Projeto 2 - Aplicação de Firewall no Mininet

Jean Bueno Karia
211055290

Marcos Alexandre da Silva Neres
211055334

Sara P Lima Campos
170045269

***Abstract*—**

***Index Terms*—Mininet, Firewall, Openflow, SDN**

1. Introdução

Utilizando um simulador para SDN(Software Defined Networking) em conjunto com o simulador Mininet, com suporte para Openflow, desenvolveremos uma topologia e um aplicativo customizado com o propósito de fortalecer a segurança da rede. O objetivo é aprimorar a segurança em ambientes de rede, tendo em vista a crescente virtualização de servidores e o aumento do uso de dispositivos móveis.

2. Fundamentação Teórica

Para a fundamentação teórica tem-se 5 principais pontos, sendo essas:

- **Redes Definidas por Software (SDN):**
  As redes Definidas por software é uma tecnologia que permite o gerenciamento de redes virtualmente por meio de um controlador, tal rede é denominada como centralizada, como foi estudado através do livro base da diciplina. Isso permite uma melhor manutenção e flexiblização da rede.
- **Protocolo OpenFlow:**
  O OpenFlow é um protocolo padronizado utilizado para a interação entre o controlador SDN e os dispositivos de rede. Ele faz com se seja possível enviar instruções para os dispositivos de rede por meio do controlador, encaminhando pacotes de forma dinâmica, como também abordado pelo livro texto da diciplina.
- **Simulador Mininet:**
  O simulador Mininet foi projetado principalmente para simular redes SDN habilitadas para OpenFlow, permitindo a criação de topologias de rede. A documentação utilizada para se obter informações sobre o mininet, pode-se se encontrar no próprio site do mininet.
- **Segurança em Redes:**
  A fundamentação teórica sobre segurança de redes, especialmente o firewall, que é uma barreira que monitora o tráfego de rede de entrada e saída e decide permitir ou bloquear tráfegos específicos de acordo com um conjunto definido de regras de segurança.
- **Camadas do Modelo TCP/IP:**
  O entedimento sobre o protocolo TPC/IP, é importante para uma boa compreensão sobre o formato das mensagens e datagramas para entender o tráfego entre os hosts, tal base teórica foi adquirida através do livro Computer Networking: A Top-Down Approach.

3. Ambiente Experimental e Análise de Resultados

3.1. Descrição do Cenário

- **Topologia:**
  Ao se inicializar é escolhido a quantidade de hosts e switches disponíveis na rede. Em seguida, os hosts são conectados através da topologia starbus, e inicializados através do metodo NAT com a porta 2224.
- **Firewall:**
  Foi desenvolvido um firewall arbitrário para o controlador POX habilitado para openflow com o objetivo de se simplificar o projeto. Ele funciona bloqueando a conexão entre hosts com endereço mac com a diferença de 1 superior. Por exemplo, supondo a existência de 3 hosts h01,h02 e h03, o firewall irá bloquear a conexão de h01 com h02 e a conexão de h02 com h03.
  Também implementamos um controle de fluxo, que identifica quando um pacote é enviado com tamanho maior que o determinado para destinos pré-definidos como limitados (neste caso, h03) e bloqueia envios seguintes daquele endereço, que é interpretado como indesejável.
- **Aplicação:**
  A aplicação é uma simples troca de mensagens, mais específicamente, ocorre o envio do index HTTP definido por padrão.

3.2. Análise de Resultados

Sem o firewall, primeiramente, podemos verificar a conectividade entre todos os nós da topologia criada utilizando o comando "pingall" dentro do CLI do mininet. Atestamos, dessa forma, que todos os hosts tem conexão entre si, portanto a toplogia é efetiva e todos os switches também estão conectados.

```
mininet> pingall
*** Ping: testing ping reachability
h01 -> h02 h03 h04 h05 h06 h07 h08 h09 h10
h02 -> h01 h03 h04 h05 h06 h07 h08 h09 h10
h03 -> h01 h02 h04 h05 h06 h07 h08 h09 h10
h04 -> h01 h02 h03 h05 h06 h07 h08 h09 h10
h05 -> h01 h02 h03 h04 h06 h07 h08 h09 h10
h06 -> h01 h02 h03 h04 h05 h07 h08 h09 h10
h07 -> h01 h02 h03 h04 h05 h06 h08 h09 h10
h08 -> h01 h02 h03 h04 h05 h06 h07 h09 h10
h09 -> h01 h02 h03 h04 h05 h06 h07 h08 h10
h10 -> h01 h02 h03 h04 h05 h06 h07 h08 h09
*** Results: 0% dropped (90/90 received)
```

Figure 1. Pingall sem firewall

Para identificar o funcionamento da rede, utilizamos a ferramenta Wireshark. Agora, queremos testar o tráfego de alguma aplicação entre dois hosts. No host h1, criamos um simples servidor http, e no host h2 fizemos uma requisição para esse servidor, de forma que um simples arquivo ”index.html” fosse transmitido a h2.

```
mininet> h01 python -m SimpleHTTPServer 80 &
mininet> h02 wget -o - h01
--2023-07-24 03:02:33--  http://192.168.1.1/
Connecting to 192.168.1.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 600 [text/html]
Saving to: 'index.html'

     0K                                                       100%  104M=0s

2023-07-24 03:02:33 (104 MB/s) - 'index.html' saved [600/600]
```

Figure 2. Tráfego da aplicação

Os resultados encontrados no wireshark foram os esperados, uma vez que a transmissão é dada em TCP: ocorre o 3-way handshake para inicializar a conexão segura entre os dois hosts e o h2 faz uma requisição HTTP com o método GET para o servidor.

2277	49.69036600	192.168.1.1	192.168.1.2	TCP	74 80 → 57642 [SYN, ACK]
2278	49.69785327	192.168.1.2	192.168.1.1	TCP	66 57642 → 80 [ACK] Seq=
2279	49.69786297	192.168.1.2	192.168.1.1	HTTP	204 GET / HTTP/1.1
2280	49.69872353	192.168.1.1	192.168.1.2	TCP	66 80 → 57642 [ACK] Seq=
2281	49.69873553	192.168.1.1	192.168.1.2	TCP	83 80 → 57642 [PSH, ACK]
2282	49.69874125	192.168.1.1	192.168.1.2	HTTP	835 HTTP/1.0 200 OK (tex
2283	49.69884750	192.168.1.2	192.168.1.1	TCP	66 57642 → 80 [ACK] Seq=
2284	49.69888093	192.168.1.2	192.168.1.1	TCP	66 57642 → 80 [ACK] Seq=

Figure 3. 3-way handshake e requisição HTTP no Wireshark

Passado as confirmações de h1 e h2 para a requisição, h1 responde enviando o arquivo texto ”index.thml”.

```
▸ Frame 21077: 835 bytes on wire (6680 bits), 835 bytes captured (6680 bits) on interface 8
▸ Ethernet II, Src: 00:00:00_00:00:01 (00:00:00:00:00:01), Dst: 00:00:00_00:00:02 (00:00:00:00:00:02)
▸ Internet Protocol Version 4, Src: 192.168.1.1, Dst: 192.168.1.2
▸ Transmission Control Protocol, Src Port: 80, Dst Port: 55810, Seq: 18, Ack: 139, Len: 769
▸ [2 Reassembled TCP Segments (786 bytes): #21076(17), #21077(769)]
▸ Hypertext Transfer Protocol
▸ Line-based text data: text/html (19 lines)
```

Figure 4. Encapsulamento da pilha TCP/IP

Ao analisar o encapsulamento, verificamos que o item mais abaixo corresponde aos dados gerados na camada de aplicação, ao texto enviado pelo servidor.

O protocolo utilizado na camada de aplicação é o HTTP (Hypertext Transfer Protocol). O overhead do encapsulamento para os dados gerados na camada de aplicação é dado abaixo:

```
▾ Hypertext Transfer Protocol
  ▾ HTTP/1.0 200 OK\r\n
    ▾ [Expert Info (Chat/Sequence): HTTP/1.0 200 OK\r\n]
        [HTTP/1.0 200 OK\r\n]
        [Severity level: Chat]
        [Group: Sequence]
      Response Version: HTTP/1.0
      Status Code: 200
      [Status Code Description: OK]
      Response Phrase: OK
    Server: SimpleHTTP/0.6 Python/2.7.12\r\n
    Date: Sun, 23 Jul 2023 01:03:06 GMT\r\n
    Content-type: text/html\r\n
  ▸ Content-Length: 600\r\n
    Last-Modified: Sat, 22 Jul 2023 22:28:48 GMT\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.002341787 seconds]
    [Request in frame: 21074]
    [Request URI: http://192.168.1.1/]
    File Data: 600 bytes
```

Figure 5. Overhead dos dados da camada de aplicação

Onde identificamos os elementos necessários para o devido encapsulamento e utilização do arquivo em h2, como os campos ”Date”, ”Content-Type” e ”Last-Modified”, que possuem usos claros; Especialmente ”Last-Modified”, uma vez que essa é uma informação necessária à criação de um arquivo e, caso não fosse formado um cabeçalho com esse campo, não seria possível adquirir essa informação.

```
▾ Transmission Control Protocol, Src Port: 80, Dst Port: 55810, Seq: 18, Ack: 139, Len: 769
    Source Port: 80
    Destination Port: 55810
    [Stream index: 5]
    [TCP Segment Len: 769]
    Sequence number: 18    (relative sequence number)
    [Next sequence number: 788    (relative sequence number)]
    Acknowledgment number: 139    (relative ack number)
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x019 (FIN, PSH, ACK)
    Window size value: 59
    [Calculated window size: 30208]
    [Window size scaling factor: 512]
    Checksum: 0x90c3 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ [SEQ/ACK analysis]
  ▸ [Timestamps]
    TCP payload (769 bytes)
    TCP segment data (769 bytes)
▸ [2 Reassembled TCP Segments (786 bytes): #21076(17), #21077(769)]
```

Figure 6. Cabeçalho do TCP no envio HTTP

O protocolo utilizado na camada de transporte é o TCP, (Transmission Control Protocol) que conta em seu cabeçalho com informações imprescindíveis para a realização do transporte do pacote - tais quais portas de destino e saída, o checksum para identificar a integridade do arquivo, bem

como itens característicos do TCP, como o uso de janela deslizante.

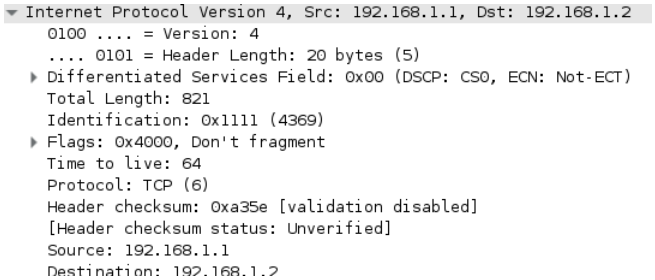

Figure 7. Cabeçalho do IPv4

A camada de rede utiliza IPv4, contendo no cabeçalho do datagrama os ips de origem e destino, o protocolo utilizado na camada de transporte, o TTL desse datagrama; Bem como o checksum, flags e a identificação, que são campos específicos a ele, descartados no IPv6.

O enlace faz uso de Ethernet II, o quadro encontrado é simples, contendo apenas a origem e destino em endereços mac e o tipo de protocolo da camada de rede (IPv4).

Com o firewall ligado, tentamos testar novamente a conectividade entre os nós da topologia. Percebemos que, agora, a conectividade entre alguns dos nós - os determinados pelas regras de bloqueio criadas entre endereços mac no firewall.

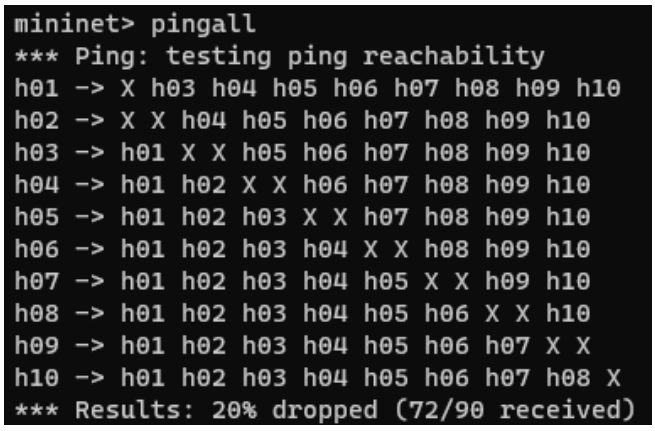

Figure 8. Pingall com firewall

Ao tentar realizar o tráfego da aplicação novamente entre h01 e h02, não é possível encontrar uma rota entre os dois, uma vez que a conexão entre ambos foi bloqueada pelo firewall.

Figure 9. Falha no tráfego da aplicação

Já a conexão entre h01 e h03 inicialmente é possível; Entretanto, envios seguintes não são possíveis, uma vez que o ”index.html” enviado possui tamanho maior que o escolhido no controle de fluxo; Se são feitos contatos com menos bits ou com hosts fora da lista de hosts limitados, o bloqueio não ocorre.

O vídeo com apresentação da topologia e firewall e análise dos resultados pode ser visto aqui (https://youtu.be/e_n7CbBPAD8).

Juntamente o github do projeto pode ser visto aqui (https://github.com/saracampss/Projeto2_Redes)

4. Conclusões

O projeto teve como objetivo a implementação de segurança em redes utilizando o simulador Mininet através da abordagem de redes definidas por software. Portanto pode-se dizer que obtemos um resultado satisfatório ao concluir todos os objetivos previamente definidos, assim como obter um melhor compreendimento dos conceito ensinados em sala de aula. Em relação à dificuldades encontradas na realização do projeto, a mais relevante foi encontrar a documentações necessárias, e também optamos em utilizar enderços MAC ao invés de endereços IP por em sua manipulação.

5. Bibliografia

References

[1] Computer Networking: A Top-Down Approach 8th edition Jim Kurose, Keith RossPearson, 2020

[2] Documentação sobre o controlador POX: https://noxrepo.github.io/pox-doc/html/

[3] Documentação sobre o mininet: https://github.com/mininet/mininet/wiki/Documentation