Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 4.1

по курсу «Компьютерные сети»

«Разработка SSH-сервера и SSH-клиента»

Студент группы ИУ9-32Б Тараканов В. Д..

Преподаватель Посевин Д. П.

Москва 2024

1 Задание

Часть 1 Задача: Реализовать ssh сервер на языке GO с применением указанных пакетов и запустить его на localhost. Проверка работы должна проводиться путем использования программы ssh в ОС Linux/Unix или PuTTY в ОС Windows. Должны работать следующие функции:

- авторизация клиента на ssh сервере;
- создание директории на удаленном сервере;
- удаление директории на удаленном сервере;
- вывод содержимого директории;
- перемещение файлов из одной директории в другую;
- удаление файла по имени;
- вызов внешних приложений, например ping.

Часть 2

Задача 1: Реализовать ssh-клиент и запустить его на localhost.

Задача 2: Протестировать соединение Go SSH-клиента к серверу реализованному в предыдущей задаче, а также к произвольному ssh серверу.

2 Результаты

Исходный код программы представлен в листингах 1- 2.

Листинг 1: sshClient.go

```
1  package main
2
3  import (
4  "fmt"
5  "golang.org/x/crypto/ssh"
6  "log"
7  "os"
8  "strings"
9  )
10
11 const (
12 sshUser       = "root"
13 sshPassword   = "1234"
14 sshServerHost = "185.104.251.226:9456"
15 )
```

```
16
17  func createSSHClient(user, password, host string) (*ssh.Client, error)
      {
18    config := &ssh.ClientConfig{
19      User: user,
20      Auth: []ssh.AuthMethod{
21        ssh.Password(password),
22      },
23      HostKeyCallback: ssh.InsecureIgnoreHostKey(),
24    }
25    client, err := ssh.Dial("tcp", host, config)
26    if err != nil {
27      return nil, fmt.Errorf(no connection to server: %v", err)
28    }
29    return client, nil
30  }
31
32  func runCommand(client *ssh.Client, command string) (string, error) {
33    session, err := client.NewSession()
34    if err != nil {
35      return "", fmt.Errorf(can't make a session %v", err)
36    }
37    defer session.Close()
38
39    var outputBuf strings.Builder
40    session.Stdout = &outputBuf
41    session.Stderr = &outputBuf
42
43    if err = session.Run(command); err != nil {
44      return "", fmt.Errorf("error while doing command %v", err)
45    }
46
47    return outputBuf.String(), nil
48  }
49
50  func main() {
51    if len(os.Args) < 2 {
52      fmt.Println("Use: go run ssh_client.go <command>")
53      os.Exit(1)
54    }
55    command := strings.Join(os.Args[1:], " ")
56
57    client, err := createSSHClient(sshUser, sshPassword, sshServerHost)
58    if err != nil {
59      log.Fatalf("Error: %v", err)
60    }
```

```
61      defer client.Close()
62
63      fmt.Printf(Do command: %s\n", command)
64      output, err := runCommand(client, command)
65      if err != nil {
66        log.Fatalf("error while doing command: %v", err)
67      }
68
69      fmt.Println("result of doing command:")
70      fmt.Println(output)
71    }
72
```

Листинг 2: sshServer.go

```
1   package main
2
3   import (
4   "fmt"
5   "github.com/gliderlabs/ssh"
6   "io"
7   "log"
8   "os/exec"
9   "strings"
10  "sync"
11  )
12
13  var (
14  users      = map[string]string{"root": "1234"}
15  usersMutex sync.Mutex
16  )
17
18  func passwordAuth(ctx ssh.Context, password string) bool {
19    usersMutex.Lock()
20    defer usersMutex.Unlock()
21
22    if pass, ok := users[ctx.User()]; ok && pass == password {
23      return true
24    }
25    return false
26  }
27
28  func handleAddUser(args []string, session ssh.Session) {
29    if session.User() != "root" {
30      io.WriteString(session, "Error: only root can add new user\n")
31      return
32    }
```

```
33      if len(args) < 2 {
34        io.WriteString(session, "Error: you must type in login and
      password \n")
35        return
36      }
37
38      username := args[0]
39      password := args[1]
40
41      usersMutex.Lock()
42      defer usersMutex.Unlock()
43
44      if _, exists := users[username]; exists {
45        io.WriteString(session, "Error: user already exist\n")
46      } else {
47        users[username] = password
48        io.WriteString(session, fmt.Sprintf("User %s added successfully\n
      ", username))
49      }
50    }
51
52    func main() {
53      server := ssh.Server{
54        Addr:            ":9456",
55        PasswordHandler: passwordAuth,
56      }
57      server.Handler = func(s ssh.Session) {
58        command := s.RawCommand()
59        args := strings.Split(command, " ")
60
61        if len(args) > 0 && args[0] == "adduser" {
62          handleAddUser(args[1:], s)
63        } else {
64          cmd := exec.Command(args[0], args[1:]...)
65          stdout, err := cmd.Output()
66          if err != nil {
67            io.WriteString(s, fmt.Sprintf("ERROR OCCURED: %s\n", err.Error
      ()))
68          } else {
69            io.WriteString(s, fmt.Sprintf("%s\n", string(stdout)))
70          }
71        }
72      }
73
74      log.Println("Launch ssh server on port 9456...")
75      log.Fatal(server.ListenAndServe())
```

```
76      }
77
```