LAB 3/4 - Construindo um *firewall* simples

Objetivo

Já trabalhamos com alguns casos de uso básicos para reforçarmos os conceitos de SDN discutidos e aprender um pouco sobre a programação do ONOS e como operá-lo como usuário final.

Vamos agora mudar o foco para uma aplicação simples de *firewall* que mostrará como criar regras de bloqueio (*drop*) e como uma aplicação pode se utilizar da CLI do ONOS, para criar novos comandos, estendendo as suas funcionalidades.

Introdução

Nessa prática, iremos utilizar um *firewall* simples, que permite a criação de regras de bloqueio por IP e porta de origem, IP e porta de destino ou ambos (a porta é opcional). Na aplicação foram implementados vários comandos, que estendem a CLI do ONOS, para gerenciar o *firewall,* como listar as regras, adicionar regras e removê-las.

Um ponto importante da prática é que essa aplicação não possui um processador de pacotes associado, ou seja, não utilizamos o serviço de *packetService* do ONOS e não há nenhum objeto do tipo *processor* ou interceptador de pacotes para a aplicação. Na prática, isso significa que a aplicação não recebe nenhum pacote pelo *pipeline* de processamento do controlador, sendo que o seu comportamento não é reativo, não reagindo aos eventos da rede.

O motivo da escolha acima é que não há muito sentido em um firewall que irá bloquear novos pacotes ao receber um novo pacote do fluxo ‘proibido’. Pense no seguinte cenário:

1. h1 tenta se comunicar com o Facebook na porta 443;
2. No momento, não há regra proibindo essa comunicação. O primeiro pacote irá para o controlador que criará as regras de fluxos de encaminhamento;
3. O operador decide bloquear o acesso WEB ao Facebook e cria uma regra no *Firewall* que será tratada pela aplicação apenas quando ela receber um pacote do fluxo ‘proibido’. No entanto, como já temos uma regra liberando a comunicação entre os dois pontos, os pacotes desse fluxo nunca irão para o controlador e a regra nunca irá tratá-los para criar o bloqueio.

A alternativa seria remover as regras que já existirem para tal fluxo e criar a regra de *drop* nos *switches*, assim que o comando de bloqueio for executado na CLI. No entanto, não há ganhos em adotar essa abordagem, além de que iremos gerar um *overhead* no *pipeline* de processamento sem necessidade, uma vez que os pacotes iriam ser encaminhados para o processamento pela aplicação. O mais simples seria apenas criar as regras de *drop* nos equipamentos da rede, assim que o comando de bloqueio for executado na CLI.

A principal desvantagem, nesse caso, é que iriamos popular os dispositivos (ex: *switches*) com regras de drop que talvez nem sejam utilizadas, se não recebermos nenhum pacote desse fluxo. Dependendo do número de regras de bloqueio e o tamanho da rede, uma abordagem proativa como esta, não seria adequada, podendo 'estourar' o número de entradas na tabela de fluxos do *switch*.

Note que existem várias aplicações de segurança que utilizam os pacotes dos novos fluxos para tomar suas decisões, como aplicações de *DPI* (inspeção profunda de pacotes), que analisam os pacotes, casam padrões, tomam decisões sobre o encaminhamento e até geram alarmes (poderíamos mandar um email para o operador sempre que um padrão de fluxos, como um ataque, for encontrado ou a assinatura de um vírus no *payload* do pacote). Reflitam no ganho que essas aplicações teriam em ambientes SDN, podendo ser possível retirar equipamentos, como *middleboxs,* da rede e criá-los em software como aplicações do controlador.

Um caso de Firewall que se utilizaria da análise de pacotes, seria o bloqueio por URLS, por exemplo. A aplicação checaria o endereço IP de destino através de uma consulta DNS e se a URL estiver na lista de bloqueio, a aplicação criaria as regras de fluxo correspondentes, bloqueando o tráfego.

Criar topologia no FIBRE

Iremos utilizar a mesma topologia no FIBRE das práticas anteriores e, portanto, não precisamos nós preocupar com essa parte. Apenas lembrando, a topologia consiste em uma rede *full-mesh* de 4 *switches*, com 5 *hosts* conectados.

Código Relevante para o aprendizado

Código do Firewall Simples

```
private final HashSet<BlockRecord> srcBlockList = new HashSet<>();
private final HashSet<BlockRecord> dstBlockList = new HashSet<>();


private final HashMultimap<BlockRecord,BlockRecord>
   pairBlockList = HashMultimap.create();


private class BlockRecord {
   private  String ip;
   private  String port;
}
```

Criação de três listas do tipo BlockRecord (armazena ip e porta) para registro das regras de bloqueio por origem, destino e ambos os pares da conexão, respectivamente.

```
Iterable<Device> devices = deviceService.getDevices();
```

Utiliza do serviço de *devices* do ONOS para obter a lista de todos os dispositivos conhecidos pela topologia, no caso, os switches ao quais aplicaremos as regras.

```
selector = selectorBuilder
      .matchIPSrc(IpAddress.valueOf(ip).toIpPrefix())
      .matchIPProtocol(ipv4Protocol)
      .matchTcpSrc(Short.parseShort(tcpPort))
      .build();

flowRuleAdd(deviceId, selector);
flowRuleRemove(deviceId, selector);
```

Define um seletor de tráfego para realizar *match* pelo protocolo IPv4 e os campos de IP de origem (***matchIPSrc***) e porta de origem (***matchTcpSrc***).

O seletor criado é repassado para a função de adicionar e remover regras de fluxos (***flowRuleAdd*** e ***flowRuleRemove***), junto com o dispositivo onde a regra será criada.

```
private void flowRuleAdd(DeviceId deviceId, TrafficSelector selector) {
   TrafficTreatment drop = DefaultTrafficTreatment.builder()
         .drop().build();
   FlowRule flowRule = DefaultFlowRule.builder()
      .fromApp(appId)
      .withTreatment(drop)
      .withSelector(selector)
      .withPriority(DROP_PRIORITY)
      .makePermanent()
      .forDevice(deviceId)
      .build();
   flowRuleService.applyFlowRules(flowRule);
   // flowRuleService.removeFlowRules(flowRule);
}
```

Função de criação de regras de fluxos. Cria um tratamento (***TrafficTreatment***) do tipo *drop* (para bloquear o tráfego) e utiliza o tratamento na definição da regra (***FlowRule flowRule***). A regra é aplicada aos dispositivos através do serviço de *flowRuleService* do ONOS. Para remover uma regra, basta trocar a função de *applyFlowRules* por *removeFlowRules* no serviço.

Importando o código no IntelliJ

Iremos importar o código da aplicação no IntelliJ. O prática se encontra no diretório **~/sci2015/aplicacoes/proactive-firewall**

Caso tenha alguma dificuldade no processo, por favor, avise a alguns dos instrutores do curso.

Compilando e carregando a aplicação no ONOS

Agora iremos compilar o código e carregá-lo para o controlador. Para tanto, execute os comandos abaixo:

```
#> cd ~/sci-2015/aplicacoes/proactive-firewall

#> mvn clean install && onos-app $( ipdocker onos1) install
target/proactive-firewall-1.0-SNAPSHOT.oar
```

Ative a aplicação no controlador e verifique que ela foi carregada corretamente.

```
onos> app activate proactive.firewall.app
onos> log:display |grep -i firewall
… … … … …
2015-09-13 16:52:27 | INFO  | h for user karaf | ProactiveFirewall
proactive.firewall.proactive-firewall - 1.0.0.SNAPSHOT | Started
2015-09-13 16:52:27 | INFO  | h for user karaf | ApplicationManager
org.onosproject.onos-core-net - 1.3.0.SNAPSHOT | Application
proactive.firewall.app has been activated
```

Testando a aplicação

Vamos verificar o funcionamento da aplicação. No entanto, como o firewall lida apenas com as regras de bloqueio, precisamos habilitar alguma aplicação de encaminhamento para lidar com a comunicação dos *hosts* na rede. Ative a aplicação padrão do ONOS para encaminhamento, conforme abaixo:

```
onos> app activate org.onosproject.fwd
onos> apps -s -a
* 3 org.onosproject.drivers   1.3.0.SNAPSHOT Builtin device driver
* 31 org.onosproject.fwd      1.3.0.SNAPSHOT Reactive forwarding
application using flow subsystem
* 32 org.onosproject.openflow 1.3.0.SNAPSHOT OpenFlow protocol
southbound providers
*  40 proactive.firewall.app  1.0.SNAPSHOT SCI-2015 Firewall App
```

A aplicação de Firewall estende a CLI do ONOS, adicionando a ela diversas novas funcionalidades para gerenciar as regras de Firewall, como:

```
onos> fw<tab>
fwadd-dstrule        fwadd-pairrule       fwadd-srcrule
fwremove-dstrule     fwremove-pairrule    fwremove-srcrule
fwrules-list         fwrules-removeall
```

Para ver uma breve descrição dos comandos, você pode utilizar o comando "**help**", que lista os comandos disponíveis e procurar pelos comandos disponibilizados pela aplicação:

```
onos> help |grep firewall
```

Para verificar a sintaxe de cada comando, podemos utilizar a opção **--help**, após o comando desejado:

```
onos> fwadd-dstrule
Error executing command proactiveFirewall:fwAdd-dstRule: argument dstIp
is required

onos> fwadd-dstrule --help
DESCRIPTION
         Adicione Regra de Bloqueio para o destino especificado-
IP/[PORTA]

SYNTAX
        proactiveFirewall:fwAdd-dstRule [options] dstIp [dstPort]

ARGUMENTS
        dstIp       Ip de Destino do Bloqueio
        dstPort     Porta a ser bloqueada [opcional]

OPTIONS
        --help      Display this help message
        -j, --json  Output JSON
```

Temos 20 regras instaladas nos *switches* (**summary)**, para o encaminhamento dos pacotes para o controlador. No entanto, elas se referem aos *requests* das demais aplicações instaladas, como a de encaminhamento. Verifique na saída do comando **flows** que não há nenhuma regra de fluxo instalada para o *Firewall*.

```
onos> summary
node=10.0.3.11, version=1.3.0.mininet nodes=1, devices=4, links=12,
hosts=0, SCC(s)=1, flows=20, intents=0

onos> flows |grep firewall
```

A aplicação de *firewall* provê um comando para listar as regras ativas. Verifique-as:

```
onos> fwlist-allrules
Listando 0 regras:
-------------------------------------------
-------------------------------------------
```

Sem regras de bloqueio, um ping entre h1 e h5 deverá funcionar normalmente:

```
h1> ping 192.168.0.5 -c2
PING 192.168.0.5 (192.168.0.5) 56(84) bytes of data.
64 bytes from 192.168.0.5: icmp_seq=1 ttl=64 time=181 ms
64 bytes from 192.168.0.5: icmp_seq=2 ttl=64 time=0.358 ms
2 packets transmitted, 2 received, 0% packet loss, time 999ms
```

Duas novas regras de fluxos foram criadas pela aplicação de encaminhamento para tratar a comunicação de h1 com h5, devido ao ping acima:

```
onos> summary
node=10.0.3.11, version=1.3.0.mininet nodes=1, devices=4, links=12,
hosts=2, SCC(s)=1, flows=22, intents=0
```

Vamos agora bloquear o *host* h5 e verificar que a regra foi devidamente aplicada:

```
onos> fwadd-dstrule 192.168.0.5
Bloqueio criado com sucesso para o destino 192.168.0.5 - A porta informada foi
null [null representa todas as portas]
Para listar as regras existentes, por favor execute o comando fwlist-allrules

onos> fwlist-allrules
Listando 1 regras:
--------------------------------------------
DstRule: 192.168.0.5
--------------------------------------------

onos> flows|grep  -A 2 firewall
   id=280000000e1781, state=PENDING_ADD, bytes=0, packets=0, duration=0,
priority=129, tableId=0 appId=proactive.firewall.app,
      selector=[IPV4_DST{ip=10.0.0.5/32}, ETH_TYPE{ethType=ipv4}]
      treatment=DefaultTrafficTreatment{immediate=[DROP{}], deferred=[],
transition=None, cleared=false, metadata=null}
--
   id=280000000e1781, state=PENDING_ADD, bytes=0, packets=0, duration=0,
priority=129, tableId=0 appId=proactive.firewall.app,
      selector=[IPV4_DST{ip=10.0.0.5/32}, ETH_TYPE{ethType=ipv4}]
      treatment=DefaultTrafficTreatment{immediate=[DROP{}], deferred=[],
transition=None, cleared=false, metadata=null}
--
   id=280000000e1781, state=PENDING_ADD, bytes=0, packets=0, duration=0,
priority=129, tableId=0 appId=proactive.firewall.app,
      selector=[IPV4_DST{ip=10.0.0.5/32}, ETH_TYPE{ethType=ipv4}]
      treatment=DefaultTrafficTreatment{immediate=[DROP{}], deferred=[],
transition=None, cleared=false, metadata=null}
--
   id=280000000e1781, state=PENDING_ADD, bytes=0, packets=0, duration=0,
priority=129, tableId=0 appId=proactive.firewall.app,
      selector=[IPV4_DST{ip=10.0.0.5/32}, ETH_TYPE{ethType=ipv4}]
      treatment=DefaultTrafficTreatment{immediate=[DROP{}], deferred=[],
transition=None, cleared=false, metadata=null}
```

Lembre que a topologia possuí quatro *switches* e, por isso, foram criadas 4 regras idênticas para o bloqueio de h5, uma em cada *switch* (saída do comando **flows**, acima). Vamos testar o ping novamente:

```
h1> ping 192.168.0.5 -c1
PING 192.168.0.5 (192.168.0.5) 56(84) bytes of data.
--- 192.168.0.5 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

h2> ping 192.168.0.5 -c1
PING 192.168.0.5 (192.168.0.5) 56(84) bytes of data.
--- 192.168.0.5 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

Remova a regra criada. Podemos apertar a tecla **<tab>**, logo após o comando **fwremove-dstrule,** para que a CLI liste e complete o comando com as regras passíveis de remoção:

```
onos> fwremove-dstrule [<tab>] 192.168.0.5
Configuraçoes do teste de velocidade removidas - Teste Desativado

onos> fwlist-allrules
Listando 0 regras:
-----------------------------------------
-----------------------------------------
```

Sem a regra de bloqueio, verifique que ping voltou a funcionar

```
h2> ping 192.168.0.5 -c2
PING 192.168.0.5 (192.168.0.5) 56(84) bytes of data.
64 bytes from 192.168.0.5: icmp_seq=1 ttl=64 time=181 ms
1 packets transmitted, 1 received, 0% packet loss, time 181ms
```

Para testarmos o bloqueio por portas, verifique se o ssh (porta 22) está em execução no *host* h4 do FIBRE e a porta acessível externamente:

```
h1> nc -vz 192.168.0.4 22
Connection to 192.168.0.4 22 port [tcp/ssh] succeeded!
```

Vamos 'brincar' um pouco com a aplicação de *Firewall*. Para tanto, crie mais algumas regras para verificar o bloqueio pela origem, pelo par de origem e destino e pela porta de destino do fluxo.

```
onos> fwadd-srcrule 192.168.0.2
Bloqueio criado com sucesso para o origem 192.168.0.2 - A porta
informada foi null [null representa todas as portas]
Para listar as regras existentes, por favor execute o comando
fwlist-allrules

onos> fwadd-pairrule 192.168.0.1 192.168.0.5
Bloqueio criado com sucesso para o par de ips - 192.168.0.1 <-->
192.168.0.5
Para listar as regras existentes, por favor execute o comando
fwlist-allrules

onos> fwadd-dstrule 192.168.0.4 22
Bloqueio criado com sucesso para o destino 192.168.0.4 - A porta
informada foi 22 [null representa todas as portas]
Para listar as regras existentes, por favor execute o comando
fwlist-allrules

onos> fwlist-allrules
Listando 3 regras:
-----------------------------------------
SrcRule: 192.168.0.2
DstRule: 192.168.0.4:22
SrcDstRule: 192.168.0.1<-->192.168.0.5
-----------------------------------------
```

Verifique que as regras foram devidamente criadas (**fwlist_allrules)** e que a porta 22 do *host* h4 foi bloqueada corretamente.

```
h1> nc -vz 192.168.0.4 22
nc: connect to 192.168.0.4 port 22 (tcp) failed: Connection refused
```

Vamos realizar alguns *pings* para verificar os bloqueios que realizamos nas demais regras criadas acima:

```
h1> ping 192.168.0.5 -c1
PING 192.168.0.5 (192.168.0.5) 56(84) bytes of data.
^C
--- 192.168.0.5 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

h3> ping 192.168.0.5 -c1
PING 192.168.0.5 (192.168.0.5) 56(84) bytes of data.
64 bytes from 192.168.0.5: icmp_seq=1 ttl=64 time=43.0 ms
--- 192.168.0.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
```

```
h2> ping 192.168.0.4 -c1
PING 192.168.0.4 (192.168.0.4) 56(84) bytes of data.
^C
--- 192.168.0.4 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

Repare que a comunicação de h1 com h5 foi bloqueada pela regra de origem e destino criada. No entanto, o *host* h3 consegue se comunicar com h5, conforme esperado. Por sua vez, h2 não consegue se comunicar com nenhum outro *host*, devido à regra de bloqueio na origem.

Podemos verificar as regras de fluxo criadas pela aplicação para ver os campos aplicados à cada *match* das regras criadas. Como as regras na aplicação são criadas em todos os *switches*, é suficiente focar apenas nas regras que a aplicação instalou em um *switch*). Dessa forma, liste os dispositivos da rede e escolha qualquer DPID disponível (no caso abaixo, escolhemos o *switch* com DPID **of:00000000000000a4**)

Lembre-se que os DPID são diferentes em cada *slice* do FIBRE.

```
onos> devices
id=of:00000000000000a1, available=true, role=MASTER, type=SWITCH, mfr=Nicira,
Inc., hw=Open vSwitch, sw=2.1.3, serial=None, protocol=OF_10,
id=of:00000000000000a2, available=true, role=MASTER, type=SWITCH, mfr=Nicira,
Inc., hw=Open vSwitch, sw=2.1.3, serial=None, protocol=OF_10,
id=of:00000000000000a3, available=true, role=MASTER, type=SWITCH, mfr=Nicira,
Inc., hw=Open vSwitch, sw=2.1.3, serial=None, protocol=OF_10,
id=of:00000000000000a4, available=true, role=MASTER, type=SWITCH, mfr=Nicira,
Inc., hw=Open vSwitch, sw=2.1.3, serial=None, protocol=OF_10,

onos> flows PENDING_ADD of:00000000000000a4
… … … … …
id=280000000e1781, state=PENDING_ADD, bytes=0, packets=0, duration=0,
priority=129, tableId=0 appId=proactive.firewall.app, payLoad=null
selector=[IPV4_DST{ip=10.0.0.5/32}, ETH_TYPE{ethType=ipv4},
IPV4_SRC{ip=10.0.0.1/32}] treatment=DefaultTrafficTreatment{immediate=[DROP{}],
deferred=[], transition=None, cleared=false, metadata=null}

id=280000000e1781, state=PENDING_ADD, bytes=0, packets=0, duration=0,
priority=129, tableId=0 appId=proactive.firewall.app, payLoad=null
selector=[TCP_DST{tcpPort=80}, IP_PROTO{protocol=6}, IPV4_DST{ip=10.0.0.4/32},
ETH_TYPE{ethType=ipv4}] treatment=DefaultTrafficTreatment{immediate=[DROP{}],
deferred=[], transition=None, cleared=false, metadata=null}

id=280000000e1781, state=PENDING_ADD, bytes=0, packets=0, duration=0,
priority=129, tableId=0 appId=proactive.firewall.app, payLoad=null
selector=[ETH_TYPE{ethType=ipv4}, IPV4_SRC{ip=10.0.0.2/32}]
treatment=DefaultTrafficTreatment{immediate=[DROP{}], deferred=[],
transition=None, cleared=false, metadata=null}
```

Na saída acima, são mostradas as regras de bloqueio de origem/destino, de destino/porta e de origem, respectivamente. Em todas as regras, o tratamento (*action*) a ser tomada é *DROP*.

Para finalizar, remova todas as regras criadas com o comando **fwremove-allrules**

```
onos> fwremove-allrules
Todas as regras do Firewall foram removidas. Não há bloqueios ativos.

onos> fwlist-allrules
Listando 3 regras:
-----------------------------------------
SrcRule: 192.168.0.2
DstRule: 192.168.0.4:80
SrcDstRule: 192.168.0.1<-->192.168.0.5
-----------------------------------------
```

Note que as regras não foram removidas. Para demonstrar um pouco sobre como os comandos da CLI podem ser construídos e integrados ao código, deixamos parte da funcionalidade da função de remover todas as regras **não implementada**.

Corrigindo o problema com o código

Siga a instruções contidas nos comentários marcados com "**TODO Lab1: Manipulando Comandos (Remover todas as Regras)**" nos códigos da definição do comando(***FwRules_RemoveALL.java***) e na implementação da funcionalidade de remoção em si (***ProactiveFirewall.java,*** **linha 201**).

Testando novamente

Basicamente, o que fizemos foi limpar os 3 registros de regras e utilizar o *flowRuleService* do ONOS para remover as regras já existentes. Corrigido o código de remoção, devemos compilar e reinstalar a aplicação no ONOS:

```
#> mvn clean install && onos-app $( ipdocker onos1) reinstall! \
target/proactive-firewall-1.0-SNAPSHOT.oar
```

Quando reinstalamos uma aplicação, o ONOS internamente desativa a aplicação carregada anteriormente e ativa a nova. Na desativação da aplicação (linhas 77 a 84) removemos as regras instaladas pelo *firewall* e limpamos o conteúdo dos registro de regras. Por isso, após a reinstalação, não há nenhuma regra na tabela de fluxos dos *switches*.

Crie algumas regras novamente, para removê-las depois e verificar se o comando **fwremove-allrules** está funcionando após as modificações realizadas:

```
** Dica: Assim como em um shell, você pode utilizar o comando   ** **
ctrl+r para realizar uma pesquisa reversa no histórico de    ** **
comandos da CLI do ONOS. Utilize o ctrl+r para procurar      **
** pelas palavras src, dst e pair para criar facilmente as      **
** regras novamente                                             **

onos> fwlist-allrules
Listando 0 regras:
-------------------------------------------
-------------------------------------------

onos> fwadd-srcrule 192.168.0.2        (ctrl+r)
Bloqueio criado com sucesso para o origem 192.168.0.2 - A porta
informada foi null [null representa todas as portas]
Para listar as regras existentes, por favor execute o comando
fwlist-allrules

onos> fwadd-dstrule 192.168.0.4 80     (ctrl+r)
Bloqueio criado com sucesso para o destino 192.168.0.4 - A porta
informada foi 80 [null representa todas as portas]
Para listar as regras existentes, por favor execute o comando
fwlist-allrules

onos> fwadd-pairrule 192.168.0.1 192.168.0.5
Bloqueio criado com sucesso para o par de ips - 192.168.0.1 <-->
192.168.0.5
Para listar as regras existentes, por favor execute o comando
fwlist-allrules

onos> fwlist-allrules
Listando 3 regras:
-------------------------------------------
SrcRule: 192.168.0.2
DstRule: 192.168.0.4:80
SrcDstRule: 192.168.0.1<-->192.168.0.5
-------------------------------------------

onos> fwremove-allrules
Todas as regras do Firewall foram removidas.

onos> fwlist-allrules
Listando 0 regras:
-------------------------------------------
-------------------------------------------
```

Para finalizarmos, execute um ping de h1 para h5 para verificar que o mesmo voltou a funcionar. Veja também que não há mais regras instaladas pela aplicação.

```
onos> flows | grep firewall

mininet> h1 ping h5 -c1
PING 10.0.0.5 (192.168.0.5) 56(84) bytes of data.
64 bytes from 192.168.0.5: icmp_seq=1 ttl=64 time=6.06 ms
--- 192.168.0.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 6.060/6.060/6.060/0.000 ms
```

Finalizando o ambiente do FIBRE

Não usaremos o FIBRE para as próximas práticas pelo fato de que necessitamos do **OpenFlow 1.3**, cujo suporte não é disponível no FIBRE, devido à limitações em sua arquitetura (mais especificamente devido ao uso do *FlowVisor*).

Finalize o projeto criado no FIBRE ou, pelo o menos, desconecte-o do controlador.

Desativando a aplicação

Por fim, devemos desativar a aplicação de Firewall, para que ela não interfira com as próximas práticas. Remova a informações da topologia no ONOS também.

```
onos> app deactivate proactive.firewall.app

onos> app deactivate org.onosproject.fwd

onos> wipe-out please
```