Universidade de Brasília

Departamento de Ciência de Computação - CIC

Servidor Proxy

Autores:

Douglas Lustosa da Silva 15/0058683

Brasília
24 de Junho de 2019

Conteúdo

1 Resumo

Este trabalho consiste no desenvolvimento de um servidor proxy com filtro de conteúdo. Para isso, o servidor receberá requisições HTTP e verificará se o endereço de destino, presente no cabeçalho do protocolo, é valido.

A validação é dada através de uma lista de sites, se o site tiver presente na lista o servidor proxy negará o aceso.

2 Introdução

Em rede de computadores, um servidor proxy serve como um intermediário entre a comunicação de um usuário e um servidor final. O usuário se conecta ao servidor proxy requisitando algum tipo de serviço então o servidor proxy analisa a requisão do usuário e decide se tal requisição é valida ou não.

Como foi mencionado na descrição do trabalho disponibilizado no moodle, o servidor proxy pode alterar a resposta do servidor ou a requisição do cliente, se configurado para isso. Também pode agir como filtro, uso mais comum para o servidor proxy, controlando o fluxo de dados na rede.

Como foi mencionado em 1, o servidor Proxy implementado receberá requisições HTTP. O protocolo HTTP (Hypertext Transfer Protocol) é um protocolo da camada de aplicação utilizado para sistemas de informação hipertexto (texto estruturado que utiliza hiperlinks) sendo considerado o protocolo base para a comunicação de dados na world wide web.Para a utilização do protocolo HTTP é utilizado a porta 80, havendo comunicação na linguagem HTML.

2.1 HTTP

HTTP funciona como um protocolo de requisição e resposta. No caso deste trabalho o browser será oo cliente que requisita aplicações web para um servidor final. A respostas dadas por esse servidor incluirão informações de estado e o conteúdo solicitado, ou seja, o HTTP será uma sequência de transações de rede de requisição-resposta. Onde, um cliente HTTP inicia a comuncação estabelecendo uma, conexão TCP para a porta 80 (porta geralmente utilizada para o HTTP), o servidor HTTP que estava ouvindo naquela porta recebe a requisição e retorna com uma mensagem de estado (por exemplo, 200 OK), além da mensagem requerida.

Para a requisição o cliente deve utilizar um dos oito métodos (GET, HEAD, POST, TRACE, OPTIONS, DELETE, PUT e CONNECT) que de-

terminarão a ação que o servidor deverá tomar. Em seguida o servidor encaminhará uma resposta de estado.

2.2 TCP

Como foi mencionado em 2.1 o protocolo HTTP utiliza o protocolo TCP (Transmission Control Protocol) para estabelecer a comunicação e haver troca de mensagens, o protocolo TCP é um dos protocolos mais importantes da camada de transporte. Através dele é permitido gerenciar os dados da camada inferior. Como o protocolo TCP é orientado a conexão ele controla o estado da transmissão de dados na comunicação entre máquinas.

Entre as características do protocolo TCP pode-se citar o fato de ser orientado a conexão, Handshake (estabelecimento de conexão de três tempos), confiabilidade, full duplex (transferência simultânea dos dois lados, tanto do cliente como também do servidor), controle de fluxo e de congestionamento.

No trabalho pode-se perceber o socket implementado utiliza o protocolo TCP para gerenciar o transporte de dados do usuário com o servidor proxy e do proxy para o servidor final.

3 Socket

o socket é o ponto final de um fluxo de comunicação entre processos, para a implementação dele foi utilizado a biblioteca socket, como pode ser verificado na imagem 1. Através da imagem 1 verifica-se que criou-se um socket com duas constantes, onde a primeira refere-se a familia do endereço, e é padrao. A segunda constante refere-se ao tipo de protocolo utilizado no transporte, protocolo TCP.

Depois dessa configuração colocou-se o servidor Proxy para ficar escutando na porta 8000 com IP padrão da máquina onde o código irá rodar. Inicialmente foi colocado para o servidor escutar 5 clientes por vez.

Foi adicionado uma excessão, onde quando não houver uma comuncação no socket implementado, o servidor Proxy deverá encerrar a comunicação.

4 Comunicação com o Usuário

Observando a imagem 2 verica-se como é feita a troca de mensagens entre o servidor proxy e o usuário. onde percebe-se que a variável $msg_conection$ receberá a página que o usuário quer acessar e a variável $end_cliente$ recebe o IP procurado pelo usuário e os encaminha para $solicitacao_cliente$, é onde

```
6
7   def inicio():
8       try:
9           server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
10          server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
11          server.bind(('',8000))
12          server.listen(5)
13          print "PRONTO PARA AS CONEXOES..."
14
15      except Exception, error:
16          print "\nNAO FOI POSSIVEL SE CONECTAR...\n"
17          #server.close()
18          sys.exit(3)
19
```

Figura 1: Implementação do Socket para a transmição de mensagens através do protocolo TCP

haverá o tratamento da informação adquirida do browser (url). Esse tratamento pode ser verificado na figura [ADICIONAR FIGURA], depois de feito o tratamento da url recebida é determinada a porta 80 para que haja troca de mensagens entre o servidor proxy e o servidor final.

```
20      while True:
21          try:
22              conexao, end_cliente = server.accept()
23              #print "Servidor conectado com o cliente", end_cliente
24
25              msg_conection = conexao.recv(4096)
26              #print "mensagem recebida: ", msg_conection
27              #msg_conection = msg_conection[:-1]
28              #if not msg_conection: break
29              thread.start_new_thread(solicitacao_cliente, (conexao, end_cliente, msg_conection))
30
31          except KeyboardInterrupt:
32              server.close()
33              print "\n Servidor finalizado..."
34              sys.exit(1)
35
```

Figura 2: Loop onde ocorre a troca de mensagens entre o servidor Proxy e o usuário

5 Servidor Proxy

Na figura 3 é mostrado como o servidor proxy realiza o encaminhamento da requisição do usuário para o servidor final, isso pode ser verificado na

linha 63 onde é utilizado o HTTPConection para acessar a página requerida.

A parte do whitelist seria implementada depois da linha 78 da figura 3, mas por problemas a serem discutidos em 6 não foi possível continuar o trabalho.

```
61  def servidor_Proxy(aux, pag_web, porta, conexao, end_cliente, msg_conection):
62      ip_host = socket.gethostbyname(pag_web)
63      conexao_site = httplib.HTTPConnection(pag_web)
64      conexao_site.request("GET", "/")
65      requisicao1 = conexao_site.getresponse()
66
67      print "\nHost: ", pag_web
68      print "\nHost_IP: ", ip_host
69      print "\nStatus: ", requisicao1.status, requisicao1.reason
70
71      conexao_site.close()
72
73      try:
74          server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
75          server.connect((pag_web, porta))
76          server.send(msg_conection)
77          while True:
78              msg = server.recv(4096)
79              if(len(msg) > 0):
80                  conexao.send(msg)
81                  conexao_server(msg)
82              else:
83                  break
84          server.close()
85          conexao.close()
86      except socket.error, (value, message):
87          server.close()
88          conexao.close()
89          sys.exit(2)
90
```

Figura 3: IP do servidor e do computador do usuário e o número das portas

6 Discursão sobre o Código

Muitas funcionalidades pedidas no roteiro do trabalho não puderam ser implementadas, principlamente pela comunicação lenta entre o proxy e o servidor final, pois apesar do status da comunicação estar OK, a demora fazia com que a página não fosse carregada. Os prints da linha 67 ate a linha 69 mostra a pagina requisitada, o ip so servidor e o status da conexão.

Para superar esses problemas de conexão foi proposto a implementação de um código que funcionaria como cliente, presente na pasta cliente do arquivo colocado no moodle, através dele é enviado a página que se deve ser acessada e um outro código (presente na mesma pasta) analisava a requisição e se conectava ao servidor final, mandando o status da conexão para o cliente.

Entretanto esse método não foi muito produtivo, pois o código do servidor apresentava erros desconhecidos.

7 Conclusão

Através do trabalho foi possível a implementação de um servido proxy simples, onde ele funciona com intermediário na comunicação entre um usuário e um servidor final, recebendo uma mensagem do usuario e depois reencaminhado para um servidor final.

8 Referência

- $https://pt.wikipedia.org/wiki/Transmission_Control_Protocol Funcionamento_do_protocolo$ (acessado em 23/06/19)
- $https://br.ccm.net/contents/284-o-protocolo-tcp$ (acessado em 22/06/19)
- roteiro do trabalho
- relatório do wireshark enviados ao moodle