Universidade Federal de Santa Maria
Cursos: Ciência da Computação e Sistemas de Informação
Alunos: Francisco das Chagas Sousa Júnior, João Victor Lisboa de Lima e Matheus Delazeri Souza
Professor Carlos Raniery Paula dos Santos
Disciplina: Redes de Computadores

Relatório do Trabalho 2

INTRODUÇÃO

Inicialmente, foi necessário definir qual seria a função de rede implementada, após analisar as opções sugeridas pelo professor, foi definida que seria um filtro de *bad words*. O ambiente escolhido para execução do trabalho foi o *mininet*, que já fui utilizado no trabalho 1 da disciplina, além disso, foi utilizada a estrutura básica que o professor disponibilizou para simular uma rede, também foi utilizada a biblioteca *scapy*.

Nesse sentido, foi utilizada a topologia básica para implementar e testar essa função, essa topologia consiste a um *host* "h1" ligado a um roteador "r", que por sua vez é ligado a um *host* "h2". Dessa forma, os *hosts* desempenham papel de cliente e o roteador tem o papel de servidor. No caso em que algum dos dois hosts deseja enviar um pacote para o outro host através do roteador, o roteador utilizará a função de filtro de rede para analisar o *payload* do pacote e verificar se há alguma *bad word* no conteúdo da mensagem, caso haja, a função substitui cada caractere da palavra por "*". Em seguida, os campos de tamanho e *checksum* do pacote são recalculados e alterados e a mensagem é encaminhada para o *host* final.

EXECUÇÃO DO PROGRAMA

1. Para inicar o ambiente e a execução do programa deve-se usar o comando: "sudo python3 topology.py";
2. Dentro do ambiente do mininet, deve-se utilizar o comando "xterm h1 h2 r", que abre os terminais de h1, h2 e r;
3. No terminal de r deve ser utilizado o comando "python3 routing.py --node r --filter bad_word", que executará o algoritmo de roteamento e o a função do filtro de *bad words.*
4. No terminal do host de destino, é necessário utilizar o comando "python3 sniff.py" para mostrar os pacotes recebidos naquele host.
5. No terminal do host da fonte, é necessário utilizar o comando "python3 send_packet.py 10.2.2.1 -- payload "m" ", m deve ser substituído pelo conteúdo da mensagem.
6. Após isso, será possível visualizar nos 3 terminais o encaminhamento dos pacotes.

TESTES E RESULTADOS

Para realizar os testes, foi implementado um arquivo test.py que deve ser executado no host da fonte com o seguinte comando "python3 test.py 10.2.2.1 --payload "m" --count "n"", sendo m o conteúdo do pacote que deseja-se enviar e n o número de vezes que deve ser enviado aquele pacote.

O primeiro teste foi realizado com a seguinte mensagem no *payload* "mas que porra!", foram considerados alguns cenários enviando o pacote 100, 300,500, 1000 e 10000 vezes. As métricas que foram consideradas para o desempenho são: pacotes recebidos, perda de pacote, tempo médio por pacotes (total), tempo médio por pacotes recebidos.

Para o primeiro teste, têm-se 100 pacotes enviados e os seguintes resultados:
Pacotes recebidos: 100
Perda de pacote: 0%
Tempo médio por pacotes (total): 0,00984s

Tempo médio por pacotes recebidos: 0,00984s
Para o segundo teste, têm-se 300 pacotes enviados e os seguintes resultados:
Pacotes recebidos: 288
Perda de pacote: 4%
Tempo médio por pacotes: 0,00915s
Tempo médio por pacotes recebidos: 0,00953s

Para o terceiro teste, têm-se 500 pacotes enviados e os seguintes resultados:
Pacotes recebidos: 421
Perda de pacote: 15,8%
Tempo médio por pacotes: 0,00861s
Tempo médio por pacotes recebidos: 0,01023s

Para o quarto teste, têm-se 1000 pacotes enviados e os seguintes resultados:
Pacotes recebidos: 763
Perda de pacote: 23,7%
Tempo médio por pacotes: 0,00812s
Tempo médio por pacotes recebidos: 0,01064s

Para o quinto teste, têm-se 10000 pacotes enviados e os seguintes resultados:
Pacotes recebidos: 7567
Perda de pacote: 24,33%
Tempo médio por pacotes: 0,00727s
Tempo médio por pacotes recebidos: 0,00960s

Para a segunda fase de testes, foi utilizada a entrada "python3 test.py 10.2.2.1 --payload "mas que merda! merda" --count n", com n variando entre 100, 300, 500, 1000, e 10000.

Para o primeiro teste dessa segunda fase, têm-se 100 pacotes enviados e os seguintes resultados:
Pacotes recebidos: 100
Perda de pacote: 0%
Tempo médio por pacotes (total): 0,01016s
Tempo médio por pacotes recebidos: 0,01016s

Para o segundo teste dessa segunda fase, têm-se 300 pacotes enviados e os seguintes resultados:
Pacotes recebidos: 260
Perda de pacote: 13,33%
Tempo médio por pacotes: 0,00921s
Tempo médio por pacotes recebidos: 0,01062s

Para o terceiro teste dessa segunda fase, têm-se 500 pacotes enviados e os seguintes resultados:
Pacotes recebidos: 364
Perda de pacote: 27,2%
Tempo médio por pacotes: 0,00795s
Tempo médio por pacotes recebidos: 0,01092s

Para o quarto teste dessa segunda fase, têm-se 1000 pacotes enviados e os seguintes resultados:
Pacotes recebidos: 704
Perda de pacote: 29,6%
Tempo médio por pacotes: 0,00765s

Tempo médio por pacotes recebidos: 0,01086s

Para o quinto teste dessa segunda fase, têm-se 10000 pacotes enviados e os seguintes resultados:
Pacotes recebidos: 7461
Perda de pacote: 25,39%
Tempo médio por pacotes: 0,00773s
Tempo médio por pacotes recebidos: 0,01036s

Para a terceira fase de testes, foi utilizada a entrada “python3 test.py 10.2.2.1 --payload “merda! caralho! porra! xambao! jaguara!” --count n”, com n variando entre 100, 300, 500, 1000, e 10000.

Para o primeiro teste dessa terceira fase, têm-se 100 pacotes enviados e os seguintes resultados:
Pacotes recebidos: 100
Perda de pacote: 0%
Tempo médio por pacotes (total): 0,00988s
Tempo médio por pacotes recebidos: 0,00988s

Para o segundo teste dessa terceira fase, tem-se 300 pacotes enviados e os seguintes resultados:
Pacotes recebidos: 290
Perda de pacote: 3,33%
Tempo médio por pacotes: 0,00994s
Tempo médio por pacotes recebidos: 0,01028s

Para o terceiro teste dessa terceira fase, tem-se 500 pacotes enviados e os seguintes resultados:
Pacotes recebidos: 407
Perda de pacote: 18,6%
Tempo médio por pacotes: 0,00896s
Tempo médio por pacotes recebidos: 0,01101s

Para o quarto teste dessa terceira fase, tem-se 1000 pacotes enviados e os seguintes resultados:
Pacotes recebidos: 751
Perda de pacote: 24,9%
Tempo médio por pacotes: 0,00801s
Tempo médio por pacotes recebidos: 0,01067s

Para o quinto teste dessa terceira fase, tem-se 10000 pacotes enviados e os seguintes resultados:
Pacotes recebidos: 7915
Perda de pacote: 20,85%
Tempo médio por pacotes: 0,00784s
Tempo médio por pacotes recebidos: 0,00991s