Budibase / **budibase** Public

Code | Issues 263 | Pull requests 13 | Discussions | Actions | Projects

Commit `5b0fe83`

melohagan authored on Mar 13 · 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

Filter files…

- 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

Search within code

...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
   6    + const DEFAULT_BLACKLIST = [
   7    +   "127.0.0.0/8",
   8    +   "10.0.0.0/8",
   9    +   "172.16.0.0/12",
   10   +   "192.168.0.0/16",
   11   +   "169.254.0.0/16",
   12   +   "0.0.0.0/8",
   13   +   "::1/128",
   14   +   "fc00::/7",
   15   +   "fe80::/10",
   16   + ] as const
   17   +
   18   + let blackList: net.BlockList | undefined
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 getIpVersion(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   +
```

```
+ function getValidSubnetPrefix(address: string, prefix: string): number | null {
+   if (!/^\d+$/.test(prefix)) {
+     return null
+   }
+
+   const parsedPrefix = Number.parseInt(prefix, 10)
+   const maxPrefixLength = getMaxPrefixLength(address)
+   if (maxPrefixLength == null || parsedPrefix > maxPrefixLength) {
+     return null
+   }
+
+   return parsedPrefix
+ }
+
+ function parseAddress(address: string) {
+   if (net.isIP(address)) {
+     return address
+   }
+   if (!address.startsWith("http")) {
+     address = `https://${address}`
+   }
+   return new URL(address).hostname.replace(/^\[|\]$/g, "")
+ }
+
+ async function lookup(address: string): Promise<string[]> {
+   address = parseAddress(address)
    const addresses = await performLookup(address, {
      all: true,
    })
    return addresses.map(addr => addr.address)
  }

+ function addEntryToBlacklist(blockList: net.BlockList, entry: string) {
+   const trimmed = entry.trim()
+   if (!trimmed) {
+     return
+   }
+
+   const segments = trimmed.split("/")
+   if (segments.length > 2) {
```

```
+     console.log(`Ignoring invalid blacklist entry: ${trimmed}`)
+     return
+   }
+
+   const [ip, prefix] = segments
+   const parsedIp = net.isIP(ip)
+   if (segments.length === 2) {
+     const parsedPrefix = getValidSubnetPrefix(ip, prefix)
+     if (parsedIp && parsedPrefix !== null) {
+       blockList.addSubnet(ip, parsedPrefix, getIpVersion(ip))
+       return
+     }
+
+     console.log(`Ignoring invalid blacklist entry: ${trimmed}`)
+     return
+   }
+
+   if (parsedIp) {
+     blockList.addAddress(ip, getIpVersion(ip))
+   }
+ }
+
  export async function refreshBlacklist() {
-   const blacklist = env.BLACKLIST_IPS
-   const list = blacklist?.split(",") || []
-   let final: string[] = []
-   for (let addr of list) {
-     const trimmed = addr.trim()
-     if (!net.isIP(trimmed)) {
-       const addresses = await lookup(trimmed)
-       final = final.concat(addresses)
-     } else {
-       final.push(trimmed)
+   const next = new net.BlockList()
+   for (const entry of DEFAULT_BLACKLIST) {
+     addEntryToBlacklist(next, entry)
+   }
+
+   const configuredBlacklist = env.BLACKLIST_IPS?.split(",") || []
+   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 () => {
```

```
                +       expect(await isBlacklisted("1.1.1.1/something")).toBe(true)
                +     })
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 +     })
```

```
                +
                +    it("should ignore entries with multiple slashes", async () => {
                +      restoreEnv = setEnv({ BLACKLIST_IPS: "4.4.4.4/24/extra" })
                +
                +      await refreshBlacklist()
                +
                +      expect(await isBlacklisted("4.4.4.4")).toBe(false)
                +      expect(await isBlacklisted("4.4.4.5")).toBe(false)
                +    })
  41    134     })
  42    135
  43          - it("should blacklist 2.2.2.2", async () => {
  44          -   expect(await isBlacklisted("2.2.2.2")).toBe(true)
                + describe("valid CIDR entries", () => {
                +   let restoreEnv: (() => void) | undefined
                +
                +   afterEach(async () => {
                +     restoreEnv?.()
                +     restoreEnv = undefined
                +     await refreshBlacklist()
                +   })
                +
                +   it("should blacklist the full configured ipv4 subnet", async () => {
                +     restoreEnv = setEnv({ BLACKLIST_IPS: "5.5.5.0/24" })
                +
                +     await refreshBlacklist()
                +
                +     expect(await isBlacklisted("5.5.5.1")).toBe(true)
                +     expect(await isBlacklisted("5.5.5.200")).toBe(true)
                +     expect(await isBlacklisted("5.5.6.1")).toBe(false)
                +   })
  45    154     })
  46    155   })
```

Comments 0

Please sign in to comment.