foxcpp / **maddy** Public

Code | Issues 123 | Pull requests 5 | Discussions | Actions | Wiki

Commit 6a06337

foxcpp committed last week Verified

```
auth/ldap: Fix GHSA-5835-4gvc-32pc

Add proper escaping when building filter expression, add
proper escaping when building DN from DN template.
```

master · v0.9.3

1 parent fd44177 commit 6a06337

6 files changed +198 -8 lines changed

Top

Filter files…

- docs/reference/auth
  - netauth.md
- go.mod
- go.sum
- internal/auth/ldap
  - ldap.go
- tests
  - conn.go
  - ghsa_5835_4gvc_32pc_test.go

6 files changed +198 -8 lines changed

Search within code

docs/reference/auth/netauth.md

```
@@ -7,10 +7,12 @@ maddy needs to know the Entity ID to use for
authentication. It must
7  7  match the string the user provides for the Local Atom part of their
```

```
 8   8    mail address.
 9   9
10       - Note that storage backends conventionally use email addresses.  Since
11       - NetAuth recommends *nix compatible usernames, you will need to map the
12       - email identifiers to NetAuth Entity IDs using `auth_map` (see
13       - documentation page for used storage backend).
     10  + Note that storage backends conventionally use email addresses.  Since NetAuth
     11  + recommends *nix compatible usernames. You will need to either map email
     12  + identifiers specified by user to NetAuth Entity IDs using `auth_map` in
     13  + endpoint.smtp/imap configuration (recommended) or you would need to use
     14  + `storage_map` in storage backend configuration to map NetAuth Entity ID
     15  + specified by user back to appropriate storage backend account names.
14   16
15   17   auth.netauth also can be used as a table module.  This way you can
16   18   check whether the account exists.
```

go.mod

```
@@ -107,6 +107,7 @@ require (
107  107      github.com/hashicorp/go-cleanhttp v0.5.2 // indirect
108  108      github.com/hashicorp/go-retryablehttp v0.7.7 // indirect
109  109      github.com/hashicorp/hcl v1.0.0 // indirect
     110  +   github.com/jimlambrt/gldap v0.1.14 // indirect
110  111      github.com/jmespath/go-jmespath v0.4.0 // indirect
111  112      github.com/josharian/intern v1.0.0 // indirect
112  113      github.com/klauspost/compress v1.17.11 // indirect
```

go.sum

```
@@ -488,6 +488,8 @@ github.com/jcmturner/gokrb5/v8 v8.4.4
     h1:x1Sv4HaTpepFkXbt2IkL29DXRf8sOfZXo8eRKh6
488  488    github.com/jcmturner/gokrb5/v8 v8.4.4/go.mod
            h1:1btQEpgT6k+unzCwX1KdWMEwPPkkgBtP+F6aCACiMrs=
489  489    github.com/jcmturner/rpc/v2 v2.0.3
            h1:7FXXj8Ti1IaVFpSAziCZWNzbNuZmnvw/i6CqLNdWfZY=
490  490    github.com/jcmturner/rpc/v2 v2.0.3/go.mod
            h1:VUJYCIDm3PVOEHw8sgt091/20OJjskO/YJki3ELg/Hc=
     491  + github.com/jimlambrt/gldap v0.1.14
            h1:InG9kldhIu6OoQK0hvfkW1Lqpc5eLJhxiiDTNmRnrDM=
```

```
                492 + github.com/jimlambrt/gldap v0.1.14/go.mod
                    h1:yobW9JIAmqe23dVNOaMWewPaff6jGaHgYjspPIIgYmg=
491             493   github.com/jmespath/go-jmespath v0.0.0-20180206201540-c2b33e8439af/go.mod
                      h1:Nht3zPeWKUH0NzdCt2Blrr5ys8VGpn0CEB0cQHVjt7k=
492             494   github.com/jmespath/go-jmespath v0.4.0
                      h1:BEgLn5cpjn8UN1mAw4NjwDrS35OdebyEtFe+9YPoQUg=
493             495   github.com/jmespath/go-jmespath v0.4.0/go.mod
                      h1:T8mJZnbsbmF+m6zOOFylbeCJqk5+pHWvzYPziyZiYoo=
```

internal/auth/ldap/ldap.go

```
                    @@ -225,7 +225,7 @@ func (a *Auth) Lookup(_ context.Context, username
                    string) (string, bool, error)
225             225           req := ldap.NewSearchRequest(
226             226               a.baseDN, ldap.ScopeWholeSubtree, ldap.NeverDerefAliases,
227             227               2, 0, false,
228             -               strings.ReplaceAll(a.filterTemplate, "{username}", username),
                228 +           strings.ReplaceAll(a.filterTemplate, "{username}",
                    ldap.EscapeFilter(username)),
229             229               []string{"dn"}, nil)
230             230           res, err := conn.Search(req)
231             231           if err != nil {
                    @@ -252,12 +252,12 @@ func (a *Auth) AuthPlain(username, password string)
                    error {
252             252
253             253       var userDN string
254             254       if a.dnTemplate != "" {
255             -           userDN = strings.ReplaceAll(a.dnTemplate, "{username}", username)
                255 +       userDN = strings.ReplaceAll(a.dnTemplate, "{username}",
                    ldap.EscapeDN(username))
256             256       } else {
257             257           req := ldap.NewSearchRequest(
258             258               a.baseDN, ldap.ScopeWholeSubtree, ldap.NeverDerefAliases,
259             259               2, 0, false,
260             -               strings.ReplaceAll(a.filterTemplate, "{username}", username),
                260 +           strings.ReplaceAll(a.filterTemplate, "{username}",
                    ldap.EscapeFilter(username)),
261             261               []string{"dn"}, nil)
262             262           res, err := conn.Search(req)
263             263           if err != nil {
```

tests/conn.go

```
@@ -211,7 +211,7 @@ func (c *Conn) SMTPPlainAuth(username, password string,
expectOk bool) {
211 211         if expectOk {
212 212             c.ExpectPattern("235 *")
213 213         } else {
214     -           c.ExpectPattern("*")
    214 +           c.ExpectPattern("5*")
215 215         }
216 216     }
217 217
@@ -282,6 +282,13 @@ func (c *Conn) Close() error {
282 282         return c.Conn.Close()
283 283     }
284 284
    285 + func (c *Conn) MustClose() {
    286 +     c.T.Helper()
    287 +     if err := c.Close(); err != nil {
    288 +         c.fatal("Close: %v", err)
    289 +     }
    290 + }
    291 +
285 292     func (c *Conn) Rebind(subtest *T) *Conn {
286 293         cpy := *c
287 294         cpy.T = subtest
```

tests/ghsa_5835_4gvc_32pc_test.go

```
@@ -0,0 +1,178 @@
    1 + //go:build integration
    2 +
    3 + package tests_test
    4 +
    5 + import (
    6 +     "strconv"
    7 +     "testing"
```

```
+     "time"
+
+     "github.com/foxcpp/maddy/tests"
+     "github.com/jimlambrt/gldap"
+     "github.com/stretchr/testify/require"
+ )
+
+ type searchEntry struct {
+     dn      string
+     options []gldap.Option
+ }
+
+ type MockLDAP struct {
+     T             *testing.T
+     SearchEntries map[string][]searchEntry
+     AllowedBinds  map[string]string
+ }
+
+ func (ml *MockLDAP) HandleBind(w *gldap.ResponseWriter, r *gldap.Request) {
+     resp := r.NewBindResponse(
+         gldap.WithResponseCode(gldap.ResultInvalidCredentials),
+     )
+
+     m, err := r.GetSimpleBindMessage()
+     if err != nil {
+         require.NoError(ml.T, w.Write(resp))
+         return
+     }
+
+     pass, ok := ml.AllowedBinds[m.UserName]
+     if ok && pass == string(m.Password) {
+         resp.SetResultCode(gldap.ResultSuccess)
+         require.NoError(ml.T, w.Write(resp))
+     }
+
+     require.NoError(ml.T, w.Write(resp))
+ }
+
+ func (ml *MockLDAP) HandleSearch(w *gldap.ResponseWriter, r *gldap.Request) {
+     resp := r.NewSearchDoneResponse()
```

```
48  +     m, err := r.GetSearchMessage()
49  +     if err != nil {
50  +         ml.T.Logf("not a search message: %s", err)
51  +         require.NoError(ml.T, w.Write(resp))
52  +         return
53  +     }
54  +     ml.T.Logf("search base dn: %s", m.BaseDN)
55  +     ml.T.Logf("search scope: %d", m.Scope)
56  +     ml.T.Logf("search filter: %s", m.Filter)
57  +
58  +     entries := ml.SearchEntries[m.Filter]
59  +     for _, entry := range entries {
60  +         ldapEntry := r.NewSearchResponseEntry(entry.dn, entry.options...)
61  +         require.NoError(ml.T, w.Write(ldapEntry))
62  +     }
63  +
64  +     resp.SetResultCode(gldap.ResultSuccess)
65  +     require.NoError(ml.T, w.Write(resp))
66  + }
67  +
68  + func (ml *MockLDAP) Run(address string) {
69  +     s, err := gldap.NewServer()
70  +     if err != nil {
71  +         ml.T.Fatalf("unable to create server: %s", err.Error())
72  +     }
73  +
74  +     // create a router and add a bind handler
75  +     r, err := gldap.NewMux()
76  +     if err != nil {
77  +         ml.T.Fatalf("unable to create router: %s", err.Error())
78  +     }
79  +     require.NoError(ml.T, r.Bind(ml.HandleBind))
80  +     require.NoError(ml.T, r.Search(ml.HandleSearch))
81  +     require.NoError(ml.T, s.Router(r))
82  +     go func() {
83  +         require.NoError(ml.T, s.Run(address))
84  +     }()
85  +     ml.T.Cleanup(func() {
86  +         require.NoError(ml.T, s.Stop())
87  +     })
```

```
+
+	for !s.Ready() {
+		ml.T.Log("Waiting for server to start")
+		time.Sleep(100 * time.Millisecond)
+	}
+ }
+
+ func TestLDAPInjectionFilter(tt *testing.T) {
+	tt.Parallel()
+	t := tests.NewT(tt)
+
+	ldapPort := t.Port("ldap")
+
+	ldapSrv := &MockLDAP{
+		T: tt,
+		AllowedBinds: map[string]string{
+			"DC=com,CN=bob":   "bob_pass",
+			"DC=com,CN=alice": "alice_pass",
+		},
+		SearchEntries: map[string][]searchEntry{
+			"(&(objectClass=inetOrgPerson)(uid=alice))": {
+				{
+					dn: "DC=com,CN=alice",
+					options: []gldap.Option{
+						gldap.WithAttributes(map[string][]string{
+							"objectClass": {"inetOrgPerson"},
+							"uid":         {"alice"},
+							"description": {"prefix_test"},
+						}),
+					},
+				},
+			},
+			"(&(objectClass=inetOrgPerson)(uid=bob))": {
+				{
+					dn: "DC=com,CN=bob",
+					options: []gldap.Option{
+						gldap.WithAttributes(map[string][]string{
+							"objectClass": {"inetOrgPerson"},
+							"uid":         {"bob"},
+							"description": {"prefix_test"},
```

```
128 +                         }),
129 +                     },
130 +                 },
131 +             },
132 +             "(&(objectClass=inetOrgPerson)(uid=bob)(description=prefix*))": {
133 +                 {
134 +                     dn: "DC=com,CN=bob",
135 +                     options: []gldap.Option{
136 +                         gldap.WithAttributes(map[string][]string{
137 +                             "objectClass": {"inetOrgPerson"},
138 +                             "uid":         {"bob"},
139 +                             "description": {"prefix_test"},
140 +                         }),
141 +                     },
142 +                 },
143 +             },
144 +         },
145 +     }
146 +     ldapSrv.Run(":" + strconv.Itoa(int(ldapPort)))
147 +
148 +     t.Port("smtp")
149 +     t.DNS(nil)
150 +     t.Config(`
151 +         hostname mx.maddy.test
152 +         tls off
153 +
154 +         auth.ldap ldap_auth {
155 +             urls ldap://127.0.0.1:{env:TEST_PORT_ldap}
156 +             bind plain "DC=com,CN=bob" "bob_pass"
157 +             base_dn "DC=com"
158 +             filter "(&(objectClass=inetOrgPerson)(uid={username}))"
159 +         }
160 +
161 +         submission tcp://0.0.0.0:{env:TEST_PORT_smtp} {
162 +             auth &ldap_auth
163 +             deliver_to dummy
164 +         }
165 +     `)
166 +     t.Run(1)
167 +     defer t.Close()
```

```
168 +
169 +     smtpConn := t.Conn("smtp")
170 +     defer smtpConn.MustClose()
171 +     smtpConn.SMTPNegotation("clieht.maddy.test", nil, nil)
172 +     smtpConn.SMTPPlainAuth("alice", "alice_pass", true)
173 +
174 +     smtpConn2 := t.Conn("smtp")
175 +     defer smtpConn2.MustClose()
176 +     smtpConn2.SMTPNegotation("clieht.maddy.test", nil, nil)
177 +     smtpConn2.SMTPPlainAuth("bob)(description=prefix*", "bob_pass", false)
178 + }
```

Comments 0

Please sign in to comment.