

Implementação da função de Rede DNS Firewall

Germano Soboroza¹, Yuri Oliveira Alves², Luhan Bavaresco³

¹Engenharia de Computação – Universidade Federal de Santa Maria (UFSM)

Abstract. *This work proposes to present an implementation of DNS Firewall network function, describing its operation, including a performance analysis on network packets.*

Resumo. *Este trabalho propõe apresentar a implementação da função de rede DNS Firewall, visando demonstrar seu funcionamento, incluindo uma análise de desempenho sobre os pacotes da rede.*

1. Introdução

O Domain Name System (DNS) Firewall é uma função de rede responsável principalmente pela segurança e controle de acesso de sites, além destas, ele também visa aumentar o desempenho, segurança e confiabilidade de um servidor, visto que pacotes indesejáveis são barrados.

Com a finalidade de demonstrar a implementação dessa função de rede, foi montado um cenário de testes, onde é simulada sua implementação. Para se aproximar de um cenário real, foi utilizada a ferramenta *packet sender*, responsável pela manipulação de pacotes de rede, o *software Wireshark* foi utilizado para análise do desempenho da função.

2. Domain Name System (DNS) Firewall

Um DNS firewall é uma solução de segurança de rede que impede que usuários e sistemas de rede se conectem a algum host indesejável, redirecionando as requisições do cliente. Ele funciona empregando políticas de bloqueio definidas pelo administrador do sistema, também pode fornecer informações sobre ameaças, ajudando a isolar dispositivos infectados até que uma ação seja tomada.

Um exemplo da aplicabilidade do emprego desta função, é evitar ataques, como por exemplo na figura 1, onde existe uma Botnet, que tenta realizar diversas requisições a um servidor, este que por possuir políticas de proteção a este tipo de ataque, os bloqueia, e alerta o administrador do sistema, para que sejam tomadas as devidas ações.

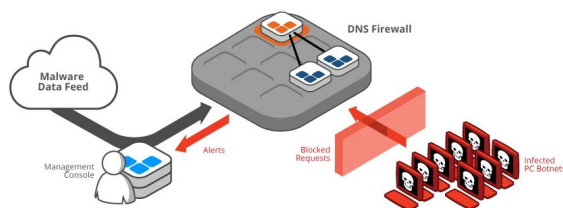


Figura 1. Visão conceitual - DNS Firewall.

3. Cenário de Testes

O cenário de testes é composto basicamente por três agentes, o cliente (VM 1), atacante (VM 2) e o roteador (Router), abaixo, nas figuras 2 e 3, respectivamente, são apresentadas as configurações de *hardware* das máquinas virtuais.

```
(germano@germano2)-[~]
$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:             Little Endian
Address sizes:          39 bits physical, 48 bits virtual
CPU(s):                 2
On-line CPU(s) list:    0,1
Thread(s) per core:     1
Core(s) per socket:     2
Socket(s):              1
NUMA node(s):           1
Vendor ID:              GenuineIntel
CPU family:             6
Model:                  142
Model name:             Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz
Stepping:                10
CPU MHz:                1992.000
BogoMIPS:               3984.00
Hypervisor vendor:      KVM
```

Figura 2. Configuração VM 1.

```
(germano@germano1)-[~]
$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:             Little Endian
Address sizes:          39 bits physical, 48 bits virtual
CPU(s):                 2
On-line CPU(s) list:    0,1
Thread(s) per core:     1
Core(s) per socket:     2
Socket(s):              1
NUMA node(s):           1
Vendor ID:              GenuineIntel
CPU family:             6
Model:                  142
Model name:             Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz
Stepping:                10
CPU MHz:                1992.000
BogoMIPS:               3984.00
Hypervisor vendor:      KVM
```

Figura 3. Configuração VM 2.

Com os respectivos endereços de IP 192.168.0.123, 192.168.0.124, as figuras 4 e 5, apresentam as configurações de IP das máquinas virtuais do cliente e do atacante.

```
(germano@germano2)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.123 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::a00:27ff:fee4:e696 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:e4:e6:96 txqueuelen 1000 (Ethernet)
    RX packets 64874 bytes 29385908 (28.0 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 16765 bytes 3005662 (2.8 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figura 4. Interface e IP da máquina do cliente.

```
(germano@germano1)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.124 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::a00:27ff:fe8b:e8b7 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:fb:e8:b7 txqueuelen 1000 (Ethernet)
    RX packets 82 bytes 8142 (7.9 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 15 bytes 1396 (1.3 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figura 5. Interface e IP da máquina do atacante.

Já a figura 6, representa a configuração do arquivo **resolv.conf** da máquina do cliente (VM 1), para forçar que essa máquina sempre tenha como servidor de DNS primário o IP do atacante (VM 2), que em nosso cenário de testes é 192.168.0.123.

```
Arquivo  Ações  Editar  Exibir  Ajuda
GNU nano 5.3 /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
# 127.0.0.53 is the systemd-resolved stub resolver.
# run "resolvectl status" to see details about the actual nameservers.

#nameserver 192.168.0.1
nameserver 192.168.0.124
```

Figura 6. Configuração do servidor DNS na máquina do cliente.

Desta forma, sempre que o cliente se conectar a rede, seu servidor DNS primário, será o endereço que se encontra no arquivo **resolv.conf**.

Como o ataque é do tipo man-in-the-middle, a topologia de rede adotada é do tipo estrela, pois obrigatoriamente, os pacotes devem passar por uma estação centralizadora, que no nosso cenário é a máquina do atacante (VM 2), abaixo, na figura 7, segue a representação do ambiente de testes.

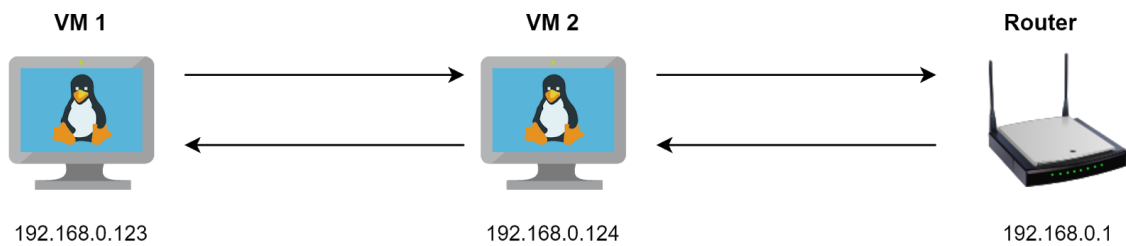


Figura 7. Esquema Man in The Middle.

Então com o ambiente montado e devidamente configurado, foi iniciado o processo de testes, a Figura 8 demonstra o *script* rodando na máquina do atacante, enquanto nenhum pacote UDP da máquina do cliente foi enviado. Perceba que, para este caso, os domínios bloqueados são *globo.com*, *facebook.com* e *uol.com.br*, como é apresentado na linha spoof domains.

```

germano@germano1: ~/Área de trabalho/DNSspoofer
Arquivo  Ações  Editar  Exibir  Ajuda

(germano@germano1)-[~/Área de trabalho/DNSspoofer]
$ sudo python3 DNSspoofer_v3.py
#####
-#-#-#-#-#-RUNNING DNS SPOOFER-#-#-#-#-#-#
#####
Interface:      eth0
Resolving to IP: 192.168.0.124
Spoof domains:  www.uol.com.br, uol.com.br, www.globo.com, globo.com, facebook.com, www.facebook.com
BPF sniff filter: udp port 53 & dst 192.168.0.124 & src 192.168.0.123
Waiting for DNS requests...
[Make sure the device you are targeting, is set to use (192.168.0.124) as its DNS server]
  
```

Figura 8. Script esperando por pacotes.

Para o envio de pacotes pela máquina do cliente, utilizou-se o *software Packet Sender*, no qual é possível criar pacotes do tipo UDP e TCP e enviá-los através de uma porta específica, além de definir um *delay* para reenvio do mesmo pacote criado.

Através da Figura 9 é possível notar que, quando um domínio não é bloqueado, a máquina do atacante repassa a requisição ao roteador e retorna a resposta à máquina do cliente.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.0.123	192.168.0.124	DNS	70	Standard query 0xe8ef A google.com
2	0.000000100	192.168.0.123	192.168.0.124	DNS	70	Standard query 0x48ec AAAA google.com
3	0.000593746	192.168.0.123	192.168.0.124	DNS	70	Standard query 0xe8ef A google.com
4	0.000593830	192.168.0.123	192.168.0.124	DNS	70	Standard query 0x48ec AAAA google.com
5	0.004154045	192.168.0.124	192.168.0.1	DNS	70	Standard query 0x5aa3 A google.com
6	0.047382277	192.168.0.1	192.168.0.124	DNS	86	Standard query response 0x5aa3 A google.com A 216.5
7	0.089496170	192.168.0.124	192.168.0.123	DNS	96	Standard query response 0xe8ef A google.com A 216.5
8	0.105362527	192.168.0.124	192.168.0.1	DNS	70	Standard query 0xf455 A google.com
9	0.115793006	192.168.0.1	192.168.0.124	DNS	86	Standard query response 0xf455 A google.com A 216.5

Figura 9. Forward de um pacote não bloqueado.

Já a Figura 10, demonstra o envio de um pacote definido pelo domínio *uol.com.br*.

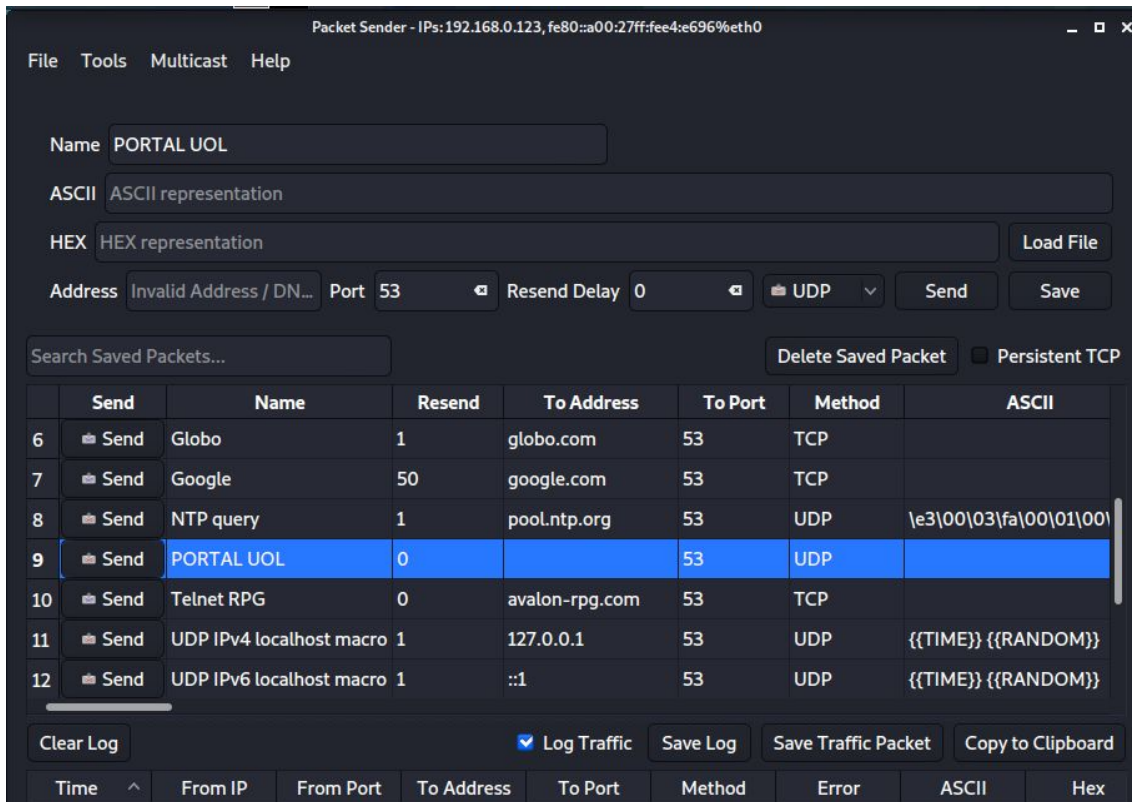


Figura 10. Envio do pacote na máquina do cliente.

No momento em que o pacote é enviado, o *script* em execução na máquina do atacante, apresenta um aviso que a requisição para o determinado domínio foi bloqueada, como visto na Figura 11.

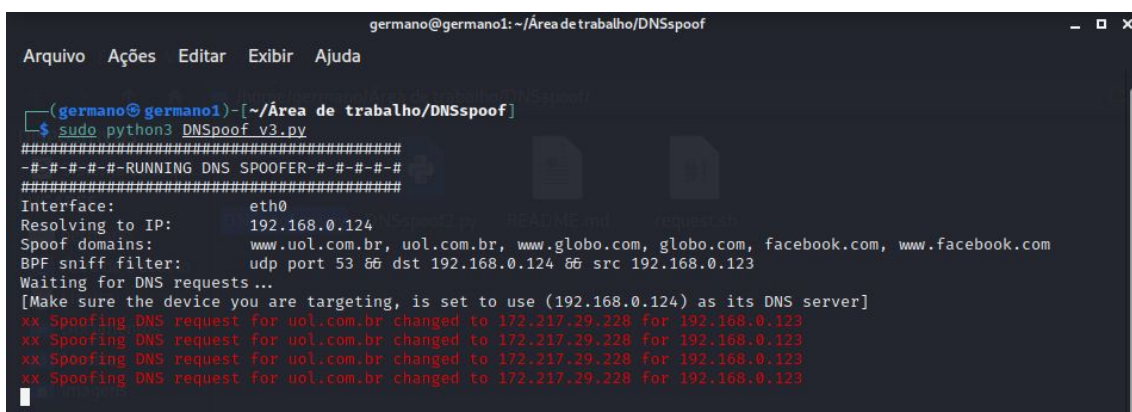
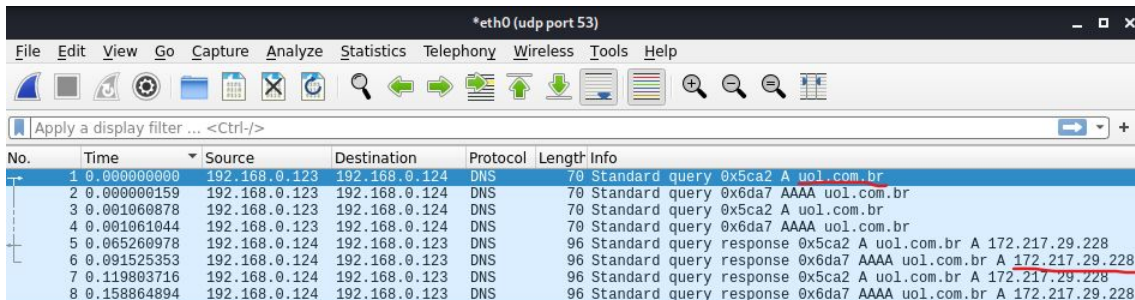


Figura 11 - Script Reconhece domínio bloqueado.

Através da Figura 12 é possível perceber que a máquina do atacante não envia a requisição da máquina do cliente ao roteador, redirecionando apenas a resolução para caso dos domínios bloqueados.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.0.123	192.168.0.124	DNS	70	Standard query 0x5ca2 A uol.com.br
2	0.000000159	192.168.0.123	192.168.0.124	DNS	70	Standard query 0x6da7 AAAA uol.com.br
3	0.001060878	192.168.0.123	192.168.0.124	DNS	70	Standard query 0x5ca2 A uol.com.br
4	0.001061044	192.168.0.123	192.168.0.124	DNS	70	Standard query 0x6da7 AAAA uol.com.br
5	0.005260978	192.168.0.124	192.168.0.123	DNS	96	Standard query response 0x5ca2 A uol.com.br A 172.217.29.228
6	0.0091525353	192.168.0.124	192.168.0.123	DNS	96	Standard query response 0x6da7 AAAA uol.com.br A 172.217.29.228
7	0.119803716	192.168.0.124	192.168.0.123	DNS	96	Standard query response 0x5ca2 A uol.com.br A 172.217.29.228
8	0.158864894	192.168.0.124	192.168.0.123	DNS	96	Standard query response 0x6da7 AAAA uol.com.br A 172.217.29.228

Figura 12. Demonstração do forward.

Ao realizarmos a tentativa de acesso ao domínio *uol.com.br* através de um navegador *web*, é possível observar pela Figura 13 que ocorre um aviso de possíveis danos de segurança ao usuário.

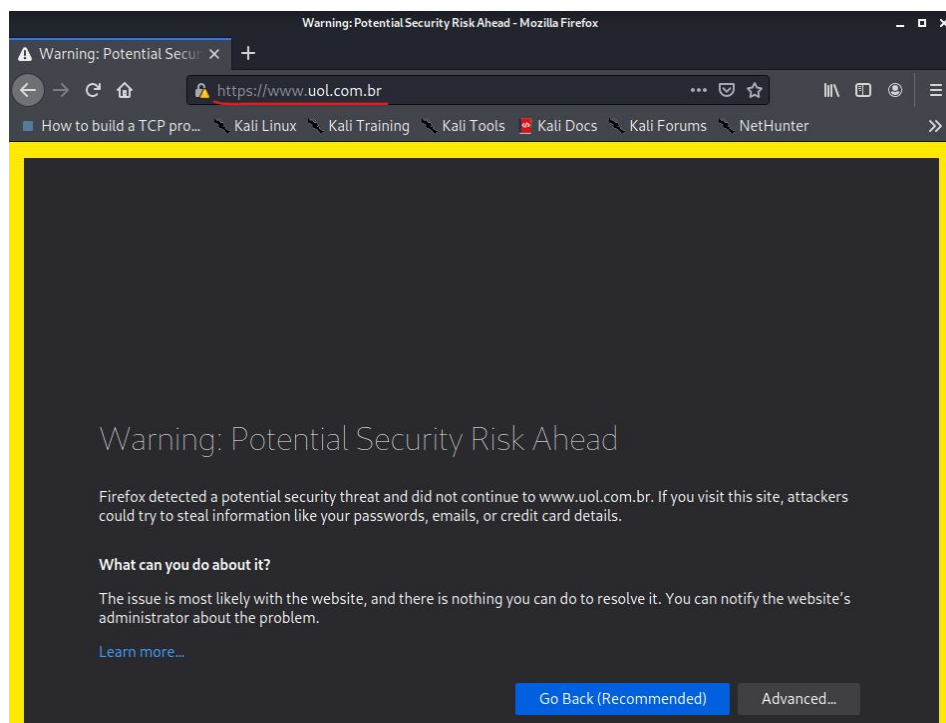


Figura 13. Página de bloqueio.

4. Análise de Desempenho

Novamente, através do *software Packet Sender*, foram gerados cerca de 2500 pacotes do protocolo UDP em um período de aproximadamente 250 segundos. Cada um destes pacotes continha um domínio, bloqueável ou não, e era enviado novamente de acordo com o *delay* estipulado na Tabela 1.

Tabela 1. Domínios enviados pelo *Packet Sender*.

Domínio	Delay para reenvio	Bloqueado
<i>facebook.com</i>	0.09	Sim
<i>globo.com</i>	0.7	Sim
<i>google.com</i>	0.01	Não
<i>uol.com.br</i>	0.4	Sim
<i>ufsm.br</i>	0.1	Não
<i>youtube.com</i>	0.05	Não

O início do envio de cada um dos pacotes é realizada de forma manual no *software*, logo, nos primeiros segundos do gráfico 1 é possível observar um menor número de pacotes por segundo. No mesmo gráfico se observa a distribuição dos pacotes bloqueados, em vermelho, e os não bloqueados, em azul.

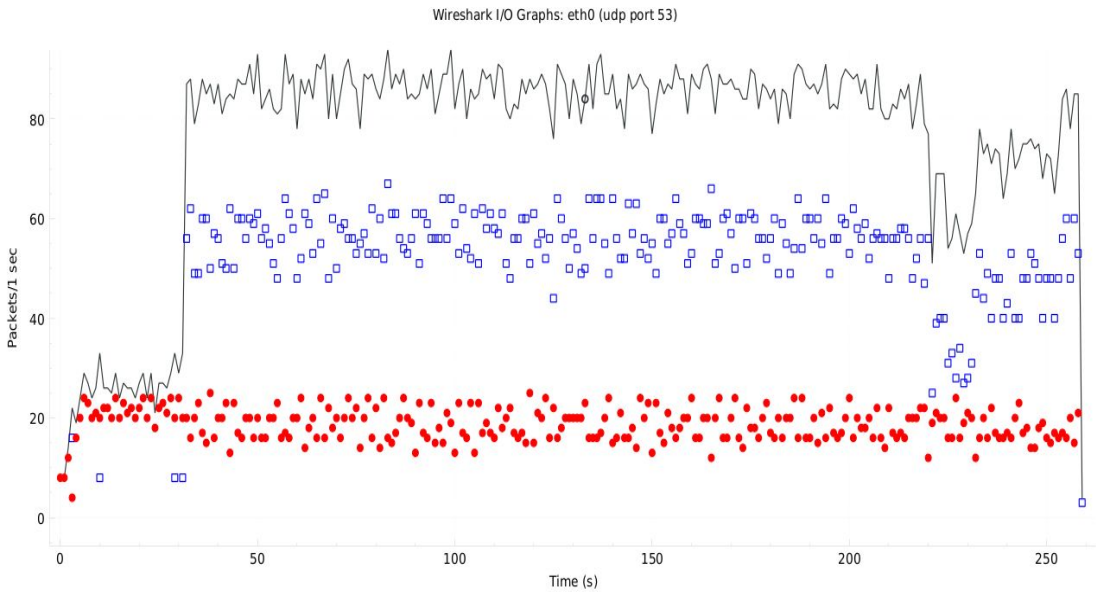


Gráfico 1. Pacotes gerados e sua distribuição.

Com os pacotes já gerados, foi possível calcular a média de bytes por segundo, onde o valor médio ficou igual a 5.788 bytes/s e 46.304 mil bits/s, o gráfico 2, apresenta a relação de bytes por segundo com relação ao tempo da análise.

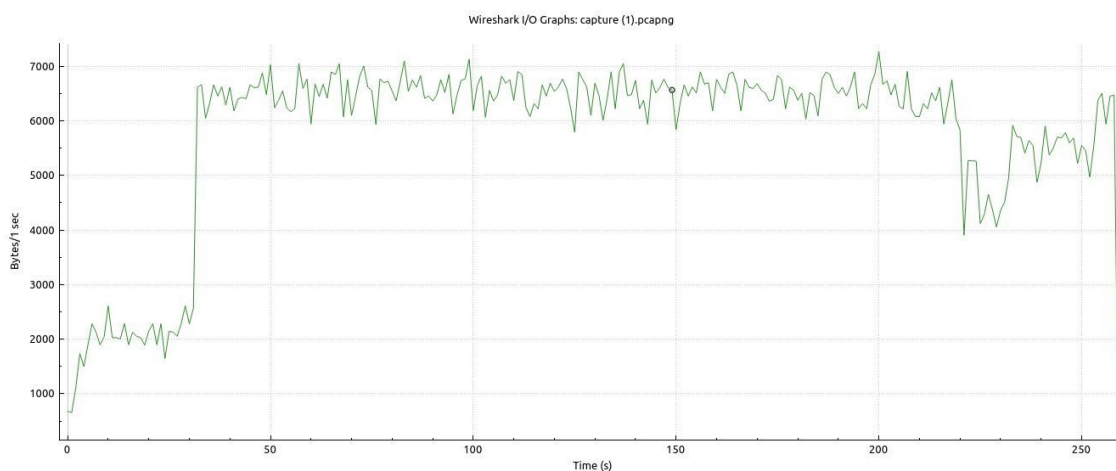


Gráfico 2. Bytes por segundo.

Através do *software Wireshark* ainda foi possível verificar o tempo médio de *delay* entre a primeira requisição da máquina do cliente e a resposta da máquina do atacante, o gráfico 3 mostra a variação do *delay* médio ao longo do tempo, com picos no início do e no fim do cenário de testes, mas mantendo um valor médio de 40 ms.

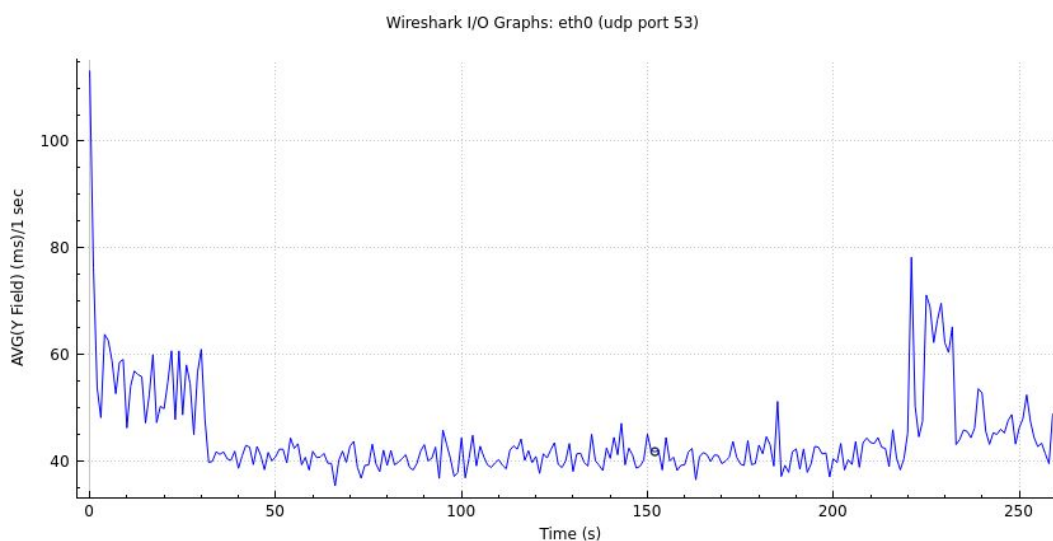


Gráfico 3. Média do delay até a resposta.

Observe através do Gráfico 4 a média dos *delays* entre a requisição e a resposta em um cenário semelhante ao anterior, porém sem o *forwarder*. O tempo *delay* do sistema se mostrou cerca 70% menor para o melhor caso e 80% para o pior caso.

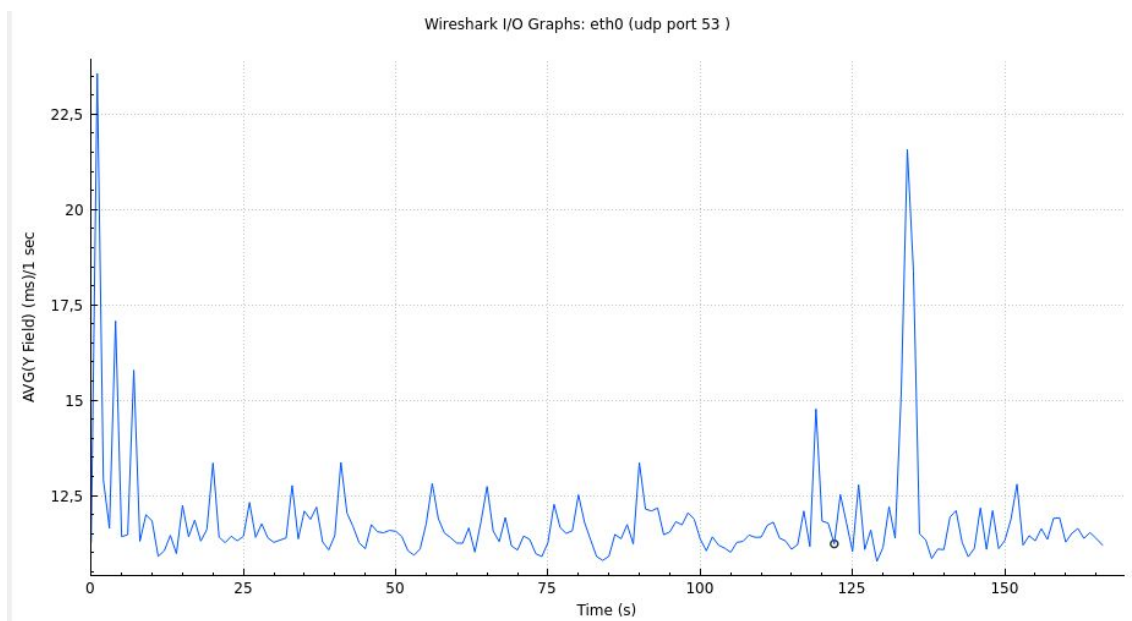


Gráfico 4. Delay médio sem o *forwarder*.

Já o gráfico 4 demonstra o delay médio dividido por domínios bloqueados e não bloqueados.

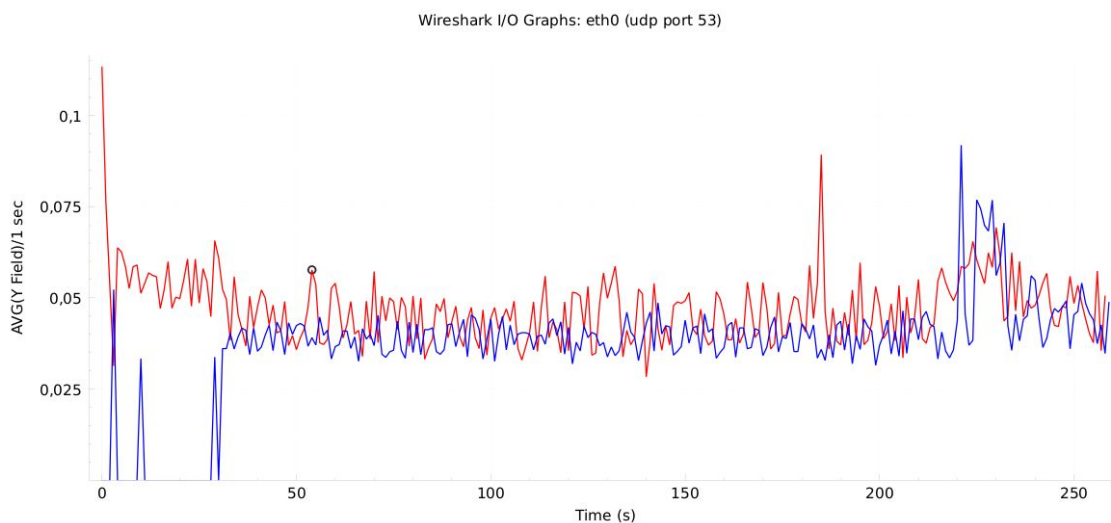


Gráfico 5. Delay médio dividido por domínios bloqueados e não bloqueados.

Número de erros

No cenário de testes implementado, com o envio de cerca de 2500 pacotes com requisições de domínios bloqueados e não bloqueados, o sistema não apresentou nem um erro. Através da ferramenta *Wireshark* se comprovou a eficácia neste panorama, como mostrado na Figura 14

Dropped packets
0 (0.0%)

Figura 14. Pacotes descartados.

Referências

Andrew Tanenbaum. 2002. Redes de Computadores (5ª ed.).

Kurose, J. F. e Ross, K. 2014. Redes de Computadores e a Internet (6ª ed.).

Robert heaton (2018) “How to build a tcp-proxy”, <https://robertheaton.com/2018/08/31/how-to-build-a-tcp-proxy-2/>. Acessado em, 2021, Fevereiro, 13.

Duke University Department of Computer Science (2016) “DNS Primer Notes”, <https://www2.cs.duke.edu/courses/fall16/compsci356/DNS/DNS-primer.pdf>. Acessado em, 2021, Fevereiro, 13.