



UNIVERSIDADE DO MINHO

LABORATÓRIOS DE ENGENHARIA INFORMÁTICA

PROJETO 36

# PRÓTOTIPO PARA DETEÇÃO DE CIBERATAQUES USANDO SDN

Norberto Sobral A60982

João Freitas A74814

João Mendes A71862

July 28, 2020

# Índice

<b>1</b>	<b>Introdução</b>	<b>iv</b>
<b>2</b>	<b>Conceitos Teóricos</b>	<b>v</b>
2.1	Redes SDN . . . . .	v
2.2	OpenFlow Protocol . . . . .	v
2.3	FloodLight . . . . .	vi
2.4	IDS . . . . .	vi
<b>3</b>	<b>Desenvolvimento</b>	<b>viii</b>
3.1	Topologia . . . . .	viii
3.2	Metodologia . . . . .	x
3.2.1	Load Balancing . . . . .	x
3.2.2	Comunicação IDS-Controlador . . . . .	x
3.2.3	Comunicação Controlador-Switch . . . . .	xi
3.3	Como montar o Modelo . . . . .	xii
3.4	Como executar . . . . .	xiv
<b>4</b>	<b>Resultados</b>	<b>xv</b>
4.1	Benigno . . . . .	xv
4.2	Maligno . . . . .	xvi
<b>5</b>	<b>Trabalho Futuro</b>	<b>xviii</b>
<b>6</b>	<b>Conclusão</b>	<b>xix</b>

# Índice de Imagens

2.1	Interação do Floodlight. . . . .	vi
3.1	Topologia1. . . . .	viii
3.2	Topologia2. . . . .	viii
3.3	Alertas destination unreachable port. . . . .	xiv
4.1	Execução do comando tcpreplay com dataset benigno. . . . .	xv
4.2	Conteúdo da pasta /tmp. . . . .	xv
4.3	Execução do comando tcpreplay com dataset maligno. . . . .	xvi
4.4	Ficheiro snort.log.*****. . . . .	xvii
4.5	Alertas. . . . .	xvii

# Acrónimos

*SDN* Redes Definidas por Software

*IDS* Intrusion Detection System

# Chapter 1

## Introdução

Embora o conceito de Software-Defined Networking *SDN* tenha surgido no contexto do encaminhamento de tráfego, nos dias de hoje tem vários âmbitos de aplicação, nomeadamente na área da segurança. Tirando partido da visão integrada e da programabilidade que um controlador *SDN* permite, a arquitetura *SDN* abriu portas para o desenvolvimento de novas aplicações com o objetivo de mitigar ciberataques.

No sentido de definir uma solução eficiente para melhorar a segurança de redes na dissertação com o título “Load Balancing Framework for Security in SDN context”, redigida pelo Bruno Machado, foi criado um protótipo envolvendo um controlador *SDN* que interage com vários Switches e Intrusion Detection Systems (*IDSs*) para balancear a detecção de ataques entre os *IDSs* e o tráfego na própria rede.

# Chapter 2

## Conceitos Teóricos

### 2.1 Redes SDN

*SDN* é um conceito da separação física do plano de controlo da rede, do plano de encaminhamento. Os nós de redes (*switches*) são programados por uma entidade central (*controller*) através de um protocolo bem definido (OpenFlow). Os *switches* fazem o encaminhamento de acordo com as tabelas (*flow tables*) que são preenchidas pelo *controller*.

Benefícios das redes *SDN*:

- Arquitetura é totalmente independente dos fabricantes: O controlador *SDN* gere qualquer dispositivo de rede, independentemente do fabricante, desde que esteja habilitado com o OpenFlow.
- Simplicidade na conceção e operação de rede: É possível desenvolver ferramentas para automatizar muitas tarefas que são feitas atualmente de forma manual.
- Maior fiabilidade e segurança: A automação reduz a probabilidade de falha de configuração ou inconsistência de políticas.
- Melhora a experiência de utilização.

### 2.2 OpenFlow Protocol

OpenFlow é um protocolo de comunicação, que nasceu de uma tese de doutoramento de um estudante, da Universidade de Stanford, Martin Casado, considerado um dos primeiros padrões da *SDN* que permite aos controladores de rede determinarem o caminho dos pacotes de rede ao longo de uma rede de *switches*.

Este protocolo permite, também, a administração remota de tabelas de encaminhamento de pacotes de um *switch* da camada de rede (*layer 3*) por adicionar, modificar e remover regras e ações de correspondência de pacotes.

## 2.3 FloodLight

Floodlight é um controlador de SDN, desenvolvido por uma comunidade de programadores, que utiliza o protocolo OpenFlow para controlar o fluxo de tráfego numa rede SDN. Este controlador é responsável por manter as regras da rede e fornecer as instruções necessárias à infraestrutura subjacente sobre como o tráfego terá de ser tratado.

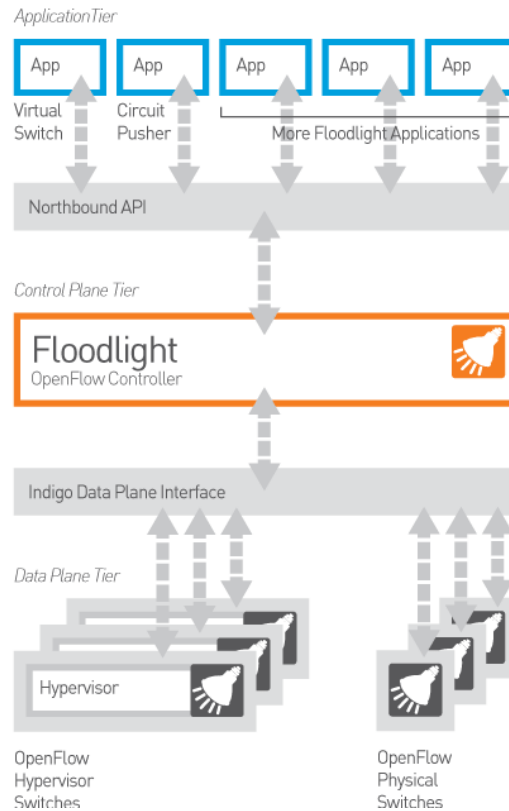


Figure 2.1: Interação do Floodlight.

## 2.4 IDS

Um Intrusion Detection System(IDS) é uma parte de software ou hardware que monitoriza o tráfego na rede de forma a detetar atividade pouco usual ou suspeita, mandando alertas ao administrador em caso de intrusão. Existem dois tipos de IDSs: Network Intrusion Detection System(NIDS) e Host Intrusion Detection System(HIDS). Num HIDS são examinados eventos que chegam apenas a um dispositivo, sendo que num NIDS são examinados os eventos que aconteçam em toda a rede. Neste trabalho usamos o Snort, pois é compatível com o Linux e capaz de ser um NIDS.

**Snort:** foi criado pela Cisco. O sistema pode ser corrido em três diferentes modos: modo sniffer, registador de pacotes, e deteção de intrusão assim como implementação de estratégias de defesa atuando como um IDS e IPS(Intrusion Prevention System). Quando atua como IDS, um módulo de análise aplica um conjunto de regras ao tráfego que passa pela rede. O Snort é software open-source com uma comunidade ativa que desenvolve vários conjuntos de regras para vários casos. As regras

detetam eventos como stealth port scan, buffer overflow attack, tentativas de DDoS, etc. Os métodos de detecção dependem das regras específicas que sejam usadas, e incluem métodos de assinatura de ataques já conhecidos na base de dados como maneira de detecção de ataques na rede, e de anomalia que detetam comportamentos anormais na rede.



## Chapter 3

# Desenvolvimento

### 3.1 Topologia

Dentro do projeto tínhamos duas topologias disponíveis:

A topologia1 que era a mais simples das duas.

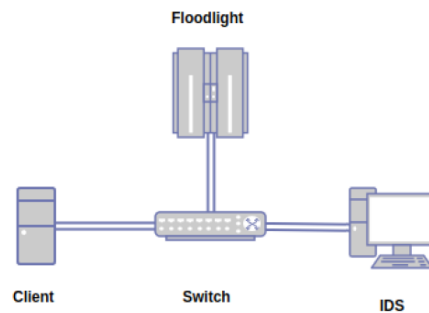


Figure 3.1: Topologia1.

E a topologia2 que devido a ter três switches permite testar algoritmos de **load-balancing** e como tal escolhemos esta para efetuar os nossos testes.

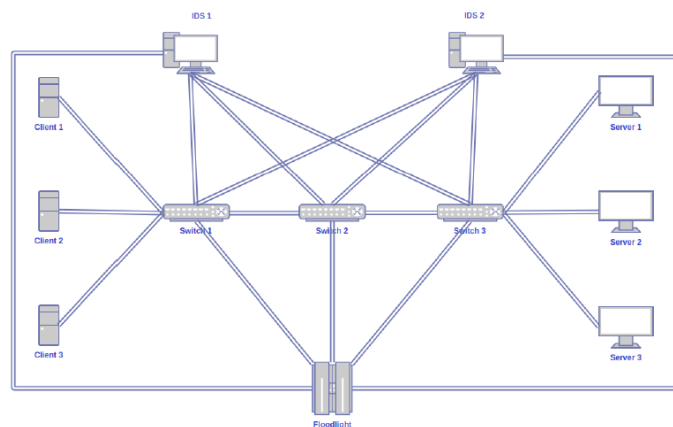


Figure 3.2: Topologia2.

Esta topologia contém os seguintes elementos.

- Três **Clientes**-*10.0.0.1->10.0.0.3*
- Três **Servidores**-*10.0.0.4->10.0.0.6*
- Dois **IDS** com três interfaces cada-*10.0.0.7->10.0.0.12*
- Três **OpenVSwitches-Switches** a correr OpenFlow
- Controlador **Floodlight**

Esta topologia contém um **SPOF** (Single Point of Failure) no **switch2** e como tal somos da opinião que devia ser adicionada uma ligação direta entre o **switch1** e o **switch3**. Esta solução vai introduzir um *loop* neste anel de switches logo, de forma evitar essa situação, é necessário utilizar o *Spanning Tree Protocol* em todas as **Bridges** de todos os **switches**.

Isso pode ser feito através dos seguintes comandos dentro da mininet:

```
1 s1 ovs-vsctl set bridge 's1' stp_enable=true
2 s1 ovs-vsctl set bridge 's2' stp_enable=true
3 s1 ovs-vsctl set bridge 's3' stp_enable=true
4 s2 ovs-vsctl set bridge 's1' stp_enable=true
5 s2 ovs-vsctl set bridge 's2' stp_enable=true
6 s2 ovs-vsctl set bridge 's3' stp_enable=true
7 s3 ovs-vsctl set bridge 's1' stp_enable=true
8 s3 ovs-vsctl set bridge 's2' stp_enable=true
9 s3 ovs-vsctl set bridge 's3' stp_enable=true
```

## 3.2 Metodologia

Todos os pacotes/fluxos são enviados de um cliente para um servidor através de switches, sendo que esses switches irão enviar os pacotes/fluxos, em **Round-Robin**, para os IDS para serem analisados. Os pacotes/fluxos irão ser bloqueados ou não de acordo com o resultado dessa análise.

Um switch irá estar responsável por um fluxo, esse fluxo é enviado ao IDS que irá dizer ao controlador se esse fluxo é seguro ou não, o controlador irá enviar mensagem ao switch de acordo com o que o IDS diz. Se um fluxo for benigno é agrupado num "Benign flow group", sendo que este grupo irá ser balanceado de acordo com o menor número de fluxos ativos num dado **Switch**.

### 3.2.1 Load Balancing

A funcionalidade de balanceamento de carga é efectuada no módulo **MACTracker** pela função *bestSwitch*. Esta função é usada para calcular o melhor **Switch** e **IDS** de acordo com o número de fluxos ativos nestes mesmos. Esta solução não é ideal visto que fluxos diferentes podem ter um número de pacotes diferentes, algo que a função não tem em conta. Por outro lado é um algoritmo que consegue escalar bem visto que é simples e tem um tempo de execução baixo.

### 3.2.2 Comunicação IDS-Controlador

A comunicação **Snort-Controlador** é feita através de *Sockets*, mais especificamente, temos as instâncias do **Snort** a enviar os alertas para o *socket snort\_alert* na pasta */tmp*. Esse *socket* é lido pelo módulo **UnixSocket** que ao ler esses alertas envia-os para o controlador que irá de seguida enviar as instruções de como prosseguir para o **Switch** que está a gerir o fluxo que gerou o alerta.

### 3.2.3 Comunicação Controlador-Switch

Quando um pacote chega a um **Switch** que não pertença a nenhum fluxo uma mensagem *packet in* é enviada do **Switch** para o **Controlador**. A função **processPacketIn** do módulo **MACTracker** irá então escolher o **Switch** e o **IDS** que irão ser responsáveis por este fluxo. Esta função tem dois modos de funcionamento dependendo se encontra o fluxo a que um pacote pertence ou não.

Se encontrar o fluxo a que o pacote pertence então vai simplesmente seguir as acções indicadas nas instruções desse fluxo.

Se não encontrar um fluxo então, como foi mencionado previamente, vão ser escolhidos um **Switch** e **IDS** para serem responsáveis por este novo fluxo. Para além disso, é criado um *match* para que outros pacotes pertencentes a esse fluxo sejam corretamente identificados como tal.

De seguida são criados:

- Dois *buckets*, um com o funcionamento normal e outro com o endereço **MAC** da interface de saída para o **IDS**
- Um **GROUP MOD** é criado com o *match* e essas acções e enviado para o respetivo **Switch**
- Um **FLOW MOD** que exercisa as acções do **GROUP MOD**
- O número de grupos é incrementado e o número de fluxos no switch atualizado

Quando um grupo já existe a prioridade desse mesmo é verificada, essa prioridade reflecte a severidade dos alertas desse fluxo. A prioridade pode variar entre 0 e 4 tendo os seguintes significados:

- **0.** Fluxo sem alertas, é adicionado ao **benignGroups**.
- **1.** Acções do Group Mod são eliminadas e é criado um Flow Mod que elimina todos os pacotes desse fluxo.
- **2.** O Group Modify é enviado para o **Switch**, tráfego passa a ser enviado apenas para IDS, o Flow Mod é criado com 4\* do *hard\_timeout* (*número de segundos que um fluxo vai estar ativo, independentemente da sua atividade*) do default.
- **3.** Funcionamento semelhante ao 4 mas com 4\* o *hard\_timeout* do default.
- **4.** É criado um novo Group Mod no switch respetivo. Um novo FlowMod é criado com o dobro do *hard\_timeout* do default.

### 3.3 Como montar o Modelo

```
1 # Security SDN Module
2 ##### Instalacao
3
4 ##### Opcao A- Usar imagem VMWARE
5 https://www.dropbox.com/s/tcdvuisnp0bn5oh/Xubuntu-18.04.7z
6 Sudo password-root
7 ##### Opcao B- Comecar de raiz
8 # Deve ser utilizado uma distribuicao linux com versao openssl 1.1.1--Ubuntu 18.04 ou
   acima
9 ### Snort - fazer a instalacao seguindo o tutorial presente no link abaixo, onde
   explica tambem como activar as community rules, apos fazer configure tem passar
   ao passo seguinte (substituir o spo_alert_unixsock.c) podendo entao voltar ao
   tutorial de instalacao do snort:
10 ## Versao 2.9.16 GRE
11 https://upcloud.com/community/tutorials/install-snort-ubuntu/
12
13 > O ficheiro Snort/spo_alert_unixsock.c deve substituir o ficheiro com o mesmo nome
   na pasta de instalacao do snort antes de compilar o snort.
14 > Snort_Instalation_Folder/src/output-plugins/
15
16 ### OpenVSwitch instalacao manual
17 ## Versao 2.11.0
18 # Criar pasta ovs no Desktop
19 cd ~/Desktop/ovs/
20 wget https://www.openvswitch.org/releases/openvswitch-2.11.0.tar.gz
21 tar -xvf openvswitch-2.11.0.tar.gz
22
23 ### Instalar Dependencias
24
25 sudo apt install build-essential fakeroot graphviz autoconf automake bzip2 debhelper
   dh-autoreconf libssl-dev libtool openssl procps python-all python-qt4 python-
   twisted-conch python-zopeinterface module-assistant dkms make libc6-dev python-
   argparse uuid-runtime netbase kmod python-twisted-web iproute2 ipsec-tools
   openvswitch-switch racoon tcpreplay
26
27 sudo dpkg-checkbuilddeps
28 ### Gerar ficheiros OpenVSwitch.deb
29 # Correr testes
30 sudo fakeroot debian/rules binary
31 # Sem correr testes
32 sudo DEB_BUILD_OPTIONS='parallel=8 nocheck' fakeroot debian/rules binary
33
```

```

34 ### Instalar os ficheiros gerados, nao e necessario instalar todos
35 cd ~/Desktop
36 sudo dpkg -i ../openvswitch-common_2.11.0-1_amd64.deb
37 sudo dpkg -i ../openvswitch-switch_2.11.0-1_amd64.deb
38
39 ### Confirmar que o daemon openvswitch esta a executar
40 ps -ef | grep ovs
41 # Se nao tiver, executar manualmente atraves do comando
42 sudo ovs-vsitchd
43 ## Adicionar link logico para a mininet conseguir detetar o controlador
44 sudo ln -s /usr/bin/ovs-testcontroller /usr/bin/ovs-controller
45
46 ### Mininet
47 ## Versao 2.2.2
48 Seguir opcao 2 (Native Installation from source) na seguinte pagina
49 http://mininet.org/download/
50
51 ### UnixSocket
52
53 > The UnixSocket.py requiere um certificado que pode ser gerado atraves do comando:
54 '''bash
55 openssl req -newkey rsa:2048 -new -nodes -x509 -days 3650 -keyout key.pem -out cert.
    pem
56
57 cat key.pem >> cert.pem
58 '''
59 ### Floodlight
60
61 # Instalar java
62 sudo apt-get default-jre
63 # Instalar Eclipse
64 sudo snap install --classic eclipse
65 # Para executar Eclipse basta escrever eclipse no terminal
66 # Apos a instalacao com sucesso, importar o projecto do floodlight contido no github
    para o eclipse e fazer a alteracao sugerida abaixo.
67
68 > Na linha 764 do projecto floodlight/src/main/java/net/floodlightcontroller/
    mactracker/MACTracker.java substituir \$Password\$ com password do root.
69 > Nota: Este processo pode ser otimizado ao nivel da seguranca e eficiencia. O
    Floodlight apresenta um servico de estatisticas de fluxo que pode ser usado
    requerendo um pouco de trabalho na sua implementacao.

```

### 3.4 Como executar

A máquina virtual que nós fornecemos já contém duas capturas para testar, uma captura que contém só fluxos benignos e outra que contém fluxos malignos.

Se for necessário, pode-se obter novas capturas através do **WireShark**. Essas novas capturas têm de ser editadas através do comando **sudo tcprewrite -infile=input.pcap -outfile=output.pcap -srcipmap=0.0.0.0/0:IPFonte -dstipmap=0.0.0.0/0:IPDestino -enet-dmac=00:00:00:00:00:01 -enet-smac=00:00:00:00:00:04** e podem ser divididas através do comando **editcap -c {nº de pacotes} input.pcap output.pcap**.

1. Executar o **Floodlight** no eclipse, valores de *Default*, *Priority\_1* e *Priority\_2* podem ser alterados no módulo **MACTracker**.
2. Executar o módulo **UnixSocket-python UnixSocket.py**.
3. Inicializar a topologia através da mininet-sudo **python2 Topologia.py**.
4. **Opcional**: Testar a conexão na *mininet* através do comando-pingall.
5. Executar o comando **tcpreplay** no host pretendido. Exemplo **cliente1 tcpreplay -intf1=cliente1-eth0 input.pcap**
6. Logs resultantes são criados na pasta */tmp* e podem ser lidos através do comando **sudo snort -r snort.log.\*\*\*\***

**Nota:** As instâncias do snort a correr nos ids, encontram-se a executar como daemon, desta forma é impossível verificar o número de pacotes analisados por cada um dos ids. Para obter esse relatório é necessário correr as instâncias com o comando **sudo snort -i ids1-eth0 -A unsock -c /etc/snort/snort.conf** nos xterms e assim o relatório será imprimido no respectivo terminal.

Para além disso, durante a execução é possível visualizar a existência de um alerta referente a *'destination unreachable port unreachable packet detected'* e que tem origem numa porta que se encontra fechada.

```
.....
PROTOCOL-ICMP destination unreachable port unreachable packet detected|10.0.0.4|10.0.0.1|1|1|402|16|3|07/08/20 - 11:22:41
PROTOCOL-ICMP destination unreachable port unreachable packet detected|10.0.0.4|10.0.0.1|1|1|402|16|3|07/08/20 - 11:22:42
PROTOCOL-ICMP destination unreachable port unreachable packet detected|10.0.0.4|10.0.0.1|1|1|402|16|3|07/08/20 - 11:22:42
PROTOCOL-ICMP destination unreachable port unreachable packet detected|10.0.0.4|10.0.0.1|1|1|402|16|3|07/08/20 - 11:22:43
PROTOCOL-ICMP destination unreachable port unreachable packet detected|10.0.0.4|10.0.0.1|1|1|402|16|3|07/08/20 - 11:22:43
PROTOCOL-ICMP destination unreachable port unreachable packet detected|10.0.0.4|10.0.0.1|1|1|402|16|3|07/08/20 - 11:22:43
PROTOCOL-ICMP destination unreachable port unreachable packet detected|10.0.0.4|10.0.0.1|1|1|402|16|3|07/08/20 - 11:22:43
PROTOCOL-ICMP destination unreachable port unreachable packet detected|10.0.0.4|10.0.0.1|1|1|402|16|3|07/08/20 - 11:22:44
PROTOCOL-ICMP destination unreachable port unreachable packet detected|10.0.0.4|10.0.0.1|1|1|402|16|3|07/08/20 - 11:22:44
PROTOCOL-ICMP destination unreachable port unreachable packet detected|10.0.0.4|10.0.0.1|1|1|402|16|3|07/08/20 - 11:22:44
PROTOCOL-ICMP destination unreachable port unreachable packet detected|10.0.0.4|10.0.0.1|1|1|402|16|3|07/08/20 - 11:22:44
PROTOCOL-ICMP destination unreachable port unreachable packet detected|10.0.0.4|10.0.0.1|1|1|402|16|3|07/08/20 - 11:22:44
PROTOCOL-ICMP destination unreachable port unreachable packet detected|10.0.0.4|10.0.0.1|1|1|402|16|3|07/08/20 - 11:22:44
PROTOCOL-ICMP destination unreachable port unreachable packet detected|10.0.0.4|10.0.0.1|1|1|402|16|3|07/08/20 - 11:22:44
PROTOCOL-ICMP destination unreachable port unreachable packet detected|10.0.0.4|10.0.0.1|1|1|402|16|3|07/08/20 - 11:22:45
PROTOCOL-ICMP destination unreachable port unreachable packet detected|10.0.0.4|10.0.0.1|1|1|402|16|3|07/08/20 - 11:22:45
.....
```

Figure 3.3: Alertas destination unreachable port.

Este alerta pode ser removido comentando a respectiva regra no ficheiros das *community rules*.

## Chapter 4

# Resultados

De seguida seguem os resultados obtidos com partes dos dois **Datasets**.

### 4.1 Benigno

Como era de esperar os o **Dataset** benigno não acusa tráfego maligno e como tal os ficheiros **snort.log.\*\*\*\*** estão vazios.

```
mininet> client1 tcpreplay --intf1=client1-eth0 safe0
Actual: 100 packets (17898 bytes) sent in 94.66 seconds
Rated: 189.0 Bps, 0.001 Mbps, 1.05 pps
Flows: 9 flows, 0.09 fps, 59 flow packets, 41 non-flow
Statistics for network device: client1-eth0
    Successful packets:      100
    Failed packets:         0
    Truncated packets:      0
    Retried packets (ENOBUFS): 0
    Retried packets (EAGAIN): 0
```

Figure 4.1: Execução do comando tcpreplay com dataset benigno.

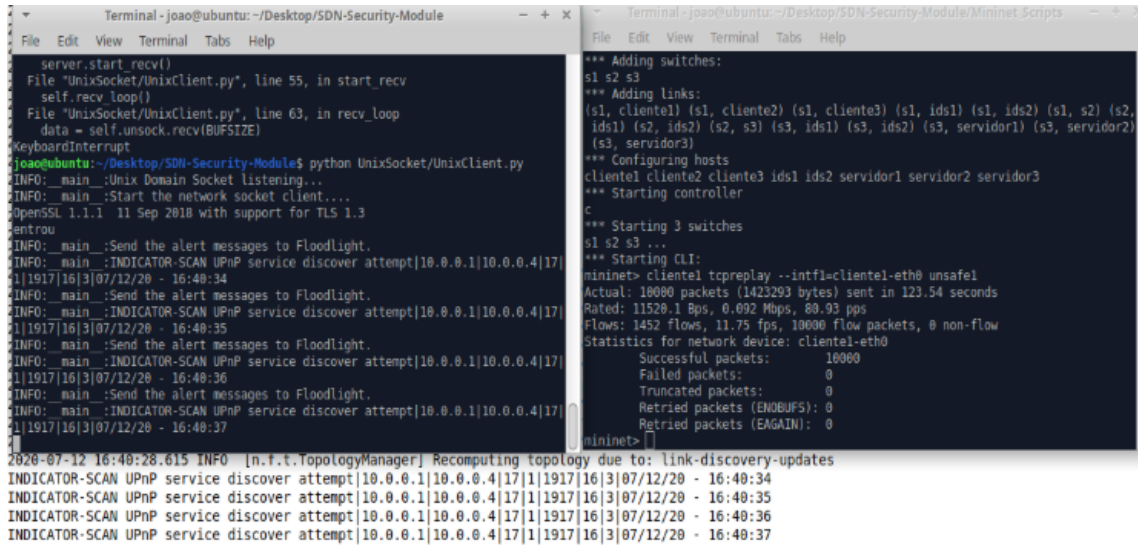
```
-rw----- 1 root root 0 Jul 12 16:27 snort.log.1594567666
-rw----- 1 root root 0 Jul 12 16:27 snort.log.1594567667
-rw----- 1 root root 0 Jul 12 16:27 snort.log.1594567668
-rw----- 1 root root 0 Jul 12 16:27 snort.log.1594567669
```

Figure 4.2: Conteúdo da pasta /tmp.



## 4.2 Maligno

Por outro lado, na segunda parte do **Dataset** maligno, temos que o **Snort** detecta quatro pacotes possivelmente malignos, sendo os respectivos alertas enviados para o **Socket** e consequentemente para o **Controlador**.



```
Terminal - joao@ubuntu: ~/Desktop/SDN-Security-Module
server.start_recv()
File "UnixSocket/UnixClient.py", line 55, in start_recv
self.recv_loop()
File "UnixSocket/UnixClient.py", line 63, in recv_loop
data = self.unsock.recv(BUFSIZE)
KeyboardInterrupt
joao@ubuntu:~/Desktop/SDN-Security-Module$ python UnixSocket/UnixClient.py
INFO: main: Unix Domain Socket listening...
INFO: main: Start the network socket client....
OpenSSL 1.1.1 11 Sep 2018 with support for TLS 1.3
Entrou
INFO: main: Send the alert messages to Floodlight.
INFO: main: INDICATOR-SCAN UPnP service discover attempt|10.0.0.1|10.0.0.4|17|
1|1917|16|3|07/12/20 - 16:40:34
INFO: main: Send the alert messages to Floodlight.
INFO: main: INDICATOR-SCAN UPnP service discover attempt|10.0.0.1|10.0.0.4|17|
2|1917|16|3|07/12/20 - 16:40:35
INFO: main: Send the alert messages to Floodlight.
INFO: main: INDICATOR-SCAN UPnP service discover attempt|10.0.0.1|10.0.0.4|17|
3|1917|16|3|07/12/20 - 16:40:36
INFO: main: Send the alert messages to Floodlight.
INFO: main: INDICATOR-SCAN UPnP service discover attempt|10.0.0.1|10.0.0.4|17|
4|1917|16|3|07/12/20 - 16:40:37

2020-07-12 16:40:28.615 INFO [n.f.t.TopologyManager] Recomputing topology due to: link-discovery-updates
INDICATOR-SCAN UPnP service discover attempt|10.0.0.1|10.0.0.4|17|1|1917|16|3|07/12/20 - 16:40:34
INDICATOR-SCAN UPnP service discover attempt|10.0.0.1|10.0.0.4|17|1|1917|16|3|07/12/20 - 16:40:35
INDICATOR-SCAN UPnP service discover attempt|10.0.0.1|10.0.0.4|17|1|1917|16|3|07/12/20 - 16:40:36
INDICATOR-SCAN UPnP service discover attempt|10.0.0.1|10.0.0.4|17|1|1917|16|3|07/12/20 - 16:40:37

Terminal - joao@ubuntu: ~/Desktop/SDN-Security-Module/Mininet-Scripts
*** Adding switches:
s1 s2 s3
*** Adding links:
(s1, cliente1) (s1, cliente2) (s1, cliente3) (s1, ids1) (s1, ids2) (s1, s2) (s2,
ids1) (s2, ids2) (s2, s3) (s3, ids1) (s3, ids2) (s3, servidor1) (s3, servidor2)
(s3, servidor3)
*** Configuring hosts
cliente1 cliente2 cliente3 ids1 ids2 servidor1 servidor2 servidor3
*** Starting controller
c
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet> cliente1 tcpreplay --intf=cliente1-eth0 unsafel
Actual: 10000 packets (1423293 bytes) sent in 123.54 seconds
Rated: 11520.1 Bps, 0.092 Mbps, 80.93 pps
Flows: 1452 flows, 11.75 fps, 10000 flow packets, 0 non-flow
Statistics for network device: cliente1-eth0
Successful packets: 10000
Failed packets: 0
Truncated packets: 0
Retried packets (ENOBUFS): 0
Retried packets (EAGAIN): 0
mininet>
```

Figure 4.3: Execução do comando tcpreplay com dataset maligno.

Como era de esperar os ficheiros **snort.log**.\*\*\* já têm conteúdo.

```

Packet I/O Totals:
Received:      4
Analyzed:     4 (100.000%)
Dropped:      0 ( 0.000%)
Filtered:     0 ( 0.000%)
Outstanding:  0 ( 0.000%)
Injected:     0

Breakdown by protocol (includes rebuilt packets):
Eth:          4 (100.000%)
VLAN:        0 ( 0.000%)
IP4:         4 (100.000%)
Frag:        0 ( 0.000%)
ICMP:        0 ( 0.000%)
UDP:         4 (100.000%)
TCP:         0 ( 0.000%)
IP6:         0 ( 0.000%)
IP6 Ext:     0 ( 0.000%)
IP6 Opts:    0 ( 0.000%)
Frag6:       0 ( 0.000%)
ICMP6:       0 ( 0.000%)
UDP6:        0 ( 0.000%)
TCP6:        0 ( 0.000%)
Teredo:      0 ( 0.000%)
ICMP-IP:     0 ( 0.000%)
IP4/IP4:     0 ( 0.000%)
IP4/IP6:     0 ( 0.000%)
IP6/IP6:     0 ( 0.000%)
IP6/IP4:     0 ( 0.000%)
GRE:         0 ( 0.000%)
GRE Eth:     0 ( 0.000%)
GRE VLAN:   0 ( 0.000%)
GRE IP4:    0 ( 0.000%)
GRE IP6:    0 ( 0.000%)
GRE IP6 Ext: 0 ( 0.000%)
GRE PPTP:   0 ( 0.000%)
GRE ARP:    0 ( 0.000%)
GRE IPX:    0 ( 0.000%)
GRE Loop:   0 ( 0.000%)
MPLS:       0 ( 0.000%)
ARP:        0 ( 0.000%)
IPX:        0 ( 0.000%)
Eth Loop:   0 ( 0.000%)
Eth Disc:   0 ( 0.000%)
IP4 Disc:   0 ( 0.000%)
IP6 Disc:   0 ( 0.000%)
TCP Disc:   0 ( 0.000%)
UDP Disc:   0 ( 0.000%)
ICMP Disc:  0 ( 0.000%)
All Discard: 0 ( 0.000%)
Other:      0 ( 0.000%)
Bad Chk Sum: 0 ( 0.000%)
Bad TTL:    0 ( 0.000%)
SS G 1:     0 ( 0.000%)
SS G 2:     0 ( 0.000%)
Total:      4

Snort exiting

```

Figure 4.4: Ficheiro snort.log.\*\*\*\*\*.

Podemos também verificar as diferentes prioridades nos alertas, sendo este resultado obtido através da terceira parte do **Dataset** maligno.

```

INDICATOR-SHELLCODE x86 inc ecx NOOP 10.0.0.1|10.0.0.4|6|1|1394|17|1|07/12/20 - 16:43:42
INDICATOR-SHELLCODE x86 inc ebx NOOP 10.0.0.1|10.0.0.4|6|1|1390|17|1|07/12/20 - 16:43:42
INDICATOR-SHELLCODE x86 inc ecx NOOP 10.0.0.1|10.0.0.4|6|1|1394|17|1|07/12/20 - 16:43:42
INDICATOR-SHELLCODE x86 inc ebx NOOP 10.0.0.1|10.0.0.4|6|1|1390|17|1|07/12/20 - 16:43:42
INDICATOR-SHELLCODE x86 inc ecx NOOP 10.0.0.1|10.0.0.4|6|1|1394|17|1|07/12/20 - 16:43:43
INDICATOR-SHELLCODE x86 inc ebx NOOP 10.0.0.1|10.0.0.4|6|1|1390|17|1|07/12/20 - 16:43:43
INDICATOR-SHELLCODE x86 inc ecx NOOP 10.0.0.1|10.0.0.4|6|1|1394|17|1|07/12/20 - 16:43:43
INDICATOR-SHELLCODE x86 inc ebx NOOP 10.0.0.1|10.0.0.4|6|1|1390|17|1|07/12/20 - 16:43:43
INDICATOR-SHELLCODE x86 inc ecx NOOP 10.0.0.1|10.0.0.4|6|1|1394|17|1|07/12/20 - 16:43:43
INDICATOR-SHELLCODE x86 inc ebx NOOP 10.0.0.1|10.0.0.4|6|1|1390|17|1|07/12/20 - 16:43:43
2020-07-12 16:43:43.784 INFO [n.f.l.i.LinkDiscoveryManager] Sending LLDP packets out of
2020-07-12 16:43:58.801 INFO [n.f.l.i.LinkDiscoveryManager] Sending LLDP packets out of
2020-07-12 16:44:13.814 INFO [n.f.l.i.LinkDiscoveryManager] Sending LLDP packets out of
2020-07-12 16:44:28.826 INFO [n.f.l.i.LinkDiscoveryManager] Sending LLDP packets out of
2020-07-12 16:44:29.262 INFO [n.f.t.TopologyManager] Recomputing topology due to: link-c
2020-07-12 16:44:43.935 INFO [n.f.l.i.LinkDiscoveryManager] Sending LLDP packets out of
2020-07-12 16:44:58.947 INFO [n.f.l.i.LinkDiscoveryManager] Sending LLDP packets out of
INDICATOR-SCAN UPnP service discover attempt 10.0.0.1|10.0.0.4|17|1|1917|16|3|07/12/20 - 16:45:08
INDICATOR-SCAN UPnP service discover attempt 10.0.0.1|10.0.0.4|17|1|1917|16|3|07/12/20 - 16:45:08
INDICATOR-SCAN UPnP service discover attempt 10.0.0.1|10.0.0.4|17|1|1917|16|3|07/12/20 - 16:45:08
INDICATOR-SCAN UPnP service discover attempt 10.0.0.1|10.0.0.4|17|1|1917|16|3|07/12/20 - 16:45:08
INDICATOR-SCAN UPnP service discover attempt 10.0.0.1|10.0.0.4|17|1|1917|16|3|07/12/20 - 16:45:10
INDICATOR-SCAN UPnP service discover attempt 10.0.0.1|10.0.0.4|17|1|1917|16|3|07/12/20 - 16:45:10
INDICATOR-SCAN UPnP service discover attempt 10.0.0.1|10.0.0.4|17|1|1917|16|3|07/12/20 - 16:45:10
INDICATOR-SCAN UPnP service discover attempt 10.0.0.1|10.0.0.4|17|1|1917|16|3|07/12/20 - 16:45:11
INDICATOR-SCAN UPnP service discover attempt 10.0.0.1|10.0.0.4|17|1|1917|16|3|07/12/20 - 16:45:11
INDICATOR-SCAN UPnP service discover attempt 10.0.0.1|10.0.0.4|17|1|1917|16|3|07/12/20 - 16:45:11
INDICATOR-SCAN UPnP service discover attempt 10.0.0.1|10.0.0.4|17|1|1917|16|3|07/12/20 - 16:45:12
INDICATOR-SCAN UPnP service discover attempt 10.0.0.1|10.0.0.4|17|1|1917|16|3|07/12/20 - 16:45:12

```

Figure 4.5: Alertas.

## Chapter 5

# Trabalho Futuro

Tal como mencionado previamente achamos que a actual *Topologia2* beneficiaria de uma ligação directa entre o **Switch1** e **Switch3** sendo que o processo de activação da **Spanning Tree Protocol** deveria ser agilizado através da modificação do script de criação da *Topologia2*.

Temos também que o actual *Dataset* é muito limitado e como tal era ideal obter *Datasets* mais variados com ainda mais pacotes.

Achamos também que podia criar-se uma *Topologia* ainda mais extensa, com mais **Switches**, **Clientes** e **IDS's** de forma a representar melhor o mundo real. Sendo estas as nossas sugestões de trabalho futuro, podemos também ter em conta as referenciadas pelo Bruno na sua dissertação são as seguintes:

- Uma das melhorias que poderia ser alcançada é escolher o switch e o IDS responsável pelo mirroring, dependendo do caminho que um pacote deve percorrer até seu destino. Este é, fazer com que o comprimento do caminho seja um fator ao escolher o switch e o IDS responsável pelo mirroring.
- Outro recurso que pode ser adicionado é a utilização dos dados armazenados em relação ao alertas, como a hora dos alertas, o número de *red flags*, etc. Desde o momento em que um alerta for encontrado e associado ao seu respectivo fluxo, ele será considerado para sempre como malicioso. Embora a decisão anterior não esteja errada, os fluxos podem ser considerados como benigno se, por um período de tempo, não mostrar qualquer sinal de comportamento malicioso.
- Os testes para a estrutura foram realizados num ambiente emulado que pode produzir resultados diferentes dos obtidos numa rede real. Então, um dos próximos passos no desenvolvimento dessa estrutura, será a instalação e testes da estrutura numa rede com tráfego real.

## Chapter 6

# Conclusão

Com a realização deste trabalho prático, aumentamos e melhoramos os nossos conhecimentos sobre o conceito de **SDNs** e dos vários softwares que permitem fazer a instalação e configuração desta tipo de arquitectura de rede, tendo conseguido realizar com sucesso a instalação, configuração dos diferentes componentes que constituem a arquitectura.

Ajudou-nos a melhor compreender o funcionamento destas arquitectura emergente, conseguindo assim saber qual o comportamento dos componentes e como estes comunicam entre si. Foi possível saber com detalhe quais os protocolos e formas de comunicação usadas neste sistema de forma a tornar possível este serviço.

Infelizmente o trabalho final não correspondeu às expectativas/objectivos inicialmente propostos visto que o nosso grupo teve imensas dificuldades a colocar o modelo a funcionar correctamente, porém achamos que conseguimos criar algo que está pronto a ser utilizado e expandido no futuro.