("one")
210 211     })
211 212
213 + it("should block localhost requests when BLACKLIST_IPS is unset", async () =>
214 + {
215 +     const resetBlacklistEnv = setCoreEnv({ BLACKLIST_IPS: undefined })
216 +     await blacklist.refreshBlacklist()
217 +     try {
218 +         await config.api.query.preview(
219 +             {
220 +                 datasourceId: datasource._id!,
221 +                 name: "test query",
222 +                 parameters: [],
223 +                 queryVerb: "read",
224 +                 transformer: "",
225 +                 schema: {},
226 +                 readable: true,
227 +                 fields: {
228 +                     path: "http://127.0.0.1:5984",
229 +                 },
230 +             },
231 +             {
232 +                 status: 400,
233 +                 body: {
234 +                     message: "Cannot connect to URL.",
235 +                 },
236 +             }
237 +         )
```

```

238 +     } finally {
239 +         resetBlacklistEnv()
240 +         await blacklist.refreshBlacklist()
241 +     }
242 + })
243 +
212 244     it("should update schema when structure changes from JSON to array", async ()
    => {
213 245         const datasource = await config.api.datasource.create({
214 246             name: generator.guid(),

```

...server/src/automations/tests/oauth2.spec.ts

```

@@ -94,7 +94,7 @@ describe("OAuth2 Automation Binding", () => {
94 94     config: {
95 95         clientId: "test_client_id",
96 96         clientSecret: "test_client_secret",
97 -         callbackURL: "http://callback.example.com",
97 +         callbackURL: "http://example.com/callback",
98 98         activated: true,
99 99     },
100 100 }

@@ -122,7 +122,7 @@ describe("OAuth2 Automation Binding", () => {
122 122
123 123     it("should make OAuth2 token available in executeQuery automation step",
    async () => {
124 124         await config.withUser(ssoUser, async () => {
125 -         const pool = getPool("https://api.example.com")
125 +         const pool = getPool("https://example.com")
126 126         let requestCount = 0
127 127
128 128         pool

@@ -144,7 +144,7 @@ describe("OAuth2 Automation Binding", () => {
144 144     })
145 145
146 146     const queryFields: RestQueryFields = {
147 -         path: "https://api.example.com/test",
147 +         path: "https://example.com/test",
148 148         queryString: "",

```

```


149 149         headers: {
150 150             Authorization: "Bearer {{ user.oauth2.accessToken }}",
@@ -174,7 +174,7 @@ describe("OAuth2 Automation Binding", () => {
174 174
175 175         it("should make OAuth2 token available in apiRequest automation step", async
176 176             () => {
176 176             await config.withUser(ssoUser, async () => {
177 -             const pool = getPool("https://api.example.com")
177 +             const pool = getPool("https://example.com")
178 178             let requestCount = 0
179 179
180 180             pool
@@ -203,7 +203,7 @@ describe("OAuth2 Automation Binding", () => {
203 203         })
204 204
205 205         const queryFields: RestQueryFields = {
206 -         path: "https://api.example.com/api-request",
206 +         path: "https://example.com/api-request",
207 207         queryString: "",
208 208         headers: {
209 209             Authorization: "Bearer {{ user.oauth2.accessToken }}",
@@ -258,7 +258,7 @@ describe("OAuth2 Automation Binding", () => {
258 258
259 259         it("should handle 401 Unauthorized and retry mechanism in apiRequest step",
260 260             async () => {
260 260             await config.withUser(ssoUser, async () => {
261 -             const apiPool = getPool("https://api.example.com")
261 +             const apiPool = getPool("https://example.com")
262 262             let firstCall = true
263 263
264 264             apiPool
@@ -293,7 +293,7 @@ describe("OAuth2 Automation Binding", () => {
293 293         })
294 294
295 295         const queryFields: RestQueryFields = {
296 -         path: "https://api.example.com/protected-endpoint",

```

```
296 + path: "https://example.com/protected-endpoint",
297 297   queryString: "",
298 298   headers: {
299 299     Authorization: "Bearer {{ user.oauth2.accessToken }}",

```

Comments 0

  
Please [sign in](#) to comment.