
Projeto Final de Teleinformática e Redes 2

Implementação de uma aplicação de Proxy Server

Nome: Amanda Aline Figueiredo Carvalho - 15/0115997

Nome: Bruno Justino Garcia Praciano - 13/0006912

Nome: Bruno Viana de Siqueira - 14/0133151

04 de Dezembro de 2017

1 Resumo

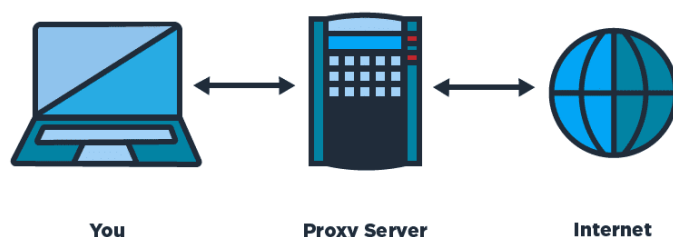
Esse trabalho tratará sobre o Proxy Server Web, TCP e o protocolo HTTP, bem como da aplicação da teoria aplicada em sala de aula.

2 Apresentação Teórica

2.1 Proxy Server Web

No contexto de redes de computadores, um servidor proxy é um servidor, seja ele um sistema de computador ou mesmo uma aplicação, que atua como intermediário para pedidos de clientes que buscam recursos de outros servidores. Um cliente se conecta ao servidor proxy, solicitando algum serviço, como um arquivo, conexão, página da Web ou outro recurso disponível de um servidor diferente e no momento em que o endereço de um site é digitado no navegador, uma solicitação é enviada ao proxy, que então realiza esta solicitação ao servidor no qual o site é hospedado, devolvendo para o cliente o resultado.

O proxy foi inventado para adicionar estrutura e encapsulamento a sistemas distribuídos. Hoje, a maioria dos proxy são web, e entre suas características estão o fácil acesso ao conteúdo da internet, proporcionar anonimato durante a navegação e possibilidade de ignorar o bloqueio de endereço IP. O aumento constante do uso de serviços web, como o acesso a sites, sistemas e diversos outros aplicativos que operam online, podem gerar contratempos ao mesmo tempo que trazem comodidade e agilidade, como questões de segurança. A utilização de servidores proxy surgiu como opção de ferramenta de segurança e controle, inclusive de acesso a páginas indesejadas.



2.2 Protocolo HTTP

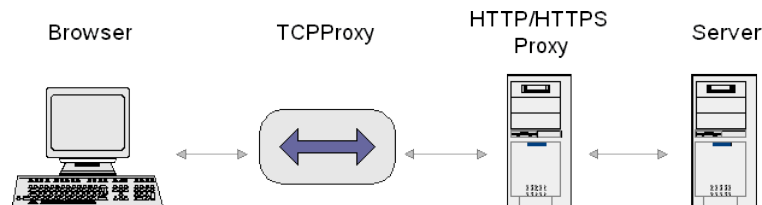
O servidor proxy faz o intermédio da conexão entre o dispositivo e o cliente, e por conta disso, tem controle total sobre o tráfego em si, podendo permitir ou bloquear de acordo com a arquitetura construída, conforme será descrito no desenvolvimento. Muitas configurações podem ser atribuídas ao proxy, fazendo com que ele permita e restrinja URLs de acordo com o interesse. O proxy web

permite controlar os serviços acessados na internet através do protocolo HTTP, sendo responsável pela gestão de acesso a sites e outras aplicações baseadas no protocolo.

Em suas requisições, o cliente envia um cabeçalho que faz a identificação dos serviços utilizados e outras informações.

```
HTTP/1.1 200 OK
Date: Mon, 31 Mar 2014 16:55:30 GMT
Server: Apache
Accept-Ranges: bytes
X-Mod-Pagespeed: 1.6.29.7-3566
Cache-Control: max-age=0, no-cache, must-revalidate
Vary: Accept-Encoding, Cookie
X-Node: askapacherackweb0
X-UA-Compatible: IE=Edge,chrome=1
Content-Length: 55069
Keep-Alive: timeout=4, max=46
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
Content-Encoding: gzip
Content-Language: en
```

O Web Proxy Caching atua no armazenamento local de páginas da internet e arquivos disponíveis em servidores remotos, tendo em vista que busca objetos primeiramente no sistema de armazenamento local, seja a memória ou disco. Assim, ao receber a solicitação, o servidor acessa o conteúdo e o armazena para utilização futura. Em um próximo acesso, o proxy responde com o conteúdo armazenado, aumentando a velocidade de resposta, uma vez que não se consome banda da internet. Ao implementar um proxy cache é possível salvar em diretórios as URLs já carregadas pelo menos uma vez, mas não por tempo indefinido, pois age de acordo com a capacidade de armazenamento da máquina. Abaixo está o exemplo de funcionamento de um proxy HTTP

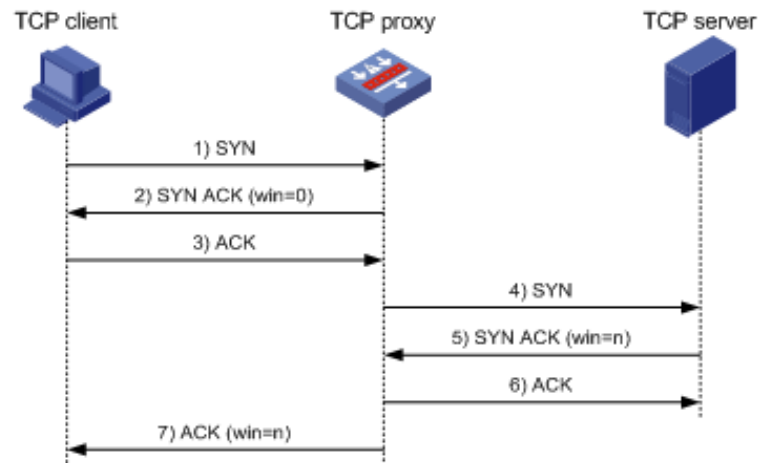


É uma característica do HTTP utilizar os serviços disponibilizados pelo TCP, pois geralmente os servidores HTTP usam a porta 80 do TCP. O HTTP/1.0 é diferente do HTTP/1.1, pois este último utiliza conexões TCP persistentes. Graças ao uso de um proxy, é possível filtrar as conexões à Internet analisando, tanto os pedidos do cliente, quanto as respostas do servidor.

2.3 TCP

O TCP é um protocolo da camada de transporte do Modelo OSI e atua na maioria das aplicações da internet, como o SSH, FTP, HTTP, portanto, a World Wide Web. O Protocolo de controle de transmissão provê confiabilidade, entrega na sequência correta e verificação de erros em pacotes de dados, entre os diferentes nós da rede, para a camada de aplicação.

Ocorre troca de pacotes entre cliente e servidor. Assim, são estabelecidos parâmetros para a conexão. Este fenômeno ocorre por meio do handshake. O handshake começa quando o cliente envia um segmento de sincronização que contém um número inicial de sequência, ao servidor. O servidor, ao receber o segmento de sincronização, envia uma mensagem de aceite da conexão (ACK). Em complemento, é enviado o número inicial de sequência do servidor. Por fim, o cliente confirma o recebimento do aceite enviando um segmento de ACK.



3 Desenvolvimento

3.1 Arquitetura do sistema desenvolvido

O sistema proxy foi desenvolvido na linguagem C. O proxy web caching está atuando conforme o esperado, assim como a filtragem de termos. O arquivo se chama proxy cache.c e contém todas as funções utilizadas no trabalho. Os requisitos do trabalho se encontram nas funções desse arquivo C. A pasta contém:

- Um arquivo C, com todas as funções implementadas documentadas.
- Um README contendo as informações de compilação e execução
- Diretório contendo os sites carregados em cache.
- Arquivos txt para filtragem dos domínios permitidos e proibidos, além de termos não permitidos.

3.2 Documentação do código

A documentação das funções está no próprio arquivo C, com uma descrição breve do funcionamento de cada função.

3.3 Implementação do Funcionamento Básico

A arquitetura do proxy foi pensada da seguinte maneira: o proxy recebe uma solicitação de um host, busca as informações necessárias na requisição, como cabeçalho, path e IP. Após, o cabeçalho do início da requisição passará a ser o proxy no host de origem. Associado a isso, há também o cache, que será abordado nas subseções a seguir.

O código foi fatiado em funções num mesmo arquivo, porém cada uma delas possui uma participação específica na execução do proxy.

- main - função principal, que chama os argumentos principais de funcionamento básico.
- verifyWhiteAndBlackList - Retorna um inteiro indicando se está na blacklist (-1), se está na whitelist(1), ou em nenhuma outra lista (0). O Buffer da dados será comparado ao arquivo de blacklst e whitelist.
- verifyDenyTerm - Retorna um inteiro indicando se o buffer contém um deny term. O Buffer da dados será comparado ao arquivo de deny terms.

- `*get in addr` - Função de passagem de endereços na socket.
- `directory` - Essa função cria os diretórios para o proxy cache. Optou-se por fazer dessa forma ao invés de salvar apenas as URLs em txt para efeitos de controle se a página já foi ou não acessada e se está gravada em memória, e verificação dos cabeçalhos. A saída fornece uma flag se a página está em cache (já foi acessada) ou não (primeira vez de acesso).
- `connect host` - Responsável por fazer a conexão com o host, setando seus parâmetros.
- `cache` - Carrega o buffer de armazenamento temporariamente para o cache.
- `request` - trabalha nas requisições de servidor, armazenamento em buffer pro cache de acordo com os requisitos de rede e filtragem.
- `start server` - Responsável por iniciar o socket. Retorna o inteiro retornado pela função `accept`. `sockfd` será iniciado pela função.
- `send file` - Passagem de arquivos para montagem de novo cabeçalho.
- `set server` - Configura o servidor chamando as funções declaradas anteriormente, bem como algumas da biblioteca `socket.h`. Utiliza como base `Socket API`.
- `deniedLogWrite` - Cria e escreve um arquivo de log com os denied terms. Denied term será escrito no arquivo de log.
- `blackLogWrite` - Cria e escreve um arquivo de log com as url blacklist. Buffer contém a URL da blacklist.
- `whiteLogWrite` - Cria e escreve um arquivo de log com as url da white list. Buffer contém a URL da whitelist.

3.4 Filtragem das requisições

A implementação foi feita baseado na premissa de que se o proxy da página acessada estiver na Whitelist ou na Blacklist, terá que verificar um terceiro termo: o deny term. Caso o domínio não estiver entre nenhuma destas listas, Black ou White, é necessário checar se esse termo pode ser acessado ou não, em `DenyTerms.txt`. Logs são emitidos para acompanhar o funcionamento.

3.5 Caching

A cache é o ambiente de armazenamento de páginas HTTP que são usadas rotineiramente. Essa técnica serve para otimizar o tempo de acesso e consumo de banda larga. As funções utilizadas foram citadas anteriormente em "Funcionamento Básico".

4 Conclusão

Esse trabalho nos permitiu aprofundar conhecimento sobre aplicações do que era visto apenas em sala de aula, de forma teórica. Foi possível ver o funcionamento dos protocolos e suas funções, como é o comportamento e importância de cada camada organizada. As maiores dificuldades enfrentadas estavam associadas à manipulação de algumas bibliotecas e desenvolver funções de filtragem. Por meio desse trabalho, foi possível colocar um proxy web para funcionar e entender melhor o funcionamento da Web, e protocolos HTTP e TCP.

5 Simulações

Foram feitas simulações utilizando o Wireshark e também capturas de tela a partir da execução do código em localhost. A seguir estão exemplos de simulações feitas e uma breve descrição de sua funcionalidade.

1) Simulação no Wireshark após acionarmos a Black List.

8	1.626571912	2804:d59:22bd:0:3bc2:9d36:7f61:b4cf	2800:3f0:4001:808::...	TCP	86	36834	.. 443 [ACK] Seq=251 Ack=47 Win=2333 Len=0 TSval=2569434741 TSecr=429183...
9	1.638040314	2800:3f0:4001:808::2003	2804:d59:22bd:0:3bc2:9d36:7f61:b4cf	TLSv1.2	155	Application Data	
10	1.638060420	2804:d59:22bd:0:3bc2:9d36:7f61:b4cf	2800:3f0:4001:808::...	TCP	86	36834	.. 443 [ACK] Seq=251 Ack=116 Win=2333 Len=0 TSval=2569434795 TSecr=42918...
11	1.638078105	2800:3f0:4001:808::2003	2804:d59:22bd:0:3bc2:9d36:7f61:b4cf	TLSv1.2	132	Application Data	
12	1.638086088	2804:d59:22bd:0:3bc2:9d36:7f61:b4cf	2800:3f0:4001:808::...	TCP	86	36834	.. 443 [ACK] Seq=251 Ack=162 Win=2333 Len=0 TSval=2569434795 TSecr=42918...
13	1.638954878	2804:d59:22bd:0:3bc2:9d36:7f61:b4cf	2800:3f0:4001:808::...	TLSv1.2	132	Application Data	
14	1.725670396	2800:3f0:4001:808::2003	2804:d59:22bd:0:3bc2:9d36:7f61:b4cf	TCP	86	443	.. 36834 [ACK] Seq=162 Ack=297 Win=472 Len=0 TSval=4291833052 TSecr=256943...
15	1.745650133	192.168.1.11	194.244.42.72	TCP	66	54864	.. 443 [ACK] Seq=1 Ack=1 Win=364 Len=0 TSval=2604317446 TSecr=3404853334
16	1.924496896	104.208.165.109	192.168.1.11	TCP	60	1199	ACKed unseen segment 443 .. 54864 [ACK] Seq=1 Ack=2 Win=195 Len=0 TSval=...
17	1.945620042	SamsungE_66:32:41	Broadcast	ARP	60	Who has 192.168.1.1? Tell 192.168.1.3	
18	3.278682441	104.208.165.109	192.168.1.11	TCP	54	443	.. 50154 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
19	3.993726312	SamsungE_66:32:41	Broadcast	ARP	60	Who has 192.168.1.1? Tell 192.168.1.3	
20	6.653239079	fe80::a28e:78ff:fed1:5bcb	fe80::a28e:78ff:fed1:5bcb	ICMPv6	86	Neighbor Solicitation for fe80::a28e:78ff:fed1:5bcb from 5c:c9:d3:53:35:74	
21	6.657317795	fe80::a28e:78ff:fed1:5bcb	fe80::a679:21aa:2b3...	ICMPv6	78	Neighbor Advertisement fe80::a28e:78ff:fed1:5bcb (rtr, sol)	
22	6.866568411	192.168.1.11	23.215.123.95	TCP	66	38898	.. 443 [ACK] Seq=1 Ack=1 Win=364 Len=0 TSval=2800239578 TSecr=3354709094
23	6.904809926	23.215.123.95	192.168.1.11	TCP	66	1199	ACKed unseen segment 443 .. 38898 [ACK] Seq=1 Ack=2 Win=980 Len=0 TSval=...
24	6.944232776	2804:d59:22bd:0:3bc2:9d36:7f61:b4cf	2800:3f0:4001:808::...	TLSv1.2	347	Application Data	
25	6.989990149	2800:3f0:4001:808::2003	2804:d59:22bd:0:3bc2:9d36:7f61:b4cf	TCP	86	443	.. 36834 [ACK] Seq=162 Ack=558 Win=479 Len=0 TSval=4291838356 TSecr=256944...
26	6.990863489	192.168.1.11	77.78.109.83	TCP	66	56474	.. 80 [FIN, ACK] Seq=1 Ack=1 Win=229 Len=0 TSval=1171786713 TSecr=409499...
27	6.990867643	192.168.1.11	77.78.109.83	TCP	66	56472	.. 80 [FIN, ACK] Seq=1 Ack=1 Win=296 Len=0 TSval=1171786713 TSecr=344004...
28	7.044462130	2800:3f0:4001:808::2003	2804:d59:22bd:0:3bc2:9d36:7f61:b4cf	TLSv1.2	155	Application Data	
29	7.045250947	2800:3f0:4001:808::2003	2804:d59:22bd:0:3bc2:9d36:7f61:b4cf	TLSv1.2	124	Application Data	
30	7.045303425	2800:3f0:4001:808::2003	2804:d59:22bd:0:3bc2:9d36:7f61:b4cf	TLSv1.2	132	Application Data	
31	7.045552705	2804:d59:22bd:0:3bc2:9d36:7f61:b4cf	2800:3f0:4001:808::...	TCP	86	36834	.. 443 [ACK] Seq=558 Ack=315 Win=2333 Len=0 TSval=2569440202 TSecr=42918...
32	7.046171945	2804:d59:22bd:0:3bc2:9d36:7f61:b4cf	2800:3f0:4001:808::...	TLSv1.2	132	Application Data	
33	7.091192796	2800:3f0:4001:808::2003	2804:d59:22bd:0:3bc2:9d36:7f61:b4cf	TCP	86	443	.. 36834 [ACK] Seq=315 Ack=604 Win=479 Len=0 TSval=4291838456 TSecr=256944...
34	7.266576606	77.78.109.83	192.168.1.11	TCP	66	80	.. 56472 [ACK] Seq=1 Ack=2 Win=8279 Len=0 TSval=3440078596 TSecr=1171786713
35	7.269390777	77.78.109.83	192.168.1.11	TCP	66	80	.. 56474 [ACK] Seq=1 Ack=2 Win=8279 Len=0 TSval=4095017463 TSecr=1171786713
36	8.990248256	SamsungE_66:32:41	Broadcast	ARP	60	Who has 192.168.1.1? Tell 192.168.1.3	
▶ Frame 19: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0							
▶ Ethernet II, Src: SamsungE_66:32:41 (bc:8c:cd:66:32:41), Dst: Broadcast (ff:ff:ff:ff:ff:ff)							
▶ Address Resolution Protocol (request)							

0000	ff ff ff ff ff ff bc 8c cd 66 32 41 08 06 00 01f2A....
0010	08 00 06 04 00 01 bc 8c cd 66 32 41 c9 a8 01 03f2A....
0020	00 00 00 00 00 00 c9 a8 01 01 00 00 00 00 00 00

2) Simulação no terminal do Linux após acionarmos a Black List.

```
[bruno@reborn src]$ ./a.out 10000
Server Start Running.....
Connected client:
server:got connection from 127.0.0.1
Term (blacklist): www.facebook.com.br
```

3) Simulação no Wireshark após utilizarmos Deny Terms.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	SamsungE_66:32:41	Broadcast	ARP	60	Who has 192.168.1.1? Tell 192.168.1.3
2	0.133361196	192.168.1.11	192.168.1.1	DNS	72	Standard query 0x6003 A gatinhos.com
3	0.133382385	192.168.1.11	192.168.1.1	DNS	72	Standard query 0xdc0c AAAA gatinhos.com
4	0.227355716	192.168.1.1	192.168.1.11	DNS	88	Standard query response 0x6003 A gatinhos.com A 176.74.176.187
5	0.522020411	192.168.1.1	192.168.1.11	DNS	156	Standard query response 0xdc0c AAAA gatinhos.com SOA ns1.uniregistrymarket.link
6	0.522194669	192.168.1.11	176.74.176.187	TCP	74	33956 .. 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2132493058 TSecr=0 WS=128
7	0.676598703	SeikoEps_8c:c3:61	Broadcast	ARP	60	Gratuitous ARP for 192.168.1.5 (Request)
8	0.683736035	fe80::a28e:78ff:fed1:5bcb	2804:d59:22bd:0:3bc2:9d36:7f61:b4cf	ICMPv6	86	Neighbor Solicitation for 2804:d59:22bd:0:3bc2:9d36:7f61:b4cf from a8:0e:78:d1:5b:cb
9	0.683795917	2804:d59:22bd:0:3bc2:9d36:7f61:b4cf	fe80::a28e:78ff:fed1:5bcb	ICMPv6	78	Neighbor Advertisement 2804:d59:22bd:0:3bc2:9d36:7f61:b4cf (sol)
10	0.776241672	176.74.176.187	192.168.1.11	TCP	74	80 .. 33956 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1452 SACK_PERM=1 TSval=3701393493 TSecr=21324930...
11	0.776323295	192.168.1.11	176.74.176.187	TCP	66	33956 .. 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=2132493312 TSecr=3701393493
12	0.776817573	192.168.1.11	176.74.176.187	TCP	66	33956 .. 80 [FIN, ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=2132493313 TSecr=3701393493
13	1.030172143	176.74.176.187	192.168.1.11	HTTP	253	HTTP/1.0 400 Bad request (text/html)
14	1.030225802	192.168.1.11	176.74.176.187	TCP	54	33956 .. 80 [RST] Seq=2 Win=0 Len=0
15	1.398742386	192.168.1.11	216.58.222.70	TCP	66	44472 .. 443 [ACK] Seq=1 Ack=1 Win=340 Len=0 TSval=3646612066 TSecr=565627012
16	1.449073303	216.58.222.70	192.168.1.11	TCP	66	[TCP ACKed unseen segment] 443 .. 44472 [ACK] Seq=1 Ack=2 Win=234 Len=0 TSval=565673113 TSecr=3646611732
17	2.025582004	SamsungE_66:32:41	Broadcast	ARP	60	Who has 192.168.1.1? Tell 192.168.1.3
18	4.117126949	SamsungE_66:32:41	Broadcast	ARP	60	Who has 192.168.1.1? Tell 192.168.1.3
19	6.167816141	SamsungE_66:32:41	Broadcast	ARP	60	Who has 192.168.1.1? Tell 192.168.1.3
20	8.111770273	SamsungE_66:32:41	Broadcast	ARP	60	Who has 192.168.1.1? Tell 192.168.1.3
21	10.158942911	SamsungE_66:32:41	Broadcast	ARP	60	Who has 192.168.1.1? Tell 192.168.1.3
22	11.490464707	192.168.1.9	224.0.0.251	MDNS	112	Standard query 0x0000 PTR _sleep-proxy._udp.local, "QU" question OPT
23	11.492326400	fe80::1830:2c19:723...	ff02::fb	MDNS	132	Standard query 0x0000 PTR _sleep-proxy._udp.local, "QU" question OPT
24	12.206854191	SamsungE_66:32:41	Broadcast	ARP	60	Who has 192.168.1.1? Tell 192.168.1.3
25	12.309510585	fe80::1830:2c19:723...	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
26	12.310728095	192.168.1.9	224.0.0.251	IGMPv3	54	Membership Report / Join group 224.0.0.251 for any sources
27	12.412128799	192.168.1.9	224.0.0.251	MDNS	112	Standard query 0x0000 PTR _sleep-proxy._udp.local, "QM" question OPT
28	12.413026263	fe80::1830:2c19:723...	ff02::fb	MDNS	132	Standard query 0x0000 PTR _sleep-proxy._udp.local, "QM" question OPT
29	13.368108307	104.208.165.109	192.168.1.11	TCP	54	443 .. 49504 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
▶ Frame 6: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0						
▶ Ethernet II, Src: Palladiu_53:35:74 (5c:c9:d3:53:35:74), Dst: Sagemcom_d1:5b:cb (a8:0e:78:d1:5b:cb)						
▶ Internet Protocol Version 4, Src: 192.168.1.11, Dst: 176.74.176.187						
▶ Transmission Control Protocol, Src Port: 33956, Dst Port: 80, Seq: 0, Len: 0						

- 4) Simulação no terminal do Linux após utilizarmos Deny Terms.

```
Connected client:
server:got connection from 127.0.0.1
host = gatinhos.com
path = (null)
Port = 80
.....
From Host
Deny Term found: gatinhos
```

- 5) Simulação no Wireshark após utilizarmos a White List.

No.	Time	Source	Destination	Protocol	Length	Info
23	0.209585392	2800:3f0:4001:805::...	2804:d59:22bd:0:3bc...	TCP	86	443 → 38982 [ACK] Seq=1196 Ack=497 Win=118 Len=0 TSval=12414512 TSecr=1950691187
21	0.117023279	2804:d59:22bd:0:3bc...	2800:3f0:4001:805::...	TCP	86	38982 → 443 [ACK] Seq=451 Ack=1196 Win=337 Len=0 TSval=1950691187 TSecr=12414458
19	0.116996554	2804:d59:22bd:0:3bc...	2800:3f0:4001:805::...	TCP	86	38982 → 443 [ACK] Seq=451 Ack=1150 Win=337 Len=0 TSval=1950691187 TSecr=12414457
17	0.116957496	2804:d59:22bd:0:3bc...	2800:3f0:4001:805::...	TCP	86	38982 → 443 [ACK] Seq=451 Ack=1112 Win=337 Len=0 TSval=1950691187 TSecr=12414457
15	0.116137669	2804:d59:22bd:0:3bc...	2800:3f0:4001:805::...	TCP	86	38982 → 443 [ACK] Seq=451 Ack=1073 Win=337 Len=0 TSval=1950691186 TSecr=12414457
13	0.115270228	2804:d59:22bd:0:3bc...	2800:3f0:4001:805::...	TCP	86	38982 → 443 [ACK] Seq=451 Ack=403 Win=319 Len=0 TSval=1950691185 TSecr=12414457
11	0.087019158	2800:3f0:4001:805::...	2804:d59:22bd:0:3bc...	TCP	86	443 → 38980 [ACK] Seq=39 Ack=216 Win=114 Len=0 TSval=4224863871 TSecr=1950691070
10	0.062888069	2800:3f0:4001:805::...	2804:d59:22bd:0:3bc...	TCP	86	443 → 38982 [ACK] Seq=39 Ack=451 Win=118 Len=0 TSval=12414404 TSecr=1950691079
9	0.053591711	2804:d59:22bd:0:3bc...	2800:3f0:4001:805::...	TCP	86	38982 → 443 [ACK] Seq=451 Ack=39 Win=300 Len=0 TSval=1950691123 TSecr=12414395
7	0.046173333	2804:d59:22bd:0:3bc...	2800:3f0:4001:805::...	TCP	86	38980 → 443 [ACK] Seq=216 Ack=39 Win=300 Len=0 TSval=1950691116 TSecr=4224863868
16652	180.732985342	2a01:4f8:172:1d86::1	2804:d59:22bd:0:3bc...	HTTP	296	HTTP/1.1 200 OK (text/plain)
16649	180.459696963	2804:d59:22bd:0:3bc...	2a01:4f8:172:1d86::1	HTTP	185	GET /check_network_status.txt HTTP/1.1
16470	117.905934587	2801:84:0:2::10	2804:d59:22bd:0:3bc...	HTTP	544	HTTP/1.1 301 Moved Permanently (text/html)
16464	117.764150937	150.162.242.95	192.168.1.11	HTTP	112	HTTP/1.0 404 Not Found (text/html)
16459	117.706778867	150.162.242.95	192.168.1.11	HTTP	112	HTTP/1.0 404 Not Found (text/html)
16458	117.677463878	2804:d59:22bd:0:3bc...	2801:84:0:2::10	HTTP	157	GET /wp-content HTTP/1.1
16451	117.546127144	192.168.1.11	150.162.242.95	HTTP	127	GET /d HTTP/1.1
16448	117.504310442	192.168.1.11	150.162.242.95	HTTP	127	GET /d HTTP/1.1
16410	113.183270477	150.162.242.95	192.168.1.11	HTTP	112	HTTP/1.0 404 Not Found (text/html)
16406	113.159097506	150.162.242.95	192.168.1.11	HTTP	112	HTTP/1.0 404 Not Found (text/html)
16401	113.144531802	150.162.242.95	192.168.1.11	HTTP	112	HTTP/1.0 404 Not Found (text/html)
16396	113.142518261	150.162.242.95	192.168.1.11	HTTP	112	HTTP/1.0 404 Not Found (text/html)
16391	113.118552933	150.162.242.95	192.168.1.11	HTTP	112	HTTP/1.0 404 Not Found (text/html)
16386	113.070596360	150.162.242.95	192.168.1.11	HTTP	112	HTTP/1.0 404 Not Found (text/html)
16382	113.037294077	2801:84:0:2::10	2804:d59:22bd:0:3bc...	HTTP	531	HTTP/1.1 301 Moved Permanently (text/html) (text/html)
16378	113.026535582	150.162.242.95	192.168.1.11	HTTP	112	HTTP/1.0 404 Not Found (text/html)
16376	112.931128173	2801:84:0:2::10	2804:d59:22bd:0:3bc...	HTTP	535	HTTP/1.1 301 Moved Permanently (text/html)
16373	112.905911894	2801:84:0:2::10	2804:d59:22bd:0:3bc...	HTTP	544	HTTP/1.1 301 Moved Permanently (text/html)
16370	112.709581189	2804:d59:22bd:0:3bc...	2801:84:0:2::10	HTTP	148	GET /wp-content HTTP/1.1
▶ Frame 11496: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0						
▶ Ethernet II, Src: Palladiu 53:35:74 (5c:c9:d3:53:74), Dst: Sagemcom_d1:5b:cb (a0:8e:78:d1:5b:cb)						
▶ Internet Protocol Version 4, Src: 192.168.1.11, Dst: 184.196.240.62						
▶ Transmission Control Protocol, Src Port: 52632, Dst Port: 443, Seq: 6465, Ack: 1037910, Len: 0						
0000	a0 8e 78 d1 5b cb 5c c9	d3 53 35 74 08 00 45 00	...x[.\..SSt..E			
0010	00 34 52 78 40 00 40 06	cd 95 c9 a8 01 0b 68 c4	.4Rx0.0.h.			
0020	f0 3e cd 98 01 bb 24 04	c4 b2 c9 e1 4e 07 80 10	.>....S.N...			

- 6) Simulação no terminal do Linux após utilizarmos a White List.

```
Connected client:
server:got connection from 127.0.0.1
host = greattreasures.org
path = gt/index.html
Port = 80
.....
From Host
No deny terms found

GET /gt HTTP/1.1
Host: greattreasures.org
Connection: close
```