JOÃO CARLOS CHIQUIM JUNIOR

UMA ANÁLISE DA SEGURANÇA DO SDN E UM POSSÍVEL ATAQUE À SUA ESTRUTURA

Trabalho de graduação apresentado à disciplina CI075 – Trabalho de Graduação em Redes de Computadores II como requisito à conclusão do curso de Ciência da Computação, Setor de Ciências Exatas da Universidade Federal do Paraná.

Orientador: Luiz Carlos Pessoa Albini.

CURITIBA

2017

RESUMO

Cada vez mais as redes de computadores estão presentes em nosso dia-a-dia, e com isso, sempre devemos nos preocupar com a segurança presente nestas redes. O SDN, ou *Software Defined Network*, é um paradigma não tão novo, mas que começou a ser mais usado e estudado há pouco tempo, e por isso, ainda existem muitos pontos para serem estudados. Para entender melhor o tema, devemos estudar um pouco sobre a segurança do SDN e os principais pontos que devem ser considerados ao pensar em uma rede segura. Por fim, será estudado um modo de elaborar e tentar simular um ataque à estrutura de uma rede SDN, visando verificar a real possibilidade de um ataque do gênero ser bem-sucedido.

Palavras-chave: SDN, *Software Defined Network*, Segurança de redes, Ataque.

ABSTRACT

More and more computer networks are present in our daily lives, and with that, we must increasingly worry about the security present in these networks. The SDN, or *Software Defined Network*, is a not-so-new paradigm, but it has started to be more used and studied recently, so there are still many points to be studied. To a better understand about the theme, we have to study a little about the security of the SDN and the main points that should be considered when thinking about a secure network. Finally, we will study a way to elaborate and try to simulate a model of attack on the structure of an SDN network, to verify the real possibility of a successful attack of the genre.

Key-words: SDN, Software Defined Network, Network security, Attack.

LISTA DE FIGURAS

FIGURA 1 – ESTRUTURA DA REDE EM ESTADO NORMAL 13
FIGURA 2 – ESTRUTURA DA REDE COMPROMETIDA 13
FIGURA 3 – REGRAS DE TRÁFEGO AO INICIAR A REDE 15
FIGURA 4 – REGRAS DE TRÁFEGO COM *HOSTS* DESCOBERTOS 16
FIGURA 5 – COMANDO PARA ADIÇÃO DE REGRA NO *SWITCH* 17
FIGURA 6 – ANÁLISE DE TRÁFEGO NORMAL 18
FIGURA 7 – ANÁLISE DE TRÁFEGO COMPROMETIDO 19
FIGURA 8 – REGRA IMPLANTADA NO *SWITCH* 19

LISTA DE TABELAS

CATEGORIZAÇÃO DOS PROBLEMAS DE SEGURANÇA

TABELA 1 – POR CAMADA/INTERFACE AFETADA 10

LISTA DE SIGLAS

SDN	-	*Software Defined Network*
DDoS	-	*Distributed Denial of Service*
VM	-	*Virtual Machine*

SUMÁRIO

1 INTRODUÇÃO 8
2 SEGURANÇA NO SDN 9
3 ESTUDO DA REDE E SIMULAÇÃO DO ATAQUE 12
3.1 MININET 14
3.2 OPENDAYLIGHT 14
3.3 CRIAÇÃO DO AMBIENTE DE TESTES 14
3.4 SIMULANDO O ATAQUE 16
4 CONCLUSÃO 20
REFERÊNCIAS 21

1 INTRODUÇÃO

Uma rede de computadores é algo que está em constante mudança, e isso normalmente é um problema para quem as gerencia. Hoje em dia temos vários mecanismos para tornar o gerenciamento mais eficiente e mais fácil, como *firewall*, sistemas para detecção de invasão, entre outros, mas ainda assim a tarefa de gerenciar uma rede é um tanto complexa. Além dos problemas comuns de uma rede de computadores, como a alta taxa de tráfego e a indisponibilidade de caminhos entre os *hosts*, cada vez mais as redes tem se tornado alvos de *cybercriminosos*, o que traz a necessidade de políticas de gerenciamento mais sofisticadas.

Rede Definida por *Software*, ou simplesmente SDN (sigla em inglês para *Software Defined Network*), é um paradigma que visa mudar a maneira com que as redes são gerenciadas. O SDN aposta no controle centralizado e na alta programabilidade [2] para melhorar o gerenciamento das redes, podendo assim fazer com que um ambiente de rede complexo seja facilmente gerenciado.

O SDN é um modelo de rede que busca separar a camada de controle (*control plane*), a camada de dados (*data plane*) e a camada de aplicação (*application plane*) [1]. Essa separação é feita atribuindo aos *switches* e roteadores apenas a função de encaminhamento de dados, deixando assim a função de controle para o controlador. O controlador, por sua vez, é um programa de computador que tem o conhecimento da rede e de seu estado atual, e atua instalando regras de controle nos dispositivos de encaminhamento de dados, fazendo com que o tráfego na rede se comporte de acordo com essas regras para o correto funcionamento das aplicações que interagem com o controlador.

A camada de dados é caracterizada por todo o *hardware* e interfaces de *software* responsáveis por fazer o encaminhamento de dados. A camada de controle é o nível responsável por implementar as regras de controle de tráfego na rede. O controlador é centralizado e baseado em *software*, o que facilita a interação com o administrador da rede, uma vez que não é necessário acessar e configurar cada dispositivo individualmente para que a rede mude de comportamento. Já o nível de aplicação, pode se comunicar com o controlador e fazer com que a rede tenha o comportamento desejado [1].

Essa separação de camadas traz vários benefícios, dentre os quais podemos citar o fato de ficar mais fácil a inserção de novas ideias e políticas de controle de rede. Isto se deve pelo fato de que o gerenciamento centralizado através de um *software* se torna mais fácil do que o gerenciamento distribuído, no qual cada dispositivo tem que ser configurado individualmente para que a rede tenha um comportamento diferente do atual. Com a presença do controlador, fica mais fácil o gerenciamento da camada de dados, uma vez que o controlador tem o conhecimento do estado geral da rede.

Existem diversos protocolos para a comunicação entre controlador e switch, dentre ele podemos citar o OpenFlow, que é considerado como um dos primeiros padrões para o SDN [3]. Ele permite que o controlador se comunique com os *switches* e roteadores, ou seja, com o plano de dados, permitindo assim que o controlador insira ou modifique as regras de tráfego nas tabelas de fluxo [16] presentes nesses dispositivos. Isso permite que a rede se adapte mais facilmente às necessidades. No contexto deste trabalho, podemos afirmar que é através do *OpenFlow* que o controlador aplicará regras nos roteadores e *switches*, possibilitando assim que o

administrador tome as decisões necessárias para eventuais particionamentos e redirecionamentos de tráfego, caso necessário. Neste trabalho será abordado um pouco da segurança no SDN, explicando alguns conceitos necessários como o que é o SDN, falando um pouco sobre o *OpenFlow*, alguns conceitos básicos sobre a segurança e, ao final, será abordado um esquema de um possível ataque a esse modelo. O esquema consiste em tentar inserir, em uma rede com um *host* comprometido, um computador configurado como controlador, que poderia ser um equipamento do atacante. Buscamos descobrir se existe a possibilidade de interagir com a rede através desses novos elementos, o que comprometeria a rede em qual ele está.

2 SEGURANÇA NO SDN

Como apontado em [6], os pilares básicos da comunicação segura em uma rede são a confidencialidade, integridade, disponibilidade de informação, autenticação e não repúdio. Essas características são essenciais para uma comunicação segura, mas com as alterações no modelo que o SDN propõe, elas têm que ser repensadas.

Como já visto, o SDN tem capacidade para proporcionar uma grande variedade de benefícios e facilidades que as redes normais não possuem, devido à centralização do controle, facilidade de inserção e remoção de políticas de tráfego e separação da camada de controle e de dados. Porém, as mesmas características que proporcionam tantos benefícios para essa estrutura, também acabam sendo fontes de muitos problemas já conhecidos, como ataques de negação de serviço, além de problemas novos que podem ser observados com essa separação, alguns deles estão listados na (TABELA 1).

Além dos problemas já conhecidos nas redes comuns, é preciso também analisar quais tipos de problemas podem ser causados especificamente pela estrutura do SDN. Sendo assim, devido à sua estrutura, podemos identificar cinco possíveis pontos que devem ser estudados. Além das três camadas já citadas anteriormente, de aplicação, controle e dados, também temos as interfaces de comunicação entre essas camadas também estão suscetíveis a problemas. Como apresentado na (TABELA 1) é possível verificar uma categorização de problemas de segurança e em qual camada ou interface esse problema pode ocorrer.

TABELA 1 – CATEGORIZAÇÃO DOS PROBLEMAS DE SEGURANÇA POR CAMADA/INTERFACE AFETADA.

Problema de Segurança/ Ataque	Camada SDN Visada ou Afetada				
	Camada de Aplicação	Interface Aplicação-Controle	Camada de Controle	Interface Controle-Dados	Camada de Dados
1. Acesso Não Autorizado					
1.1 Acesso Não Autorizado no Controlador			✓	✓	✓
1.2 Aplicação Não Autenticada	✓	✓	✓		
2. Vazamento de Dados					
2.1 Descoberta de Regras de Fluxo (Side-Channel Attack no Buffer de Entrada					✓
2.2 Descoberta de Políticas de Encaminhamento (Analise de Tempo de Processamento de Pacotes)					✓
3. Modificação de Dados					
3.1 Modificação de Regra de Fluxo para Modificar Pacotes			✓	✓	✓
4. Aplicações Maliciosas					
4.1 Inserção de Regras Fraudulentas	✓	✓	✓		
4.2 Sequestro do Controlador			✓	✓	✓
5. Negação de Serviço					
5.1 Flood na Comunicação Controlador-Switch			✓	✓	✓
5.2Floding na Tabela de Fluxo do Switch					✓
6. Problemas de Configuração					
6.1 Pouca Adoção de TLS (ou outras técnicas de autenticação)			✓	✓	✓
6.2 Aplicação de Políticas	✓	✓	✓		

FONTE: Adaptado de [6].

Como podemos observar, a camada de dados é a maior prejudicada, pois acaba sendo afetada pela maioria dos problemas apresentados, com isso, os dados que deveriam ser confidenciais e confiáveis, acabam sendo comprometidos. Esse comprometimento, seja por problemas nas camadas superiores que acabam influenciando no gerenciamento do tráfego e dos dados, quanto problemas nas camadas inferiores, que acabam interferindo diretamente com os dados, faz com que os mesmos não sejam mais confiáveis.

Um dos tópicos mais estudados no campo da segurança do SDN são os ataques de negação de serviço, também conhecidos por DDoS (sigla em inglês para *Distributed Denial of Service*, ou Negação de Serviço Distribuída). Este tipo de ataque possui várias pesquisas referentes a técnicas de detecção [7][8] e prevenção ou mitigação [7][9]. Os ataques DDoS são bastante preocupantes no ambiente SDN pelo fato de que o ponto mais prejudicado da rede será o controlador, que é o responsável pela avaliação e definição de regras para todo o tráfego da rede. Com o controlador indisponível para o tráfego real, toda a rede fica indisponível, pois novos fluxos de dados que ainda não tem regras definidas, por exemplo, não poderão ser atendidos, e consequentemente, não conseguirão trafegar na rede. Porém, DDoS não é a única preocupação que se deve ter quando se fala em SDN, existem vários outros problemas que podem ser estudados e explorados, incluindo alguns pontos que derivam da estrutura do SDN.

O controle centralizado proporcionado pela arquitetura SDN é um dos motivos de preocupação nesse modelo de rede [11]. Essa centralização é algo bastante importante do ponto de vista prático, visto que se torna mais fácil administrar uma rede através de *software*, sem precisar configurar cada roteador ou *switch* separadamente, um a um. A desvantagem é que, se o controlador for comprometido, a rede inteira fica comprometida. Um exemplo é que se um atacante consegue ter acesso ao controlador de uma rede SDN, ele pode fazer com que a rede se comporte como quiser. Visto o

fato de o controlador ser o principal elemento da arquitetura SDN, ele é o alvo mais visado para ataques e também o elemento com mais possibilidades de problemas eventualmente exploráveis.

A programabilidade disponibilizada pelo SDN é outro ponto de preocupação [11]. É uma funcionalidade fundamental para a arquitetura, pois proporciona a possibilidade de alterações no ambiente de rede em tempo real, de acordo com a necessidade do momento. Por outro lado, a programabilidade deve ser pensada como uma possível ferramenta para pessoas ou programas mal-intencionados, visto que com um acesso indevido, é muito fácil fazer com que a rede se comporte de acordo com o desejo do atacante. É importante reforçar as políticas de acesso ao controlador, através de diferentes níveis de acesso, por exemplo, para diminuir a exposição do controlador à possíveis ameaças. Na (TABELA 1), podemos citar a inserção de regras fraudulentas, item 4.1, e a modificação de regras de fluxo, item 3.1, como exemplos de ataques que tem como base a programabilidade do SDN.

A conexão de diferentes domínios é mais um ponto de preocupação [11]. Essa conexão é fundamental, porém pode envolver a conexão entre controladores. Devem ser avaliadas as políticas e níveis de acesso, bem como a segurança da conexão entre os controladores.

Um ambiente seguro requer muito estudo sobre os possíveis riscos e ameaças, mas existem cinco pontos que devem ser sempre observados, como apontado por [10], quando se trata da segurança em SDN. Deve existir sempre a segurança do controlador, que pelo fato de ser o ponto de controle responsável por toda a rede, deve ter uma política rígida de acesso e permissões. Deve existir a proteção do controlador contra indisponibilidade, visto que se ele não estiver disponível, novas rotas não poderão ser descobertas, e eventuais medidas para controle ou ajuste não poderão ser tomadas. Se o controlador ficar indisponível por algum motivo, por consequência, a rede toda fica indisponível. A rede deve ser confiável, ou seja, deve-se assegurar que o controlador, as aplicações que nele rodam e os dispositivos que ele gerencia são confiáveis e estão todos funcionando como deveriam. Se um destes três pontos não for confiável, a rede inteira fica comprometida, já que o comportamento da rede não pode mais ser previsto. Uma estrutura de políticas robusta é necessária para que se tenha certeza de que o controlador esteja fazendo exatamente o que é esperado que ele faça, e com isso, não acabe deixando nenhuma brecha imprevista na rede. Por fim, mas não menos importante, deve-se ter uma conduta forense e de remediação de incidentes para que, quando um incidente aconteça, se tenha a capacidade de determinar qual foi o incidente, o motivo pelo qual ele ocorreu, poder recuperar os problemas causados por ele e para que possam ser tomadas precauções para evitar o mesmo tipo de incidente no futuro.

As características do SDN podem ajudar na segurança de uma rede [12], como por exemplo: o controle dinâmico dos fluxos no SDN, fundamental para a arquitetura, torna configuração de rotas de acordo com as necessidades muito mais fácil. Esse controle também pode ser aplicado como auxílio na segurança de uma rede, como um *firewall*, que pode ser implementado para inserir dinamicamente as regras de fluxo no roteador de acordo com uma análise dos pacotes recebidos, ou então um sistema de detecção e redirecionamento de tráfego malicioso. O controle centralizado, também é uma funcionalidade importante. Com esse controle, sabendo-se que outra característica do SDN é o conhecimento “global” da rede, ou seja, ter

conexão com todos os roteadores e *switches*, é fácil conseguir estatísticas e informações dos dispositivos que integram a rede. A coleta rápida dessas informações pode ajudar na detecção e mitigação de tráfego suspeito, pois com os dados gerados pelas estatísticas, é fácil rastrear e apontar qual tráfego está fora do normal e onde pode existir um problema. Com isso podem ser tomadas medidas como a reconfiguração da rede para redirecionar o tráfego desejado para alguma aplicação de segurança, ou até mesmo o isolamento do ponto de origem do tráfego suspeito.

3 ESTUDO DA REDE E SIMULAÇÃO DO ATAQUE

Como mostrado na (TABELA 1), a camada de dados é a mais afetada com os problemas que possivelmente possam ocorrer no SDN, e justamente são os dados que visamos proteger quando pensamos em soluções para segurança de redes e de computadores. Sendo assim, sempre é necessário pensar quais os caminhos que podem levar ao comprometimento dos dados que temos em nossa rede e, no caso do SDN, quais camadas são afetadas por cada tipo de problema.

Partindo de uma situação na qual um atacante consegue acesso à uma rede SDN, assim como em qualquer outra rede, podemos pensar nas possibilidades que o atacante tem para o comprometimento da rede. Sabemos que o número de possibilidade para o comprometimento da rede, seja para interceptação de dados, seja para o comprometimento do serviço prestado, são inúmeras e, com isso, podemos começar a traçar os caminhos possíveis para o atacante. Para os propósitos deste trabalho, partiremos de um cenário no qual um atacante conseguiu acesso físico à uma rede SDN. Esse atacante então conseguiu conectar à um *switch* da rede um computador que tem como função rodar um controlador implementado pelo atacante. O atacante também já conseguiu acesso a um computador na rede através de alguma vulnerabilidade presente no mesmo.

Como visto anteriormente, uma das grandes vantagens do SDN é a característica de programabilidade da rede. Sendo assim, se um atacante conseguir conectar e controlar um *switch* da rede através de seu controlador, ele poderá usar essa característica da rede para ter controle sobre o fluxo de dados que passa pelo *switch*. Com isso, poderá impor suas próprias regras de fluxo como, por exemplo, fazer com que qualquer pacote que passe por esse *switch* se perca.

Vamos trabalhar com uma topologia simples, para facilitar o entendimento e a simulação. Primeiro teremos um controlador conectado à um *switch*, e esse *switch* conectado à três *hosts*, como mostrado na (FIGURA 1). Em seguida, tentaremos utilizar a mesma topologia, porém com um controlador implantado pelo atacante e um dos *hosts* da rede dominado pelo atacante, sendo o *host* conectado ao *switch* comprometido, como mostrado na (FIGURA 2) em vermelho.

FIGURA 1 – ESTRUTURA DA REDE EM ESTADO NORMAL

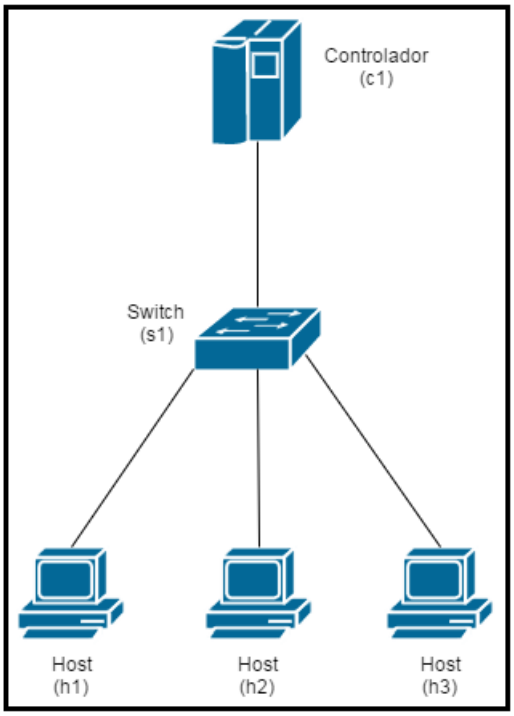
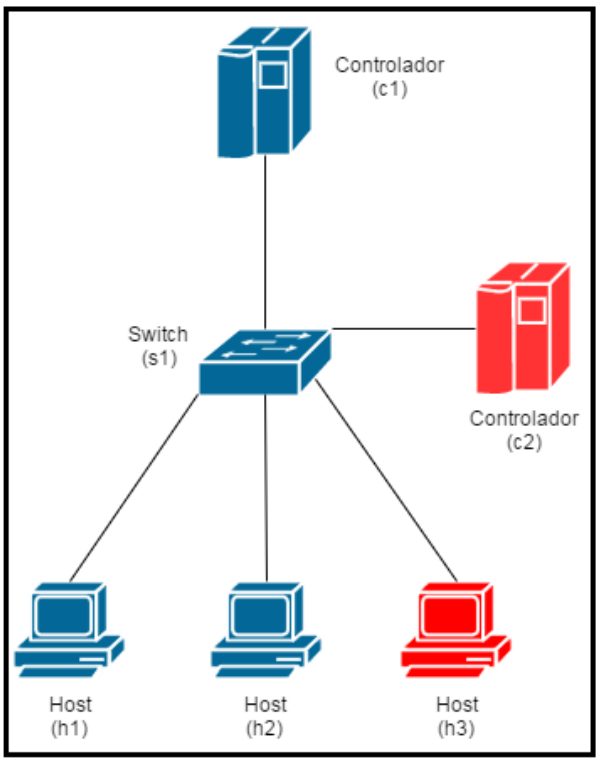


FIGURA 2 – ESTRUTURA DA REDE COMPROMETIDA



Para os testes necessários, será usado o *VirtualBox*, que é um sistema de virtualização *open source*. Através dele, serão usadas 2 máquinas virtuais, ou simplesmente VM, sendo uma VM responsável pelo Mininet e uma VM responsável pelo *OpenDaylight*. Será usado o *OpenDaylight* versão 4.4 (Beryllium-SR4), o *Open vSwitch* versão 2.5 [15], e também o *OpenFlow* versão 1.0.

3.1 MININET

Mininet é um emulador de redes *open source* capaz de rodar diversos *hosts*, *switches*, links e roteadores em um único *Kernel* Linux [4], e tem a capacidade de suportar redes SDN. Usa virtualização para obter tal fim, fazendo com que um único computador pareça uma rede completa. Também suporta a funcionalidade da utilização de scripts em *Python* para a criação de ambientes de trabalho, sendo ideal para os testes abordados neste trabalho.

Os *hosts*, *switches*, roteadores, e *links* usados no Mininet tem um comportamento bem próximo da realidade, com a diferença de que são baseados em *software* e não em *hardware*. Sendo assim, uma aplicação que roda em uma rede Mininet, pode ser facilmente implantada em um ambiente real, e aplicações que rodam em um ambiente real também podem ser facilmente implantadas em um ambiente Mininet [4].

Vale ressaltar que o Mininet não vem acompanhado de um controlador *OpenFlow*, então para o correto uso é necessário utilizá-lo em conjunto com um ou mais controladores, sejam eles controladores já existentes ou escrever um controlador. Neste trabalho usaremos um controlador já existente, o *OpenDaylight*, bem como um controlador bem simples escrito para o nosso propósito.

3.2 OPENDAYLIGHT

OpenDaylight [5] é um dos mais conhecidos controladores SDN *open source* existentes, e foi concebido para possibilitar a implantação de soluções SDN em ambientes heterogêneos, com equipamentos e protocolos diversos. Será usado neste trabalho apenas para termos um ponto de partida como uma rede completa.

3.3 CRIAÇÃO DO AMBIENTE DE TESTES

Em primeiro lugar criamos a rede de testes com um controlador, que será o *OpenDaylight*, e que rodará em uma VM separada, a rede também terá um *switch*, que rodará com a implementação do *Open vSwitch*, e também teremos 3 *hosts*, h1 (10.0.0.1), h2 (10.0.0.2), h3 (10.0.0.3). O Mininet nos dá uma possibilidade interessante, a de definir o endereço MAC de cada um dos *hosts*, o que deixa mais fácil a identificação. Durante o decorrer do trabalho, o endereço MAC de cada *host* será definido de acordo com seu número, ou seja, o *host* h1 terá endereço

00:00:00:00:00:01, o *host* h2 terá endereço 00:00:00:00:00:02 e o host h3 terá endereço 00:00:00:00:00:03.

Com isso podemos simular o comportamento padrão da nossa rede, sem a presença do atacante, podendo verificar o correto funcionamento do nosso ambiente de trabalho. A simulação é feita através do envio de pacotes com o comando *ping*, que nos permite verificar que os pacotes estão sendo enviados corretamente ao destino, e estão gerando a resposta esperada. Verificamos que o *host* destino está recebendo as mensagens através do comando tcpdump e, com o mesmo comando, verificamos que o terceiro *host* não está recebendo as mensagens do *ping*. Executamos o *ping* do *host* h1 para o *host* h2, e verificamos o tráfego tanto em h2 quanto em h3, obtendo o resultado esperado, com apenas h2 recebendo os pacotes.

Uma parte fundamental para o correto funcionamento do *OpenFlow* são as tabelas de fluxo presentes nos *switches*. Estas tabelas servem para o armazenamento das regras de tráfego que serão usadas nas comunicações, ou seja, regras que verificam determinados parâmetros para decidir qual o melhor caminho para a mensagem, podendo assim definir rotas específicas para mensagens de determinadas origens, ou para determinados destinos. Quando iniciamos nossa rede, num primeiro momento, nosso *switch* não possui regras específicas, apenas regras especificando que mensagens vindas de uma porta devem ser enviadas para todas as demais, ou seja, deve ser feito um broadcast, como mostrado na (FIGURA 3).

FIGURA 3 – REGRAS DE TRÁFEGO AO INICIAR A REDE

```
joao@mininet-vm:~$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
cookie=0x2b00000000000005, duration=9.194s, table=0, n_packets=0, n_bytes=0,
idle_age=10, priority=100,dl_type=0x88cc actions=CONTROLLER:65535
cookie=0x2b00000000000006, duration=6.669s, table=0, n_packets=2, n_bytes=140,
idle_age=1, priority=2,in_port=3 actions=output:2,output:1,CONTROLLER:65535
cookie=0x2b00000000000007, duration=6.669s, table=0, n_packets=2, n_bytes=140,
idle_age=1, priority=2,in_port=2 actions=output:3,output:1,CONTROLLER:65535
cookie=0x2b00000000000008, duration=6.669s, table=0, n_packets=2, n_bytes=140,
idle_age=1, priority=2,in_port=1 actions=output:3,output:2,CONTROLLER:65535
cookie=0x2b00000000000005, duration=9.194s, table=0, n_packets=14, n_bytes=
1164, idle_age=8, priority=0 actions=drop
```

Após isso, para inserir as regras que de fato nos interessam, usamos outro comando fornecido pelo Mininet, o *pingall*. Este comando efetua um *ping* de cada *host* para todos os outros para descobrir as rotas entre os *hosts*, na tela é exibido em formato texto quais *hosts* estão e quais não estão acessíveis a partir de cada um deles. Podemos usar então o comando ovs-ofctl, um comando que acompanha o *Open vSwitch*, e serve para o monitoramento e administração dos *switches*, que funciona com qualquer *switch OpenFlow* [14], e não só com o *Open vSwitch*. Usamos a opção dump-flows**,** que já foi usada anteriormente para obter o resultado mostrado na (FIGURA 3), do comando ovs-ofctl, para que possamos ver quais as regras implantadas em um determinado *switch*, usamos o comando para verificar quais as regras de fluxo implantadas no *switch* s1, como mostrado na (FIGURA 4)

FIGURA 4 – REGRAS DE TRÁFEGO COM *HOSTS* DESCOBERTOS

```
joao@mininet-vm:~$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
cookie=0x2b00000000000005, duration=196.871s, table=0, n_packets=0, n_bytes=0,
idle_age=198, priority=100,dl_type=0x88cc actions=CONTROLLER:65535
cookie=0x2a000000000000e, duration=10.184s, table=0, n_packets=1, n_bytes=42,
idle_timeout=300, hard_timeout=600, idle_age=5, priority=10,dl_src=
00:00:00:00:00:02,dl_dst=00:00:00:00:00:03 actions=output:3
cookie=0x2a000000000000f, duration=10.184s, table=0, n_packets=1, n_bytes=42,
idle_timeout=300, hard_timeout=600, idle_age=5, priority=10,dl_src=
00:00:00:00:00:03,dl_dst=00:00:00:00:00:02 actions=output:2
cookie=0x2a00000000000014, duration=5.192s, table=0, n_packets=0, n_bytes=0,
idle_timeout=300, hard_timeout=600, idle_age=5, priority=10,dl_src=
00:00:00:00:00:03,dl_dst=00:00:00:00:00:01 actions=output:1
cookie=0x2a00000000000015, duration=5.192s, table=0, n_packets=0, n_bytes=0,
idle_timeout=300, hard_timeout=600, idle_age=5, priority=10,dl_src=
00:00:00:00:00:01,dl_dst=00:00:00:00:00:03 actions=output:3
cookie=0x2a00000000000016, duration=5.192s, table=0, n_packets=0, n_bytes=0,
idle_timeout=300, hard_timeout=600, idle_age=5, priority=10,dl_src=
00:00:00:00:00:02,dl_dst=00:00:00:00:00:01 actions=output:1
cookie=0x2a00000000000017, duration=5.192s, table=0, n_packets=0, n_bytes=0,
idle_timeout=300, hard_timeout=600, idle_age=5, priority=10,dl_src=
00:00:00:00:00:01,dl_dst=00:00:00:00:00:02 actions=output:2
-- trecho omitido --
```

A parte que nos interessa nas regras está em negrito, é a parte que nos informa o que deve acontecer com cada mensagem de acordo com sua origem e seu destino. O trecho omitido corresponde às regras para *broadcast* já observadas na listagem 3.1.

Como podemos observar, as regras baseiam-se na comparação do endereço de origem, do endereço de destino, e se a mensagem se encaixar nessa regra, é tomada a ação correspondente, no nosso caso, a mensagem é enviada para a porta específica do *switch* onde o *host* está. Olhando num ambiente mais amplo, temos a seguinte situação, quando iniciamos a nossa rede, nenhuma regra de fluxo está instalada no *switch*, as regras vão sendo instaladas a medida em que rotas vão sendo descobertas via ARP, ou podem ser instaladas manualmente pelo administrador da rede.

3.4 SIMULANDO O ATAQUE

Agora que temos uma noção do funcionamento normal da nossa rede, podemos tentar explorar esse funcionamento para obter acesso a mensagens que não nos são destinadas. Uma forma de conseguir isso é tentando instalar regras “maliciosas” no *switch*, ou seja, a ideia é implantar no *switch* regras que alteram o comportamento normal da nossa rede. Um exemplo seria tentar implantar uma regra que verifica mensagens vindas de h1 com destino h2 e, ao detectar uma mensagem que se encaixe nesse padrão, redirecioná-la para h3, e com isso h3 receberia as

mensagens e h2 não. Outra abordagem possível seria fazer um ARP *Spoofing*[1] de um *host* implantado na rede tentando se passar por algum dos outros *hosts*, essas são duas das inúmeras possibilidades. Como podemos ver, quando se conhece o funcionamento da rede, vários são os caminhos possíveis para buscar alcançar o objetivo de obter acesso e controle sobre tudo o que passa pelo *switch* que estamos usando como alvo.

A abordagem que iremos seguir é a de inserir regras maliciosas no *switch*, desviando o tráfego desejado para um computador que está sob nosso controle. Podemos usar a característica do SDN que nos permite ter mais de um controlador numa rede e tentar inserir um controlador falso na rede, conectado apenas ao nosso *switch* alvo.

O nosso controlador, que seria o controlador que o atacante conseguiu conectar ao *switch*, será simples e terá como única tarefa o redirecionamento dos pacotes desejados. Usaremos o POX como plataforma para escrever nosso controlador. O comportamento básico do nosso controlador será, ao receber um pacote, instalar uma regra que fará com que o tráfego vindo de h1 com destino h2 seja mandado para h3, e não para h2. Esse comportamento será obtido através do comando descrito na (FIGURA 5), que estará presente no nosso controlador.

FIGURA 5 – COMANDO PARA ADIÇÃO DE REGRA NO *SWITCH*

```
event.connection.send(
of.ofp_flow_mod(command=of.OFPFC_ADD,
                actions=of.ofp_action_output(port=3),
                priority=32,
                match=of.ofp_match(dl_type=0x800,
                                   nw_src="10.0.0.1",
                                   nw_dst="10.0.0.2")))
```

O *Open vSwitch* é um *switch* virtual que suporta, entre muitos outros protocolos, o *OpenFlow* [15]. Ele nos disponibiliza uma variedade de comandos, entre eles o ovs-vsctl, que é um utilitário para verificação e configuração do *switch*, esse comando será usado em nossos testes.

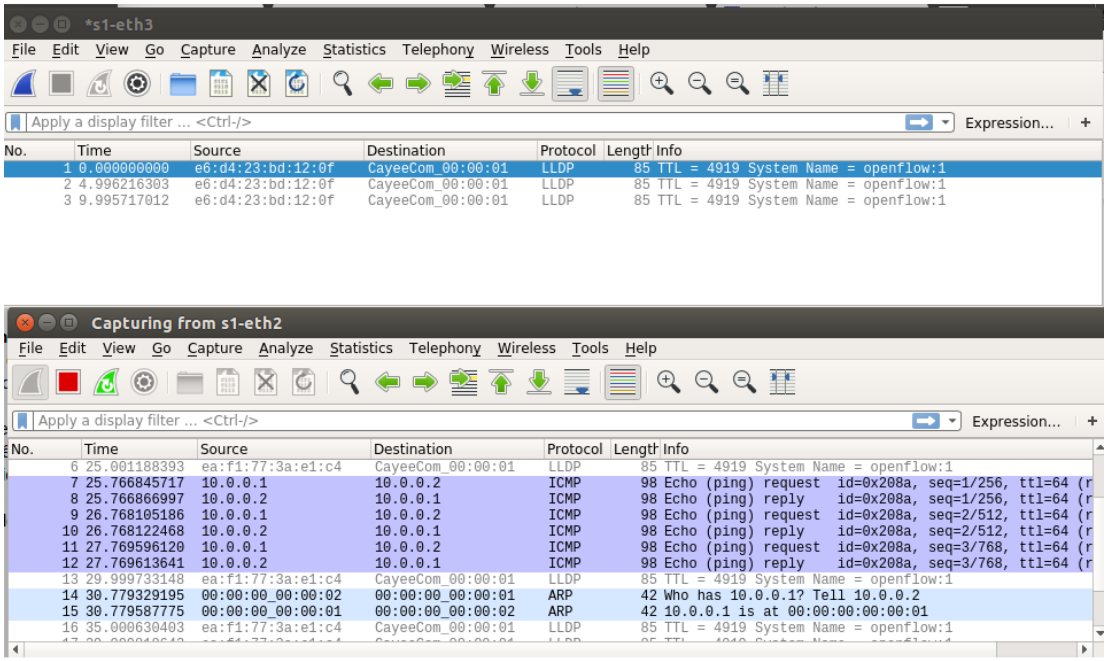
O primeiro passo é iniciar a nossa rede normalmente, como já fizemos anteriormente, tendo apenas um controlador, um *switch* e três *hosts*, e executar o comando *pingall* do Mininet para que todas as rotas sejam estabelecidas, a saída do comando *pingall* será constituída de três linhas, sendo elas h1 -> h2 h3, h2 -> h1 h3, h3 -> h1 h2, mostrando que a partir de cada *host* todos os demais estão acessíveis. Após isso, executamos o comando *xterm* h1 h2 h3 no console Mininet, isso fará com que sejam abertos três terminais, um para h1, um para h2 e um para h3, cada um deles executando como *root*. A partir daí executamos, no terminal h2 o comando *tcpdump* -XX -n -i h2-eth0, no terminal h3 o comando *tcpdump* -XX -n -i h3-eth0 e no terminal h1 o comando *ping* -c3 10.0.0.2.

[1] Arp Spoofing é um tipo de ataque no qual o atacante envia repetidamente mensagens ARP falsas com o objetivo de enganar a máquina ou *switch* alvo, para que este redirecione ao atacante mensagens que, na verdade, pertencem a outro computador. [13]

Nos *hosts* h1 e h2 executamos o *tcpdump*, que serve para fazer o *dump* do tráfego da rede. A opção -XX do *tcpdump* serve para especificarmos que, além dos cabeçalhos, desejamos exibir o conteúdo dos pacotes em hexadecimal e em ASCII, a opção -n serve para que os endereços não sejam convertidos para nomes, assim podemos ter um acompanhamento melhor, e finalmente a opção **-i**, que serve para informarmos uma interface específica, no nosso caso, h2-eth0 para h2 e h3-eth0 para h3. O comando *ping* -c3 10.0.0.2, executado em h1, serve para fazermos o *ping* de h1 para h2 (10.0.0.2), com a opção -c3 para definirmos que queremos fazer o envio de 3 pacotes.

Por uma análise visual através do *tcpdump* em h2 e h3, ao executarmos o *ping* em h1, vemos claramente que apenas o *host* h2 está recebendo os pacotes enviados pelo *ping*, essa confirmação vem quando executamos duas instâncias do *Wireshark* e monitoramos as interfaces de h2 e h3, o que nos possibilita ver claramente, como mostrado na (FIGURA 6), que apenas a interface h2-eth0 recebe os pacotes enviados pelo *ping*.

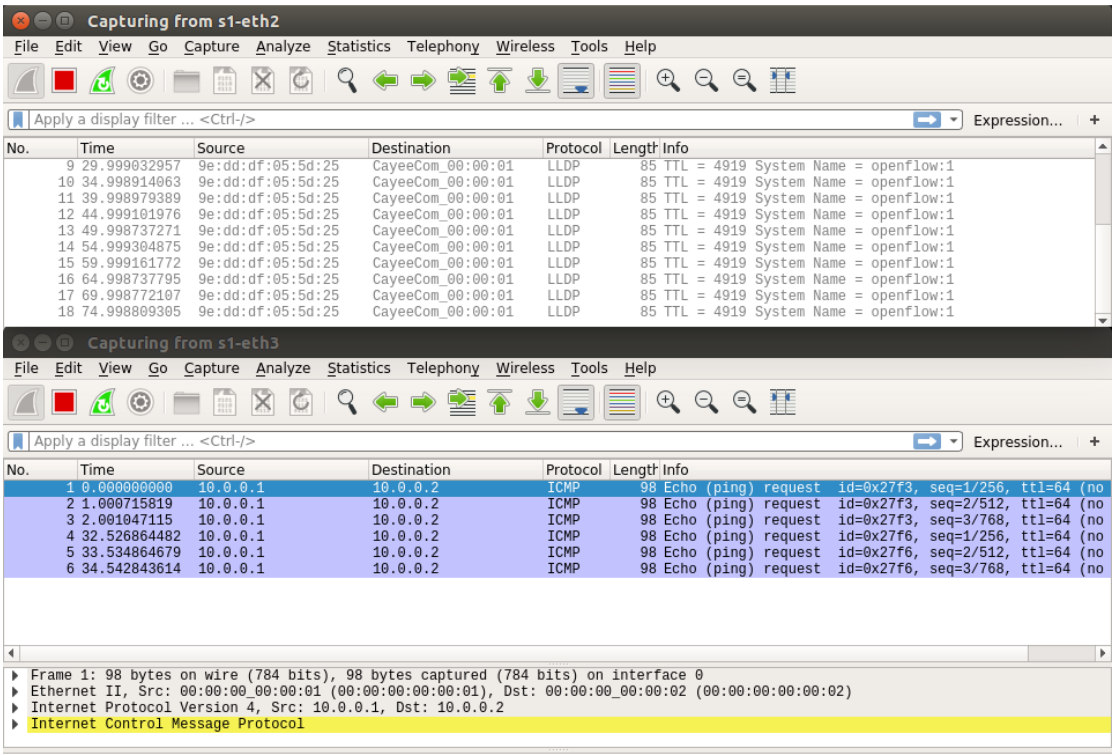
FIGURA 6 – ANÁLISE DE TRÁFEGO NORMAL



Após, iniciamos novamente nossa rede, como feito anteriormente. Usamos novamente o comando *pingall* para calcular as rotas entre os *hosts* e verificar que tudo está funcionando corretamente. Usamos o comando *xterm* para abrir novamente os terminais para h1, h2 e h3, e executamos novamente os comandos anteriores, os dois *tcpdump* em h2 e h3, e o *ping* em h1, e com isso verifica-se o correto funcionamento da rede. Após isso, temos separadamente o nosso controlador “de ataque”, que seria o computador que o atacante conseguiu conectar ao *switch*. Usando o acesso que o atacante conseguiu anteriormente a um dos computadores da rede, que está conectado ao *switch* comprometido, executamos o comando sudo ovs-vsctl set-controller s1 tcp:127.0.0.1:6633 tcp:192.168.56.102:6633 no nosso terminal de h1. O comando faz com que s1 seja configurado para ter dois controladores, sendo que o

endereço 192.168.56.101:6633 é o do nosso controlador *OpenDaylight* e o 127.0.0.1:6633 é o do nosso controlador malicioso que, para o nosso teste, está rodando na mesma máquina virtual que o simulador Mininet, mas que poderia estar rodando em outra máquina. Ao executar o comando, verifica-se, através do terminal, que nosso controlador é conectado à rede, e a partir daí, foi obtido o comportamento que estávamos buscando. Executamos novamente o comando *ping* -c3 10.0.0.2 a partir do *host* h1, porém vemos através do *tcpdump* que está rodando nos terminais de h1 e h2 que quem recebe as mensagens do *ping* não é mais h2, e sim h3, fazendo com que h1 não obtenha resposta de h2 para seu *ping*. Podemos também observar esse comportamento através do *Wireshark*, quando vemos que após executar o *ping* para h2, quem recebe as mensagens do *ping* é na verdade h3-eth0, que é a interface de rede do *host* h3, como mostrado na (FIGURA 7).

FIGURA 7 – ANÁLISE DE TRÁFEGO COMPROMETIDO



O comportamento apresentado pode ser explicado pelo fato de que o nosso controlador preparado está funcionando, fazendo sua função de criar uma regra a cada pacote enviado, caso a regra ainda não exista, como verificado ao executar o comando sudo ovs-ofctl dump-flows s1, que nos exibe a saída mostrada na (FIGURA 8), que representa as regras instaladas em s1.

FIGURA 8 – REGRA IMPLANTADA NO *SWITCH*

```
joao@mininet-vm:~$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=312.529s, table=0, n_packets=3, n_bytes=294, idle_age=283,
priority=32,ip,nw_src=10.0.0.1,nw_dst=10.0.0.2 actions=output:3
```

Com isso conseguimos alterar as configurações de s1 para que funcione como desejamos, comprometendo assim todo o tráfego que passa por s1. A partir daí, poderíamos tentar explorar outras opções de ações para serem inseridas na regra de fluxo e outras abordagens, como tentar redirecionar as mensagens interceptadas para o seu destino real, assumindo um papel de *man-in-the-middle* na rede, entre outros. As possibilidades são inúmeras, e muito disso depende da criatividade e do conhecimento sobre o funcionamento da rede e de tudo que está envolvido para o correto funcionamento, porém essas variações estão fora do escopo do trabalho.

Podemos então definir o nosso ataque como sendo realizado em três etapas, sendo elas a “invasão”, o comprometimento da rede e a coleta ou comprometimento dos dados. O primeiro passo, a “invasão”, é o momento em que o atacante consegue conectar o controlador malicioso ao *switch* e consegue invadir um *host* que está conectado a esse *switch*. O segundo passo, o comprometimento da rede, é o momento em que o atacante consegue fazer com que seu controlador assuma uma função na rede, podendo assim controlar o comportamento da mesma. O terceiro passo, a coleta ou comprometimento dos dados, é dado pelo fato do invasor conseguir, através do seu controlador funcional, fazer com que o *switch*, no caso da coleta de dados, desvie o fluxo de dados para o *host* dominado pelo atacante.

4 CONCLUSÃO

Através dos testes realizados observou-se que, no ambiente de simulação, foi possível alterar a tabela de regras de fluxo do *switch* para que ele redirecionasse indevidamente mensagens para um *host* que não é o destino das mensagens, mas devemos lembrar que o ambiente de simulação nos proporciona bastante liberdade para trabalharmos. Pelo fato de esse experimento envolver modificações suportadas pela arquitetura do modelo SDN e não a utilização de falhas específicas de *softwares* usados, podemos assumir que o ataque seria possível com qualquer versão dos *softwares* usados.

Para que o ataque em questão fosse de fato viável em um ambiente real, uma série de fatores devem ser levados em consideração, dentre eles podemos citar a validade dos comandos usados, ou seja, se os comandos usados na simulação realmente podem ser usados em *switches* reais, ou se existem comandos equivalentes para tais fins. Outro ponto que devemos avaliar é que, em uma rede com políticas de tráfego bem definidas, com *softwares* de controle e com um acompanhamento eficiente da dinâmica da rede, esse ataque seria facilmente percebido e combatido, o que o torna extremamente ineficaz no modelo proposto neste trabalho.

Tendo em mente os pontos de atenção citados anteriormente, podemos usar estes testes como um ponto de partida, seja para tentar explorar uma rede de uma

maneira mais eficiente, seja para tentar buscar soluções eficientes para o tipo de ataque em questão, proporcionando assim um leque de possibilidades para futuros estudos na área.

REFERÊNCIAS

[1] NISHTHA; SOOD, M. Software Defined Network – Architectures. **2014 International Conference on Parallel, Distributed and Grid Computing**, p. 451-456. Dezembro de 2014

[2] Open Networking Foundation (ONF), Software-Defined Networking (SDN) Definition. Disponível em <https://www.opennetworking.org/sdn-resources/sdn-definition>. Acesso em: 02 de agosto de 2016.

[3] Open Networking Foundation (ONF), OpenFlow. Disponível em <https://www.opennetworking.org/sdn-resources/openflow>. Acesso em: 02 de agosto de 2016.

[4] Mininet, Introduction to Mininet. Disponível em <https://github.com/mininet/mininet/wiki/Introduction-to-Mininet#what>. Acesso em: 05 de agosto de 2016.

[5] The OpenDaylight Platform. Disponível em <https://www.opendaylight.org/>. Acesso em: 1 de setembro de 2016.

[6] SCOTT-HAYWARD, S.; O'CALLAGHAN, G.; SEZER, S. SDN Security: A Survey. **2013 IEEE SDN for Future Networks and Services (SDN4FNS)**. Novembro de 2013.

[7] DHARMA, N. I G. et al. Time-based DDoS detection and mitigation for SDN controller. **2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)**, Agosto de 2013.

[8] MOUSAVI, S. M.; ST-HILAIRE, M. Early detection of DDoS attacks against SDN controllers. **2015 International Conference on Computing, Networking and Communications (ICNC)**, p. 77-81. Fevereiro de 2015.

[9] YAN, Q. et al. Software-Defined Networking (SDN) and Distributed Denial of Service (DDoS) Attacks in Cloud Computing Environments: A Survey, Some Research Issues, and Challenges. **IEEE Communications Surveys & Tutorials**, v.18, n. 1, p. 602-622. Outubro de 2015.

[10] SDX Central, SDN Security Challenges in SDN Environments. Disponível em < https://www.sdxcentral.com/security/definitions/security-challenges-sdn-software-defined-networks>. Acesso em: 28 de outubro de 2016.

[11] OPEN NETWORKING FOUNDATION (ONF). Principles and Practices for Securing Software Defined Networks. Janeiro de 2015. Disponível em: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/Principles_and_Practices_for_Securing_ Software-Defined_Networks_applied_to_OFv1.3.4_V1.0.pdf>. Acessado em: 01 de novembro de 2016.

[12] SHIN, S. et al. Enhancing Network Security through Software Defined Networking (SDN). **2016 25th International Conference on Computer Communication and Networks (ICCCN)**. Agosto de 2016.

[13] WEIDMAN, G. **Penetration Testing: A Hands-On Introduction to Hacking**, 1. ed. San Francisco, CA. No Starch Press, 2014.

[14] Open vSwitch, Open vSwitch Manual. Disponível em <http://openvswitch.org/support/dist-docs/ovs-ofctl.8.txt>. Acessado em: 29 de outubro de 2016.

[15] Open vSwitch. Disponível em <http://openvswitch.org>. Acessado em: 29 de outubro de 2016.

[16] RISDIANTO, A. C.; MULYANA, E. Implementation and analysis of control and forwarding plane for SDN. **2012 7th International Conference on Telecommunication Systems, Services and Applications (TSSA)**, p. 227-231. Outubro 2012.