steveiliop56 / **tinyauth** Public

Code | Issues 13 | Pull requests 6 | Discussions | Actions | Projects

Commit `f26c217`

steveiliop56 authored 2 weeks ago · 6 / 6 · Verified

```
refactor: oauth flow (#726)

* wip

* feat: add oauth session impl in auth service

* feat: move oauth logic into auth service and handle multiple sessions

* tests: fix tests

* fix: review comments

* fix: prevent ddos attacks in oauth rate limit
```

`main` (#726) · v5.0.6 ••• nightly

1 parent `d71a8e0` commit `f26c217`

15 files changed +519 -557 lines changed

Top

Filter files…

- internal
  - bootstrap
    - app_bootstrap.go
    - router_bootstrap.go
    - service_bootstrap.go
  - config
    - config.go
  - controller
    - oauth_controller.go

- proxy_controller_test.go
- user_controller_test.go
- service
  - auth_service.go
  - generic_oauth_service.go
  - github_oauth_service.go
  - google_oauth_service.go
  - oauth_broker_service.go
  - oauth_extractors.go
  - oauth_presets.go
  - oauth_service.go

15 files changed +519 -557 lines changed

Search within code

internal/bootstrap/app_bootstrap.go

```
@@ -22,16 +22,17 @@ import (
22 22  type BootstrapApp struct {
23 23      config  config.Config
24 24      context struct {
25    -         appUrl             string
26    -         uuid               string
27    -         cookieDomain       string
28    -         sessionCookieName  string
29    -         csrfCookieName     string
30    -         redirectCookieName string
31    -         users              []config.User
32    -         oauthProviders     map[string]config.OAuthServiceConfig
33    -         configuredProviders []controller.Provider
34    -         oidcClients        []config.OIDCClientConfig
   25 +         appUrl                string
   26 +         uuid                  string
   27 +         cookieDomain          string
   28 +         sessionCookieName     string
   29 +         csrfCookieName        string
   30 +         redirectCookieName    string
   31 +         oauthSessionCookieName string
```

```
     32 +         users                []config.User
     33 +         oauthProviders       map[string]config.OAuthServiceConfig
     34 +         configuredProviders  []controller.Provider
     35 +         oidcClients          []config.OIDCClientConfig
35   36         }
36   37         services Services
37   38     }
          @@ -113,6 +114,7 @@ func (app *BootstrapApp) Setup() error {
113  114         app.context.sessionCookieName = fmt.Sprintf("%s-%s",
                config.SessionCookieName, cookieId)
114  115         app.context.csrfCookieName = fmt.Sprintf("%s-%s", config.CSRFCookieName,
                cookieId)
115  116         app.context.redirectCookieName = fmt.Sprintf("%s-%s",
                config.RedirectCookieName, cookieId)
     117 +       app.context.oauthSessionCookieName = fmt.Sprintf("%s-%s",
                config.OAuthSessionCookieName, cookieId)
116  118
117  119         // Dumps
118  120         tlog.App.Trace().Interface("config", app.config).Msg("Config dump")
          @@ -190,12 +192,12 @@ func (app *BootstrapApp) Setup() error {
190  192
191  193         // Start db cleanup routine
192  194         tlog.App.Debug().Msg("Starting database cleanup routine")
193       -       go app.dbCleanup(queries)
     195 +       go app.dbCleanupRoutine(queries)
194  196
195  197         // If analytics are not disabled, start heartbeat
196  198         if app.config.Analytics.Enabled {
197  199             tlog.App.Debug().Msg("Starting heartbeat routine")
198       -           go app.heartbeat()
     200 +           go app.heartbeatRoutine()
199  201         }
200  202
201  203         // If we have an socket path, bind to it
          @@ -226,7 +228,7 @@ func (app *BootstrapApp) Setup() error {
226  228         return nil
227  229     }
```

```
228  230
229       - func (app *BootstrapApp) heartbeat() {
     231  + func (app *BootstrapApp) heartbeatRoutine() {
230  232       ticker := time.NewTicker(time.Duration(12) * time.Hour)
231  233       defer ticker.Stop()
232  234
          @@ -280,7 +282,7 @@ func (app *BootstrapApp) heartbeat() {
280  282       }
281  283   }
282  284
283       - func (app *BootstrapApp) dbCleanup(queries *repository.Queries) {
     285  + func (app *BootstrapApp) dbCleanupRoutine(queries *repository.Queries) {
284  286       ticker := time.NewTicker(time.Duration(30) * time.Minute)
285  287       defer ticker.Stop()
286  288       ctx := context.Background()
```

internal/bootstrap/router_bootstrap.go

```
          @@ -77,12 +77,13 @@ func (app *BootstrapApp) setupRouter() (*gin.Engine,
          error) {
77  77        contextController.SetupRoutes()
78  78
79  79        oauthController :=
          controller.NewOAuthController(controller.OAuthControllerConfig{
80        -         AppURL:             app.config.AppURL,
81        -         SecureCookie:       app.config.Auth.SecureCookie,
82        -         CSRFCookieName:     app.context.csrfCookieName,
83        -         RedirectCookieName: app.context.redirectCookieName,
84        -         CookieDomain:       app.context.cookieDomain,
85        -     }, apiRouter, app.services.authService, app.services.oauthBrokerService)
      80  +         AppURL:                 app.config.AppURL,
      81  +         SecureCookie:           app.config.Auth.SecureCookie,
      82  +         CSRFCookieName:         app.context.csrfCookieName,
      83  +         RedirectCookieName:     app.context.redirectCookieName,
      84  +         CookieDomain:           app.context.cookieDomain,
      85  +         OAuthSessionCookieName: app.context.oauthSessionCookieName,
      86  +     }, apiRouter, app.services.authService)
86  87
87  88        oauthController.SetupRoutes()
```

```
88  89
```

internal/bootstrap/service_bootstrap.go

```
@@ -58,6 +58,16 @@ func (app *BootstrapApp) initServices(queries
*repository.Queries) (Services, er
58  58
59  59          services.accessControlService = accessControlsService
60  60
    61  +       oauthBrokerService :=
    service.NewOAuthBrokerService(app.context.oauthProviders)
    62  +
    63  +       err = oauthBrokerService.Init()
    64  +
    65  +       if err != nil {
    66  +           return Services{}, err
    67  +       }
    68  +
    69  +       services.oauthBrokerService = oauthBrokerService
    70  +
61  71          authService := service.NewAuthService(service.AuthServiceConfig{
62  72              Users:              app.context.users,
63  73              OauthWhitelist:     app.config.OAuth.Whitelist,
@@ -70,7 +80,7 @@ func (app *BootstrapApp) initServices(queries
*repository.Queries) (Services, er
70  80              SessionCookieName:  app.context.sessionCookieName,
71  81              IP:                 app.config.Auth.IP,
72  82              LDAPGroupsCacheTTL: app.config.Ldap.GroupCacheTTL,
73      -       }, dockerService, services.ldapService, queries)
    83  +       }, dockerService, services.ldapService, queries,
    services.oauthBrokerService)
74  84
75  85          err = authService.Init()
76  86
@@ -80,16 +90,6 @@ func (app *BootstrapApp) initServices(queries
*repository.Queries) (Services, er
80  90
81  91          services.authService = authService
82  92
```

```
83      -     oauthBrokerService :=
          service.NewOAuthBrokerService(app.context.oauthProviders)
84      -
85      -     err = oauthBrokerService.Init()
86      -
87      -     if err != nil {
88      -         return Services{}, err
89      -     }
90      -
91      -     services.oauthBrokerService = oauthBrokerService
92      -
93  93          oidcService := service.NewOIDCService(service.OIDCServiceConfig{
94  94              Clients:        app.config.OIDC.Clients,
95  95              PrivateKeyPath: app.config.OIDC.PrivateKeyPath,
```

internal/config/config.go

```
    @@ -73,6 +73,7 @@ var BuildTimestamp = "0000-00-00T00:00:00Z"
73  73  var SessionCookieName = "tinyauth-session"
74  74  var CSRFCookieName = "tinyauth-csrf"
75  75  var RedirectCookieName = "tinyauth-redirect"
    76  + var OAuthSessionCookieName = "tinyauth-oauth"
76  77
77  78  // Main app config
78  79
```

internal/controller/oauth_controller.go

```
    @@ -21,26 +21,25 @@ type OAuthRequest struct {
21  21  }
22  22
23  23  type OAuthControllerConfig struct {
24      -     CSRFCookieName     string
25      -     RedirectCookieName string
26      -     SecureCookie       bool
27      -     AppURL             string
28      -     CookieDomain       string
    24  +     CSRFCookieName         string
    25  +     OAuthSessionCookieName string
```

```
      26  +     RedirectCookieName     string
      27  +     SecureCookie           bool
      28  +     AppURL                 string
      29  +     CookieDomain           string
29    30      }
30    31
31    32      type OAuthController struct {
32    33          config OAuthControllerConfig
33    34          router *gin.RouterGroup
34    35          auth   *service.AuthService
35          -     broker *service.OAuthBrokerService
36    36      }
37    37
38          - func NewOAuthController(config OAuthControllerConfig, router *gin.RouterGroup,
              auth *service.AuthService, broker *service.OAuthBrokerService) *OAuthController
              {
      38  + func NewOAuthController(config OAuthControllerConfig, router *gin.RouterGroup,
              auth *service.AuthService) *OAuthController {
39    39          return &OAuthController{
40    40              config: config,
41    41              router: router,
42    42              auth:   auth,
43          -         broker: broker,
44    43          }
45    44      }
46    45
              @@ -63,21 +62,30 @@ func (controller *OAuthController) oauthURLHandler(c
              *gin.Context) {
63    62              return
64    63          }
65    64
66          -     service, exists := controller.broker.GetService(req.Provider)
      65  +     sessionId, session, err := controller.auth.NewOAuthSession(req.Provider)
67    66
68          -     if !exists {
69          -         tlog.App.Warn().Msgf("OAuth provider not found: %s", req.Provider)
70          -         c.JSON(404, gin.H{
71          -             "status":  404,
72          -             "message": "Not Found",
      67  +     if err != nil {
```

```
+         tlog.App.Error().Err(err).Msg("Failed to create OAuth session")
+         c.JSON(500, gin.H{
+             "status":  500,
+             "message": "Internal Server Error",
+         })
+         return
+     }
+
+     authUrl, err := controller.auth.GetOAuthURL(sessionId)
+
+     if err != nil {
+         tlog.App.Error().Err(err).Msg("Failed to get OAuth URL")
+         c.JSON(500, gin.H{
+             "status":  500,
+             "message": "Internal Server Error",
          })
          return
      }

-     service.GenerateVerifier()
-     state := service.GenerateState()
-     authURL := service.GetAuthURL(state)
-     c.SetCookie(controller.config.CSRFCookieName, state, int(time.Hour.Seconds()), "/", fmt.Sprintf(".%s", controller.config.CookieDomain), controller.config.SecureCookie, true)
+     c.SetCookie(controller.config.OAuthSessionCookieName, sessionId, int(time.Hour.Seconds()), "/", fmt.Sprintf(".%s", controller.config.CookieDomain), controller.config.SecureCookie, true)
+     c.SetCookie(controller.config.CSRFCookieName, session.State, int(time.Hour.Seconds()), "/", fmt.Sprintf(".%s", controller.config.CookieDomain), controller.config.SecureCookie, true)

      redirectURI := c.Query("redirect_uri")
      isRedirectSafe := utils.IsRedirectSafe(redirectURI, controller.config.CookieDomain)
@@ -95,7 +103,7 @@ func (controller *OAuthController) oauthURLHandler(c *gin.Context) {
      c.JSON(200, gin.H{
          "status":  200,
          "message": "OK",
```

```
98        -          "url":      authURL,
      106 +          "url":      authUrl,
99    107          })
100   108      }
101   109
          @@ -112,6 +120,17 @@ func (controller *OAuthController)
          oauthCallbackHandler(c *gin.Context) {
112   120              return
113   121          }
114   122
      123 +      sessionIdCookie, err := c.Cookie(controller.config.OAuthSessionCookieName)
      124 +
      125 +      if err != nil {
      126 +          tlog.App.Warn().Err(err).Msg("OAuth session cookie missing")
      127 +          c.Redirect(http.StatusTemporaryRedirect, fmt.Sprintf("%s/error",
          controller.config.AppURL))
      128 +          return
      129 +      }
      130 +
      131 +      c.SetCookie(controller.config.OAuthSessionCookieName, "", -1, "/",
          fmt.Sprintf(".%s", controller.config.CookieDomain),
          controller.config.SecureCookie, true)
      132 +      defer controller.auth.EndOAuthSession(sessionIdCookie)
      133 +
115   134          state := c.Query("state")
116   135          csrfCookie, err := c.Cookie(controller.config.CSRFCookieName)
117   136
          @@ -125,28 +144,15 @@ func (controller *OAuthController)
          oauthCallbackHandler(c *gin.Context) {
125   144          c.SetCookie(controller.config.CSRFCookieName, "", -1, "/",
          fmt.Sprintf(".%s", controller.config.CookieDomain),
          controller.config.SecureCookie, true)
126   145
127   146          code := c.Query("code")
128       -      service, exists := controller.broker.GetService(req.Provider)
      147 +      _, err = controller.auth.GetOAuthToken(sessionIdCookie, code)
129   148
130       -      if !exists {
131       -          tlog.App.Warn().Msgf("OAuth provider not found: %s", req.Provider)
```

```
132      - 		c.Redirect(http.StatusTemporaryRedirect, fmt.Sprintf("%s/error", controller.config.AppURL))
133      - 		return
134      - 	}
135      - 
136      - 	err = service.VerifyCode(code)
137  149 		if err != nil {
138      - 		tlog.App.Error().Err(err).Msg("Failed to verify OAuth code")
     150 + 		tlog.App.Error().Err(err).Msg("Failed to exchange code for token")
139  151 			c.Redirect(http.StatusTemporaryRedirect, fmt.Sprintf("%s/error", controller.config.AppURL))
140  152 			return
141  153 		}
142  154 
143      - 	user, err := controller.broker.GetUser(req.Provider)
144      - 
145      - 	if err != nil {
146      - 		tlog.App.Error().Err(err).Msg("Failed to get user from OAuth provider")
147      - 		c.Redirect(http.StatusTemporaryRedirect, fmt.Sprintf("%s/error", controller.config.AppURL))
148      - 		return
149      - 	}
     155 + 	user, err := controller.auth.GetOAuthUserinfo(sessionIdCookie)
150  156 
151  157 		if user.Email == "" {
152  158 			tlog.App.Error().Msg("OAuth provider did not return an email")
@@ -192,13 +198,21 @@ func (controller *OAuthController) oauthCallbackHandler(c *gin.Context) {
192  198 			username = strings.Replace(user.Email, "@", "_", 1)
193  199 		}
194  200 
     201 + 	service, err := controller.auth.GetOAuthService(sessionIdCookie)
     202 + 
     203 + 	if err != nil {
     204 + 		tlog.App.Error().Err(err).Msg("Failed to get OAuth service for session")
     205 + 		c.Redirect(http.StatusTemporaryRedirect, fmt.Sprintf("%s/error", controller.config.AppURL))
     206 + 		return
     207 + 	}
```

```
                208 +
195             209         sessionCookie := repository.Session{
196             210             Username:    username,
197             211             Name:        name,
198             212             Email:       user.Email,
199             213             Provider:    req.Provider,
200             214             OAuthGroups: utils.CoalesceToString(user.Groups),
201             -               OAuthName:   service.GetName(),
                215 +           OAuthName:   service.Name(),
202             216             OAuthSub:    user.Sub,
203             217         }
204             218
```

internal/controller/proxy_controller_test.go

```
@@ -85,7 +85,7 @@ func setupProxyController(t *testing.T, middlewares
[]gin.HandlerFunc) (*gin.Eng
85  85              LoginTimeout:       300,
86  86              LoginMaxRetries:    3,
87  87              SessionCookieName:  "tinyauth-session",
88      -       }, dockerService, nil, queries)
    88  +       }, dockerService, nil, queries, &service.OAuthBrokerService{})
89  89
90  90          // Controller
91  91          ctrl := controller.NewProxyController(controller.ProxyControllerConfig{
```

internal/controller/user_controller_test.go

```
@@ -71,7 +71,7 @@ func setupUserController(t *testing.T, middlewares *
[]gin.HandlerFunc) (*gin.Eng
71  71              LoginTimeout:       300,
72  72              LoginMaxRetries:    3,
73  73              SessionCookieName:  "tinyauth-session",
74      -       }, nil, nil, queries)
    74  +       }, nil, nil, queries, &service.OAuthBrokerService{})
75  75
76  76          // Controller
77  77          ctrl := controller.NewUserController(controller.UserControllerConfig{
```

internal/service/auth_service.go

```
@@ -17,8 +17,21 @@ import (
        "github.com/gin-gonic/gin"
        "github.com/google/uuid"
        "golang.org/x/crypto/bcrypt"
+       "golang.org/x/exp/slices"
+       "golang.org/x/oauth2"
)

+const MaxOAuthPendingSessions = 256
+const OAuthCleanupCount = 16
+
+type OAuthPendingSession struct {
+       State     string
+       Verifier  string
+       Token     *oauth2.Token
+       Service   *OAuthServiceImpl
+       ExpiresAt time.Time
+}
+
type LdapGroupsCache struct {
        Groups  []string
        Expires time.Time
@@ -45,28 +58,34 @@ type AuthServiceConfig struct {
}

type AuthService struct {
-       config          AuthServiceConfig
-       docker          *DockerService
-       loginAttempts   map[string]*LoginAttempt
-       ldapGroupsCache map[string]*LdapGroupsCache
-       loginMutex      sync.RWMutex
-       ldapGroupsMutex sync.RWMutex
-       ldap            *LdapService
-       queries         *repository.Queries
+       config              AuthServiceConfig
+       docker              *DockerService
+       loginAttempts       map[string]*LoginAttempt
+       ldapGroupsCache     map[string]*LdapGroupsCache
+       oauthPendingSessions map[string]*OAuthPendingSession
+       loginMutex          sync.RWMutex
+       ldapGroupsMutex     sync.RWMutex
+       oauthPendingMutex   sync.RWMutex
+       ldap                *LdapService
+       queries             *repository.Queries
}

func NewAuthService(config AuthServiceConfig, docker *DockerService, ldap *LdapService, queries *repository.Queries) *AuthService {
        return &AuthService{
-               config:          config,
-               docker:          docker,
-               loginAttempts:   make(map[string]*LoginAttempt),
-               ldapGroupsCache: make(map[string]*LdapGroupsCache),
-               loginMutex:      sync.RWMutex{},
-               ldapGroupsMutex: sync.RWMutex{},
-               ldap:            ldap,
-               queries:         queries,
+               config:              config,
+               docker:              docker,
+               loginAttempts:       make(map[string]*LoginAttempt),
+               ldapGroupsCache:     make(map[string]*LdapGroupsCache),
+               oauthPendingSessions: make(map[string]*OAuthPendingSession),
+               loginMutex:          sync.RWMutex{},
+               ldapGroupsMutex:     sync.RWMutex{},
+               oauthPendingMutex:   sync.RWMutex{},
+               ldap:                ldap,
+               queries:             queries,
        }
}
```

```
      65 +     oauthPendingSessions map[string]*OAuthPendingSession
      66 +     oauthMutex           sync.RWMutex
      67 +     loginMutex           sync.RWMutex
      68 +     ldapGroupsMutex      sync.RWMutex
      69 +     ldap                 *LdapService
      70 +     queries              *repository.Queries
      71 +     oauthBroker          *OAuthBrokerService
56    72     }
57    73
58       - func NewAuthService(config AuthServiceConfig, docker *DockerService, ldap
           *LdapService, queries *repository.Queries) *AuthService {
      74 + func NewAuthService(config AuthServiceConfig, docker *DockerService, ldap
           *LdapService, queries *repository.Queries, oauthBroker *OAuthBrokerService)
           *AuthService {
59    75         return &AuthService{
60       -         config:          config,
61       -         docker:          docker,
62       -         loginAttempts:   make(map[string]*LoginAttempt),
63       -         ldapGroupsCache: make(map[string]*LdapGroupsCache),
64       -         ldap:            ldap,
65       -         queries:         queries,
      76 +         config:               config,
      77 +         docker:               docker,
      78 +         loginAttempts:        make(map[string]*LoginAttempt),
      79 +         ldapGroupsCache:      make(map[string]*LdapGroupsCache),
      80 +         oauthPendingSessions: make(map[string]*OAuthPendingSession),
      81 +         ldap:                 ldap,
      82 +         queries:              queries,
      83 +         oauthBroker:          oauthBroker,
66    84         }
67    85     }
68    86
69    87     func (auth *AuthService) Init() error {
      88 +     go auth.CleanupOAuthSessionsRoutine()
70    89         return nil
71    90     }
72    91
           @@ -553,3 +572,177 @@ func (auth *AuthService) IsBypassedIP(acls
           config.AppIP, ip string) bool {
```

```
553  572          tlog.App.Debug().Str("ip", ip).Msg("IP not in bypass list, continuing with authentication")
554  573          return false
555  574      }
     575  +
     576  + func (auth *AuthService) NewOAuthSession(serviceName string) (string, OAuthPendingSession, error) {
     577  +     auth.ensureOAuthSessionLimit()
     578  +
     579  +     service, ok := auth.oauthBroker.GetService(serviceName)
     580  +
     581  +     if !ok {
     582  +         return "", OAuthPendingSession{}, fmt.Errorf("oauth service not found: %s", serviceName)
     583  +     }
     584  +
     585  +     sessionId, err := uuid.NewRandom()
     586  +
     587  +     if err != nil {
     588  +         return "", OAuthPendingSession{}, fmt.Errorf("failed to generate session ID: %w", err)
     589  +     }
     590  +
     591  +     state := service.NewRandom()
     592  +     verifier := service.NewRandom()
     593  +
     594  +     session := OAuthPendingSession{
     595  +         State:     state,
     596  +         Verifier:  verifier,
     597  +         Service:   &service,
     598  +         ExpiresAt: time.Now().Add(1 * time.Hour),
     599  +     }
     600  +
     601  +     auth.oauthMutex.Lock()
     602  +     auth.oauthPendingSessions[sessionId.String()] = &session
     603  +     auth.oauthMutex.Unlock()
     604  +
     605  +     return sessionId.String(), session, nil
     606  + }
     607  +
```

```
608 + func (auth *AuthService) GetOAuthURL(sessionId string) (string, error) {
609 +     session, err := auth.getOAuthPendingSession(sessionId)
610 +
611 +     if err != nil {
612 +         return "", err
613 +     }
614 +
615 +     return (*session.Service).GetAuthURL(session.State, session.Verifier), nil
616 + }
617 +
618 + func (auth *AuthService) GetOAuthToken(sessionId string, code string)
      (*oauth2.Token, error) {
619 +     session, err := auth.getOAuthPendingSession(sessionId)
620 +
621 +     if err != nil {
622 +         return nil, err
623 +     }
624 +
625 +     token, err := (*session.Service).GetToken(code, session.Verifier)
626 +
627 +     if err != nil {
628 +         return nil, fmt.Errorf("failed to exchange code for token: %w", err)
629 +     }
630 +
631 +     auth.oauthMutex.Lock()
632 +     session.Token = token
633 +     auth.oauthMutex.Unlock()
634 +
635 +     return token, nil
636 + }
637 +
638 + func (auth *AuthService) GetOAuthUserinfo(sessionId string) (config.Claims,
      error) {
639 +     session, err := auth.getOAuthPendingSession(sessionId)
640 +
641 +     if err != nil {
642 +         return config.Claims{}, err
643 +     }
644 +
645 +     if session.Token == nil {
```

```
646 +         return config.Claims{}, fmt.Errorf("oauth token not found for session:
    %s", sessionId)
647 +     }
648 +
649 +     userinfo, err := (*session.Service).GetUserinfo(session.Token)
650 +
651 +     if err != nil {
652 +         return config.Claims{}, fmt.Errorf("failed to get userinfo: %w", err)
653 +     }
654 +
655 +     return userinfo, nil
656 + }
657 +
658 + func (auth *AuthService) GetOAuthService(sessionId string) (OAuthServiceImpl,
    error) {
659 +     session, err := auth.getOAuthPendingSession(sessionId)
660 +
661 +     if err != nil {
662 +         return nil, err
663 +     }
664 +
665 +     return *session.Service, nil
666 + }
667 +
668 + func (auth *AuthService) EndOAuthSession(sessionId string) {
669 +     auth.oauthMutex.Lock()
670 +     delete(auth.oauthPendingSessions, sessionId)
671 +     auth.oauthMutex.Unlock()
672 + }
673 +
674 + func (auth *AuthService) CleanupOAuthSessionsRoutine() {
675 +     ticker := time.NewTicker(30 * time.Minute)
676 +     defer ticker.Stop()
677 +
678 +     for range ticker.C {
679 +         auth.oauthMutex.Lock()
680 +
681 +         now := time.Now()
682 +
683 +         for sessionId, session := range auth.oauthPendingSessions {
```

```
684 +             if now.After(session.ExpiresAt) {
685 +                 delete(auth.oauthPendingSessions, sessionId)
686 +             }
687 +         }
688 +
689 +         auth.oauthMutex.Unlock()
690 +     }
691 + }
692 +
693 + func (auth *AuthService) getOAuthPendingSession(sessionId string)
      (*OAuthPendingSession, error) {
694 +     auth.ensureOAuthSessionLimit()
695 +
696 +     auth.oauthMutex.RLock()
697 +     session, exists := auth.oauthPendingSessions[sessionId]
698 +     auth.oauthMutex.RUnlock()
699 +
700 +     if !exists {
701 +         return &OAuthPendingSession{}, fmt.Errorf("oauth session not found:
      %s", sessionId)
702 +     }
703 +
704 +     if time.Now().After(session.ExpiresAt) {
705 +         auth.oauthMutex.Lock()
706 +         delete(auth.oauthPendingSessions, sessionId)
707 +         auth.oauthMutex.Unlock()
708 +         return &OAuthPendingSession{}, fmt.Errorf("oauth session expired: %s",
      sessionId)
709 +     }
710 +
711 +     return session, nil
712 + }
713 +
714 + func (auth *AuthService) ensureOAuthSessionLimit() {
715 +     auth.oauthMutex.Lock()
716 +     defer auth.oauthMutex.Unlock()
717 +
718 +     if len(auth.oauthPendingSessions) >= MaxOAuthPendingSessions {
719 +
720 +         cleanupIds := make([]string, 0, OAuthCleanupCount)
```

```
721 +
722 +         for range OAuthCleanupCount {
723 +             oldestId := ""
724 +             oldestTime := int64(0)
725 +
726 +             for id, session := range auth.oauthPendingSessions {
727 +                 if oldestTime == 0 {
728 +                     oldestId = id
729 +                     oldestTime = session.ExpiresAt.Unix()
730 +                     continue
731 +                 }
732 +                 if slices.Contains(cleanupIds, id) {
733 +                     continue
734 +                 }
735 +                 if session.ExpiresAt.Unix() < oldestTime {
736 +                     oldestId = id
737 +                     oldestTime = session.ExpiresAt.Unix()
738 +                 }
739 +             }
740 +
741 +             cleanupIds = append(cleanupIds, oldestId)
742 +         }
743 +
744 +         for _, id := range cleanupIds {
745 +             delete(auth.oauthPendingSessions, id)
746 +         }
747 +     }
748 + }
```

internal/service/generic_oauth_service.go

Load Diff

This file was deleted.

internal/service/github_oauth_service.go

Load Diff

This file was deleted.

internal/service/google_oauth_service.go

Load Diff

This file was deleted.

internal/service/oauth_broker_service.go

```
@@ -1,60 +1,48 @@
 package service

 import (
-    "errors"
-
     "github.com/steveiliop56/tinyauth/internal/config"
     "github.com/steveiliop56/tinyauth/internal/utils/tlog"

     "golang.org/x/exp/slices"
+    "golang.org/x/oauth2"
 )

-type OAuthService interface {
-    Init() error
-    GenerateState() string
-    GenerateVerifier() string
-    GetAuthURL(state string) string
-    VerifyCode(code string) error
-    Userinfo() (config.Claims, error)
-    GetName() string
+type OAuthServiceImpl interface {
+    Name() string
+    NewRandom() string
```

```
      14  +     GetAuthURL(state string, verifier string) string
      15  +     GetToken(code string, verifier string) (*oauth2.Token, error)
      16  +     GetUserinfo(token *oauth2.Token) (config.Claims, error)
20    17      }
21    18
22    19      type OAuthBrokerService struct {
23          -     services map[string]OAuthService
      20  +     services map[string]OAuthServiceImpl
24    21          configs  map[string]config.OAuthServiceConfig
25    22      }
26    23
      24  + var presets = map[string]func(config config.OAuthServiceConfig) *OAuthService{
      25  +     "github": newGitHubOAuthService,
      26  +     "google": newGoogleOAuthService,
      27  + }
      28  +
27    29      func NewOAuthBrokerService(configs map[string]config.OAuthServiceConfig)
              *OAuthBrokerService {
28    30          return &OAuthBrokerService{
29          -         services: make(map[string]OAuthService),
      31  +         services: make(map[string]OAuthServiceImpl),
30    32              configs:  configs,
31    33          }
32    34      }
33    35
34    36      func (broker *OAuthBrokerService) Init() error {
35    37          for name, cfg := range broker.configs {
36          -         switch name {
37          -         case "github":
38          -             service := NewGithubOAuthService(cfg)
39          -             broker.services[name] = service
40          -         case "google":
41          -             service := NewGoogleOAuthService(cfg)
42          -             broker.services[name] = service
43          -         default:
44          -             service := NewGenericOAuthService(cfg)
45          -             broker.services[name] = service
46          -         }
47          -     }
48          -
```

```
49      -     for name, service := range broker.services {
50      -         err := service.Init()
51      -         if err != nil {
52      -             tlog.App.Error().Err(err).Msgf("Failed to initialize OAuth service: %s", name)
53      -             return err
        38  +         if presetFunc, exists := presets[name]; exists {
        39  +             broker.services[name] = presetFunc(cfg)
        40  +             tlog.App.Debug().Str("service", name).Msg("Loaded OAuth service from preset")
        41  +         } else {
        42  +             broker.services[name] = NewOAuthService(cfg)
        43  +             tlog.App.Debug().Str("service", name).Msg("Loaded OAuth service from config")
54      44            }
55      -         tlog.App.Info().Str("service", name).Msg("Initialized OAuth service")
56      45        }
57      -
58      46        return nil
59      47    }
60      48
          @@ -67,15 +55,7 @@ func (broker *OAuthBrokerService) GetConfiguredServices() []string {
67      55        return services
68      56    }
69      57
70      - func (broker *OAuthBrokerService) GetService(name string) (OAuthService, bool) {
        58  + func (broker *OAuthBrokerService) GetService(name string) (OAuthServiceImpl, bool) {
71      59        service, exists := broker.services[name]
72      60        return service, exists
73      61    }
74      -
75      - func (broker *OAuthBrokerService) GetUser(service string) (config.Claims, error) {
76      -     oauthService, exists := broker.services[service]
77      -     if !exists {
78      -         return config.Claims{}, errors.New("oauth service not found")
79      -     }
80      -     return oauthService.Userinfo()
```

```
81  - }
```

internal/service/oauth_extractors.go

```
    @@ -0,0 +1,102 @@
1   + package service
2   +
3   + import (
4   +     "encoding/json"
5   +     "errors"
6   +     "fmt"
7   +     "io"
8   +     "net/http"
9   +     "strconv"
10  +
11  +     "github.com/steveiliop56/tinyauth/internal/config"
12  + )
13  +
14  + type GithubEmailResponse []struct {
15  +     Email   string `json:"email"`
16  +     Primary bool   `json:"primary"`
17  + }
18  +
19  + type GithubUserInfoResponse struct {
20  +     Login string `json:"login"`
21  +     Name  string `json:"name"`
22  +     ID    int    `json:"id"`
23  + }
24  +
25  + func defaultExtractor(client *http.Client, url string) (config.Claims, error) {
26  +     return simpleReq[config.Claims](client, url, nil)
27  + }
28  +
29  + func githubExtractor(client *http.Client, url string) (config.Claims, error) {
30  +     var user config.Claims
31  +
32  +     userInfo, err := simpleReq[GithubUserInfoResponse](client,
      "https://api.github.com/user", map[string]string{
33  +         "accept": "application/vnd.github+json",
34  +     })
35  +     if err != nil {
```

```
36  +         return config.Claims{}, err
37  +     }
38  +
39  +     userEmails, err := simpleReq[GithubEmailResponse](client,
      "https://api.github.com/user/emails", map[string]string{
40  +         "accept": "application/vnd.github+json",
41  +     })
42  +     if err != nil {
43  +         return config.Claims{}, err
44  +     }
45  +
46  +     if len(userEmails) == 0 {
47  +         return user, errors.New("no emails found")
48  +     }
49  +
50  +     for _, email := range userEmails {
51  +         if email.Primary {
52  +             user.Email = email.Email
53  +             break
54  +         }
55  +     }
56  +
57  +     // Use first available email if no primary email was found
58  +     if user.Email == "" {
59  +         user.Email = userEmails[0].Email
60  +     }
61  +
62  +     user.PreferredUsername = userInfo.Login
63  +     user.Name = userInfo.Name
64  +     user.Sub = strconv.Itoa(userInfo.ID)
65  +
66  +     return user, nil
67  + }
68  +
69  + func simpleReq[T any](client *http.Client, url string, headers
      map[string]string) (T, error) {
70  +     var decodedRes T
71  +
72  +     req, err := http.NewRequest("GET", url, nil)
73  +     if err != nil {
```

```
74  +         return decodedRes, err
75  +     }
76  +
77  +     for key, value := range headers {
78  +         req.Header.Add(key, value)
79  +     }
80  +
81  +     res, err := client.Do(req)
82  +     if err != nil {
83  +         return decodedRes, err
84  +     }
85  +     defer res.Body.Close()
86  +
87  +     if res.StatusCode < 200 || res.StatusCode >= 300 {
88  +         return decodedRes, fmt.Errorf("request failed with status: %s",
      res.Status)
89  +     }
90  +
91  +     body, err := io.ReadAll(res.Body)
92  +     if err != nil {
93  +         return decodedRes, err
94  +     }
95  +
96  +     err = json.Unmarshal(body, &decodedRes)
97  +     if err != nil {
98  +         return decodedRes, err
99  +     }
100 +
101 +     return decodedRes, nil
102 + }
```

internal/service/oauth_presets.go

```
...   @@ -0,0 +1,23 @@
1   + package service
2   +
3   + import (
4   +     "github.com/steveiliop56/tinyauth/internal/config"
5   +     "golang.org/x/oauth2/endpoints"
6   + )
7   +
```

```
+ func newGoogleOAuthService(config config.OAuthServiceConfig) *OAuthService {
+     scopes := []string{"openid", "email", "profile"}
+     config.Scopes = scopes
+     config.AuthURL = endpoints.Google.AuthURL
+     config.TokenURL = endpoints.Google.TokenURL
+     config.UserinfoURL = "https://openidconnect.googleapis.com/v1/userinfo"
+     return NewOAuthService(config)
+ }
+
+ func newGitHubOAuthService(config config.OAuthServiceConfig) *OAuthService {
+     scopes := []string{"read:user", "user:email"}
+     config.Scopes = scopes
+     config.AuthURL = endpoints.GitHub.AuthURL
+     config.TokenURL = endpoints.GitHub.TokenURL
+     return NewOAuthService(config).WithUserinfoExtractor(githubExtractor)
+ }
```

internal/service/oauth_service.go

```
@@ -0,0 +1,78 @@
+ package service
+
+ import (
+     "context"
+     "crypto/tls"
+     "net/http"
+     "time"
+
+     "github.com/steveiliop56/tinyauth/internal/config"
+     "golang.org/x/oauth2"
+ )
+
+ type UserinfoExtractor func(client *http.Client, url string) (config.Claims,
  error)
+
+ type OAuthService struct {
+     serviceCfg        config.OAuthServiceConfig
+     config            *oauth2.Config
+     ctx               context.Context
+     userinfoExtractor UserinfoExtractor
+ }
```

```
21  +
22  + func NewOAuthService(config config.OAuthServiceConfig) *OAuthService {
23  +     httpClient := &http.Client{
24  +         Timeout: 30 * time.Second,
25  +         Transport: &http.Transport{
26  +             TLSClientConfig: &tls.Config{
27  +                 InsecureSkipVerify: config.Insecure,
28  +             },
29  +         },
30  +     }
31  +     ctx := context.Background()
32  +     ctx = context.WithValue(ctx, oauth2.HTTPClient, httpClient)
33  +
34  +     return &OAuthService{
35  +         serviceCfg: config,
36  +         config: &oauth2.Config{
37  +             ClientID:     config.ClientID,
38  +             ClientSecret: config.ClientSecret,
39  +             RedirectURL:  config.RedirectURL,
40  +             Scopes:       config.Scopes,
41  +             Endpoint: oauth2.Endpoint{
42  +                 AuthURL:  config.AuthURL,
43  +                 TokenURL: config.TokenURL,
44  +             },
45  +         },
46  +         ctx:              ctx,
47  +         userinfoExtractor: defaultExtractor,
48  +     }
49  + }
50  +
51  + func (s *OAuthService) WithUserinfoExtractor(extractor UserinfoExtractor)
      *OAuthService {
52  +     s.userinfoExtractor = extractor
53  +     return s
54  + }
55  +
56  + func (s *OAuthService) Name() string {
57  +     return s.serviceCfg.Name
58  + }
59  +
```

```
60  + func (s *OAuthService) NewRandom() string {
61  +     // The generate verifier function just creates a random string,
62  +     // so we can use it to generate a random state as well
63  +     random := oauth2.GenerateVerifier()
64  +     return random
65  + }
66  +
67  + func (s *OAuthService) GetAuthURL(state string, verifier string) string {
68  +     return s.config.AuthCodeURL(state, oauth2.AccessTypeOnline,
      oauth2.S256ChallengeOption(verifier))
69  + }
70  +
71  + func (s *OAuthService) GetToken(code string, verifier string) (*oauth2.Token,
      error) {
72  +     return s.config.Exchange(s.ctx, code, oauth2.VerifierOption(verifier))
73  + }
74  +
75  + func (s *OAuthService) GetUserinfo(token *oauth2.Token) (config.Claims, error) {
76  +     client := oauth2.NewClient(s.ctx, oauth2.StaticTokenSource(token))
77  +     return s.userinfoExtractor(client, s.serviceCfg.UserinfoURL)
78  + }
```

Comments 0

Please sign in to comment.