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

по курсу «Компьютерные системы и сети»

«Прокси сервер HTTP»

Студент группы ИУ9-32Б Федуков А. А.

Преподаватель Посевин Д. П.

15 октября 2024 г.

Цель работы

Целью данной работы является создание проки сервера на языке Go.

Задание

Заблокировать ресурсы локально:

- http://www.gnuplot.info/
- https://netlib.sandia.gov/
- https://putty.org/
- https://www.openbsd.org/
- https://netbsd.org/
- https://www.freebsd.org/

После чего при помощи созданного прокси сервера получить доступ к сайтам.

Реализация

Я заблокировал ресурсы перопределив адреса через /etc/hosts на localhost, после чего создал несколько файлов: download.go (для скачивания сайта); server.go (для взаимодействия с http клиентами); parse.go (для находения ссылок и переопределения их запроса на прокси сервер); main.go (для инициализации значений и запуска функций), а также файл starts.sh для сборки и запуска программы.

Код

Листинг 1: Файл download.go

```
1 package main
2
3 import (
4   "log"
```

```
5    "net/http"
6    "os"
7  )
8
9
10 func downloadURL(url string) string{
11
12   // Make the GET request
13   resp, err := http.Get(url)
14
15   log.Println("Downloading HTML")
16   if err != nil{
17     msg := "Cannot download html of " + url
18     log.Println(msg)
19     log.Println("ERROR: \n  ", err)
20     return msg
21   }
22   defer resp.Body.Close()
23
24   os.RemoveAll("site")
25   os.Mkdir("site", 0755)
26
27   return parsedHTML(resp.Body, url)
28 }
```

Листинг 2: Файл server.go

```
1  package main
2
3  import (
4    "fmt"
5    "log"
6    "net/http"
7  )
8
9  func handler(w http.ResponseWriter, r *http.Request) {
10   // Get full address of request
11   scheme := "http"
12   if r.TLS != nil {
13     scheme = "https"
14   }
15   fullAddress := fmt.Sprintf("%s://%s%s", scheme, r.Host, r.RequestURI)
16   log.Println("Handle link:", fullAddress)
17
18   // Have to have url query parameter
19   url := r.URL.Query().Get("url")
20   if url == "" {
```

```
21      http.Error(w, fmt.Sprintf("Wrong adress: %s. \nRewrite it like
        example: http://%s?url=%s", r.URL, SERVER, "http://gnuplot.info/"),
        http.StatusBadRequest)
22    } else {
23      w.Header().Set("Content-Type", "text/html")
24      html := downloadURL(url)
25      fmt.Fprint(w, html)
26    }
27    fmt.Println()
28  }
29
30  func startServer() {
31    http.HandleFunc("/", handler)
32    http.Handle("/site/", http.StripPrefix("/site/", http.FileServer(http.
        Dir("./site"))))
33    log.Println("Server started: http://" + SERVER + "?url=")
34    log.Fatal(http.ListenAndServe(SERVER, nil))
35  }
```

Листинг 3: Файл parse.go

```
1   package main
2
3   import (
4     "fmt"
5     "io"
6     "log"
7     "net/url"
8     "path/filepath"
9
10    "github.com/PuerkitoBio/goquery"
11    "github.com/cavaliergopher/grab/v3"
12  )
13
14  // changeURL add "link" to "attrLink" via SERVER proxy if needed
15  func changeURL(link string, attrLink string) string {
16    parsed_attrLink, err := url.Parse(attrLink)
17    if err != nil {
18      log.Fatalf("Failed to parse URL: %s\n", err)
19      return ""
20    }
21
22    parsed_link, err := url.Parse(link)
23    if err != nil {
24      log.Fatalf("Failed to parse URL: %s\n", err)
25      return ""
26    }
```

```
27
28    // Set file location
29    d := filepath.Dir(parsed_link.Path)
30    if d[len(d)-1] != '/' {
31      d += "/"
32    }
33    parsed_link.Path = d
34
35    // Amend local paths
36    if parsed_attrLink.IsAbs() {
37      return fmt.Sprintf("http://%s?url=%s", SERVER, attrLink)
38    } else {
39      return fmt.Sprintf("http://%s?url=%s", SERVER, parsed_link.String()+
       attrLink)
40    }
41  }
42
43  func makeLinkAbsolute(link string, server string) string {
44    parsed_link, err := url.Parse(link)
45    if err != nil {
46      log.Fatalf("Failed to parse URL: %s\n", err)
47      return ""
48    }
49
50    server_link, err := url.Parse(server)
51    if err != nil {
52      log.Fatalf("Failed to parse URL: %s\n", err)
53      return ""
54    }
55
56    server_link.Path = ""
57
58    // Amend local paths
59    if parsed_link.IsAbs() {
60      return link
61    } else {
62      return fmt.Sprintf(server_link.String() + "/" + link)
63    }
64  }
65
66  // source and link from [http.Get](link)
67  func parsedHTML(source io.Reader, link string) string {
68
69    // Load the HTML document using goquery
70    doc, err := goquery.NewDocumentFromReader(source)
71    if err != nil {
```

```
72      log.Fatal("no gq\n", err)
73      return ""
74    }
75
76    // Parse document
77    doc.Find("iframe").Each(func(i int, s *goquery.Selection) {
78      param := "src"
79      v, exists := s.Attr(param)
80      if exists {
81        s.SetAttr(param, changeURL(link, v))
82      }
83    })
84
85    doc.Find("img").Each(func(i int, s *goquery.Selection) {
86      param := "src"
87      v, exists := s.Attr(param)
88      if exists {
89        aboba := makeLinkAbsolute(v, link)
90        fmt.Println(aboba)
91        resp, err := grab.Get("./site/", aboba)
92        if err != nil {
93          log.Println(err)
94        }
95        fmt.Println("Download saved to", resp.Filename)
96
97        s.SetAttr(param, fmt.Sprintf("http://%s/%s", SERVER, resp.Filename
      ))
98      }
99    })
100
101   doc.Find("link").Each(func(i int, s *goquery.Selection) {
102     param := "href"
103     v, exists := s.Attr(param)
104     if exists {
105       resp, err := grab.Get("./site/", makeLinkAbsolute(v, link))
106       if err != nil {
107         log.Println(err)
108       }
109       fmt.Println("Download saved to", resp.Filename)
110
111       s.SetAttr(param, fmt.Sprintf("http://%s/%s", SERVER, resp.Filename
      ))
112     }
113   })
114
115   doc.Find("a").Each(func(i int, s *goquery.Selection) {
```

```
116       param := "href"
117       v, exists := s.Attr(param)
118       if exists {
119         s.SetAttr(param, changeURL(link, v))
120       }
121     })
122
123     // Convert the modified document back to HTML
124     html, err := doc.Html()
125     if err != nil {
126       log.Fatal("no inj", err)
127       return ""
128     }
129     return html
130 }
```

Листинг 4: Файл main.go

```
1  package main
2  import "os"
3  /* Example urls
4  http://127.0.0.1:1818/?url=http://gnuplot.info
5
6  http://gnuplot.info/
7  https://netlib.sandia.gov/
8  https://putty.org/
9  https://www.openbsd.org/
10 https://netbsd.org/
11 https://www.freebsd.org/
12 */
13
14 var SERVER_ADDRESS = os.Getenv("MyIP")
15
16 var SERVER_PORT = "1819"
17 var SERVER = SERVER_ADDRESS + ":" + SERVER_PORT
18
19 func main() {
20   startServer()
21 }
```

Листинг 5: Файл start.sh

```
1 #!/bin/bash
2
3 MyIP=$(hostname -I | awk '{ print $1 }' | tr -d '[:space:]')
4 export MyIP
5
```

```
6  echo "Building..."
7  go build -o fedukov_lab4
8  echo "Running..."
9  killall fedukov_lab4
10 ./fedukov_lab4
```

Вывод программы

После загрузки программы на сервер, переходя по http://{SERVER_IP}:1818/?url=http://gnuplot.info (передавая query параметр url как целевой адрес для проксирования) можно было увидить целостный сайт. После получения http запроса на сервере скачивался сайт по адресу url в папку site, после чего он обрабатывался и передавался клиенту, сделавшему запрос.

Вывод

В этот лабораторной я научился на Go скачивать сайты и парсить их, попутно меняя интересуемый меня контент.