【GO-Micro】 micro API 网关增加 JWT 鉴权功能

作者：Allenxuxu
原文链接：https://ld246.com/article/1561381232613
来源网站：链滴

github完整代码地址

micro API网关

micro API网关是基于go-micro开发的，具有服务发现，负载均衡和RPC通信的能力。

业界普遍做法是将鉴权，限流，熔断等功能也纳入API网关。micro API网关本身是可插拔的，可以通新增插件的方式加入其他功能。

JWT (JSON Web Token)

JWT是是微服务中常用的授权技术，关于JWT的技术原理可以参考阮一峰的博文

JWT库封装

- lib/token 目录下封装了JWT的库。有一点特殊的是，库中利用consul的KV存储和micro的go-config库实现了动态更新JWT的PrivateKey功能，实际生产中还是应该使用拥有发布和权限管理的配置中。
- go-config 是micro作者实现的一个可动态加载、可插拔的配置库，可以从多种格式文件或者远服务获取配置。详情可以参考文档中文文档|英文文档
- PrivateKey是JWT在编解码时使用的私钥，一旦泄漏，客户端便可以利用这个私钥篡改、伪造Toen。所以一般生产环境中都必须具备动态更新私钥的能力，一旦发现泄漏可以立即更改，或者定期更私钥，提高安全性。

```go
// InitConfig 初始化
func (srv *Token) InitConfig(address string, path ...string) {
    consulSource := consul.NewSource(
        consul.WithAddress(address),
    )
    srv.conf = config.NewConfig()
    err := srv.conf.Load(consulSource)
    if err != nil {
        log.Fatal(err)
    }

    value := srv.conf.Get(path...).Bytes()
    if err != nil {
        log.Fatal(err)
    }

    srv.put(value)
    log.Println("JWT privateKey:", string(srv.get()))
    srv.enableAutoUpdate(path...)
}

func (srv *Token) enableAutoUpdate(path ...string) {
    go func() {
        for {
            w, err := srv.conf.Watch(path...)
            if err != nil {
```

```
            log.Println(err)
        }
        v, err := w.Next()
        if err != nil {
            log.Println(err)
        }

        value := v.Bytes()
        srv.put(value)
        log.Println("New JWT privateKey:", string(srv.get()))
    }
}()
}
```

作者已经实现了consul的KV配置的插件，所以只需要导入这个库"github.com/micro/go-config/source/consul"，便可以直接读取consul中的配置。

动态跟新实现就是利用go-config的watch方法，当consul KV里的配置更改，Watch函数返回再通过ext方法读取新数据。将watch 读取的操作起一个协程循环执行（没有考虑优雅退出），通过读写锁保证操作安全。

实现API网关插件

将JWT Token在HTTP头中携带，通过HTTP中间件过滤每一个HTTP请求，提取头中的Token鉴权，过则继续执行，不通过就直接返回。

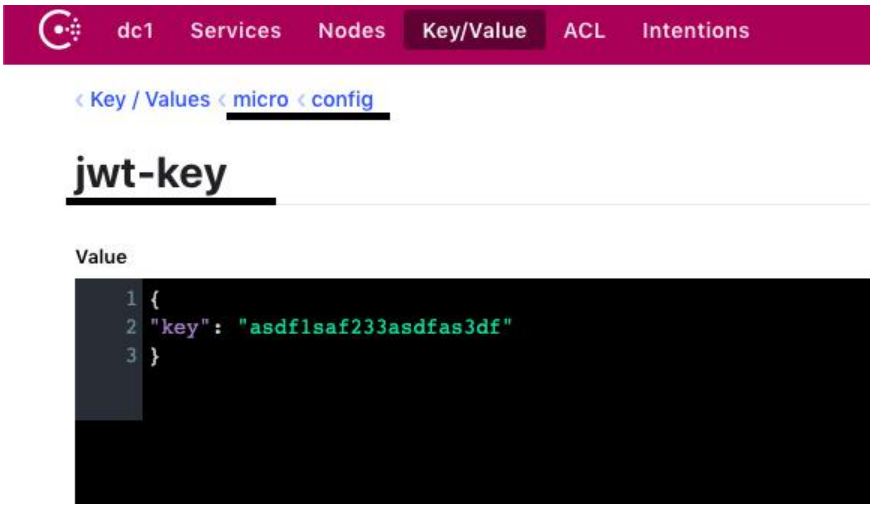
```
//microservices/lib/wrapper/auth

// JWTAuthWrapper JWT鉴权Wrapper
func JWTAuthWrapper(token *token.Token) plugin.Handler {
    return func(h http.Handler) http.Handler {
        return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
            log.Println("auth plugin received: " + r.URL.Path)
            // TODO 从配置中心动态获取白名单URL
            if r.URL.Path == "/user/login" || r.URL.Path == "/user/register"{
                h.ServeHTTP(w, r)
                return
            }

            tokenstr := r.Header.Get("Authorization")
            userFromToken, e := token.Decode(tokenstr)

            if e != nil {
                w.WriteHeader(http.StatusUnauthorized)
                return
            }

            log.Println("User Name : ", userFromToken.UserName)
            r.Header.Set("X-Example-Username", userFromToken.UserName)
            h.ServeHTTP(w, r)
        })
    }
}
 ...
```

```go
// main.go
func init() {
    token := &token.Token{}
    token.InitConfig("127.0.0.1:8500", "micro", "config", "jwt-key", "key")

    plugin.Register(plugin.NewPlugin(
        plugin.WithName("auth"),
        plugin.WithHandler(
            auth.JWTAuthWrapper(token),
        ),
    ))
}
const name = "API gateway"

func main() {
    cmd.Init()
}
```

- 初始化我们封装JWT Token

```go
func (srv *Token) InitConfig(address string, path ...string)
token.InitConfig("127.0.0.1:8500", "micro", "config", "jwt-key", "key")
```

"127.0.0.1:8500" 是本地consul 监听地址，path是可变参数，传递consul KV中的配置路径：micro/onfig/jwt-key。



- 注册插件

```go
func Register(plugin Plugin) error //全局注册一个插件
func NewPlugin(opts ...Option) Plugin //生成一个插件
func WithName(n string) Option //设置插件的名字
func WithHandler(h ...Handler) Option  //http handler中间件
```

注册一个新插件的时候，还可以定制其他操作，具体可以看作者的文档英文文档|中文文档

在hander中将Token进行校验，如果鉴权成功，则调用 h.ServeHTTP(w, r) ，此时micro会调用下个hander。

如果鉴权失败，就修改状态码w.WriteHeader(http.StatusUnauthorized)， 不调用 h.ServeHTTP(w,

)，此时链式调用中断，micro框架不会调用剩下的hander。

github完整代码地址