

SSH任务总结

一个测试Demo：

Demo背景：前端主动像后端发起websocket请求并且携带相关数据，后端先进性判断，如果验证成功就可以放行进行websocket连接，此时前后端可以畅通无阻的通话。

Demo代码：

```
1 package main
2
3 import (
4     "fmt"
5     "github.com/gorilla/websocket"
6     "net/http"
7 )
8
9 var upgrader = websocket.Upgrader{
10     CheckOrigin: func(r *http.Request) bool {
11         return true // 允许所有CORS请求
12     },
13 }
14
15 func serves(w http.ResponseWriter, r *http.Request) {
16     // 从查询参数获取验证信息
17     username := r.URL.Query().Get("username")
18     password := r.URL.Query().Get("password")
19
20     // 验证逻辑
21     if username != "user" || password != "pass" {
22         http.Error(w, "错误请求", http.StatusUnauthorized)
23         return
24     }
25
26     // 升级HTTP连接到WebSocket连接
27     ws, err := upgrader.Upgrade(w, r, nil)
28     if err != nil {
29         fmt.Println("Error upgrading to WebSocket:", err)
30         return
31     }
32     defer ws.Close()
33
34     for {
35         var msg map[string]string
36         // 读取消息
37         err := ws.ReadJSON(&msg)
38         if err != nil {
39             fmt.Println("Error reading json:", err)
40             break
41         }
42
43         // 打印并回复消息
44         fmt.Printf("Received message: %s\n", msg["data"])
```

```

45     err = ws.WriteJSON(map[string]string{"data": "收到你的消息啦"+
msg["data"]}))
46     if err != nil {
47         fmt.Println("Error writing json:", err)
48         break
49     }
50 }
51 }
52
53 func main() {
54     http.HandleFunc("/ws", servews)
55     fmt.Println("Server started on :8080")
56     err := http.ListenAndServe(":8080", nil)
57     if err != nil {
58         fmt.Println("ListenAndServe: ", err)
59     }
60 }

```

注意这个Demo的参数在url上。

下图是用ApiFox建立连接

localhost:8080/ws

请求 接口说明 预览文档 接口名称

Message

Params 2

Header 6

Cookie

设置

Query 参数

参数名	参数值	类型	说明
✓ username	= user	string	
✓ password	= pass	string	
添加参数			

连接建立完成后，在message中输入信息（下图）

请求

接口说明

预览文档

接口名称

Message

Params 2

Header 6

Cookie

设置

Text ▾

1 { "mes": "用户发送的数据" }

此时，后端会自动返回数据。上箭头代表用户（发送数据），下箭头是后端（接受后端数据）

↑	{ "mes": "用户发送的数据" }	17:02:21
↓	{"data": "收到你的消息啦"}	17:02:21

在后端可以收到数据：

```
Received message: r
Received message: lalalalal
Received message: 前端发送数据了
```

关于远程连接

请求：

websocket + ... 开 开发环境

localhost:8080/api/terminal/sshConnect 连接 保存 ...

请求 接口说明 预览文档 websocket

Message Params 4 Header 6 Cookie 设置

Query 参数

参数名	参数值	类型	说明
✓ username	= tangjie	string	更多
✓ password	[REDACTED]	string	更多
✓ port	= 22	string	更多
✓ host	= 192.168.163.129	string	更多

添加参数

Messages Header 3 Cookie 实际请求

请求 URL:

ws://localhost:8080/ssh?
username=tangjie&password=[REDACTED]&port=22&host=192.168.163.129

Header:

Pretty Auto utf8

```
1 {"Code": "200",  
  "cmdResponseData": "(base)  
  \\u001b]0;tangjie@ubuntu:  
  ~\\u0007tangjie@ubuntu:~$  
  11",  
  "connectID": "co0f09mpnb96s03  
  r10kg", "message": ""}  
2
```

在ApiFox中有ws关键字用于websocket请求。

在后端代码中，我会根据你传递的虚拟机信息先进行连接测试（看看能否正确连接连接虚拟机）

```

func sshConnect(w http.ResponseWriter, r *http.Request) {
    1 usage

    host := r.URL.Query().Get(key: "host")
    port := r.URL.Query().Get(key: "port")
    password := r.URL.Query().Get(key: "password")
    username := r.URL.Query().Get(key: "username")

    vmInfo := model.VMConnectRequest{
        Host:    host,
        Port:    port,
        Username: username,
        Password: password,
    }

    config := &ssh.ClientConfig{
        User: vmInfo.Username,
        Auth: []ssh.AuthMethod{
            ssh.Password(vmInfo.Password),
        },
        HostKeyCallback: ssh.InsecureIgnoreHostKey(),
    }

    //测试连接
    flag, connectID := utils.VerifyConnect(&vmInfo)
    if flag != true {
        //todo 返回错误码
        log.Fatalln(v... "001测试连接失败")
        return
    }
}

```

如果不能就直接退出程序，如果可以连接就会响应成功（code：200）

要注意，在建立完websocket连接后，后端会将终端的初始信息 发送到websocket中

The screenshot shows a web browser's developer console with the 'Messages' tab selected. It displays a successful connection to 'ws://localhost:8080/api/terminal...' at 12:51:36. Below this, a message is shown with a red arrow pointing to it, indicating it was automatically sent. The message is a JSON object: `{ "Code": "200", "connectID": "co0g5lupnb96i..." }`. To the right, the terminal output is visible, showing the Ubuntu 20.04.6 LTS welcome message and system information.

上图 只有一个下箭头表示这个数据是自动发送的。

Messages

Header 3

Cookie

实际

...

已连接到 ws://localhost:8080/api/terminal... 12:51:36

↓ {"Code": "200", "connectID": "co0g5lupnb96i... 12:51:36

↑ { "data": " " } 12:53:14

↓ {"Code": "200", "connectID": "co0g5lupnb96i... 12:53:14

Pretty Auto utf8

1 {"Code": "200", "connectID": "co0g5lupnb96i34tvtr0", "message": "", "resData": "(base \u001b)0;tangjie@ubuntu: ~\u0007tangjie@ubuntu:~\$"}
2

下图展示了发送命令