Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Colegiado de Engenharia de Computação

Testes de segurança em SDN's, utilizando Honeypot

Helton Ribeiro Lustosa

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:
Theo Silva Lins

Setembro, 2016
João Monlevade/MG

Helton Ribeiro Lustosa

Testes de segurança em SDN's, utilizando Hóneypot

Orientador: Theo Silva Lins

Monografia apresentada ao curso de Engenharia de Computação do Departamento de Computação e Sistemas da Universidade Federal de Ouro Preto como requisito parcial para obtenção do grau de Bacharel em Engenharia de Computação

Universidade Federal de Ouro Preto

João Monlevade

Setembro de 2016

L972t Lustosa, Helton Ribeiro.
Testes de segurança em SDN utilizando Honeypot [manuscrito] / Helton Ribeiro Lustosa. - 2016.

68f.: il.: color; tabs.

Orientador: Prof. Me. Theo Silva Lins.

Monografia (Graduação). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Departamento de Computação e Sistemas de Informação.

1. Segurança de sistemas. 2. Redes - Segurança. 3. Rede Definida por Software (SDN) . 4. Recursos de rede de computador (HoneyPot). I. Lins, Theo Silva . II. Universidade Federal de Ouro Preto. III. Titulo.

CDU: 004.056.53

Catalogação: ficha@sisbin.ufop.br

FOLHA DE APROVAÇÃO DA BANCA EXAMINADORA

Testes de segurança em SDN's, utilizando Honeypot

Helton Ribeiro Lustosa

Monografia apresentada ao curso de Engenharia de Computação do Departamento de Computação e Sistemas da Universidade Federal de Ouro Preto como requisito parcial para obtenção do grau de Bacharel em Engenharia de Computação aprovada pela Banca Examinadora abaixo assinada:

Theo Silva Lins
Mestre em Ciência da Computação
Orientador
DECSI - UFOP

Erik de Britto e Silva
Mestre em Ciência da Computação
Examinador
DECSI - UFOP

Vinícius Fernandes Soares Mota
Doutor em Ciências da Computação
Examinador
DECSI - UFOP

João Monlevade, 28 de setembro de 2016

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS
COLEGIADO DE ENGENHARIA DE COMPUTAÇÃO

ANEXO VII – Ata de Defesa

ATA DE DEFESA

Aos 28 dias do mês de Setembro de 2016, às 17 horas e 00 minutos, no Laboratório de BD/Redes do ICEA/UFOP - C304, foi realizada a defesa de Monografia pelo aluno Helton Ribeiro Lustosa, sendo a Comissão Examinadora constituída pelos professores: Prof. MSc. Theo Silva Lins , Prof. MSc. Erik de Britto e Silva e Prof. Dr. Vinícius Fernandes Soares Mota. O candidato apresentou a monografia intitulada: "Testes de segurança em SDN's, utilizando Honeypot". A comissão examinadora deliberou, por unanimidade, pela aprovação do candidato, concedendo-lhe o prazo de 15 dias para incorporação no texto final das alterações sugeridas. Na forma regulamentar, foi lavrada a presente ata que é assinada pelos membros da Comissão Examinadora e pelo formando.

João Monlevade, 28 de Setembro de 2016.

Prof. MSc. Theo Silva Lins
Professor Orientador/Presidente

Prof. MSc. Erik de Britto e Silva
Professor Convidado

Prof. Dr. Vinícius Fernandes Soares Mota
Professor Convidado

Helton Ribeiro Lustosa
Formando

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS
COLEGIADO DE ENGENHARIA DE COMPUTAÇÃO

ANEXO II – TERMO DE RESPONSABILIDADE

Curso Engenharia de Computação

TERMO DE RESPONSABILIDADE

Eu, Heleton Ribeiro Fonseca ,
declaro que o texto do trabalho de conclusão de curso intitulado
"Teste de segurança em SDN's utilizando Honeypot
" é de minha inteira responsabilidade e que não há utilização de texto, material fotográfico, código fonte de programa ou qualquer outro material pertencente a terceiros sem as devidas referências ou consentimento dos respectivos autores.

João Monlevade, 21 de Setembro de 2016

Heleton Ribeiro Fonseca
Assinatura do aluno

Este trabalho é dedicado à todos que desejam ter uma base sobre SDN e verificar a sua viabilidade considerando sua vulnerabilidade.

Agradecimentos

Gostaria de agradecer principalmente a minha família que me forneceu apoio durante todos estes anos e a meus amigos e colegas que contribuíram tanto para meu crescimento pessoal e profissional durante a realização do curso. Ao meu orientador Theo Silva Lins por ter dado esta oportunidade de desenvolver mais um trabalho juntamente a ele, aos demais professores que contribuíram para a aquisição de conhecimento aplicado neste e em diversos outros trabalhos ao longo da graduação, fica o meu muito obrigado.

"Que os vossos esforços desafiem as impossibilidades,
lembrai-vos de que as grandes coisas do homem foram
conquistadas do que parecia impossível."
(Charles Chaplin)

Resumo

O rápido crescimento tecnológico influenciou profundamente na demanda por soluções que possam dar suporte ao crescente número de usuários da grande rede de computadores. Isso, sem ainda contar os novos padrões de exigência dos consumidores e das aplicações que cada vez mais requisitam qualidade e agilidade, temos atualmente gamas de usuários em jogos multiplayers online, serviços de streaming, pesquisa e muitos outros que necessitam de escalabilidade e QoS. Falhas podem gerar prejuízos milionários como constantemente é noticiado por toda mídia e claro, isso nunca é desejado. As SDN's surgem desta necessidade, onde devido à complexidade e à quantidade de protocolos utilizados em tecnologias atuais, muitos usuários passam por cenários de ineficiência do serviço devido ao fato de que a rede não da conta de gerir todos os dados. Neste trabalho de conclusão de curso, apresenta-se testes através de uma honeypot para análise de fatores externos reais e ataques forçados realizados contra três dos principais controladores OpenFlow no mercado de forma a identificar prováveis vulnerabilidades, ineficiências ou lentidão e se possível apresentar soluções.

Palavras-chaves: SDN. Honeypot. Controlador. Segurança.

Abstract

The fast growth technological influenced deeply by the request for solutions that can give support the growing number of users of the large computer network. This, without even counting the new patterns of consumer and applications demand that increasingly ordering quality and speed, we currently have ranges of users in online multiplayer games, streaming services, research and many others that need scalability and QoS. Failures can generate millionaires losses as is constantly reported by all social media and of course, it is never desired. SDN's come this need, which due to the complexity and amount of protocols used in current technologies, many users go through the service inefficiency scenarios due to the fact that the network does not support manage all data. In this conclusion work of course, presents tests through a honeypot for analysis of real external factors and forced attacks carried out against three major OpenFlow controllers in the market to identify potential vulnerabilities, inefficiencies or slowness and can provide solutions.

Key-words: SDN. Honeypot. Controller. Security.

Lista de ilustrações

Figura 1 – Arquitetura do Pox 25
Figura 2 – Arquitetura do Floodlight 26
Figura 3 – Arquitetura do OpenDayLight 27
Figura 4 – Representação da long tail 31
Figura 5 – SDN com OpenFlow 33
Figura 6 – Sistema Operacional Utilizado 42
Figura 7 – Execução do OpenDayLight 42
Figura 8 – Sistema Web do OpenDayLight 43
Figura 9 – Topologia da Rede do OpenDayLight 44
Figura 10 – Execução do POX 45
Figura 11 – Execução do FloodLight 45
Figura 12 – Tabela Iptables em um dos servidores 46
Figura 13 – Tabelas antes de iniciar controlador 47
Figura 14 – Tabelas após iniciar controlador 48
Figura 15 – Tabelas antes de iniciar controlador Pox 48
Figura 16 – Tabelas após iniciar controlador Pox 49
Figura 17 – Tabelas antes de iniciar controlador FloodLight 49
Figura 18 – Tabelas após iniciar controlador FloodLight 50
Figura 19 – Diretório de Backup com arquivos temporários 52
Figura 20 – Arquivo de registro com Login e Senha 52
Figura 21 – Diretório com registro temporário da aplicação maliciosa 53
Figura 22 – Lista good2 53
Figura 23 – Exemplo de execução do T50 56
Figura 24 – Registro de SYN flooding no ODL 56
Figura 25 – Registro de SYN flooding no POX 57
Figura 26 – Registro de SYN flooding no FloodLight 57
Figura 27 – Arquivo de configuração SSH 61
Figura 28 – Arquivo de configuração do Block Hosts 62
Figura 29 – Arquivo de configuração do KNOCKD 64

Lista de tabelas

Tabela 1 – Tabela de informação sobre os controladores 24
Tabela 2 – Tabela com descrição dos computadores utilizados 41
Tabela 3 – Tabela com teste de banda utilizando Iperf3 55

Lista de abreviaturas e siglas

ACL	Access Control List
API	Application Programming Interface
ASIC	Application Specific Integrated Circuits
CPU	Central Processing Unit
DAC	Discretionary Access Control
DoS	Denial of Service
DDoS	Distributed Denial of Service
FPGA	Field-Programmable Gate Array
IP	Internet Protocol
MAC	Media Access Control
NFV	Virtualização das funções de rede
NMAP	Network Mapper
NOS	Network Operating System
ODL	OpenDaylight
QoS	Qualidade de Serviço
RBAC	Role-based Access Control
RPC	Remote Procedure Call
SDN	Software-Defined Network
SSH	Secure Socket Shell
TCP	Transmission Control Protocol
UFRJ	Universidade Federal do Rio de Janeiro
WiFi	Wireless Fidelity

Sumário

I INTRODUÇÃO 17

1 INTRODUÇÃO . . . 18
1.1 PROBLEMA . . . 18
1.2 OBJETIVOS . . . 19
1.2.1 Objetivo geral . . . 19
1.2.2 Objetivos específicos . . . 19
1.3 JUSTIFICATIVA . . . 19
1.4 ESTRUTURA DO TRABALHO . . . 20
1.4.1 Metodologia . . . 20
1.4.2 Organização do Trabalho . . . 21

II CONCEITOS GERAIS E REVISÃO DA LITERATURA 22

2 REVISÃO BIBLIOGRÁFICA . . . 23
2.1 Preparação de pesquisa . . . 23
2.1.1 Arquitetura de um SDN . . . 23
2.1.1.1 Camada de Aplicação . . . 23
2.1.1.2 Camada de Plano de Controle . . . 23
2.1.1.3 Camada de Plano de Dados . . . 24
2.1.2 Controladores . . . 24
2.1.2.1 Pox: . . . 24
2.1.2.2 Floodlight: . . . 25
2.1.2.3 OpenDayLight: . . . 27
2.2 Definições . . . 28
2.3 Modelos de Implantação de SDN . . . 28
2.3.1 SDN de pesquisa . . . 28
2.3.2 SDN empresarial . . . 29
2.4 Modelos de Serviço . . . 30
2.4.1 Mobile Service Provider(Provedor de serviço móvel) . . . 30
2.4.2 Carrier(Portadora) . . . 30
2.4.3 Web 2.0 . . . 31
2.4.4 Nuvem pública ou privada . . . 31
2.4.5 Serviços financeiros . . . 32
2.5 Principais Características de uma SDN . . . 32
2.5.1 Protocolo OpenFlow . . . 32

2.5.2 Alterações na gerência da rede . . . 33
2.5.3 Abstrações . . . 34
2.5.4 Facilidade de criação de features e protocolos . . . 34
2.5.5 Configuração da rede conforme a demanda . . . 34

III REFERENCIAIS TEÓRICOS 36

3 TRABALHOS RELACIONADOS . . . 37

IV RESULTADOS 39

4 TESTES . . . 40
4.1 Plataforma de trabalho . . . 41
4.1.1 Honeypot com OpenDayLight . . . 42
4.1.2 Honeypot com POX . . . 44
4.1.3 Honeypot com Floodlight . . . 45
4.2 Levantamento de informações . . . 46
4.2.1 IPTABLES . . . 46
4.2.2 NETSTAT . . . 47
4.2.2.1 Tabelas de roteamento com OpenDayLight . . . 47
4.2.2.2 Tabelas de roteamento com Pox . . . 48
4.2.2.3 Tabelas de roteamento com FloodLight . . . 49
4.2.3 NMAP . . . 50
4.2.4 TCPDump . . . 50

5 EXPLORAÇÃO DE VULNERABILIDADES . . . 51
5.1 Ataques externos . . . 51
5.1.1 Ataque por SSH . . . 51
5.2 Simulações . . . 54
5.2.1 Utilizando IPerf . . . 54
5.2.2 Utilizando o T50 . . . 55
5.3 Problemas enfrentados . . . 58

6 SOLUÇÕES ENCONTRADAS . . . 60
6.1 Alterar porta padrão e configurações padrões . . . 60
6.2 Block Hosts . . . 61
6.3 KNOCKD . . . 63
6.4 Bash Script . . . 64
6.5 Mecanismo de Autenticação(AuthFlow) . . . 65

Conclusão **67**

REFERÊNCIAS **68**

Parte I

Introdução

1 INTRODUÇÃO

Atualmente vivemos em um mundo onde as tecnologias de comunicação avançaram em um ritmo exacerbado, em parte isso se deve a uma daquelas que é considerada uma das maiores revoluções tecnológicas da humanidade por diversos autores como o estudo de Ana Maria Nicolaci-da-Costa (COSTA, 2002) sobre revoluções tecnológicas e o trabalho de Pierre Lévy (LÉVY, 2000) sobre revolução de comunicações, cujo trabalho foi traduzido pelo professor Brasileiro Juremir Machado da Silva, ambos datam do inicio do século e podemos atualmente constatar de fato o que foi verificado naquela época. A internet possibilitou não só a disponibilização de conteúdo de diversos tipos com acesso em tempo real, mas também possibilitou que novas tecnologias e serviços surgissem. As redes definidas por software surgiram neste meio como uma forma de atender novas necessidades, infelizmente todo hardware possui um dado limite e ocorre constantemente de ficarem obsoletos com pouco tempo. Escalabilidade, desempenho, baixo custo, todos são fatores primordiais em qualquer setor, não é diferente quando tratamos de soluções voltadas a redes de computadores. A SDN(*Software Defined Network*) possibilita tirar a carga do hardware e transferir funções para um sistema programável. Porém, como se trata de um projeto novo a tendência é de que apareçam muitos problemas e vulnerabilidades. Para tratar isto, existem alguns métodos para enrijecer a segurança ou testa-la, neste trabalho a proposta é de explorar possíveis falhas e brechas em um ambiente controlado utilizando simulações de invasão ou outra forma condenar o sistema e também disponibilizando o ambiente para ataques externos, simulando assim toda a rede em uma honeypot/honeynet e a medida do possível prover ou apontar soluções.

1.1 PROBLEMA

Como foi comentado anteriormente, boa parte da tecnologia que cerca a implementação de uma SDN é consideravelmente recente. Devido a isso, ainda existe pouca pesquisa já concluída e disponível, principalmente na internet, com relação ao assunto e menor ainda a quantidade com relação à aquelas que tratam do quesito de segurança. Por se tratar de um ferramenta de grande importância, afinal a partir dele é gerenciado uma grande quantidade de recursos e caso seja suscetível a falhas de segurança, muito pode se estar em risco, principalmente se for utilizado na rede principal da organização em que estiver sendo utilizada, já que informações ou até mesmo a segurança com relação a acesso ao servidor pode ser prejudicada.

1.2 OBJETIVOS

Este estudo pretende verificar algumas possíveis vulnerabilidades que algumas ferramentas utilizadas na criação de uma Rede Definida por Software(SDN), principalmente com relação aos controladores. Como base serão utilizados alguns controladores já conhecidos e criada então uma Honeypot para análise de ações tomadas por invasores e realização de testes. Dessa forma, pretende-se atingir os objetivos gerais e específicos considerados adiante.

1.2.1 Objetivo geral

Inicialmente devemos conhecer alguns dos controladores OpenFlow de código aberto mais conhecidos e utilizados para estudos acadêmicos e por algumas organizações. Serão, então, testados em uma rede definida dentro de uma honeypot. Desta forma, podemos avaliar algumas questões ligadas a segurança através de testes realizados em laboratório e análise de possíveis tentativas de invasão. Assim o objetivo ao final é poder apontar casos onde foram verificadas falhas e apontar possíveis soluções caso sejam encontradas.

1.2.2 Objetivos específicos

Para realizar a pesquisa deverão ser levantadas informações sobre cada controlador e diferenciar cada implementação, para assim poder testa-los e identificar algumas soluções para problemas aparentes. Ainda, haverá de ter algumas implementações como, por exemplo, de módulos que executem determinadas ações que inutilizem ou amenizem ataques. Para tanto, deve-se levantar uma pesquisa por referências de soluções já conhecidas e que possam ser implementadas para uma SDN além de articular um meio de enrijecer a segurança.

1.3 JUSTIFICATIVA

Claro que muitas dúvidas podem surgir com base na importância deste tipo de pesquisa, afinal, como já foi dito anteriormente são ferramentas que ainda estão em fase inicial de teste e desenvolvimento, desta forma possuem apenas algumas versões. Porém, deve-se perceber que esta é uma tendência, como já foi citado em outros trabalhos como o Aprovisionamento de QoS de (ARAÚJO, 2013) e o trabalho de (RUIZ, 2015) que proporciona uma visão geral do assunto além de comparações com uma estrutura de rede comum. Podemos ainda citar que serão analisadas questões de vulnerabilidade em redes virtuais rodando juntamente à controladores OpenFlow para casos de banda cheia e ataques de negação de serviço realizados em laboratório utilizando o Iperf e T50 respectivamente, além da análise de dados de invasores. Um trabalho atualmente que

reúne estas características dificilmente é encontrado, principalmente casos que contam com utilização de honeypot.

1.4 ESTRUTURA DO TRABALHO

Nesta etapa será realizada uma breve descrição de como o trabalho foi organizado, o conteúdo de cada capítulo descrito neste trabalho, além dos motivos da escolha das referências, a forma com que foi abordada questão de segurança de redes e metodologia adotada.

1.4.1 Metodologia

Foi elaborado um cronograma de pesquisa na primeira etapa de desenvolvimento deste trabalho para entender a estrutura de uma SDN; como essa tecnologia funciona; seus desafios; suas vulnerabilidades, além da busca por informações referentes a instalação e configuração de uma honeypot. Assim, foram feitas pesquisas utilizando artigos, trabalhos de pesquisa e livros que tratam sobre o tema, seguido pela análise de dados resultante de técnicas de ataque em diferentes controladores OpenFlow. Logo, podemos dizer que foi realizada neste trabalho uma pesquisa de abordagem qualitativa, onde todo o processo foi definido e descrito, a teoria foi definida a partir da elaboração dos testes além do fato de que os dados são interpretados de forma subjetiva assim como a proposta de soluções. Para a proposta deste trabalho foram selecionadas alguns controladores já conhecidos e estáveis, logicamente com diversas diferenças e até mesmo propostas diferentes.

Como o conceito do trabalho é uma análise de segurança, partiu-se do princípio de que o melhor método seria através de uma Honeypot. Isso, porque a proposta de se instalar uma é o de que ataques externos sejam identificados e através do monitoramento possam ser identificadas os métodos e ferramentas utilizadas e assim expor as fragilidades que possam surgir. Alguns autores nomeiam uma honeypot que possui como característica englobar uma rede inteira como uma honeynet, como no caso estamos simulando uma rede e utilizando um controlador OpenFlow talvez seja uma nomenclatura aplicável. Outro ponto, seria o fato de que não podemos depender apenas do acaso e de ataques que podem vir a ocorrer em pesquisas deste âmbito, também foram estipulados simulações para levantar dados com ferramentas já conhecidas e utilizadas.

Para realizar os testes e avaliar os desafios de segurança dos controladores, foram utilizados as informações coletadas com os dados das simulações em complemento com os dados colhidos pelos ataques externos. Além da possibilidade de comparação entre as plataformas que surgiu da comparação entre os resultados individuais de cada uma nas simulações utilizando a mesma metodologia de ataque e mesmo ambiente para todas.

1.4.2 Organização do Trabalho

A divisão do trabalho se dá como segue: no capítulo 2 encontra-se uma revisão bibliográfica que conta com uma preparação de pesquisa e breve descrição dos controladores e da SDN. O capítulo 3 expõe alguns trabalhos relacionados que inclusive foram utilizados como base para realização deste e tratam de temas semelhantes ao abordado neste trabalho. São trabalhos realizados por outros autores que tenham como tema segurança em redes, utilização de controladores Openflow, implementação de SDN e/ou honeypot. A revisão conta com uma coleção de diversas referências para explorar o tema de segurança em SDN e implementação de uma honeypot mais a fundo, descrevendo suas diversas definições, formas de implementação e configuração, modelos de serviço, principais características e os desafios encontrados neste ambiente. No capítulo 4 se encontram a configuração do ambiente de trabalho e testes realizados com algumas ferramentas. Capítulo 5 contém a etapa de exploração de vulnerabilidades utilizando como base análise de ataques externos e simulações. O capítulo 6 contém algumas possíveis soluções para melhoria de segurança. Por fim, temos uma dissertação que expõe as conclusões acerca do trabalho.

Parte II

Conceitos Gerais e Revisão da Literatura

2 Revisão bibliográfica

Neste capítulo o foco e voltado à prover uma breve contextualização contendo o modo como foi preparada a pesquisa, uma descrição em camadas de uma rede definida por software, além de uma descrição de cada controlador utilizado e modelos tanto de implantação quanto de serviço.

2.1 Preparação de pesquisa

Antes de qualquer outra coisa, devemos deixar claro como foram realizados os testes, assim como o ambiente, porque da escolha dos controladores utilizados e sistemas utilizados. Primeiramente, vamos considerar que o trabalho foi desenvolvido basicamente em um laboratório de computação da Universidade Federal de Ouro Preto onde foram utilizados computadores com configurações modestas, porém com acesso a internet através de *link* dedicado e cada máquina possuía um IP externo.

2.1.1 Arquitetura de um SDN

Uma rede definida por software comumente é dividida em três camadas, isso devido a sua arquitetura e a sua apresentação da ferramenta. Desta forma, os desenvolvedores como (MCCAULEY, 2016), (FLOODLIGHT, 2016) e (FOUNDATION, 2016) normalmente fazem uma breve descrição em suas páginas, como será exibido abaixo:

2.1.1.1 Camada de Aplicação

Quando falamos de aplicação em SDN, devemos comentar sobre *Northbound* APIs. Esses API's são interfaces programáveis, situadas entre a plataforma do controlador e as aplicações rodando *"on top"*. Ainda segundo (FLOODLIGHT, 2016) a arquitetura de 3 camadas incide sobre essas API's , que expõem os modelos de dados de abstração de rede universais e funcionalidade dentro do controlador para uso por aplicativos de rede. Essas interfaces são publicadas e aberto para a comunidades *OpenSource* a fim de desenvolvimento.

2.1.1.2 Camada de Plano de Controle

Cada controlador OpenFlow pode utilizar ferramentas diferentes, assim como ser desenvolvido em linguagens distintas. OpenFlow é um padrão aberto gerido pela fundação Open Networking. Ele especifica um protocolo através do qual um controlador remoto pode modificar o comportamento dos dispositivos de rede através de um conjunto de

instruções de encaminhamento bem definida. Em geral são concebidos para trabalhar com o crescente número de switch's, roteadores e pontos de acesso que suportam o padrão OpenFlow.

2.1.1.3 Camada de Plano de Dados

O controlador OpenFlow geralmente está fisicamente dissociado do plano de transmissão de dados, ou seja, um switch físico ou virtual pode encaminhar pacotes e um servidor separado pode executar o plano de controle de rede. Esta dissociação permite que o plano de controle citado anteriormente possa ser implementado utilizando um modelo de distribuição diferente do que o usado no plano de dados. Em segundo lugar, permite o desenvolvimento do plano de controle e do ambiente de execução, para gerar uma plataforma diferente do que a de CPU's, tradicionalmente de baixa potência de gestão, encontradas no hardware de switches e roteadores. Para que isso funcione, é necessário que haja alguma forma estabelecida para o controlador se comunicar com o plano de dados. O mais popular desses mecanismos é o OpenFlow que fornece uma interface padrão para controlar switches.

2.1.2 Controladores

Cada controlador possui uma característica distinta, seja pela linguagem adotada, neste caso optamos por trabalhar com alguns modelos descritos na Tabela 1 que possuem porte comercial e/ou acadêmico.

Tabela 1 – Tabela de informação sobre os controladores

Controlador	Código Aberto	Linguagem	Multi-threaded	Autor
OpenDaylight	sim	Java	sim	Vários
POX	sim	Python	sim	Nicira Networks
Floodlight	sim	Java	-	Big Switch Networks

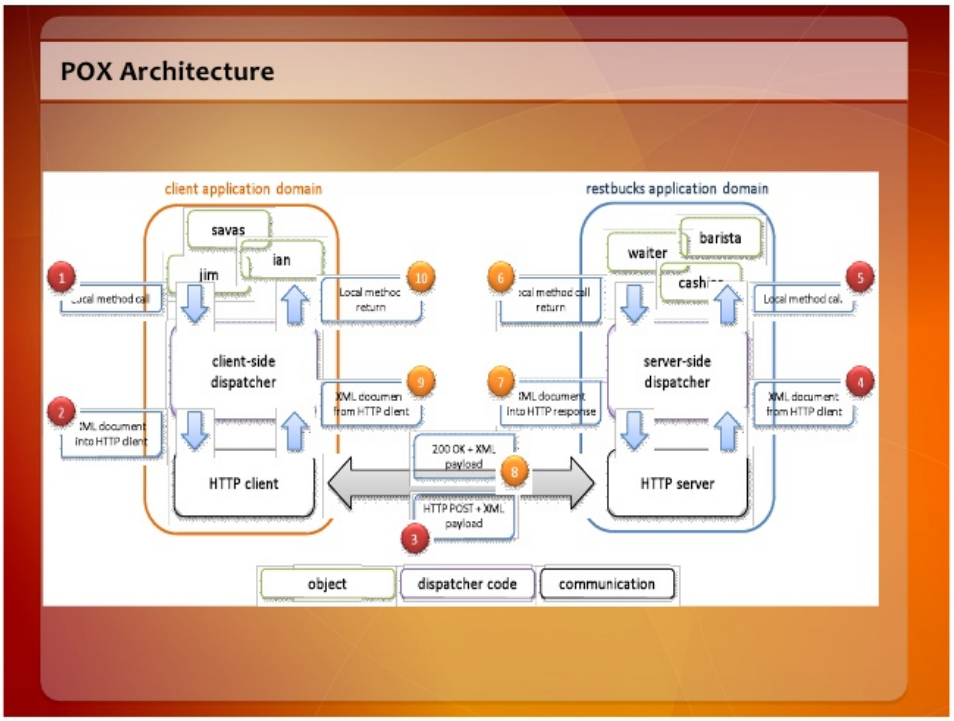
Fonte: Elaborada pelo autor

2.1.2.1 Pox:

Segundo informações do próprio site (MCCAULEY, 2016) este controlador é escrito em linguagem python e é bastante difundido por conta dos pesquisadores originais manterem suas atualizações. A sua principal utilização é acadêmica, ou seja, seu intuito é ser uma ferramenta para pesquisa, devido a isto talvez este seja o mais fácil de se utilizar

dentre os que foram testados. Possui uma execução direta e os seus módulos escritos em python são fáceis de serem alterados e redigidos, o que facilita os testes de módulos de segurança. Alterações como endereço ip, arquitetura, entre outros são simples e podem ser realizados com os módulos nativos e o processo é explicado na documentação da ferramenta em (MCCAULEY, 2016). A arquitetura se adapta a qualquer serviço que a rede pode oferecer, na Figura 1 - Arquitetura do Pox temos um exemplo da arquitetura para uma rede sendo utilizada por servidor e cliente HTTP, percebe-se claramente a distinção dos planos com relação ao tratamento dos objetos, do código de despacho definido pelo controlador e a comunicação entre ambos os domínios.

Figura 1 – Arquitetura do Pox



Fonte: Caelum[1]

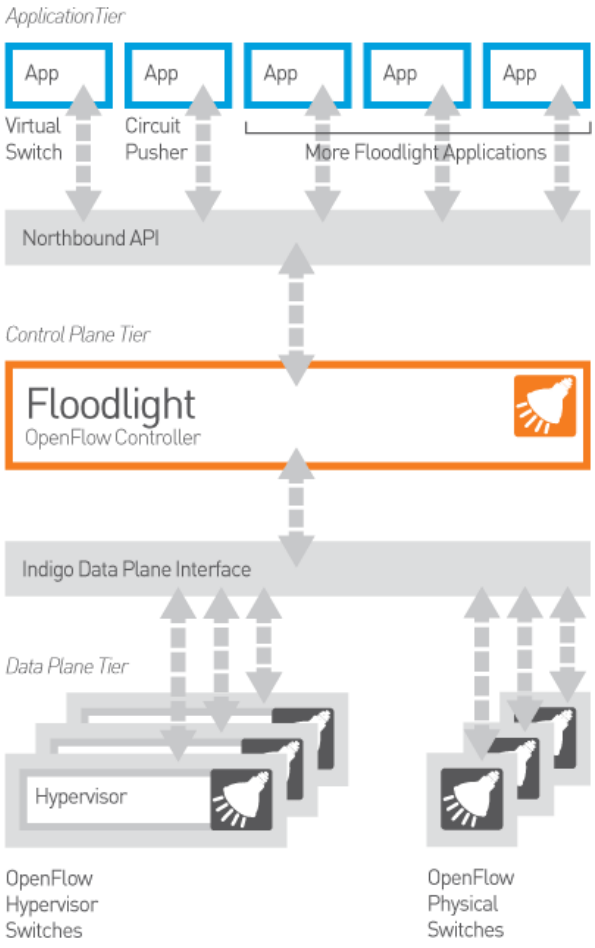
2.1.2.2 Floodlight:

Segundo informações fornecidas no próprio portal do projeto (FLOODLIGHT, 2016) este controlador OpenFlow é desenvolvido em linguagem Java, possui licenciamento Apache onde, segundo o grupo, proporciona aos clientes e desenvolvedores de aplicativos uma ferramenta com a máxima segurança de investimento e a sua arquitetura SDN permanecerá sempre independentes dos interesses do fornecedor, logo o foco é voltado ao ramo empresarial. Possui o apoio de comunidade de desenvolvedores, que inclui um

[1] Disponível em: <http://www.slideshare.net/guilhermecaelum/rest-in-practice> Acesso em Abril de 2016.

bom número de engenheiros do Big Switch Networks (NETWORKS, 2016). A Figura 2 - Arquitetura do Floodlight exibe o modo como o controlador se relaciona e como as informações são entregues, temos então o Plano de Gerenciamento que basicamente é uma interface entre a SDN e o administrador da rede onde as já citadas API's exercem a função de comunicação com o controlador. A interface *Northbound* é uma API que se encarrega da função de realizar a comunicação entre o plano de gerenciamento e o plano de controle, este ultimo plano é composto pelo NOS, também conhecido como Sistema Operacional da Rede de carácter centralizado que fornece APIs para desenvolvimento, abstração, além de outros serviços essenciais. A ponte entre os elementos de controle e encaminhamento de dados normalmente é feita por uma API conhecida como *Southbound*, no caso do Floodlight temos a *Indigo Data Plane Interface*, o plano de dados que aparece logo a seguir tem como papel encaminhar os dados na rede e para isso, diferentemente de roteadores e comutadores convencionais, ele não carrega implementações de protocolos e programas complexos(CAMPOS, 2016).

Figura 2 – Arquitetura do Floodlight



Fonte: Project Floodlight[2]

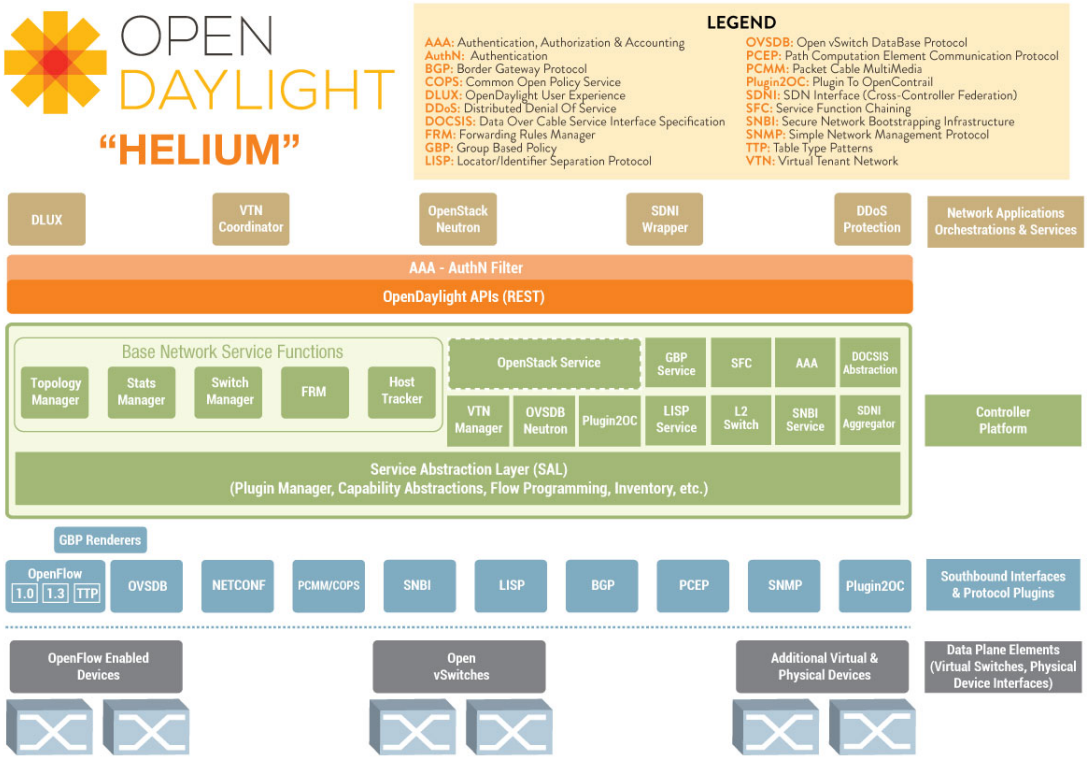
[2] Disponível em: <http://www.projectfloodlight.org/floodlight> Acesso em Abril de 2016.

2.1.2.3 OpenDayLight:

Nesta sessão temos a descrição do projeto de mais um controlador de código aberto, fundado recentemente em 2013 com a colaboração de desenvolvedores, usuários e demais membros para a produção de programas relevantes, eventos e recursos para a plataforma.

O programa conta com a parceria entre as maiores empresas do ramo de soluções para redes como Cisco, IBM, HP, NEC, Dell, Huawei, dentre muitas outras. O ODL, como costuma ser chamado por seus desenvolvedores, permite serviços de rede através de um espectro de hardware em ambiente com múltiplos fornecedores. A arquitetura presente ilustrada na Figura 3 - Arquitetura do OpenDayLight permite aos usuários controlar aplicações, *plugins* e protocolos, além de fornecer conexão entre consumidores externos e fornecedores. Ainda segundo seus desenvolvedores, ODL não é um controlador SDN comum, mas sim uma plataforma comum que pode ser configurado de diversas formas para resolver problemas.

Figura 3 – Arquitetura do OpenDayLight



Fonte: SDN Hub[3]

[3] Disponível em: <http://sdnhub.org/tutorials/opendaylight> Acesso em Abril de 2016.

2.2 Definições

(SPITZNER, 2001), define que Honeynets são ferramentas de pesquisa que consistem de uma rede projetada especificamente para ser comprometida.

(PROJECT, 2004) aborda que uma Honeynet nada mais é do que um tipo de honeypot. Especificamente, é um honeypot de alta interatividade, projetado para pesquisa e obtenção de informações dos invasores. É conhecido também como "honeypot de pesquisa".

(GUEDES et al., 2012) podemos dizer que uma Rede Definida por Software (SDN) é caracterizada pela existência de um sistema de controle (software) que pode controlar o mecanismo de encaminhamento dos elementos de comutação da rede por uma interface de programação bem definida.

Segundo a definição de (ALMEIDA, 2016) criada utilizando com base o material de (MCKEOWN, 2011), temos que SDN é uma arquitetura emergente onde a funcionalidade de controle da rede (plano de controle) é destacada do hardware que realiza a comutação (plano de dados), permitindo a programabilidade da rede. O controle, que era integrado fortemente nos equipamentos individuais de rede, passa a ser realizado por controladores SDN logicamente centralizados, permitindo a abstração da infraestrutura de rede para as aplicações e serviços de rede. Dessa forma, aplicações podem tratar a rede como uma entidade lógica ou virtual.

2.3 Modelos de Implantação de SDN

Uma SDN não necessariamente é implementada apenas em empresas, existem implementações para diversos casos onde suas funcionalidades são úteis e nesses casos sempre é preciso se considerar algumas questões de restrições de acesso ao recurso e segurança dos dados.

2.3.1 SDN de pesquisa

Como se trata de uma ferramenta consideravelmente recente, muitas implementações surgiram no âmbito de pesquisa. Na verdade, muitos consideram o NOX que é um projeto anterior ao POX, porém desenvolvido em C++ como o primeiro controlador SDN baseado em OpenFlow. Dentre as vantagens temos, um ambiente escalável onde um único controlador suporta rodar enumeras aplicações e switch's, podem ser implementadas GUI's para assessoria no uso e monitoramento da rede, possibilidade de criar um número exorbitante de nós que aparentemente só se encerra quando não houver mais memória RAM disponível para criação de novos nós ou descritores de registro, logo um número absurdo realmente.

Logo, percebemos que uma rede desenvolvida para pesquisa pode seguir diversas frentes diferentes, logicamente vai depender da pesquisa aplicada, da aplicação que será utilizada e do estudo que está sendo elaborado. Podemos, assim ter uma SDN onde ela faz parte do projeto ou o módulo executado no controlador será o alvo da pesquisa, infinitas possibilidades surgem e diversos modelos de serviços podem ser aplicados.

2.3.2 SDN empresarial

No ramo dos negócios diversas aplicações já existem e muitas das vezes são utilizadas outros conceitos como computação em nuvens e NFV. Neste sentido temos mudanças na maneira como as empresas de telecom projetam e gerenciam suas redes, além de alterações no modelo de negócios. Isso porque, novas oportunidades de produtos e serviços mais eficientes e flexíveis podem ser elaborados, porém a ferramenta por si só não gera receita e sim oferece uma nova maneira de arquitetar redes que permite gerenciar infraestrutura e entregar serviços de uma nova forma.

Dentre as enumeras vantagens para serviços de rede em comunicações temos a possibilidade de melhorar o gerenciamento da rede, flexibilidade, permite gestão da automação do tráfego, engenharia de banda o que é muito importante principalmente em servidores, redução drástica de custos de TI e a possibilidade de ter uma rede que se ajuste às necessidades dos clientes com disponibilização, distribuição e mobilidade de aplicações e serviços de forma automatizada, sob demanda e em grande escala. Segundo um artigo publicado no *Computerworld*[4], temos alguns exemplos de empresas tanto nacionais como internacionais agindo para adequar o uso da nova tecnologia:

- A AT&T, dos Estados Unidos, tem feito uma transição agressiva de sua infraestrutura para o gerenciamento virtualizado de rede. Seu serviço de rede sob demanda, por exemplo, é capaz de consumir banda da mesma maneira como consumidores consomem serviços na nuvem: escalando de acordo com suas necessidades, de maneira dinâmica e em tempo real.

- A NTT Communications, do Japão, começou a usar SDN há cerca de cinco anos para gerenciar interconexões entre seus próprios data centers. Recentemente a tecnologia passou a ser um produto corporativo: a infraestrutura de SDN suporta serviços na nuvem privada global da NTT, entregue por meio de 130 data centers espalhados em vários países do mundo.

- No ano passado(2015) a Tim Brasil revelou que está realizando SDN, assim como outras tecnologias ligadas a NFV e nuvem em sua rede para 2020. O maior obstáculo

[4] <http://computerworld.com.br/sdn-e-nfv-abrem-novas-oportunidades-empresas-de-telecom-0> Acesso em Abril 2016.

para a empresa são as redes com tecnologia defasada, que limitam a inovação e o "time to market".

- Telefônica também está avançando em virtualização e prevê alcançar 30% da rede virtualizada em 2016, como parte de seu programa de unificação global.
- A OI havia anunciado também que estaria realizando pilotos de SDN e NFV e deve capacitar suas áreas de engenharia e operações de redes de telecomunicações.

2.4 Modelos de Serviço

Neste ambiente programável, diversas aplicações podem surgir. Logo a seguir serão citados com base em que alguns autores especialistas na área e grande parte das empresas que prestam serviços utilizando a tecnologia afirmam ser casos onde a sua adoção pode alterar todo o cenário do mercado influenciando não só na flexibilidade dos serviços, mas em redução de custos, inovação e resolução de problemas restritivos.

2.4.1 Mobile Service Provider(Provedor de serviço móvel)

Não é surpresa o tremendo sucesso que é a venda de smartphones e outros dispositivos móveis em todo o mundo. Segundo a revista Pequenas Empresas & Grandes Negócios(NEGOCIOS, 2016), amplamente conhecida e premiada, as vendas globais de smartphones cresceram 3,9% no primeiro trimestre de 2016 com relação ao mesmo trimestre do ano passado. Isso se traduz em números para mais de 349 milhões, ou seja, cerca de 3,8 milhões de aparelhos comercializados por dia, neste contexto temos a demanda por serviços cada vez mais robustos e em grande maioria através de aplicações que utilizam a internet ou alguma de suas ferramentas.

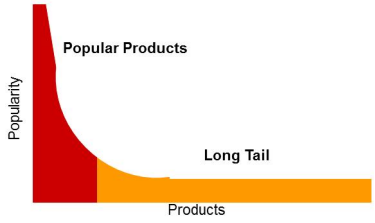
Obviamente, neste cenário empresa alguma deseja manter um espectro e cobertura de serviço limitados além de alto custos e consequentemente uma ruim experiência para o usuário, a SDN surge então como uma ferramenta que permite flexibilizar a dinâmica entre aparelhos móveis e a rede Wi-Fi a fim de refinar o tipo de serviço conhecido como *"Mobile data offloading"*, que é o uso complementar de tecnologias de rede para entregar dados originalmente voltados para redes móveis.

2.4.2 Carrier(Portadora)

O problema de dependência com relação ao fornecedor do produto, seja ela referente a software ou hardware é algo recorrente no contexto de redes. A SDN proporciona um ambiente onde é possível se escrever ou adquirir seu próprio software de forma livre e de baixo custo, afim de criar novas classes de data center ou serviços hospedados e assim atender a todas as necessidades dos usuários, inclusive aqueles que se encontram em uma

faixa distinta pois necessitam de serviços característicos conhecidos como *"long tail"*(Figura 4 - Representação da long tail). O nome vem da forma como essa parcela é representada que aparenta com uma longa calda que vai se afilando e representam em um gráfico de distribuição numérico a porção da distribuição que tem um grande número de ocorrências longe da parte central da distribuição.

Figura 4 – Representação da long tail



Fonte: OPS Rules[5]

2.4.3 Web 2.0

Para muitos, quando se fala em banda de rede cheia pode caracterizar um problema, pois significa que há um grande uso e dependendo da tarefa que será executada pode não haver banda suficiente disponível. Porém, para aqueles que são provedores de serviço como as gigantes Google e Akamai que possuem um grande número de servidores em operação constante o real interesse é manter a banda sempre cheia, uma vez que os custos de se manter o serviço sempre disponível é alto e caso operem com baixo uso seria um desperdício de recursos. Além do fato de que, se por acaso em uma empresa que presta este tipo de serviço alguns de seus data center's estiverem com baixa utilização e outros com sobrecarga de requisições a empresa pode interpretar de forma superficial o caso, o que pode ocasionar a construção de novas estruturas desnecessariamente. Desta forma, ao se utilizar uma estrutura de rede programável o que permite a virtualização de rede e maior agilidade para se gerenciar a utilização dos servidores, podemos atrasar ou até mesmo evitar a construção de novos data center's.

2.4.4 Nuvem pública ou privada

Como já foi brevemente comentado neste trabalho, a computação em nuvem é um tipo de serviço que demanda de flexibilidade e escalabilidade e atualmente um dos fatores que limitam as ferramentas são as arquiteturas de rede que possuem baixa flexibilidade e acabam por inibir a conquista de novos clientes, isso sem contar o fator de ainda haver diversos domínios isolados o que é de inadequado ao contexto de nuvem. A ferramenta em

[5] Disponível em: <http://www.opsrules.com/supply-chain-optimization-blog/bid/175579/Long-Tail-Analysis-for-Supply-Chain-Transformation> Acesso em Abril 2016.

teste neste trabalho surge então como uma possível solução onde uma rede física pode ser virtualizada e de forma teoricamente segura prover a cada cliente um domínio que pode ser personalizado em uma nuvem pública ou privada.

2.4.5 Serviços financeiros

Serviços financeiros estão em um caso a parte, o tipo de aplicação que são utilizados demandam uma rede segura e possui diversas exigências de conformidade para redes isoladas isso sem contar as grandes, complexas e dispendiosas redes físicas. Logo, as vantagens são claras, temos uma forma de simplificar a conformidade além de permitir o isolamento de rede.

2.5 Principais Características de uma SDN

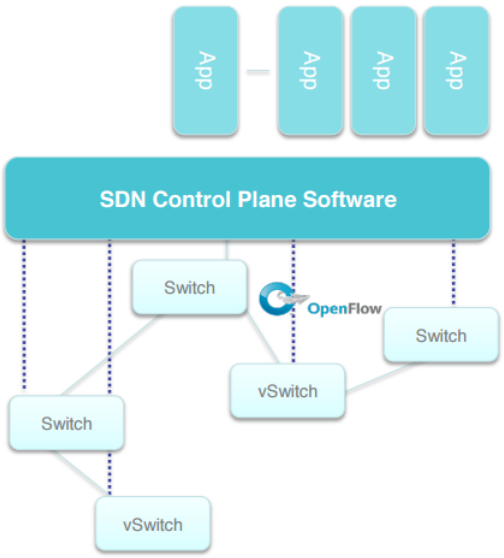
Realmente o número de vantagens é enorme, mas em síntese podemos destacar que a partir da implementação sairíamos de um contexto onde trabalhávamos com uma rede de caráter estático, inflexível e limitante para o surgimento de inúmeras iniciativas, isso sem contar a baixa flexibilidade e agilidade para um novo contexto onde temos uma ferramenta programável e dessa forma habilitada para atingir novas iniciativas de negócios, flexível, ágil e passiva de virtualização.

2.5.1 Protocolo OpenFlow

Proposto por Mckeown (MCKEOWN et al., 2008), é um exemplo de protocolo para se implantar uma Rede Definida por Software inclusive é o protocolo utilizado pelos controladores apresentados neste trabalho. Diego Henrique Duque (DUQUE, 2012) define em seu trabalho o OpenFlow como uma interface de comunicação entre o plano de encaminhamento, que se encontra nos equipamentos de rede, sejam eles switch's ou roteadores além do plano de controle como podemos visualizar na Figura 5 - SDN com OpenFlow. No caso das Redes Definidas por Software ele fica externo ao dispositivo de rede alocado numa máquina física ou virtual.

Ele fornece então aos fabricantes um conjunto de regras padrão, o que permite a adição de novas ferramentas e protocolos e o mais importante é que para isso o software presente nos switch's pouco importam. Se tratando do fluxo de dados, o OpenFlow utiliza a "flow-table"que é uma tabela de dados, além de permitir também a criação de um grupo de tabelas o que define um fluxo com entrada específica representando um método adicional de encaminhamento.

Figura 5 – SDN com OpenFlow



Fonte: Big Switch[6]

Mas então, como isso funciona? Podemos considerar um cenário onde roteadores atuais possuem uma tabela de fluxo onde é possível implementar funcionalidades como firewalls, QoS e a coleta de estatísticas. Porém, como já foi mencionado, tais tabelas diferem de uma para outra de acordo com fabricante e modelo do equipamento, assim o que pode ser feito é utilizar o conjunto de funções que são comuns à maioria dos switches e roteadores para manipular as tabelas de fluxo. Dessa forma, com o OpenFlow pode-se programar a tabela e o administrador da rede consegue agora dividir o fluxo em dois novos, sendo um deles possível tratar a rota e o que cada pacote irá receber e o outro continua isolado e processado normalmente.

Se considerarmos, por exemplo, um pesquisador utilizando o OpenFlow seria possível testar experimentos como novos protocolos de roteamento e modelos de segurança, na mesma rede onde o trafego de fluxo modificado e o normal estão presentes e sem interferir no funcionamento adequado.

2.5.2 Alterações na gerência da rede

Alterações sutis são aparentes ao ver do usuário, porém temos influências diversas como, por exemplo:

- Hardware/Aplicações são agora influenciados por software de código aberto.

[6] Disponível em: <http://www.bigswitch.com/blog/2012/03/16/open-software-defined-networking-sdn-ensuring-you-get-all-the-benefits-of-sdn> Acesso em Abril 2016.

- O plano de controle que anteriormente era distribuído, agora é centralizado.
- ASICs/FPGAs entre outros dispositivos de rede que anteriormente eram configurados e personalizados de acordo com o fabricante dão espaço para qualquer equipamento comercial programável.
- Como dito anteriormente agora os antigos protocolos, mesmo que ainda utilizados com a implantação da SDN, terão atividades substituídas por API's.

2.5.3 Abstrações

Já é comum se atribuir o conceito de abstração da infraestrutura de rede a esse tipo de implementação, onde pode ser relacionado ao encaminhamento de pacotes, o modo de distribuição dos dados ou a gestão global da rede. Por algumas vezes, encontramos até mesmo analogias, onde esta separação do plano de controle e plano de dados oferece ao administrador algo parecido ao que o hypervisor fornece à equipe de TI com relação à autonomia de criação de servidores virtuais. Nesse contexto, temos:

- API comum para hardware de rede programável.
- Algoritmo com estado de distribuição único e simplificado.
- Programador agora interage com toda a rede, em vez de nós individuais.

2.5.4 Facilidade de criação de features e protocolos

Utilizando o protocolo OpenFlow descrito anteriormente, podemos considerar que a criação de API's se torna simplificada, logo é possível se desenvolver funcionalidades e protocolos, independentemente dos fabricantes. Neste sentido, os principais fornecedores de equipamentos de rede estão correndo atrás de ofertar seus produtos com software aberto e compatíveis com o modelo de SDN.

2.5.5 Configuração da rede conforme a demanda

A escalabilidade já é uma tendência em diversos campos como no caso dos serviços em nuvem que podem ser ofertados como software, plataforma ou infraestrutura, sendo que em qualquer caso demanda ser escalável. Neste sentido, surge agora um outro conceito de redes como um serviço onde podemos encontrar uma solução para o sub-sequente problema referente a mão de obra especializada para operar com o grande parque de equipamentos, sendo eles em grande maioria de fabricantes distintos, pois nem sempre é possível de se encontrar um profissional para ocupar o cargo devido ao alto nível técnico exigido, como o plano de controle agora é parcialmente separado do plano de dados o especialista terá

que focar, basicamente, em poucas aplicações e não na grande variedade de aplicações oferecidas por diversos fabricantes.

Outro ponto seria a deficiência que redes *ethernet* possuem para prover suporte à serviços escaláveis como os já citados serviços em nuvem. Isso, se deve principalmente à questões de segurança, QoS, escalabilidade e custos operacionais, claramente as vantagens da SDN ficam evidentes neste contexto, porém surge a dúvida quanto a segurança. Já sabemos que uma rede enquanto programável pode prover QoS, escalabilidade e custos bem mais baixos, porém ainda é cedo para afirmar se realmente são seguras, principalmente por ser implementado em cima de software e ainda ser uma ferramenta muito nova e com poucas versões estáveis, seja com relação ao protocolo OpenFlow que está na presente data na versão 1.2 estável ou com relação aos controladores que também são projetos recentes.

Parte III

Referenciais teóricos

3 Trabalhos relacionados

Neste capítulo serão apresentados alguns trabalhos que utilizaram como base a implantação ou caracterização de Honeypot/Honeynet, redes definidas por software, segurança em redes de computadores, enfim, diversas abordagens que convergem ao assunto abordado neste assunto. Claramente observamos que o fato de a análise se basear no estudo de ferramentas recentemente desenvolvidas, ainda especificamente na questão de segurança, acabamos por enfrentar o problema de ainda existir um número pequeno de artigos publicados e menor ainda é o número de trabalhos que utilizam mais de um controlador e/ou realizam testes práticos nestes, além de também haverem poucos que se aprofundam nos testes em cima de Honeypots. Desta forma, a partir de tudo o que pode ser encontrado durante a realização deste trabalho foram retiradas análises de problemas e desafios encontrados pelos autores, assim como as soluções que foram propostas.

Em seu trabalho Niels Provos(PROVOS, 2003) faz uma análise dos principais problemas para implementação de um honeypot, além de apresentar a Honeyd que é um *framework* para honeypots virtuais. Para mostrar tal ferramenta o autor esclarece algumas questões que esta utiliza como uma forma de enganar ferramentas utilizadas para recolher impressões digitais de rede e a utilização de topologias de roteamento virtuais.

Em sua pesquisa, categorizada como um estudo sobre o "estado da arte"de redes programáveis, Andrés Felipe Ruiz (RUIZ, 2015) realiza uma apanhado histórico relacionado a este tipo de rede, apresenta tanto a arquitetura SDN quanto o protocolo OpenFlow e examina algumas aplicações que existem atualmente e outras que poderão surgir futuramente através de estudos em diferentes linhas de investigação. O trabalho apresentado neste artigo não foca em examinar puramente as aplicações existentes, tão pouco em novas linhas de aplicações, o foco é avaliar alguns dos mais conhecidos controladores no quesito segurança.

O grupo de estudiosos em redes sobre a questão de:"uma abordagem sistêmica para o desenvolvimento de pesquisas em Redes de Computadores"(GUEDES et al., 2012), realiza neste um modelo de minicurso oferecendo diversas informações na área. Este minicurso aborda o assunto de forma teórica discutindo os diversos componentes de um sistema de rede definido por software e apontando soluções para virtualização, sistemas operacionais de rede e aplicações recentes. Para exemplificar, utilizam o POX em experimentos práticos. Porém, vale considerar que o enfoque dos exercícios práticos não se baseiam na questão de segurança e apenas um dos diversos sistemas apontados é utilizado de forma prática. Logo, temos um trabalho de muito boa qualidade e com uma boa quantidade de informações que serão úteis para a elaboração deste novo trabalho.

Um dos trabalhos que mais chamaram a atenção foi: "Um Mecanismo de Autenticação e Controle de Acesso para Redes Definidas por Software"(MATTOS; DUARTE, 2014). O artigo traz um estudo realizado por um grupo da Universidade Federal do Rio de Janeiro (UFRJ), onde o mecanismo nomeado como "AuthFlow"possui características de fornecer autenticação da estação diretamente na camada de enlace em uma rede Open-Flow utilizando o MAC(Media Access Control). Isso, segundo os autores, se traduz em uma baixa sobrecarga e controle de acesso refinado, além de permitir utilizar a credencial de autenticação para realizar o controle de acesso de acordo com o nível de privilégio de cada estação. Neste trabalho também será discutido o quanto o controle de acesso é fundamental com relação a segurança, e ferramentas deste tipo são fundamentais.

Há também um outro estudo sobre a mesma temática do AuthFlow iniciado em um programa de pós graduação na Universidade Federal do Paraná (UFPR) por Adi Nascimento Marcondes(MARCONDES, 2014). Segundo o próprio autor, o mecanismo de Autenticação e Controle de Acesso tem como objetivo facilitar a descoberta de um nó (hospedeiro) malicioso presente na rede através da associação entre a tabela de fluxo presente no comutador e a autenticação quando um nó comutador(Switch) se conecta à rede.

O trabalho de um grupo de estudantes da PUC-MG e CEFET-MG(QUINTAO et al., 2015) utilizaram o mesmo conceito de bloqueio de fluxos não autorizados a partir do endereço MAC dos hosts. Porém, neste caso as simulações foram conduzidas através do Mininet-Wifi e foram analisados dados obtidos com relação ao bloqueio de fluxos e latência da rede, a fim de mostras se é possível se definir de forma eficaz e eficiente, funções de *firewall* através de controladores SDN. Mesmo que neste trabalho não foram realizados pesquisas em cima de redes virtuais wi-fi a ideia de controle por *firewall* e a forma como foi inserido diretamente por código através da chamada do módulo do controlador são interessantes e podem apontar uma solução.

O trabalho de estudantes da Escola Politécnica da Universidade de São Paulo(ROJAS; CARVALHO, 2013) utiliza uma abordagem distinta para mecanismos de controle de acesso como o RBAC, DAC, MAC e outros diversos como híbridos(MAC e DAC). Em suas pesquisas utilizaram Rede de Petri Colorida que é uma ferramenta de modelagem matemática que possibilitou a criação de um metamodelo para o sistema, validar este modelo através de testes utilizando os espaço de estados gerados do sistema modelado.

Parte IV

Resultados

4 Testes

Neste capítulo e nos outros subsequentes o intuito é apresentar todas as questões de segurança abordadas durante o trabalho. O fato principal seria que para poder garantir a segurança de uma rede utilizando uma ferramenta tão recente é imprescindível a realização de testes. A questão é que ferramentas recentes com alto grau de complexidade, isso devido a natureza complexa dos softwares com grande número de linhas de códigos, interoperabilidade com componentes externos e forte interação entre seus componentes internos, enfim, para todo caso é necessário ficar atento, pois erros ou falta de atenção a pequenos detalhes durante a implementação podem ocasionar falhas e possíveis quebras de segurança. Pode não parecer, mas este tipo de situação é muito comum e existem grupos, sites e comunidades em redes sociais que expõem estes tipos de vulnerabilidades como o caso dos "exploit's". O site Exploit Database(DATABASE, 2016) possui praticamente um banco de dados de exploit's apontados por usuários e la é possível visualizar a quantidade que existe tanto para diversas ferramentas quanto para uma mesma ferramenta. Basicamente é uma questão de tempo até que algo seja encontrado, isso independente de quão bem construído foi o código, pelo simples fato de que atualmente com o que temos disponível é impossível testar todas as questões de um sistema do tamanho, por exemplo, de um controlador que está sendo utilizado neste trabalho.

Ainda na questão de testes, devemos apontar que a função de apresentar erros e vulnerabilidades não é o único fator, neste trabalho logicamente será o de maior importância, porém podemos ainda destacar outras virtudes como a melhoria do produto em questão e neste sentido tornar a ferramenta mais segura e proteger os dados da rede. Um terceiro ponto seria entender, ajustar e documentar a postura operacional de uma organização como citado por Clarice Pereira em seu trabalho sobre segurança em nuvens(RIBEIRO, 2015).

Neste artigo é documentado uma análise de resultados obtidos por influência de fatores externos(tentativas de invasão e/ou derrubar a rede) e internos através de testes de laboratório em diferente controladores. Para tanto, iremos realizar a análise em diferentes etapas em que cada uma conterá um objetivo distinto, assim será mais fácil de se interpretar os resultados obtidos. Temos então uma divisão dos seguintes pontos discutidos neste e nos próximos capítulos: plataforma de trabalho; levantamento de informações; exploração das vulnerabilidades; apresentação de soluções.

4.1 Plataforma de trabalho

O trabalho foi desenvolvido em um laboratório com acesso a internet através de link dedicado como já foi exposto anteriormente, porém o acesso é regulado por um servidor e uma infra-estrutura de rede com switch e cabos de baixa capacidade de banda o que é um fator de interferência no trabalho. Os computadores utilizados como provedores do serviço de rede não possuem bom desempenho, mas para fim de testes é o suficiente, logicamente se ao criar a rede forem adicionados muitas funcionalidades para se executar dentro do controlador ou paralelamente a ele poderíamos ter problemas, assim como a adoção de um número muito grande de nós na rede o que poderia acarretar por exemplo no uso exacerbado de memória principal.

Foram utilizados ao todo três computadores com configurações idênticas, sendo um para cada controlador e em cada uma rede foi simulada, logo abaixo(Tabela 2) temos a descrição destes equipamentos utilizado para o teste laboratorial:

Tabela 2 – Tabela com descrição dos computadores utilizados

Máquina	CPU	Memória principal	Adaptador de rede
server1	Intel® Core™2 Duo E7200	1 GB	Realtek RTL8111
server2	Intel® Core™2 Duo E7200	1 GB	Realtek RTL8111
server3	Intel® Core™2 Duo E7200	1 GB	Realtek RTL8111

Fonte: Elaborada pelo autor

O sistema em que os testes foram executados é o linux Mint 17.3 Rosa, que é uma distribuição derivada do Ubuntu, porém mais leve e com boa estabilidade. Pode-se visualizar tais informações e alguns outros detalhes na Figura 6 - Sistema Operacional Utilizado abaixo retirada de uma das máquinas:

Figura 6 – Sistema Operacional Utilizado

Fonte: Elaborada pelo autor

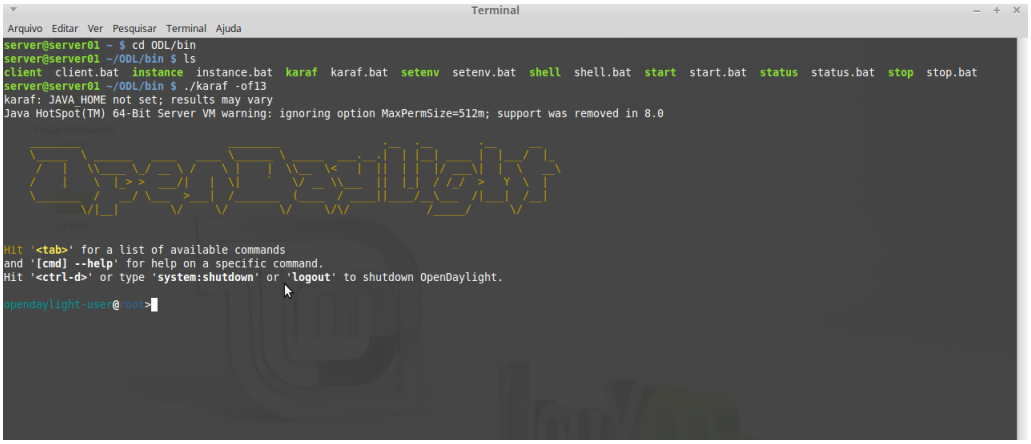
4.1.1 Honeypot com OpenDayLight

Como já mencionado anteriormente este é um dos controladores utilizados, a versão Beryllium está mais completa e relativamente fácil de utilizar se comparada à versões anteriores. Para iniciar a aplicação temos que realizar a instalação dos pacotes do Maven e OpenJDK para construir a aplicação, uma vez que ela é desenvolvida em Java. O próximo passo fica por conta de descompactar o pacote do controlador e acessar o diretório "\diretorioODL\bin", assim o comando exibido abaixo irá iniciar o controlador como representado pela Figura 7 - Execução do OpenDayLight:

```
# ./karaf -of13
```

Ao se iniciar temos uma interface simples como a representada abaixo:

Figura 7 – Execução do OpenDayLight

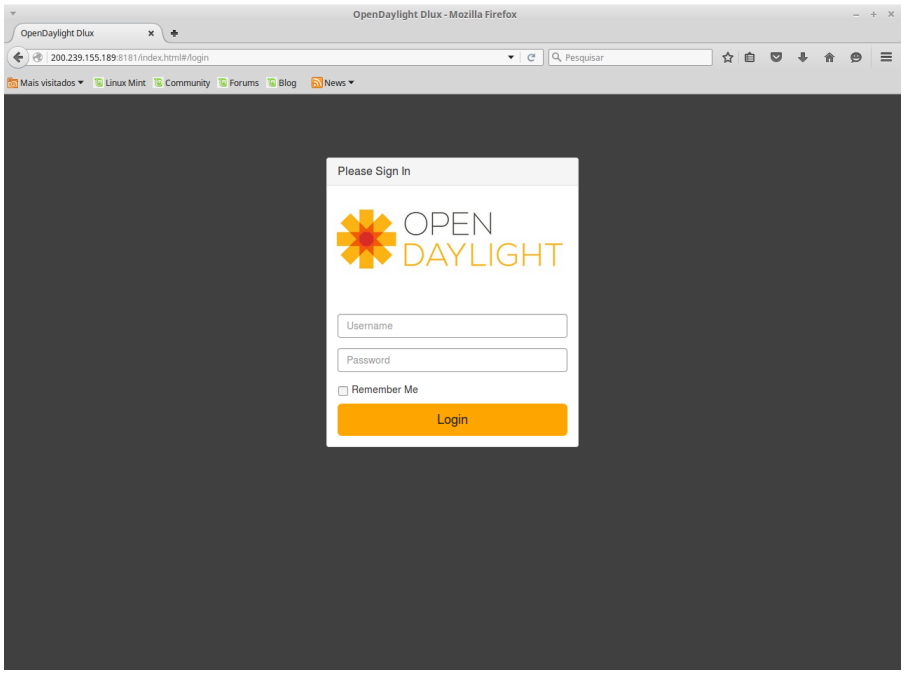


Fonte: Elaborada pelo autor

O sistema inicialmente não carrega nenhum módulo e não apenas inicializamos os módulos como fazemos com outros controladores, mas instalamos ele utilizando uma ferramenta interna. É possível instalar uma interface de usuário web para se trabalhar com o sistema, lembrando que para simular a rede foi utilizada a aplicação Mininet, assim obtemos os resultados ilustrados nas Figuras 8 - Sistema Web do OpenDayLight e 9 - Topologia da Rede do OpenDayLight a seguir. Para tanto, utilizamos então:

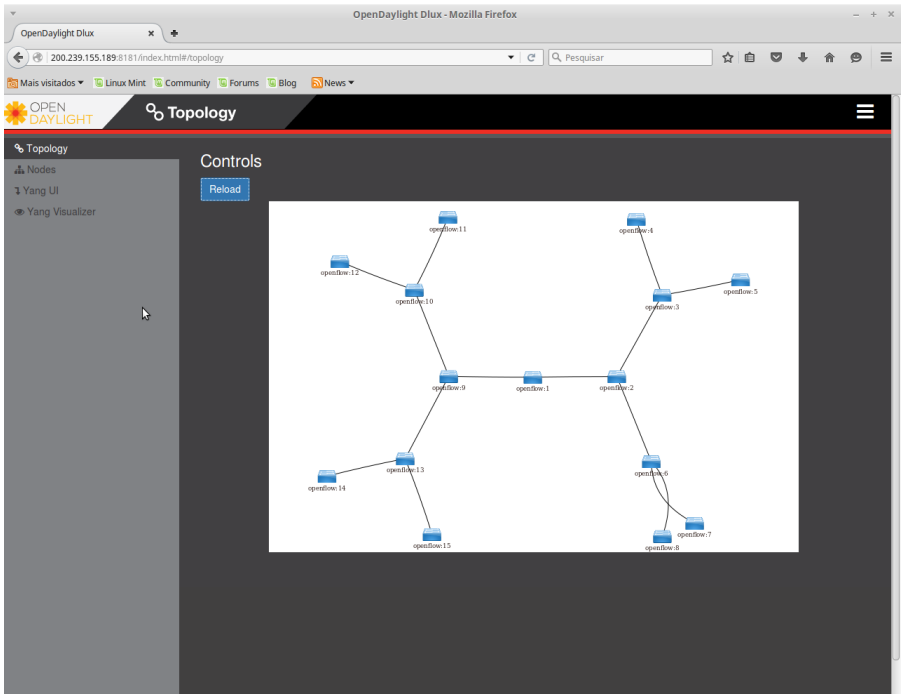
- feature:install odl-l2switch-switch-all para instalar os módulos assistentes para configuração da rede e interface.
- feature:install odl-dlux-all para instalar uma UI web.

Figura 8 – Sistema Web do OpenDayLight



Fonte: Elaborada pelo autor

Figura 9 – Topologia da Rede do OpenDayLight



Fonte: Elaborada pelo autor

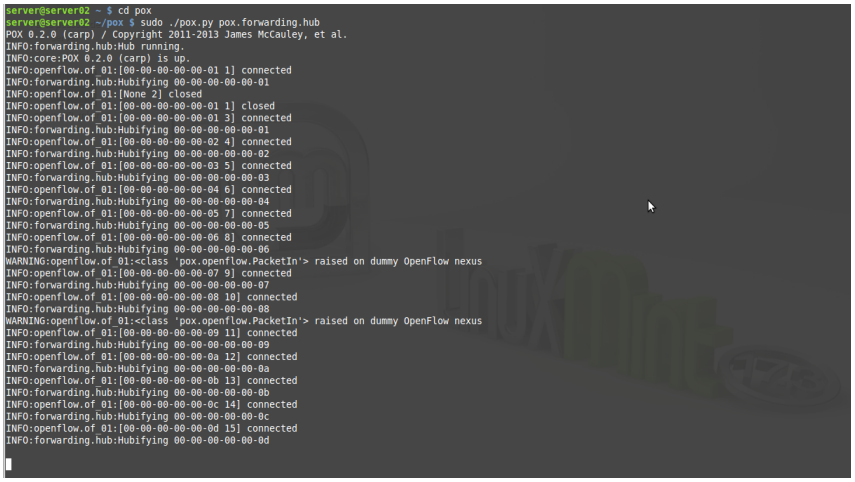
4.1.2 Honeypot com POX

O POX como mencionado anteriormente é voltado para pesquisa, para algumas versões mais antigas foram desenvolvidas pela comunidade interfaces de usuário assim como a do ODL exibida anteriormente, porém a versão mais recente que é utilizada para teste neste trabalho até a presente data não recebeu algo do tipo. Mesmo assim, a utilização é bem simples e o pacote após descompactado apenas precisa ser executado, isso sem um módulo especifico ou utilizando algum módulo disponível como demonstrado no comando abaixo e ilustrado na Figura 10 - Execução do POX.

```
# ./pox.py pox.forwarding.hub
```

Também foi utilizado a Mininet para estruturar uma rede virtual e módulos que são script's escritos em Python como este exibido anteriormente. Os pontos positivos do POX é a sua facilidade para utilização e configuração dos módulos, além da sensível leveza se comparado ao ODL, por exemplo.

Figura 10 – Execução do POX



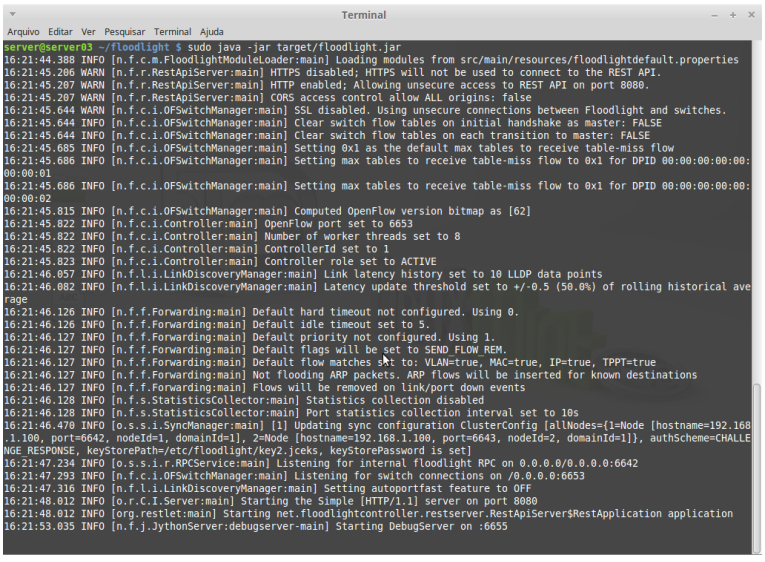
Fonte: Elaborada pelo autor

4.1.3 Honeypot com Floodlight

O floodlight como é uma ferramenta desenvolvida em java e precisamos construir o projeto temos que, assim como ocorre com a ODL, instalar o Java Development Kit(JDK) versão 7 ou superior. A Mininet também é utilizada para criar uma rede virtual e após extrair o pacote temos apenas que construir o projeto, ao iniciar temos algo como ilustrado na Figura 11 - Execução do FloodLight a seguir:

```
# sudo java -jar target/floodlight.jar
```

Figura 11 – Execução do FloodLight



Fonte: Elaborada pelo autor

Para construção de módulos é necessário uma abordagem um pouco mais complexa se comparado ao POX. É preciso instalar:

```
build-essential
default-jdk
ant
python-dev
eclipse
```

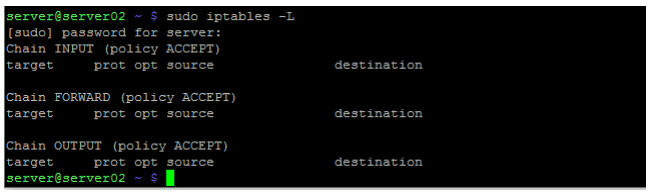
Após isso, usando o comando: *ant eclipse*. Selecionamos no eclipse ou outra ide o pacote do FloodLight podemos adicionar outras classes no pacote "net.floodlightcontroller.headerextract", estas serão os novos módulos e terão de instanciar as Interfaces "IOFMessageListener" e "IFloodlightModule", pois os métodos implementados nessas interfaces são necessários.

4.2 Levantamento de informações

4.2.1 IPTABLES

É uma aplicação executável em espaço de usuário que permite que o administrador do sistema, neste caso do sistema que suportará a honeypot, configure o firewall provido pelo Kernel do Linux através da alteração das regras presentes nas tabelas como pode-se visualizar na Figura 12 - Tabela Iptables em um dos servidores.

Figura 12 – Tabela Iptables em um dos servidores



Fonte: Elaborada pelo autor

Podemos verificar que o sistema utilizado não possui regras rígidas para controle do tráfego, e permitem basicamente que muitas coisas passem. Isso difere para cada sistema, o OpenSuse que também foi brevemente testado apresentou melhores resultados neste ponto, devido ao seu firewall mais robusto controlado diretamente pelo Yast que é uma ferramenta presente no sistema.

4.2.2 NETSTAT

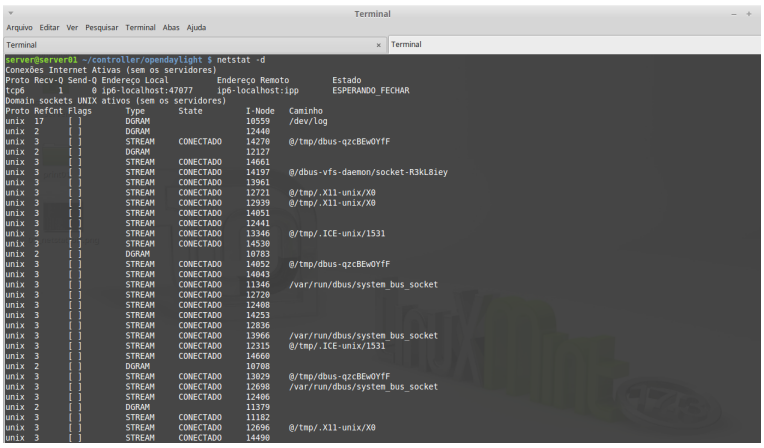
Com o comando "netstat"poderemos descobrir informações sobre tabelas de roteamento, conexões de rede, estatísticas das interfaces de rede, conexões mascaradas. Segundo o próprio desenvolvedor(LYON, 2009) há alguns comando da ferramenta para retirar informações distintas, logo abaixo se encontram algumas interessantes para o trabalho:

- # netstat -a: Mostra todos sockets, incluindo de servidores.
- # netstat -d: Mostra o estado de todas interfaces que usam DHCP.
- # netstat -s: Mostra estatísticas da rede.
- # netstat -M: Mostra todas as sessões mascaradas.

4.2.2.1 Tabelas de roteamento com OpenDayLight

Nesta sessão iremos destacar um fator importante em relação a segurança, podemos observar na Figura 13 - Tabelas antes de iniciar controlador o estado normal da máquina, ou seja, sem ter algum controlador sendo executado ou a Mininet. Entretanto, na Figura 14 - Tabelas após iniciar controlador a seguir, vemos um considerável numero de conexões TCP que surgiram ao se ligar o controlador juntamente com nossa topologia simulada. Este é um fator alarmante pois portas ficam abertas e suscetíveis a ataques.

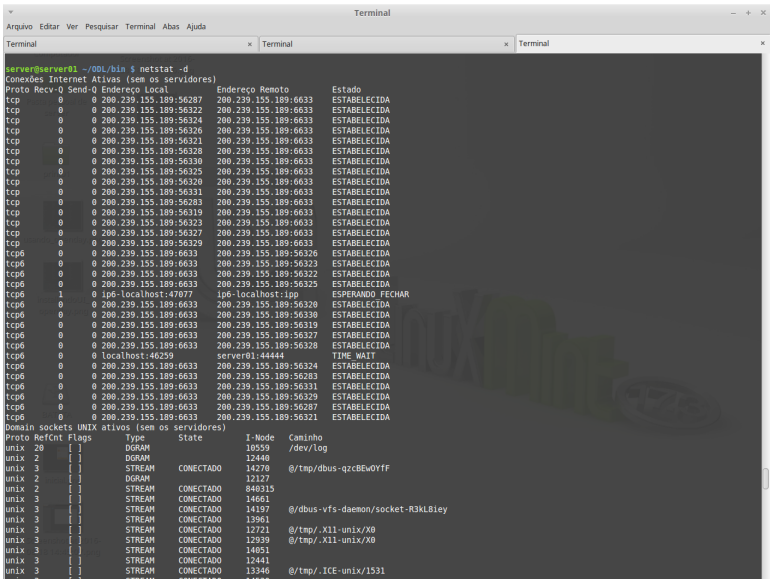
Figura 13 – Tabelas antes de iniciar controlador



Fonte: Elaborada pelo autor

Percebe-se que múltiplas conexões foram estabelecidas e muitos tipos de ataques utilizam tal situação como oportunidade para infiltrar.

Figura 14 – Tabelas após iniciar controlador

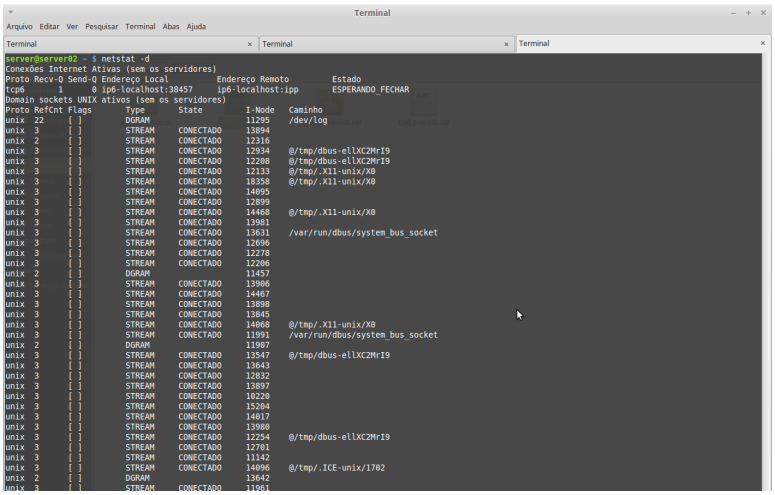


Fonte: Elaborada pelo autor

4.2.2.2 Tabelas de roteamento com Pox

Logo mais teremos um log das tabelas de roteamento, assim como feito anteriormente com o primeiro servidor, porém neste serão exibido os estados da tabela com o controlador Pox sendo executado.

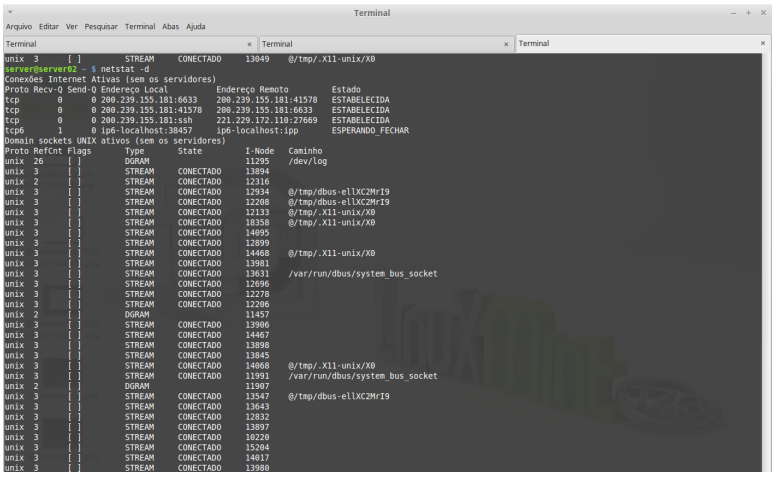
Figura 15 – Tabelas antes de iniciar controlador Pox



Fonte: Elaborada pelo autor

A Figura 15 - Tabelas antes de iniciar controlador Pox registra o estado inicial da máquina com este *controller* , é um caso semelhante ao visto anteriormente na máquina que suporta o controlador ODL e ao iniciar o POX, novamente portas são abertas como era de se esperar, isto pode ser visto na Figura 16 - Tabelas após iniciar controlador Pox.

Figura 16 – Tabelas após iniciar controlador Pox

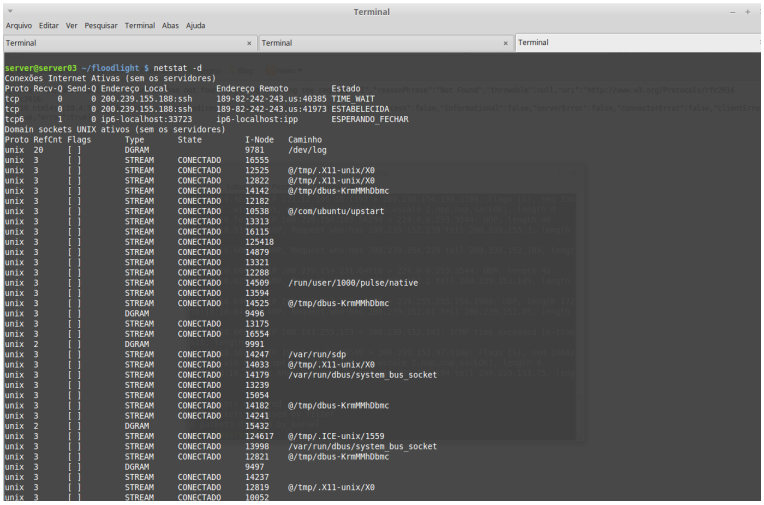


Fonte: Elaborada pelo autor

4.2.2.3 Tabelas de roteamento com FloodLight

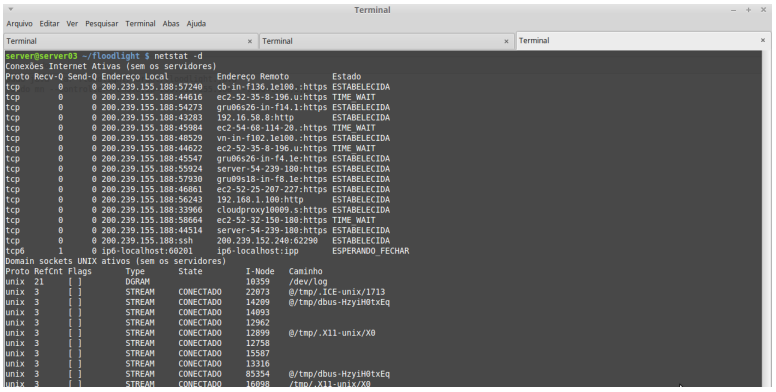
Assim como nos casos anteriores, nesta sessão estará disponível os dados dos estados da tabela de roteamento do terceiro servidor que como podemos visualizar na Figura 15 - Tabelas antes de iniciar controlador FloodLight possui um estado inicial semelhante aos anteriores. Ao se executar o controlador Floodlight, percebe-se a mesma característica com relação a portas sempre abertas visualizada nos outros controladores, a Figura 18 - Tabelas após iniciar controlador FloodLight nos da uma amostra obtida pelo netstat após a execução a fim de comparação com os *controllers* anteriores.

Figura 17 – Tabelas antes de iniciar controlador FloodLight



Fonte: Elaborada pelo autor

Figura 18 – Tabelas após iniciar controlador FloodLight



Fonte: Elaborada pelo autor

4.2.3 NMAP

Temos uma ferramenta utilizada em testes de segurança em redes e servidores, o Nmap é um scanner de portas muito eficiente que nos permite descobrir em uma rede, por exemplo, quais hosts estão disponíveis, quais firewalls estão sendo utilizados, portas abertas e quais serviços elas proveem. Ele possui papel primordial neste trabalho tanto para monitoramento das portar quanto para futuras soluções que serão apresentadas adiante.

```
# nmap localhost
```

Basta este simples comando descrito acima para visualizar todas as portas abertas no host local.

4.2.4 TCPDump

O TCPDump é conhecido como um "sniffer"assim como o WireShark e desenvolvido em C/C++, porém ele é mais simplista com relação a usabilidade e interface por se tratar de um pacote de análise em linha de comando. Neste trabalho, foi adotado por ser uma excelente ferramente para a proposta de uma honeypot, e a partir de seu uso podemos fazer com que a placa de rede entre em modo promíscuo(capture todos os pacotes da rede). Os dados são salvos em um arquivo ".cap"que posteriormente podem ser traduzidos em arquivos de texto e analisados utilizando um editor, o próprio TCPDump ou o que pode ser a melhor opção utilizando o Wireshark, pois este possui uma interface com filtros e indicadores com cores que facilitam a análise.

5 Exploração de vulnerabilidades

Neste capítulo serão apresentadas as considerações relacionadas às vulnerabilidades observadas durante o desenvolvimento do trabalho. Temos então uma avaliação de atividades externas registrada pelas Honeypot's, além de testes laboratoriais executados com ferramentas como o Iperf3 e o T50.

5.1 Ataques externos

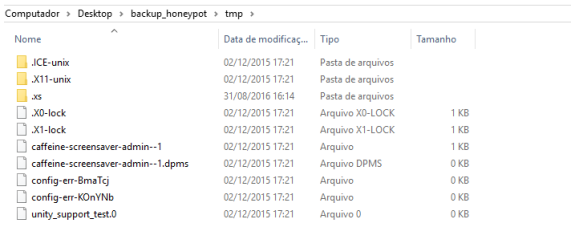
Nesta etapa o foco é analisar tudo o que foi obtido com o resultado das honeypot's. O objetivo inicial deste trabalho é testar todo tipo de influência com intensão maliciosa externa que uma rede utilizando controladores OpenFlow pode sofrer, por isso o ambiente representa uma honeypot simples onde temos simulações de redes utilizando os já citados controladores juntamente com a Mininet e para captura de dados utilizamos as ferramentas citadas anteriormente como o TCPDump, nmap, entre outros. Como em testes deste tipo é necessário algum tempo, uma vez que depende de influência externa, o serviço foi disponibilizado para ataques durante parte de quase dois semestres letivos o que se traduz em praticamente 6 meses, porém é preciso levar em conta que devido a quedas de força e outros problemas frequentemente havia indisponibilidade.

5.1.1 Ataque por SSH

Poucos foram os ataques obtidos, cerca de três do mesmo gênero para ser mais preciso o que remete a grande possibilidade de se tratar da mesma aplicação, em um dos registrados pode-se obter alguns detalhes específicos como a utilização da porta aberta pelo serviço de acesso remoto para ter acesso ao computador. A porta comumente utilizada para este serviço é a "porta 22", durante os testes pode-se observar que a grande maioria dos programas maliciosos trabalham de maneira bem simples, pois são feitos para ataques em máquinas com diversos propostas, sejam servidores de grandes empresas ou computadores pessoais. Neste caso em especifico a aplicação utiliza a porta aberta deixada pelo serviço SSH e utiliza um ataque de força bruto testando combinações de login e senha básicos, que normalmente são utilizados em máquinas com nenhuma política de segurança ou uma muito fraca como podemos visualizar na Figura 20 - Arquivo de registro com Login e Senha. A questão é que este tipo de programa malicioso normalmente é utilizado para instalar serviços na máquina e muitas das vezes se destinam a criar sistemas para ataques conhecidos como BotNets, na Figura 19 - Diretório de Backup com arquivos temporários temos o diretório de arquivos temporários com uma pasta ".xs"onde se encontram os

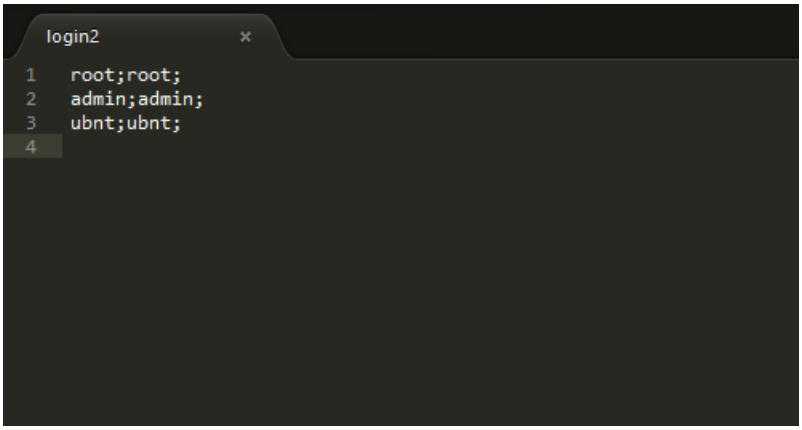
registros deixados pela aplicação maliciosa, a partir deles podemos ter uma noção de como ele executa e qual seu objetivo.

Figura 19 – Diretório de Backup com arquivos temporários



Fonte: Elaborada pelo autor

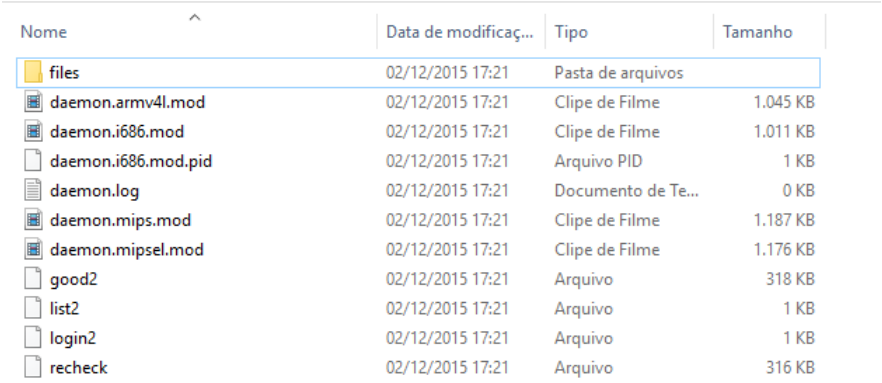
Figura 20 – Arquivo de registro com Login e Senha



Fonte: Elaborada pelo autor

Podemos visualizar nas imagens anteriores que este é um Script bem simples que testa algumas poucas combinações de login e senha, porém podem haver casos onde o invasor queira de fato encontrar o login e senha válidos e para tal utiliza um código mais complexo que tenta criar combinações de forma aleatória e realiza diversas tentativas de acesso. Logo adiante temos uma terceira imagem(Figura 21 - Diretório com registro temporário da aplicação maliciosa) referente a este diretório onde podemos visualizar os resquícios deixados pela aplicação, encontramos também uma lista com o registro de máquinas com o serviço disponível(Figura 22 - Lista good2) e uma lista de recheck no mesmo formato, onde estavam disponíveis o endereço atacado, o login e senha utilizado, a porta de acesso e demais informações do computador. Uma outra lista simples salva no arquivo "list2"onde continham apenas o endereço IP da máquinas e a porta foi salvo aquelas que puderam ser de fato acessadas e provavelmente tiveram o mesmo programa malicioso rodando.

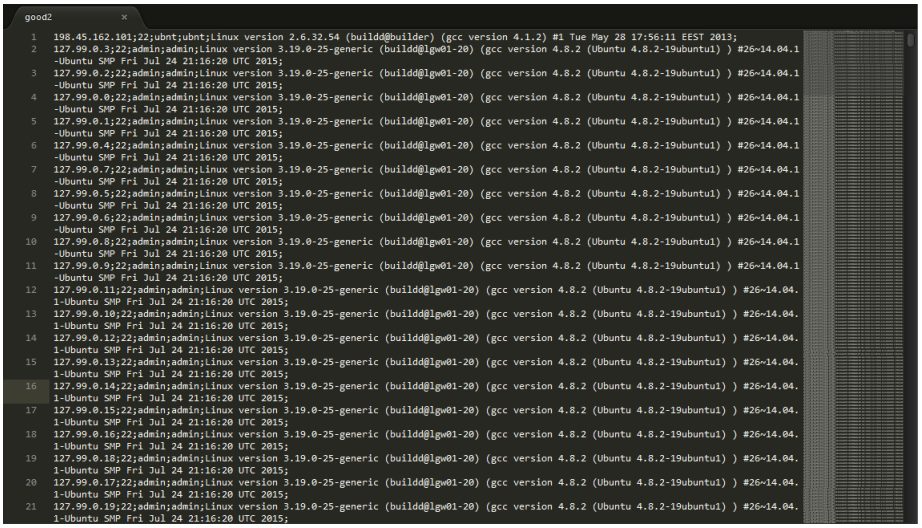
Figura 21 – Diretório com registro temporário da aplicação maliciosa



Nome	Data de modificaç...	Tipo	Tamanho
files	02/12/2015 17:21	Pasta de arquivos	
daemon.armv4l.mod	02/12/2015 17:21	Clipe de Filme	1.045 KB
daemon.i686.mod	02/12/2015 17:21	Clipe de Filme	1.011 KB
daemon.i686.mod.pid	02/12/2015 17:21	Arquivo PID	1 KB
daemon.log	02/12/2015 17:21	Documento de Te...	0 KB
daemon.mips.mod	02/12/2015 17:21	Clipe de Filme	1.187 KB
daemon.mipsel.mod	02/12/2015 17:21	Clipe de Filme	1.176 KB
good2	02/12/2015 17:21	Arquivo	318 KB
list2	02/12/2015 17:21	Arquivo	1 KB
login2	02/12/2015 17:21	Arquivo	1 KB
recheck	02/12/2015 17:21	Arquivo	316 KB

Fonte: Elaborada pelo autor

Figura 22 – Lista good2



```
good2
1  198.45.162.101;22;ubnt;ubnt;Linux version 2.6.32.54 (buildd@builder) (gcc version 4.1.2) #1 Tue May 28 17:56:11 EEST 2013;
2  127.99.0.3;22;admin;admin;Linux version 3.19.0-25-generic (buildd@lgw01-20) (gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1) ) #26~14.04.1
   -Ubuntu SMP Fri Jul 24 21:16:20 UTC 2015;
3  127.99.0.2;22;admin;admin;Linux version 3.19.0-25-generic (buildd@lgw01-20) (gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1) ) #26~14.04.1
   -Ubuntu SMP Fri Jul 24 21:16:20 UTC 2015;
4  127.99.0.0;22;admin;admin;Linux version 3.19.0-25-generic (buildd@lgw01-20) (gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1) ) #26~14.04.1
   -Ubuntu SMP Fri Jul 24 21:16:20 UTC 2015;
5  127.99.0.1;22;admin;admin;Linux version 3.19.0-25-generic (buildd@lgw01-20) (gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1) ) #26~14.04.1
   -Ubuntu SMP Fri Jul 24 21:16:20 UTC 2015;
6  127.99.0.4;22;admin;admin;Linux version 3.19.0-25-generic (buildd@lgw01-20) (gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1) ) #26~14.04.1
   -Ubuntu SMP Fri Jul 24 21:16:20 UTC 2015;
7  127.99.0.7;22;admin;admin;Linux version 3.19.0-25-generic (buildd@lgw01-20) (gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1) ) #26~14.04.1
   -Ubuntu SMP Fri Jul 24 21:16:20 UTC 2015;
8  127.99.0.5;22;admin;admin;Linux version 3.19.0-25-generic (buildd@lgw01-20) (gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1) ) #26~14.04.1
   -Ubuntu SMP Fri Jul 24 21:16:20 UTC 2015;
9  127.99.0.6;22;admin;admin;Linux version 3.19.0-25-generic (buildd@lgw01-20) (gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1) ) #26~14.04.1
   -Ubuntu SMP Fri Jul 24 21:16:20 UTC 2015;
10 127.99.0.8;22;admin;admin;Linux version 3.19.0-25-generic (buildd@lgw01-20) (gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1) ) #26~14.04.1
   -Ubuntu SMP Fri Jul 24 21:16:20 UTC 2015;
11 127.99.0.9;22;admin;admin;Linux version 3.19.0-25-generic (buildd@lgw01-20) (gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1) ) #26~14.04.1
   -Ubuntu SMP Fri Jul 24 21:16:20 UTC 2015;
12 127.99.0.11;22;admin;admin;Linux version 3.19.0-25-generic (buildd@lgw01-20) (gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1) ) #26~14.04.
   1-Ubuntu SMP Fri Jul 24 21:16:20 UTC 2015;
13 127.99.0.10;22;admin;admin;Linux version 3.19.0-25-generic (buildd@lgw01-20) (gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1) ) #26~14.04.
   1-Ubuntu SMP Fri Jul 24 21:16:20 UTC 2015;
14 127.99.0.12;22;admin;admin;Linux version 3.19.0-25-generic (buildd@lgw01-20) (gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1) ) #26~14.04.
   1-Ubuntu SMP Fri Jul 24 21:16:20 UTC 2015;
15 127.99.0.13;22;admin;admin;Linux version 3.19.0-25-generic (buildd@lgw01-20) (gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1) ) #26~14.04.
   1-Ubuntu SMP Fri Jul 24 21:16:20 UTC 2015;
16 127.99.0.14;22;admin;admin;Linux version 3.19.0-25-generic (buildd@lgw01-20) (gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1) ) #26~14.04.
   1-Ubuntu SMP Fri Jul 24 21:16:20 UTC 2015;
17 127.99.0.15;22;admin;admin;Linux version 3.19.0-25-generic (buildd@lgw01-20) (gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1) ) #26~14.04.
   1-Ubuntu SMP Fri Jul 24 21:16:20 UTC 2015;
18 127.99.0.16;22;admin;admin;Linux version 3.19.0-25-generic (buildd@lgw01-20) (gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1) ) #26~14.04.
   1-Ubuntu SMP Fri Jul 24 21:16:20 UTC 2015;
19 127.99.0.18;22;admin;admin;Linux version 3.19.0-25-generic (buildd@lgw01-20) (gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1) ) #26~14.04.
   1-Ubuntu SMP Fri Jul 24 21:16:20 UTC 2015;
20 127.99.0.17;22;admin;admin;Linux version 3.19.0-25-generic (buildd@lgw01-20) (gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1) ) #26~14.04.
   1-Ubuntu SMP Fri Jul 24 21:16:20 UTC 2015;
21 127.99.0.19;22;admin;admin;Linux version 3.19.0-25-generic (buildd@lgw01-20) (gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1) ) #26~14.04.
   1-Ubuntu SMP Fri Jul 24 21:16:20 UTC 2015;
```

Fonte: Elaborada pelo autor

A partir do que foi visto anteriormente, fica claro o quão importante é utilizar conceitos de segurança, o controlador SDN proporciona um controle via software de todas as máquinas da rede, e logicamente para isso o computador que comporta o controlador está frequentemente se comunicando com as demais máquinas na rede, caso o sistema não esteja protegido todas as máquinas conectadas na rede podem se tornar vitimas, isso considerando Script's simples como este visto. Senhas simples e portas abertas para outras atividades tem de ser monitoradas pois podem tornar todo o sistema vulnerável e afetar o serviço fornecido pelo controlador SDN e isso não é especificamente para acesso remoto, mas também para servidor http, sql, ftp ou qualquer outro serviço que não esteja incluso

em um controle por firewall, utilize portas padrões e permita utilização de login e senha simples.

5.2 Simulações

5.2.1 Utilizando IPerf

O iPerf3 é a uma ferramenta nova que segue a linha do iperf e iperf2, porém não compartilha nenhum código com o iperf original e também não é compatível com versões anteriores. O iperf foi originalmente desenvolvido pela NLANR/DAST, já o iPerf3 é desenvolvido principalmente pela ESnet/Lawrence Berkeley National Laboratory contando com diversos contribuintes, segundo a definição dos próprios desenvolvedores (DUGAN SETH ELLIOTT, 2016) é uma ferramenta para realizar medições ativas e a largura de banda máxima alcançável em uma determinada rede. Suporta ajuste de vários parâmetros relacionados ao timing, buffers e protocolos(TCP, UDP, SCTP com IPv4 e IPv6). Para cada teste, ele informa a largura de banda, perda, throughput, dentre outros parâmetros, assim podemos destacar dentre suas características:

- Para TCP:

  Suporta TCP e SCTP

  Obtêm largura de banda

  Reporta tamanho MSS/MTU e observa os números de leitura.

  Possui suporte para o tamanho da janela TCP via buffers de soquete.

- Para UDP:

  Cliente pode criar transmissões UDP para um bandwidth específico.

  Mede número de pacotes perdidos

  Mede a taxa de delay jitter

  Suporta Multicast

- Suporte á multiplos sistemas(Windows, Linux, Android, MacOS X,...).
- O cliente pode realizar múltiplas conexões simultâneas.
- Pode funcionar por tempo especificado(opção -t), ou com uma quantidade de dados pré-definida para transferir(-n ou opção-k).

Como comentado anteriormente na Tabela 2 as máquinas são idênticas, a rede externa é exatamente assim como a versão do sistema, Iperf3 e demais aplicações. A

única diferença foi a utilização de controladores distintos e assim utilizando esta aplicação para todas as máquinas como servidores do serviço e a partir de um outro computador realizar trafego como cliente utilizando também o Iperf3, temos então que mesmo com diversas oscilações em testes realizados com pacotes de 40.0 MBytes de dados e executados repetidamente por 10 vezes, uma vez que a partir deste número percebeu-se que quase não existia modificação no valor das médias, podemos tirar os resultados da Tabela 3 a seguir:

Tabela 3 – Tabela com teste de banda utilizando Iperf3

Máquina	Controlador	Average Bandwidth	Serviço lentidão/caiu?
server1	OpenDaylight	29.5 Mbits/sec	Não
server2	POX	33.6 Mbits/sec	Não
server3	Floodlight	34.8 Mbits/sec	Não

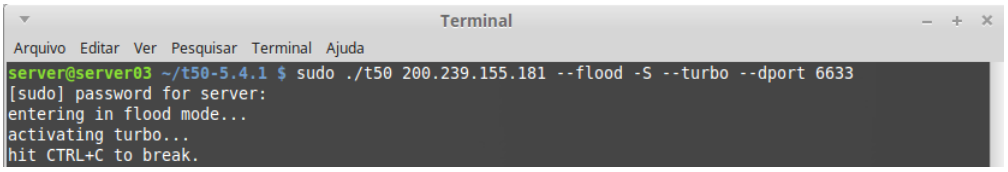
Fonte: Elaborada pelo autor

5.2.2 Utilizando o T50

O T50 Sukhoi PAK FA Mixed Packet Injector é considerada por muitos como uma excelente ferramenta de packet injection utilizada para realizar testes de stress devido ao seu bom desempenho, principalmente com relação a ataques de força bruta como DoS/DDoS. Desenvolvido pelo brasileiro Nelson Brito, consegue injetar mais de 1.000.000 pacotes por segundo em uma rede Gigabit Ethernet(IEEE 802.3ab). Mesmo não sendo propósito para o qual foi criada, algumas pessoas com intenções maliciosas podem utilizar a ferramenta para ataques DoS(Ataque de negação de serviço) que se caracteriza não como um ataque onde a intenção é invadir ou infectar o sistema alvo, mas utiliza-se de um grande número de requisições ao alvo, até que a vítima não consiga atender a mais nenhuma requisição e comece a descartar pacotes o que resulta em lentidão no serviço e em alguns casos queda do serviço por inacessibilidade.

Botnets podem utilizar este tipo de ataque para gerar uma forma de ataque DoS distribuído conhecido como DDoS, neste caso temos diversas máquinas executando a mesma ação de requisições a um mesmo alvo. Muitos malwares passam a incorporar ferramentas como o t50 que podem ser utilizados para tal e computadores domésticos que geralmente possuem um sistema de segurança mais brando são os alvos deste tipo de aplicação maliciosa que infectam o computador e instalam o serviço para ser posteriormente utilizado em ataques.

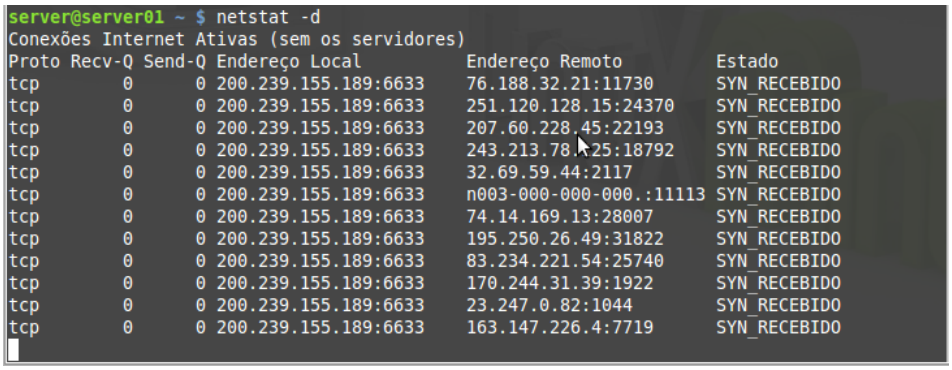
Figura 23 – Exemplo de execução do T50



Fonte: Elaborada pelo autor

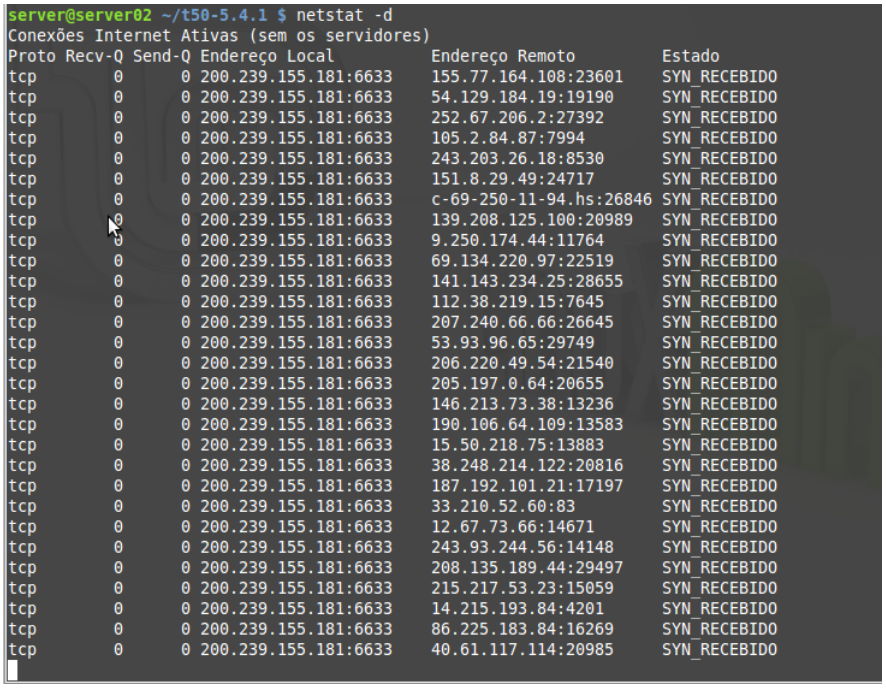
Neste trabalho o programa foi utilizado em simples simulações de ataque DoS como ilustrado na Figura 23 - Exemplo de execução do T50 a fim de verificar o comportamento dos controladores com relação a este tipo de ataque. A simulação foi realizada cerca de 5 vezes para cada caso e em ambos os casos o resultado foi semelhante e pode-se observar que em toda rede local houve um pouco de lentidão a ponto de que a máquina alvo e qualquer outra na rede tivessem dificuldade de abrir páginas de internet, porém em nenhum dos três servidores o serviço caiu ou teve-se problemas com relação a acesso ao controlador. Logicamente este é um caso em um ambiente controlado e com apenas uma máquina executando o ataque, em um sistema distribuído de ataque o resultado pode acusar mais que apenas uma lentidão. Como teste foram realizados ataques a portas que ofereciam serviços específicos como o Dlux do OpenDayLight na porta "8181"e o http-proxy na porta "8080"utilizada tanto pelo ODL quanto pelo FloodLight, entretanto atacando ambos os controladores e portas não foi verificado inatividade.

Figura 24 – Registro de SYN flooding no ODL



Fonte: Elaborada pelo autor

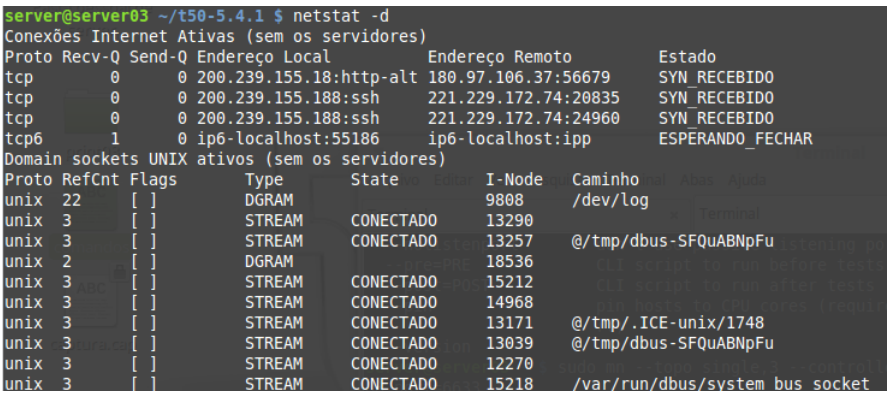
Figura 25 – Registro de SYN flooding no POX



```
server@server02 ~/t50-5.4.1 $ netstat -d
Conexões Internet Ativas (sem os servidores)
Proto Recv-Q Send-Q Endereço Local           Endereço Remoto         Estado
tcp        0      0 200.239.155.181:6633    155.77.164.108:23601    SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    54.129.184.19:19190     SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    252.67.206.2:27392      SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    105.2.84.87:7994        SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    243.203.26.18:8530      SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    151.8.29.49:24717       SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    c-69-250-11-94.hs:26846 SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    139.208.125.100:20989   SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    9.250.174.44:11764      SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    69.134.220.97:22519     SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    141.143.234.25:28655    SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    112.38.219.15:7645      SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    207.240.66.66:26645     SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    53.93.96.65:29749       SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    206.220.49.54:21540     SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    205.197.0.64:20655      SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    146.213.73.38:13236     SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    190.106.64.109:13583    SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    15.50.218.75:13883      SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    38.248.214.122:20816    SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    187.192.101.21:17197    SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    33.210.52.60:83         SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    12.67.73.66:14671       SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    243.93.244.56:14148     SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    208.135.189.44:29497    SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    215.217.53.23:15059     SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    14.215.193.84:4201      SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    86.225.183.84:16269     SYN_RECEBIDO
tcp        0      0 200.239.155.181:6633    40.61.117.114:20985     SYN_RECEBIDO
```

Fonte: Elaborada pelo autor

Figura 26 – Registro de SYN flooding no FloodLight



```
server@server03 ~/t50-5.4.1 $ netstat -d
Conexões Internet Ativas (sem os servidores)
Proto Recv-Q Send-Q Endereço Local          Endereço Remoto         Estado
tcp        0      0 200.239.155.18:http-alt 180.97.106.37:56679     SYN_RECEBIDO
tcp        0      0 200.239.155.188:ssh     221.229.172.74:20835    SYN_RECEBIDO
tcp        0      0 200.239.155.188:ssh     221.229.172.74:24960    SYN_RECEBIDO
tcp6       1      0 ip6-localhost:55186     ip6-localhost:ipp       ESPERANDO_FECHAR
Domain sockets UNIX ativos (sem os servidores)
Proto RefCnt Flags       Type       State         I-Node   Caminho
unix  22     [ ]         DGRAM                    9808     /dev/log
unix  3      [ ]         STREAM     CONECTADO     13290
unix  3      [ ]         STREAM     CONECTADO     13257    @/tmp/dbus-SFQuABNpFu
unix  2      [ ]         DGRAM                    18536
unix  3      [ ]         STREAM     CONECTADO     15212
unix  3      [ ]         STREAM     CONECTADO     14968
unix  3      [ ]         STREAM     CONECTADO     13171    @/tmp/.ICE-unix/1748
unix  3      [ ]         STREAM     CONECTADO     13039    @/tmp/dbus-SFQuABNpFu
unix  3      [ ]         STREAM     CONECTADO     12270
unix  3      [ ]         STREAM     CONECTADO     15218    /var/run/dbus/system_bus_socket
```

Fonte: Elaborada pelo autor

Como mencionado anteriormente lentidões ocorreram nos três casos, porém o controlador FloodLight se saiu melhor em um dos casos de teste, onde o ataque foi concentrado na porta utilizada pela Mininet para executar e ouvir. O ataque utilizado é bastante comum e pode ser realizado através do protocolo TCP/IP, o SYN flooding tenta estabelecer uma conexão com o alvo por meio de um sinal TCP conhecido como SYN que podemos visualizar nas Figuras 24 - Registro de SYN flooding no ODL, 25 - Registro de SYN flooding no POX e 26 - Registro de SYN flooding no FloodLight. De modo geral as honeypots rodando controladores distintos porém a mesma versão da aplicação de virtualização de rede(Mininet) obtiveram comportamentos distintos como, por exemplo, o

POX sofreu uma perceptiva lentidão com o registro de muitas confirmações de SYN, o ODL teve grande lentidão para confirmar cada SYN e ainda registrou uma quantidade considerável, entretanto o FloodLight parecia descartar a grande quantidade de conexões e confirmava apenas alguns sinais.

5.3 Problemas enfrentados

Alguns ajustes foram necessários para que o serviço se mantivesse ativo. Primeiramente a configuração da Honeypot, para aproveitar o máximo possível o tempo disponível, pois parte do trabalho depende de atividades externas e para contornar possíveis problemas como uma queda de energia, perda de dados, entre outros. Para a questão de quedas de energia, que foram muito comuns e tiveram forte influência na má disponibilidade dos serviços para potenciais ataques, foi utilizado o artifício de configuração das máquinas para religamento automático nas configurações da Bios e para tratar a perda de informações os serviços eram inicializados de forma com que os dados obtidos eram salvos em arquivos. O problema é que realizar backup destes dados em outra mídia, que não fosse a da própria honeypot poderia acarretar deficiência tanto no desempenho da máquina quanto na rede caso fosse necessário mover utilizando a conexão de rede, por isso os dados de captura do TCPDump e os arquivos temporários tiveram que ser salvos no próprio computador onde rodava o serviço. Em sistemas linux são comuns a utilização de Bash Script's para inicializar serviços e realizar operações, neste caso a inicialização dos controladores foi deixado para ser feito de forma manual, porém inicializar o registro de captura do TCPDump e outros serviços podem ser realizados utilizando este tipo de ação.

Os computadores utilizados serviram para executar testes mais brandos, porém como a rede utilizada é emulada houveram momentos que ocorreram certos "engasgos"e travamentos, principalmente quando ao se iniciar a Mininet com configurações não muito alarmantes relacionadas à topologia da árvore, profundidade e fanout. Viu-se então que ao se rodar o controlador juntamente com o mininet em determinados casos o serviço pode vir a cair devido a stress por falta de recursos. Grandes problemas surgiram para implementar os serviços e realizar os teste. Primeiramente devemos observar que temos 3 opções de controladores distintas e consequentemente a configuração e todo o processo é distinto.

O POX foi o controlador relativamente mais simples, talvez devido a sua atribuição ser principalmente de pesquisa, entretanto a criação de módulos e a manipulação ainda careceu de certa pesquisa. O OpenDayLight foi o mais complexo de se trabalhar, este requeria diversas ferramentas para sua execução que foi aprimorada na ultima versão a Beryllium-SR2 e portanto é a utilizada neste trabalho. Porém, em versões anteriores houveram diversas divergências referentes a uso e instalação o que custaram um certo tempo, fato que foi simplificado na versão final. O FloodLight também requer certas

ferramentas para sua execução, entretanto a sua utilização é simples e não ocorreram tantos contratempos como o que ocorreram com as versões do ODL utilizada anteriormente a versão Beryllium-SR2.

6 Soluções encontradas

Uma das propostas deste trabalho é apontar soluções para problemas de segurança e vulnerabilidade quando possível. Como mencionado anteriormente nas mais diversas etapas deste trabalho, tão importante quando identificar problemas e sugerir ferramentas ou ações que possam suprimir ou elimina-los. Hoje em dia com a popularização da tecnologia de redes definidas por software e dos protocolos OpenFlow já é possível de se encontrar alguns trabalhos que buscam testar e/ou resolver problemas recorrentes da plataforma. No capítulo III temos diversos exemplos de pesquisas onde empenham-se em encontrar soluções para alguns casos, principalmente para o problema de controle de acesso que é um dos mais sucessivos nesta área. Neste capítulo o foco é apresentar algumas das possíveis soluções que podem ser adotadas para enrijecer o sistema de segurança ao se adotar uma SDN.

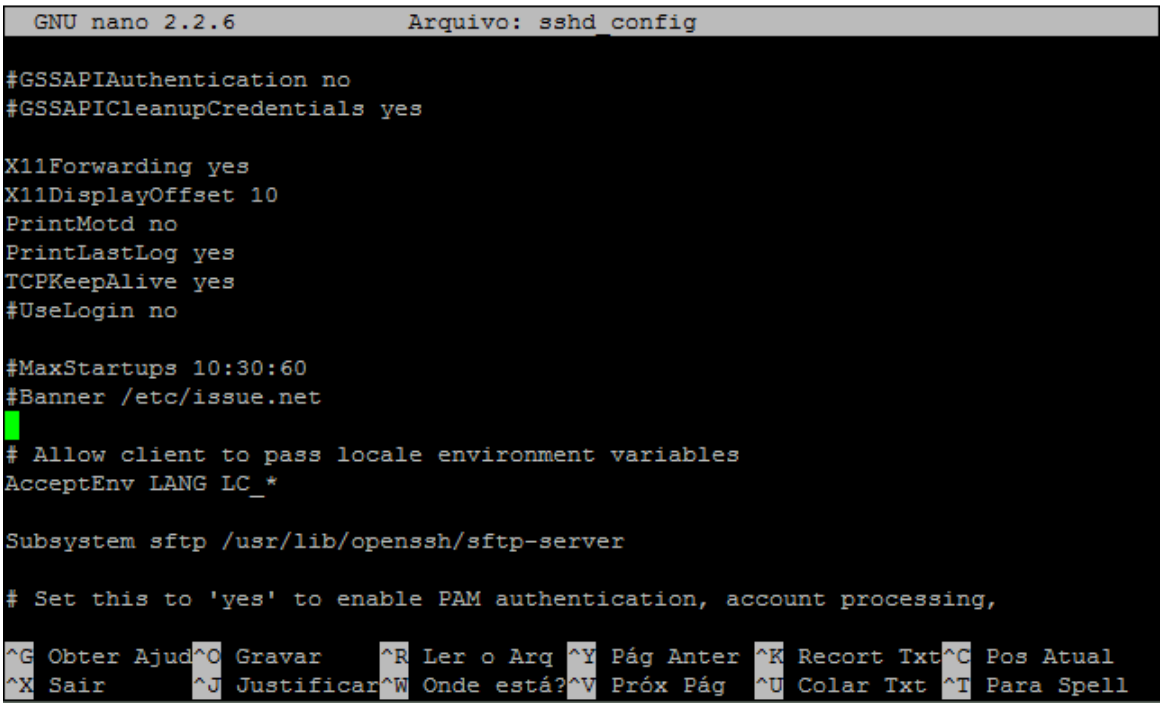
6.1 Alterar porta padrão e configurações padrões

A primeira vista parece algo muito simplista e realmente não é nada complexo, visto que muitas ferramentas por padrão permitem alterar a porta de serviço. Normalmente aplicações maliciosas mais simples buscam explorar portas padrões de serviço, assim como temos 22 para SSH, 80 para HTTP e para os controladores, por exemplo, temos a 8181 utilizada pelo API do ODL. Como citado anteriormente script's simples executam ataques utilizando dados pré-definidos que são padrões para aquela ferramenta, como logins e portas padrões. Alterar este tipo de dado é fundamental para eliminar qualquer potencial ataque feito por programas simples. Podemos ainda utilizar algumas configurações próprias das ferramentas, caso as atividades maliciosas incluam tentativa por força bruta, é comum que ocorram erros de tentativas e utilizar esta característica como um dos métodos é fundamental. O próprio SSH que serviu de porta de entrada para a invasão ocorrida, possui um arquivo de configuração(Figura 27 - Arquivo de configuração SSH) onde é possível configurar o parâmetro *MaxStartups 10:30:60* que por padrão utiliza três números configuráveis conforme a necessidade e definem:

- O número 10 corresponde a um número limite de tentativas sem sucesso. Após alcançar tal valor, o servidor SSH recusará as tentativas de conexão referentes ao endereço IP.
- No caso o segundo valor é a percentagem de conexões a serem recusadas ao se extrapolar o limite mínimo estipulado anteriormente.

- O último número representa o limite máximo de tentativas erradas que um IP pode chegar a fazer. Ao ser alcançando, o servidor recusará todas as conexões referentes ao endereço IP.

Figura 27 – Arquivo de configuração SSH



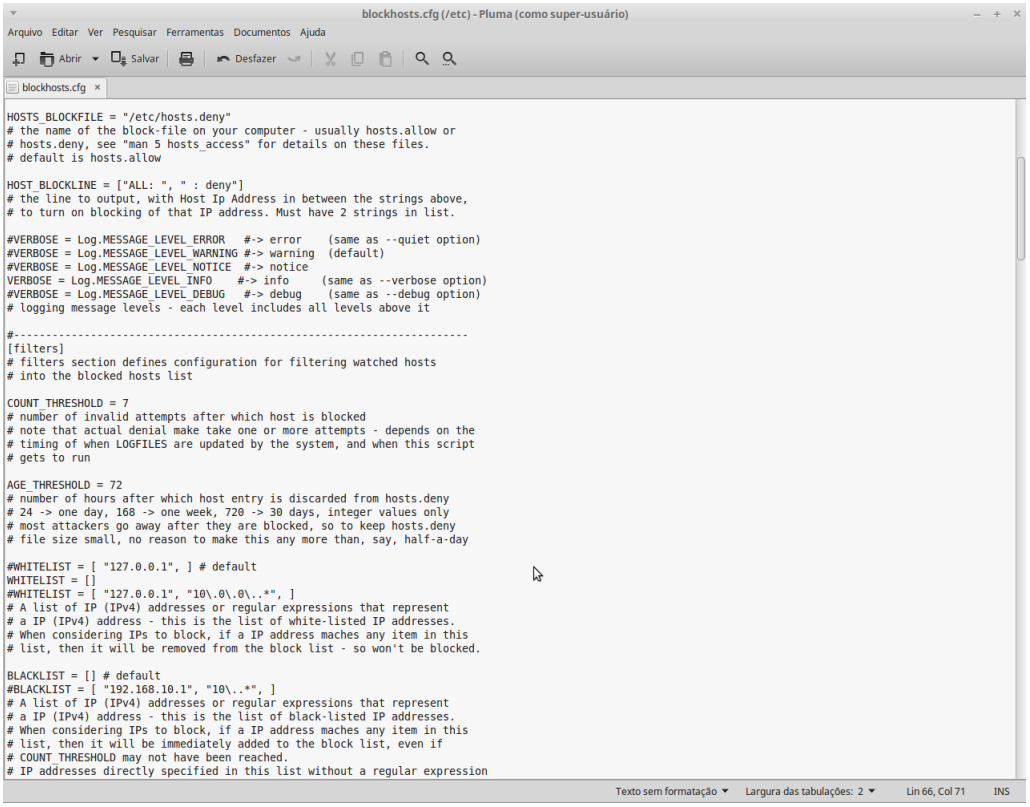
Fonte: Elaborada pelo autor

6.2 Block Hosts

O Block Hosts é uma ferramenta que se destina a bloquear ataques de força-bruta em FTP, SSH e outros serviços, sendo inclusive possível de ser utilizado para as portas abertas pelo controlador. Ela possui a função de monitorar os logs dos serviços e bloquear através do TCP_WRAPPERS casos em que existam muitas conexões inválidas de uma mesma origem. Um das grandes vantagens está na possibilidade de configurar quaisquer tipo de ação como o tempo que o suposto endereço do atacante deve ficar bloqueado, número máximo de erros de senha, configurar um "Whitelist"com endereços que nunca poderão ser bloqueados ou um "Blacklist"onde os endereços deverão ser sempre bloqueados, entre algumas outras. O TCP_WRAPPERS comentado anteriormente é um sistema ACL de rede baseado em host, usado para filtrar o acesso à rede para servidores de protocolo de internet. A configuração dessa aplicação é bem simples e após a instalação basta

acessar o arquivo de configuração onde por default todas as opções estão comentadas, ao se retirar o comentário(#) da linha a opção estará valida, logo a seguir na Figura 28 - Arquivo de configuração do Block Hosts temos um exemplo do arquivo de configuração usando configurações padrões. Percebe-se que há uma grande flexibilidade para criar e/ou modificar o sistema de segurança.

Figura 28 – Arquivo de configuração do Block Hosts



Fonte: Elaborada pelo autor

Podemos visualizar então que temos os seguintes parâmetros em principal para configuração, vale ressaltar alguns pontos:

- HOSTS_BLOCKFILE = "/etc/hosts.deny- Indica onde o arquivo com os endereços dos hosts permitidos ou não estão.
- HOST_BLOCKLINE = ["ALL: ", ": deny"] - Linha de bloqueio que será inserida no arquivo "/etc/hosts.deny".
- VERBOSE = Log.MESSAGE_LEVEL_INFO - Define o nível de log, ou seja, como será reportado.
- COUNT_THRESHOLD = 7 - É o número de vezes que um usuário pode errar a senha. Caso erre o número correspondente de vezes ele será bloqueado.

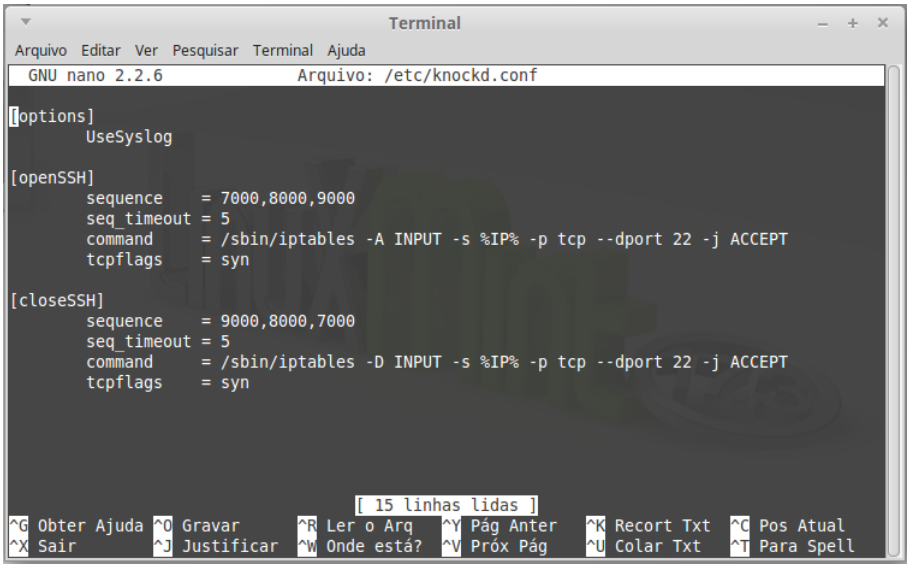
- AGE_THRESHOLD = 72 - Corresponde ao tempo em qu o IP de origem ficará bloqueado em horas, neste caso: 72 horas.
- WHITELIST = [ ] - Lista de IPs ou range de IPs que não devem ser bloqueados. Quando usamos uma rede interna real ou virtual(usando Mininet) é interessante listar aqui os componentes da rede.
- BLACKLIST = [ ] - O mesmo caso do item anterior, porém a idéia e de obter uma lista com IP ou range de IPs que sempre serão bloqueados.
- LOGFILES = ["/var/log/auth.log", "/var/log/proftpd/proftpd.log",] - Aqui serão definidos quais os logs que serão analisados, podemos então selecionar os arquivos de qualquer ferramenta de acesso remoto, HTTP, FTP...
- LOCKFILE = "/tmp/blockhosts.lock- Arquivo temporário de lock(trava).

Ainda vale ressaltar que é preciso configurar uma aplicação de agendamento e inicialização para executar periodicamente. O linux derivados do Debian como o Ubuntu, Mint, dentre outros possuem como padrão para este tipo de atividade o agendador de tarefas Cron, sendo assim basta adicionar uma linha no arquivo de *"/etc/crontab"* e verificar os logs gerados pelo BlockHosts em *"/var/log/blockhosts.log"*.

6.3 KNOCKD

Esta aplicação trouxe um conceito distinto de bloqueio, entretanto está disponível apenas para distribuições Linux. A idéia consiste em permitir a execução de um serviço somente após o usuário enviar uma sequência previamente conhecida de pacotes, que são configurados no arquivo *"knockd.conf"* ilustrado na Figura 29 - Arquivo de configuração do KNOCKD. Para isso é preciso que exista um servidor para emitir os pacotes e um cliente que conheça a combinação de pacotes a ser enviado, assim este será responsável por enviar os pacotes da forma como a aplicação knockd espera receber no servidor. Percebe-se que é como se fosse um método de autenticação por código e claramente o seu ponto fraco é que a rede não pode ser comprometida a fim de permitir que um usuário externo possa observar a rede e conseguir identificar o padrão. Podemos para criar a sequencia utilizar ainda pacotes TCP ou UDP em quaisquer portas que estejam livres, sendo prioridade evitar portas padrões que estejam sendo utilizadas por outras aplicações. Para finalizar podemos definir uma outra sequência para que o Knockd reconheça, normalmente é utilizada a sequência de entrada ao contrário.

Figura 29 – Arquivo de configuração do KNOCKD



Fonte: Elaborada pelo autor

6.4 Bash Script

Podemos utilizar meios de segurança que se utilizam da implementação de programas para adaptar a segurança, temos por exemplo a possibilidade em sistemas linux de se utilizar Bash scripts que realizam tarefas de monitorar e/ou dar drop em atividades suspeitas. Para isso pode-se criar o script da forma com a qual deseja e utilizar quaisquer ferramenta disponível na máquina. Entretanto, é importante considerar que esta é uma atividade bastante massiva e requer ser feita manualmente e pode ser muito complexa, pois além da elaboração do script será preciso adiciona-lo a rotina do sistema, dar as permissões necessárias para sua execução além de configurar as demais ferramentas que serão utilizadas e os arquivos de log. Um exemplo simples encontrado foi esta aplicação(BLEDSOE, 2011) criada por Greg Bledsoe que após inserido roda continuadamente utilizando *sleep* como temporizador, basicamente ele realiza um tratamento e executa drop utilizando a tabela do "iptables"em tentativas de conexões de um mesmo endereço que esteja atuando de forma excessiva. A ideia de Greg Bledsoe como ele mesmo sugere é utilizar a simplicidade, pois segundo seus próprios testes e experiência a melhor solução costuma ser a mais simples.

#! /bin/bash

while [1];

do

for ip in ‘lsof -ni | grep httpd | grep -iv listen | awk ‘print $8’ | cut -d : -f 2 | sort | uniq | sed s/”http->”//‘;

the line above gets the list of all connections and connection attempts, and

produces a list of uniq IPs

and iterates through the list

do

noconns=‘lsof -ni | grep $ip | wc -l‘;

This finds how many connections there are from this particular IP address

echo $ip : $noconns ;

if [“$noconns” -gt “10”] ;

if there are more than 10 connections established or connecting

from this IP

then

echo More;

echo ‘date‘ “*iphas*noconns connections. Total connections

to prod spider: ‘lsof -ni | grep httpd | grep -iv listen | wc -l‘” »

/var/log/Ddos/Ddos.log

to keep track of the IPs uncomment the above two lines and make sure you can write to the appropriate place

iptables -I INPUT -s $ip -p tcp -j REJECT –reject-with tcp-reset

for these connections, add an iptables statement to send resets on any packets recieved

else

echo Less;

fi;

done

sleep 60

done

6.5 Mecanismo de Autenticação(AuthFlow)

Este tipo de ferramenta é frequentemente abordada em trabalhos de pesquisa como uma excelente opção para este paradigma de redes. Em seu trabalho baseado em segurança utilizando AuthFlow(MATTOS; DUARTE, 2014), o grupo de Teleinformática e Automação da UFRJ define como: "um mecanismo de autenticação e controle de acesso de

estações finais baseado na credencial da estação ". E segundo os próprios autores oferece clara vantagem com relação à:

- Autentica a estação diretamente na camada de enlace em uma rede Open-Flow, o que introduz uma baixa sobrecarga de controle e assegura um controle de acesso refinado.
- Usa a credencial de autenticação para realizar o controle de acesso de acordo com o nível de privilegio de cada estação, através da associação da credencial ao conjunto de fluxos pertencentes à estação.

Não iremos adentrar no uso desta ferramenta mais a fundo, pois basicamente isto renderia um outro trabalho e comprovar a eficácia apenas desta ferramenta não é foco deste estudo, além de que sua eficiência já foi comprovada em outros trabalhos relacionados como aqueles citados anteriormente. O importante então é identificar que não apenas controladores mais robustos mas novas ferramentas surgem neste contexto com a proposta de complementar o serviço, assim cada vez mais temos algo que atenda as novas expectativas de uma estrutura de rede emergente. Neste caso em específico, como podemos visualizar através do artigo citado, temos um mecanismo que não apenas é capaz de impedir que estações não autorizadas acessem recursos da rede, mas também pode revogar privilégios mesmo quando já estão autenticadas.

Conclusão

A partir da elaboração deste trabalho, foi possível observar algumas questões básicas de segurança que devem estar presentes em uma implementação deste tipo de rede. A grande parte das ferramentas e controladores possuem código livre, mesmo quando o projeto é realizado com iniciativa privada ainda sim pode-se obter todo o material gratuitamente, esse tipo de modelo de desenvolvimento já se mostrou favorável em diversos aspectos pois a comunidade de desenvolvimento e usuários se tornam eficientes meios de teste. Neste ponto temos ferramentas que já possuem uma boa maturidade de implementação, logo, instabilidade e vulnerabilidades costumam ser apontadas rapidamente, fato que por si só já fornece alguma segurança. A questão principal é que o controlador fornece meios para que sejam definidos políticas de segurança eficientes, mas pelo que podemos notar a medida de segurança adotada no sistema pode ser um fator tão importante quanto ou até superior pois este pode comprometer qualquer serviço disponível na rede além do próprio controlador. Aplicações maliciosas como vistas anteriormente ao infectar o servidor que roda o controlador podem ter acesso, por exemplo, ao endereço de todos os hosts da rede.

Vale ainda ressaltar que cada implementação de rede definida por software apresenta características que distinguem das demais, pois são um tipo de serviço e como qual são configuradas de modo que atendam necessidades específicas. Basicamente, pode-se construir plataformas para diversos tipos de usuários e elaborar um ambiente seguro depende muitas vezes daquilo que o administrador espera, quais tipos de tecnologias e ferramentas serão utilizadas, além da forma como essa plataforma de rede será utilizada.

Com relação as honeypot's houveram pontos interessantes que puderam ser observados, entretanto não houveram casos intrigantes que apontassem elementos mais profundos relacionados a segurança deste tipo de rede. Isso, talvez se deve ao fato da baixa visibilidade da rede, dos problemas recorrentes de estrutura e o baixo número de ataques que utilizam meios mais sofisticados de invasão e em consequência uma simples alteração de login e senha reduz drasticamente o sucesso de invasão. Por fim, mesmo com o número de ataques bem-sucedidos abaixo do esperado para análise o conhecimento e experiência adquiridos tanto por parte da implementação quanto de testes e resultados foram bastante proveitosos. Sem contar que tal experimento foi um dos principais incentivos para busca por ferramentas e medidas que auxiliem no enrijecimento da segurança.

Referências

ALMEIDA, G. de. Pathflow: uma soluç ao sdn para o problema de migraç ao de máquinas virtuais em data centers. 2016. Citado na página 28.

ARAÚJO, M. R. A. G. de. Uma abordagem para aprovisionamento de qos em redes definidas por software baseadas em openflow. 2013. Citado na página 19.

BLEDSOE, G. Back from the dead: Simple bash for complex ddos. *Linux Journal*, Belltown Media, 2011. Citado na página 64.

CAMPOS, V. S. *Arquitetura SDN*. 2016. Web site. Disponível em: <http://www.gta.ufrj.br/ensino/eel879/trabalhos_v1_2015_2/SDN/architecture.html>. Acesso em: 23 mar. 2016. Citado na página 26.

COSTA, A. M. Nicolaci-da. Revoluções tecnológicas e transformações subjetivas. *Psicologia: teoria e pesquisa*, SciELO Brasil, v. 18, n. 2, p. 193–202, 2002. Citado na página 18.

DATABASE, E. *Offensive Security Exploit Database Archive*. 2016. Web site. Disponível em: <https://www.exploit-db.com/>. Acesso em: 24 jul. 2016. Citado na página 40.

DUGAN SETH ELLIOTT, B. A. M. J. P. K. P. J. Iperf: The tcp/udp bandwidth measurement tool. *https://iperf.fr*, 2016. Citado na página 54.

DUQUE, D. H. Redes definidas por software. 2012. Citado na página 32.

FLOODLIGHT, P. *FloodLight*. 2016. Web site. Disponível em: <http://www.projectfloodlight.org/floodlight/>. Acesso em: 23 mar. 2016. Citado 2 vezes nas páginas 23 e 25.

FOUNDATION, L. *OpenDayLight*. 2016. Web site. Disponível em: <https://www.opendaylight.org/>. Acesso em: 23 mar. 2016. Citado na página 23.

GUEDES, D. et al. Redes definidas por software: uma abordagem sistêmica para o desenvolvimento de pesquisas em redes de computadores. *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC 2012*, v. 30, n. 4, p. 160–210, 2012. Citado 2 vezes nas páginas 28 e 37.

LÉVY, P. A revolução contemporânea em matéria de comunicação. *Para navegar no século XXI: tecnologias do imaginário e cibercultura*, v. 2, p. 195–216, 2000. Citado na página 18.

LYON, G. F. *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. [S.l.]: Insecure, 2009. Citado na página 47.

MARCONDES, A. N. Um mecanismo para gerência de segurança da autenticaç ao e controle de acesso em redes sdn. 2014. Citado na página 38.

MATTOS, D. M. F.; DUARTE, O. C. M. B. Authflow: Um mecanismo de autenticaç ao e controle de acesso para redes definidas por software. 2014. Citado 2 vezes nas páginas 38 e 65.

MCCAULEY, J. *NOXRepo*. 2016. Git do POX. Disponível em: <http://www.noxrepo.org/>. Acesso em: 23 mar. 2016. Citado 3 vezes nas páginas 23, 24 e 25.

MCKEOWN, N. How sdn will shape networking. *Open Networking Summit*, 2011. Citado na página 28.

MCKEOWN, N. et al. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, ACM, v. 38, n. 2, p. 69–74, 2008. Citado na página 32.

NEGOCIOS, E. P. empresas e G. *VENDAS DE SMARTPHONES CRESCEM NO MUNDO TODO*. 2016. Web site. Disponível em: <http://revistapegn.globo.com/Noticias/noticia/2016/06/vendas-de-smartphones-crescem-no-mundo-todo.html>. Acesso em: 23 mar. 2016. Citado na página 30.

NETWORKS, B. S. *BigSwitch*. 2016. Web site. Disponível em: <http://www.bigswitch.com/>. Acesso em: 23 mar. 2016. Citado na página 26.

PROJECT, H. *Know Your Enemy: Learning about Security Threatsr*. 2. ed. [S.l.]: Addison-Wesley, 2004. Acesso em: 25 Jun 2016. Citado na página 28.

PROVOS, N. Honeyd-a virtual honeypot daemon. In: *10th DFN-CERT Workshop, Hamburg, Germany*. [S.l.: s.n.], 2003. v. 2, p. 4. Citado na página 37.

QUINTAO, T. et al. Avaliaç ao, através de simulaç ao e teste real, de um firewall baseado em sdn. 2015. Citado na página 38.

RIBEIRO, C. P. Análise e implementação de mecanismos de segurança para computação em nuvem. p. 40–41, 2015. Citado na página 40.

ROJAS, M. A. T.; CARVALHO, T. C. M. de B. Proposta de mecanismo de controle de acesso para o ambiente sdn/openflow. *Sistemas e Tecnologias de Informação*, p. 355, 2013. Citado na página 38.

RUIZ, A. F. Estado del arte redes definidas por software (sdn). Universidad Católica de Pereira, 2015. Citado 2 vezes nas páginas 19 e 37.

SPITZNER, L. *Know your enemy: Revealing the security tools, tactics, and motives of the blackhat community*. [S.l.]: Addison-Wesley New York, NY, 2001. Citado na página 28.