Uma Arquitetura de Detecção e Prevenção de Intrusão para Redes Definidas por Software

Martín Esteban Andreoni Lopez

PEE / COPPE / UFRJ

Dissertação submetida para a obtenção do título de
Mestre em Engenheira Elétrica
ao Programa de Engenharia Elétrica/COPPE/UFRJ

UMA ARQUITETURA DE DETECÇÃO E PREVENÇÃO DE INTRUSÃO PARA REDES DEFINIDAS POR SOFTWARE

Martín Esteban Andreoni Lopez

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Otto Carlos Muniz Bandeira Duarte

Rio de Janeiro
Maio de 2014

UMA ARQUITETURA DE DETECÇÃO E PREVENÇÃO DE INTRUSÃO PARA REDES DEFINIDAS POR SOFTWARE

Martín Esteban Andreoni Lopez

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

Prof. Otto Carlos Muniz Bandeira Duarte, Dr.Ing.

Prof. Miguel Elias Mitre Campista, D.Sc.

Prof. Igor Monteiro Moraes, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
MAIO DE 2014

Lopez, Martín Esteban Andreoni

Uma Arquitetura de Detecção e Prevenção de Intrusão para Redes Definidas por Software/Martín Esteban Andreoni Lopez. – Rio de Janeiro: UFRJ/COPPE, 2014.

XV, 77 p.: il.; 29, 7cm.

Orientador: Otto Carlos Muniz Bandeira Duarte

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2014.

Referências Bibliográficas: p. 72 – 77.

1. Detecção de Intrusão. 2. Redes Definidas por Software. 3. Ataques Cibernéticos. 4. Internet do Futuro. 5. Segurança. 6. OpenFlow. 7. Negação de Serviço. I. Duarte, Otto Carlos Muniz Bandeira. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

À minha família.

Agradecimentos

Agradeço a Mamá, Papá, Dani e Mora, que sempre estiveram ao meu lado, por todo carinho e compreensão. Em especial, agradeço aos meus pais, pelo apoio que me dão em todos os momentos e por sempre me motivarem a seguir em frente.

Agradeço aos amigos, em especial, Lyno Ferraz, Diogo Menezes, Andrés Murillo, Govinda Mohini pela amizade e companheirismo e por tornarem o mestrado uma experiência repleta de aprendizagem e boas lembranças. Agradeço, também, a todos os amigos que fiz no Grupo de Teleinformática e Automação, pois sempre contribuíram positivamente para a conclusão desse trabalho. Uma menção especial de agradecimento a Antonio Gonzalez Pastana Lobato e Ulisses da Rocha Figueiredo pela ajuda e discussões nesta dissertação.

Agradeço, ainda, a todos os professores que participaram da minha formação. Em especial, agradeço ao meu orientador, professor Otto Carlos Duarte, por todos os conselhos, dedicação e principalmente paciência, durante a orientação no mestrado. Gostaria de agradecer também aos professores Luís Henrique Maciel Kosmalski Costa, Miguel Elias Mitre Campista e Aloysio de Castro Pinto Pedroza do GTA/UFRJ.

Agradeço ao professor Igor Monteiro Moraes pela participação na banca examinadora.

Agradeço aos funcionários do Programa de Engenharia Elétrica da COPPE/UFRJ, Maurício, Daniele, Rosa e Arthur pela presteza no atendimento na secretaria do Programa.

Agradeço a todos que participaram de forma direta ou indireta da minha formação profissional. Por fim, agradeço a FINEP, FUNTTEL, CNPq, CAPES, FAPERJ e UOL pelo financiamento deste trabalho.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

UMA ARQUITETURA DE DETECÇÃO E PREVENÇÃO DE INTRUSÃO PARA REDES DEFINIDAS POR SOFTWARE

Martín Esteban Andreoni Lopez

Maio/2014

Orientador: Otto Carlos Muniz Bandeira Duarte

Programa: Engenharia Elétrica

Os Sistemas de Detecção e Prevenção de Intrusão são fundamentais para inspecionar o tráfego da rede em tempo real, em busca de padrões anômalos causados por intrusos ou abusos de usuários internos, para garantir a segurança dos sistemas de comunicação. Nesta dissertação de mestrado é proposto o BroFlow, um Sistema de Detecção e Prevenção de Intrusão baseado nas características da ferramenta de análise de tráfego Bro e na visão de rede global da Interface de Programação de Aplicação OpenFlow. As principais contribuições do BroFlow são: (i) detecção de intrusão através de algoritmos simples implementados através de uma arquitetura modular e flexível; (ii) reação imediata a um ataque e descarte dos pacotes atacantes o mais perto possível da origem; (iii) posicionamento estratégico de sensores para detecção de ataques em uma infraestrutura de rede compartilhada com diversos inquilinos; e (iv) o fornecimento de recursos em demanda de forma dinâmica e elástica ante a sobrecarga da máquina de análises. Um protótipo do sistema proposto foi implementado e avaliado no ambiente de redes virtuais Future Internet Testbed with Security (FITS). A avaliação do sistema sob ataque mostra que o BroFlow garante o encaminhamento de pacotes legítimos na rede na taxa máxima do enlace e reduz, em até dez vezes, o atraso na rede provocado pelo ataque, mesmo quando os atacantes são inquilinos legítimos agindo em conluio.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

AN INTRUSION DETECTION AND PREVENTION ARCHITECTURE FOR SOFTWARE DEFINED NETWORKING

Martín Esteban Andreoni Lopez

May/2014

Advisor: Otto Carlos Muniz Bandeira Duarte

Department: Electrical Engineering

Intrusion Detection and Prevention Systems are fundamental to inspect real-time network traffic, seeking abnormal patterns caused by intruders or insider misuse, to ensure communication systems security. Moreover, this is the only effective mechanism to detect attacks from internal authenticated users. We propose BroFlow, an Intrusion Detection and Prevention System based on Bro traffic analyzer, and on the global network-view feature of OpenFlow Application Programming Interface. BroFlow main contributions are: (i) intrusion detection through simple algorithms implemented by a modular and flexible architecture; (ii) immediate reaction to an attack and malicious packets dropping closest from its origin; (iii) strategic sensor positioning for attack detection in an infrastructure network shared by multi-tenants; and (iv) dynamic and elastic resource provision of analyses machines under demand. A system prototype was implemented and evaluated in the virtual environment Future Testbed Internet with Security (FITS). A system evaluation under attack shows that BroFlow guarantees the forwarding of legitimate packets at the maximal link rate and reduces, up to ten times, the maximal network delay caused by the attack, even when the attackers are legitimate tenants acting in collusion.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Abreviaturas

API	*Application Programming Interface*, p. 41
BROCOLI	*Bro Client Communications LIbrary*, p. 42
DDoS	*Distributed Denial of Service*, p. 13
DPI	*Deep Packet Inspection, p. 8*
DoS	*Denial of Service, p. 7*
FN	*Falsos Negativos, p. 10*
FP	*Falsos Positivo, p. 10*
HIDS	*Host-based Intrusion Detection System, p. 14*
HTTP	*Hypertext Transfer Protocol, p. 3*
ICMP	*Internet Control Message Protocol, p. 3*
IDPS	*Intrusion Detection and Prevention System, p. 3, 11*
IDS	*Intrusion Detection System, p. 2*
IPS	*Intrusion Prevention System, p. 2*
NBA	*Network Behavior Analysis, p. 15*
NFAT	*Network Forensic Analysis Tool, p. 19*
NGFW	*Next Generation Firewall, p. 19*
NIC	*Network Interface Controller, p. 40*
NIDS	*Network-based Intrusion Detection System, p. 14*
OF	*OpenFlow, p. 4*
SDN	*Software Defined Networking , p. 3, 44*

SPAN *Switched Port Analyzer, p. 16*

SSH *Secure Shell, p. 18*

SSL *Secure Sockets Layer, p. 18*

SYN *Synchronization, p. 3*

TCP *Transmission Control Protocol, p. 3*

UDP *User Datagram Protocol, p. 3*

VN *Verdadeiros Negativo, p. 11*

VP *Verdadeiros Positivo, p. 10*

Capítulo 1

Introdução

A Internet é um conjunto de redes físicas heterogêneas com controle descentralizado, as quais funcionam como uma rede lógica única de alcance mundial. O grande e contínuo crescimento da Internet gerou um aumento da sua complexidade, que a expõe a diversas vulnerabilidades. Hoje, atacantes exploram essas vulnerabilidades para realizar diversos tipos de ataque com diferentes objetivos. Neste cenário inseguro, os atacantes criam diferentes ameaças tendo como alvos desde as agências de serviços financeiros até grandes sistemas de controle industrial como no caso do vírus *Stuxnet* [1] que causou danos e interrompeu a geração de energia em uma usina nuclear no Irã. Os métodos de ataques variam amplamente, utilizando técnicas simples para explorar as vulnerabilidades de protocolos ou mediante operações combinadas para a utilização de múltiplos *Bots* ou máquinas zumbis. A Figura 1.1 mostra as ameaças mais frequentes registradas durante o ano de 2013 nas redes de computadores. Entre os mais importantes encontram-se a injeção nas bases de dados SQL e o ataque de desconfiguração web, também conhecido como *Defacement*. Sendo o ataque mais frequente durante o ano de 2013 os ataques de Negação de Serviço em suas diversas formas. Para lidar com a complexidade da Internet diversas propostas modificam a estrutura da rede. Um novo conceito, conhecido como Redes Pluralistas, propõe redes virtualizadas que permitem diversas máquinas virtuais compartilhar os recursos das máquinas físicas. Nesta proposta, cada rede virtual tem a suas próprias caraterísticas dependendo das necessidades de aplicações de redes. Permitindo a utilização de diferentes protocolos, o que permite uma compatibilidade com a Internet atual, e brindando diferentes serviços como segurança, mobilidade ou qualidade de serviço [2]. O isolamento das redes virtuais garante que uma rede virtual não afete as outras que executam na mesma infraestrutura [3]. Além de novos desafios como o isolamento de redes virtuais, problemas tradicionais relacionados com a segurança da Internet continuam presentes em ambientes virtualizados. Por exemplo, os ataques de Negação de Serviço contra uma rede física em um ambiente virtualizado, poderia afetar o funcionamento de todas

as redes virtuais hospedadas nessa rede física. Os impactos dos ataques variam de acordo com as ameaças e os riscos, podem chegar a provocar prejuízos incalculáveis e até mesmo catástrofes quando direcionados a infraestruturas críticas como, por exemplo, as redes elétricas inteligentes [4].

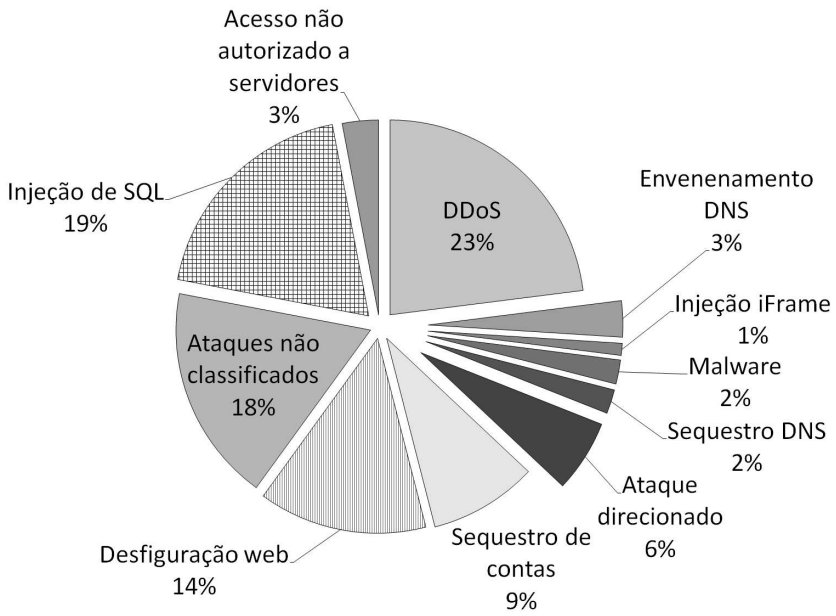


Figura 1.1: Ataques durante o ano de 2013. Fonte: www.wordpress.com/top-attacks-2013

Durante muitos anos os sistemas de segurança se baseiam em barreiras de proteção, tais como firewalls e mecanismos de controle de acesso contra as ameaças. No entanto, hoje em dia com o avanço das ameaças, os sistemas de segurança do tipo barreiras de proteção e mecanismos de controle de acesso não são mais suficientes e deixam as redes sem segurança e indefensas ante muitos tipos de ataques, precisando de uma complementação entre ferramentas de segurança [5]. Acrescente-se a isso que 70% dos ataques são originados por usuários internos e legítimos [6] da rede e, portanto, os métodos convencionais de segurança baseados em barreiras são completamente ineficazes. Uma das formas de se proteger desses ataques internos é monitorar o tráfego em busca de atividades maliciosas ou violação de políticas. Para realizar o monitoramento de pacotes da rede, a ferramenta mais utilizada são os Sistemas de Detecção de Intrusão (*Intrusion Detection System* - IDS) que realiza o monitoramento passivo dos pacotes na rede [7]. Os IDS são em sua maioria utilizados para identificar as causas dos ataques após a sua ocorrência, então a análise dos pacotes ocorre de forma *offline*, mas dependendo da tecnologia utilizada é possível fazer uma análises em tempo real ou *online*. Os Sistemas de Prevenção de Intrusão (*Intrusion Prevention System* - IPS) são considerados uma extensão dos sistemas de Detecção de Intrusão, pois atuam sobre o sistema monitorado uma vez que as ameaças ou atividades maliciosas são detectadas. Logo, os sistemas que atuam con-

juntamente na detecção e na prevenção são mais completos e denominados Sistemas de Detecção e Prevenção de Intrusão (*Intrusion Detection and Prevention System* - IDPS).

Os Ataques de Negação de Serviço (*Denial of Service* - DoS) visam consumir uma grande quantidade de recursos da vítima, de forma a interromper os serviços prestados ou evitar que usuários legítimos sejam atendidos com a qualidade de serviço adequada. Existem registros de ataques de negação de serviço por inundação que causaram perdas milionárias [8]. No começo do ano de 2013, *hackers* geraram ataques de DoS contra sítios *web* comerciais como Visa, Mastercard e PayPal, e causaram perdas de mais de 130 milhões de reais. Nesse período, se registrou o maior volume de tráfego de ataque da história da Internet, com um volume maior que 300 Gb/s[1].

Os ataques por inundação se aproveitam de vulnerabilidades na implementação e na arquitetura da rede, assim como também exploram falhas nas especificações dos protocolos. Exemplos de ataques de negação de serviço por inundação são: inundação por SYN, ICMP, UDP, HTTP, entre outros. A Figura 1.2 mostra os Ataques de Negação de Serviço por Inundação registrados durante o ano de 2013. Durante esse ano, o ataque de inundação de TCP SYN foi o mais comum. Este ataque explora a vulnerabilidade do procedimento de abertura de conexão que se efetua com três trocas de mensagens (*three way handshake*) do protocolo de transporte TCP. O ataque consiste em enviar à vítima diversos pacotes de SYN, de pedidos de conexão, provocando a abertura simultânea de múltiplas conexões, consumindo recursos de processamento, memória e número de conexões do servidor atacado. O procedimento *three way handshake* requer uma "confirmação" do solicitante da conexão, que no caso de ser um atacante nunca é enviada. O grande desafio para combater ataques de DoS é que é difícil distinguir um pacote legítimo de um pacote de ataque. Além disso, por falta de uma visão global de controle da rede, a detecção do ataque por caracterização de tráfego ocorre perto do destino, sendo mais difícil bloquear o ataque perto das fontes.

O paradigma das Redes Definidas por Software (*Software Defined Networking* - SDN) é um paradigma para desenvolver a Internet do Futuro [9]. SDN propõe a separação de planos em plano de controle centralizado e plano de dados distribuído [10]. Assim, SDN oferece a facilidade de visão global da rede no o plano de controle, o que permite uma maior programabilidade e gerenciamento da segurança. A programabilidade do plano de controle centralizado com a visão global da rede dos elementos encaminhadores se constitui em uma poderosa facilidade que proporciona um processamento ativo e dinâmico de todos os fluxos da rede [11].

[1]Reportagem de revista especializada em segurança informando que ataques com tráfego de 200-400 Gb/s passaram a ser comuns www.krebsonsecurity.com/200-400-gbps-ddos-attacks

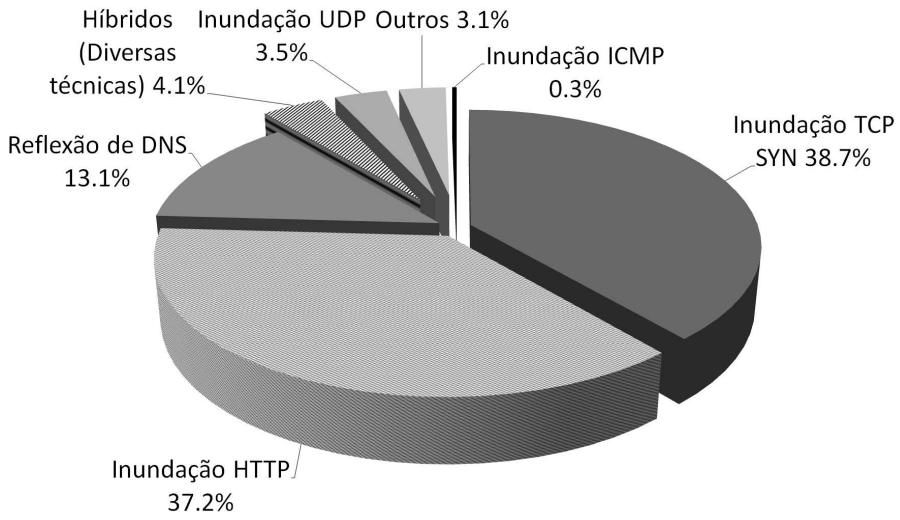


Figura 1.2: Classificação Ataques de Negação de Serviço por inundação durante o ano de 2013. Fonte: Artigo sobre os ataques de DoS da companhia nsfocus, especialistas em segurança informática. www.en.nsfocus.com/SecurityReport/2013

O OpenFlow (OF) é a implementação de maior sucesso do paradigma SDN e seus elementos encaminhadores são chamados de comutadores OpenFlow. O OpenFlow fornece instruções básicas para modificar, encaminhar e bloquear os fluxos dentro de uma rede. Assim, é possível criar aplicações de segurança que reagem contra ataques, atuando diretamente nos fluxos. Para garantir a segurança são usados como características fundamentais a visão global da rede, a facilidade de programação do controle da rede e a possibilidade de reagir a ataques. Além disso, independentemente de onde possa ocorrer a detecção do ataque, a eliminação do fluxo atacante é eficaz em todo o caminho desde a origem do ataque até à vítima, o que resulta em uma eficiente forma de eliminação de tráfego de ataque.

Na arquitetura OpenFlow, existe uma tabela de fluxos nos comutadores que é atualizada pelo plano de controle. Quando chega um pacote de um fluxo que já está definido nessa tabela, ele é simplesmente comutado. Caso chegue um pacote de um fluxo não definido, ele é enviado para o controlador, onde é definido qual será o próximo destino desse pacote. O destino é escolhido de acordo com as aplicações sendo executadas e também de acordo com o cabeçalho do pacote. O controlador então adiciona essa regra na tabela de fluxo do elemento encaminhador. É importante ressaltar que só o primeiro pacote de um fluxo não definido é encaminhado ao controlador, sendo a partir dele criada a regra na tabela de fluxo, o que faz que os demais pacotes desse fluxo sejam somente comutados, afetando pouco o desempenho da rede. Em contrapartida, a comunicação entre o plano de dados e o plano de controle sofre de um problema de escalabilidade, já que pode existir um excesso de comunicação entre os dois planos durante a chegada de pacotes de novos fluxos. Esta vulnerabilidade pode ser explorada mediante Ataques de Negação de Serviço

com endereços falsificados, o qual representa um possível excesso de entradas nas tabelas de fluxos.

Esta dissertação propõe o BroFlow [12], um sistema de Detecção e Prevenção de Intrusão (IDPS) elástico e distribuído para a defesa contra Ataques de Negação de Serviço (DoS). Para tanto, o BroFlow é baseado na Interface de Programação de Aplicação de redes OpenFlow [10] e na ferramenta de análise de tráfego Bro [13]. Foram implementados diferentes algoritmos para a detecção de anomalias que correspondem a ataques de negação de serviço por inundação. Os sensores do BroFlow enviam alarmes ao controlador POX através de canais seguros e uma aplicação OpenFlow efetua as contramedidas a serem realizadas, de modo a bloquear o ataque de DoS o mais perto possível da fonte e, assim, eliminar o fluxo malicioso no comutador OpenFlow já na entrada do fluxo na rede. Também foi desenvolvido com os colegas Figueiredo e Lobato, uma facilidade para replicação do sistema de análise de tráfego. Logo, se o sistema de análise de tráfego estiver sobrecarregado, ele envia uma mensagem ao sistema de gerência da rede, que disponibiliza mais recursos físicos [14]. Isso é realizado com a criação de máquinas virtuais específicas que realizam a análise de tráfego, que podem ser rapidamente replicadas e desligadas para atender às demandas variáveis. O controlador OpenFlow é então informado para que seja instaurada uma nova regra para a distribuição dos fluxos espelhados. Caso após algum tempo o tráfego diminua e não sejam mais necessárias tantas máquinas de análise, outro alerta é enviado para que estas máquinas sejam desativadas, garantindo assim um melhor aproveitamento de recursos.

As principais contribuições do sistema BroFlow são: (i) a detecção em tempo real de ataques de negação de serviço através de algoritmos simples implementados em uma linguagem específica para eventos de rede; (ii) a reação instantânea para a interrupção de ataques de negação de serviço; (iii) o posicionamento estratégico de sensores na rede para garantir o correto funcionamento da infraestrutura física, mesmo em cenários em que redes virtuais legítimas agem em conluio para realizar um ataque e (iv) o fornecimento de recursos sob demanda de forma elástica ante a sobrecarga do IDS.

A proposta BroFlow de detecção e prevenção de intrusão se serve da característica de visão global da rede para agir diretamente no controle do encaminhamento dos fluxos na rede, eliminando de forma eficaz o fluxo malicioso ainda na sua origem. Também é possível redirecionar o fluxo malicioso para uma réplica do servidor atacado ou para um servidor de pote de mel (*honeypot*). A eficiência e flexibilidade oferecida pela ferramenta Bro de análise de tráfego permite ao sistema proposto a criação de políticas eficientes de segurança individuais para cada inquilino da rede. Com a ferramenta Bro é possível detectar e bloquear os ataques de maneira local, enquanto que o controle e gerenciamento global da rede fornecida pelo OpenFlow

permite aplicar essa informação em todos os comutadores da rede eliminando o ataque desde a origem.

Um protótipo do sistema BroFlow foi desenvolvido, avaliado e testado na plataforma *Future Internet Testbed with Security* (FITS), uma plataforma de teste de redes baseada na técnica de virtualização de redes. Os resultados mostram a eficácia do sistema para detectar e reagir ante ataques de DoS por inundação.

1.1 Organização do Texto

O restante deste trabalho está organizado da seguinte forma. O Capítulo 2 apresenta os fundamentos de Sistemas de Detecção e Prevenção de Intrusão (IDPS). A ferramenta para o desenvolvimento de algoritmos de detecção de ataques é apresentado no Capítulo 3. O Capítulo 4 descreve o sistema BroFlow para a detecção e prevenção de ataques de negação de serviço nas redes definidas por software. O Capítulo 5 conclui este trabalho e apresenta as direções de trabalhos futuros.

Capítulo 2

Os Sistemas de Detecção e Prevenção de Intrusão

As atuais soluções para a segurança em redes, como por exemplo, os *firewalls*, não foram projetadas para controlar ataques às camadas de rede e de aplicação. Ataques como a Negação de Serviço (*Denial of Service* - DoS), vermes (*worms*), cavalos de Tróia são alguns exemplos que ultrapassam os atuais sistemas de segurança. Além disso, as atividades não autorizadas são efetuadas por atacantes externos que invadem os sistemas, mas também por usuários internos autenticados ilegítimos, sendo este último aproximadamente 70% dos ataques detectados [6]. Assim, os atuais sistemas de segurança são totalmente vulneráveis ante ataques sofisticados, precisando de uma complementação com outros mecanismos modernos de segurança. A detecção de intrusão é o mecanismo de segurança mais eficaz para detectar ataques internos e consiste no processo de monitorar e analisar eventos que ocorrem em um sistema ou rede de computação em busca de sinais de possíveis incidentes de segurança [7]. Esses incidentes de segurança são violações ou ameaças às políticas de segurança definido como tentativas de comprometer a confiabilidade, a integridade ou a disponibilidade dos recursos do sistema. Uma ameaça de segurança é considerada como a possibilidade ou tentativa potencial de um intento "não autorizado" de acessar e/ou manipular a informação ou ainda de tornar um sistema não confiável ou não utilizável. Os IDS inspecionam e analisam informação relativa aos eventos observados e emitem alarmes para ao administrador de rede ou são armazenados em arquivos para uma análises.

Os Sistemas de Prevenção de Intrusão (*Intrusion Prevention System* -IPS) atuam uma vez que as ameaças ou atividades maliciosas são detectadas e por isto são considerados uma extensão dos sistemas de Detecção de Intrusão (*Intrusion Detection System* -IDS) [15]. As ações dos IPS são o descarte dos pacotes maliciosos, a reinicialização da conexão, o bloqueio o tráfego especifico do atacante entre outras. Consequentemente, os sistemas que atuam conjuntamente na detecção e na pre-

venção se denominam Sistemas de Detecção e Prevenção de Intrusão (IDPS). As principais funções dos IDPSs são: i) identificar uma ameaça ou atividade maliciosa, ii) registrar a informação relevante dessa atividade maliciosa e iii) reagir para interromper ou bloquear seu efeito e e reportar o evento ao administrador da rede.

Existem diferentes técnicas de respostas relacionadas aos Sistemas de Detecção e Prevenção de Intrusão, as quais podem ser divididas em três grandes grupos:

- interrupção do ataque: o IDPS pode terminar a conexão ou finalizar a sessão que está sendo usada para o ataque, mas também pode fazer um bloqueio de acesso do atacante ao dispositivo ou à vítima atacada, ou seja, impedimento do acesso à conta, ao endereço IP ou a qualquer outro atributo do atacante;

- alteração do ambiente de segurança: o IDPS pode modificar a configuração de controle de outros dispositivos de segurança para interromper o ataque, como por exemplo, alterar a configuração do firewall, dos roteadores ou dos comutadores, para impedir o acesso ao atacante. Alguns IDPS comerciais, podem até gerar correções ou *patches* para aplicar nos hospedeiros uma vez detectada a vulnerabilidade. Também nesta categoria se inclui desviar o fluxo atacante até uma área segura onde não afete nenhum dispositivo;

- alteração do conteúdo do ataque: os IDPS podem remover ou substituir porções do ataque para converte-lo em ação benigna, esta ação é conhecida como higienizado. Um IDPS pode remover um arquivo infestado dentro de um correio eletrônico ou atuar como um `proxy` removendo o conteúdo de um pacote em ataques mais complexos.

Os IDPS podem implementar a técnica denominada Inspeção Profunda de Pacotes (*Deep Packet Inspection* -DPI) que não se limita a verificar os cabeçalhos dos pacotes, mas também inspeciona o seu conteúdo em busca de parâmetros específicos, tal como uma sequência especifica de caracteres. Assim, dependendo da implementação do IDPS, a inspeção de um pacote pode ir da camada de enlace até a camada de aplicação [16]. Portanto, a inspeção inclui a estrutura e o conteúdo dos cabeçalhos de protocolos, assim como a carga útil ou conteúdo da mensagem. Desse modo, os dispositivos baseados em DPI são mais complexos e efetivos, pois também identificam e classificam tráfego baseado no conteúdo do pacote, o que confere a esse sistema um controle mais fino e acurado quando comparados aos sistemas que analisam apenas os cabeçalhos dos pacotes [17].

Algumas definições e terminologias relativas aos IDPS que são usada nesta dissertação são: [18]

- **Ameaça:** possibilidade ou tentativa potencial de um intento “não autorizado”

de: acesso e/ou manipulação da informação ou de tornar um sistema não confiável ou não utilizável;

- **Risco:** exposição acidental ou imprevisível da informação, ou violação da integridade das operações devido ao mau funcionamento do *hardware* ou projeto incompleto ou incorreto do *software*;
- **Vulnerabilidade:** falha conhecida ou suspeita no projeto de *hardware* ou *software*, ou operação que expõe o sistema a penetração ou obtenção da sua informação;
- **Ataque:** formulação específica ou execução de um plano para realizar ou executar uma ameaça;
- **Invasão no Controle de Segurança:** também conhecido como penetração. Sucesso na tentativa do ataque, e capacidade de obter um acesso não autorizado ao sistema.

2.1 Componentes dos Sistemas de Detecção e Prevenção de Intrusão

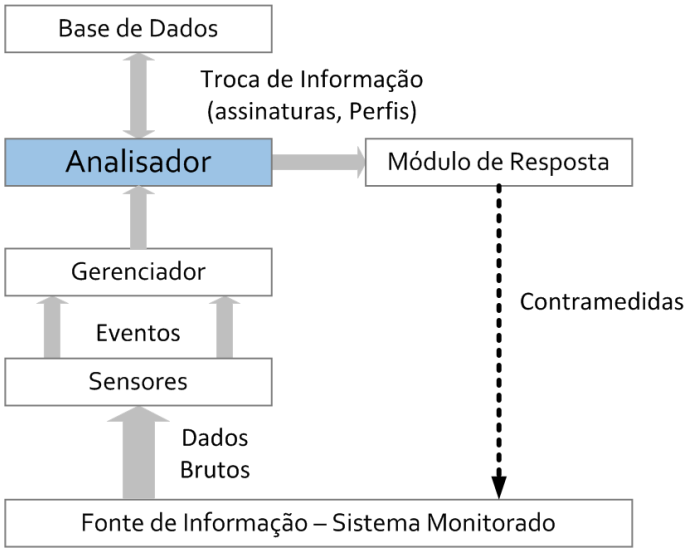


Figura 2.1: Esquema Básico dos sistemas de Detecção e Prevenção de Intrusão. Fonte *Lazarevic et al.* [19].

Basicamente os Sistemas IDPS são constituídos de cinco módulos, como mostra a Figura 2.1: sensores, gerenciador, base dados, analisador e resposta. Os sensores são responsáveis por obter os dados no local onde ocorrem os eventos e convertê-los em informação coerente para o analisador. Os dados obtidos geralmente são conhecidos

como informação auditada. O módulo gerenciador recebe as informações básicas dos sensores, gerando eventos de mais alto nível de abstração, os quais são passados ao analisador. Portanto, o gerenciador correlaciona informações de distintos sensores para gerar eventos. Em alguns sistemas o gerenciador e o analisador podem estar juntos em um mesmo módulo, realizando as análises imediatamente após receber a informação do sensor. Em sistemas IDPS de grande escala de sensoriamento podem existir mais de um gerenciador, inclusive pode existir uma camada de gerenciamento para correlacionar grandes volumes de informações [20]. O módulo de base de dados contém a informação processada coletada pelos sensores. Aqui são armazenadas as assinaturas ou perfis de comportamento, dependendo do método de detecção. O módulo analisador é o núcleo dos IDPS, o qual analisa a informação auditada a fim de detectar ataques. Quando o analisador detecta um comportamento que ele interpreta como ataque, ele produz registros ou `logs` com a informação do ataque correspondente. Por fim, o módulo de resposta é responsável pelo mecanismo de reação e, portanto, determina como responder e que contramedida adotar quando o analisador detecta um ataque. Dependendo das políticas de segurança estabelecidas pelo administrador da rede, o módulo de resposta acionará notificações do ataque conhecidas como alertas ou executará ações tais como o bloqueio do trafego de rede.

Como os IDPS são baseados na hipótese de que o comportamento dos intrusos difere do comportamento de um usuário legítimo, a detecção de um ataque não é garantida e também não é garantido que aquele ataque detectado pelo sistema seja realmente um ataque. Assim, a acurácia desses sistemas é baseada em duas métricas:

- Falsos Positivo (FP) que são comportamentos legítimos que o sistema classificou como comportamento malignos e, portanto, intrusivo. Logo, o sistema gerou um alerta ou alarme por engano, também conhecido como um falso alarme;

- Falsos Negativo (FN) são intrusões ou comportamentos malignos que o sistema não detectou e, portanto, os classificou como comportamento normal ou inofensivo. Isso significa que o ataque não é detectado pelo sistema. Além disso, com esse erro são relacionados dois conceitos enquanto a eficiência do sistema.

Também são usados como medida de eficiência do sistema de detecção de intrusão os:

- Verdadeiros Positivo (VP) é a classificação de uma intrusão como um evento malicioso, como se espera que aconteça. A taxa de Verdadeiros Positivos é sinônimo de taxa de detecção de intrusão ou sensibilidade do sistema;

- Verdadeiros Negativo (VN) é a classificação correta dos eventos legitimos como normais. A taxa de Verdadeiros Negativos é também conhecida na literatura como especificidade do sistema.

Assim, o desempenho dos Sistemas de Detecção e Prevenção de Intrusão é avaliado usando a métrica de acurácia, Ac, expressa por

$$Ac = \frac{VP - VN}{VP - FP + VN + FN} \tag{2.1}$$

que indica o número total de eventos classificados corretamente como intrusões, incluindo eventos normais e intrusos e a métrica de precisão, ou taxa de detecção, TD, expressa por

$$TD = \frac{VP}{VP + FN} \tag{2.2}$$

que indica a relação entre o número de eventos maliciosos corretamente detetados e o número total de ataques.

As variáveis utilizadas, extraídas do campo da inteligência artificial, levam a uma ferramenta utilizada para a ajuda na visualização dos resultados. A ferramenta denominada matriz de confusão, como mostra a Figura 2.2 faz uma comparação entre os valores previstos e os valores detectados pelos IDS.

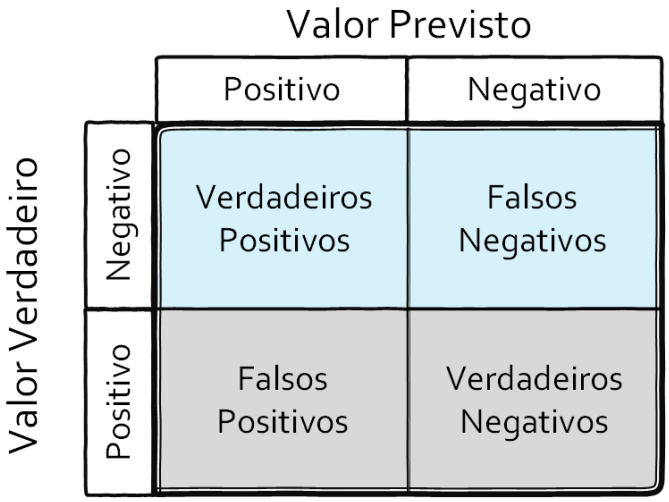


Figura 2.2: Matriz de confusão dos Sistemas de Detecção de Intrusão.

2.2 Classificação dos Sistemas IDPS

Os Sistemas de Detecção e Prevenção de Intrusão (*Intrusion Detection and Prevention System* - IDPS) podem ser classificados de diferentes formas, sendo as cinco principais ilustradas na Figura 2.3, que são: i) o método de detecção: assinatura ou anomalia; ii) a forma de aquisição de dados: IDS por estação (*Host based Intrusion*

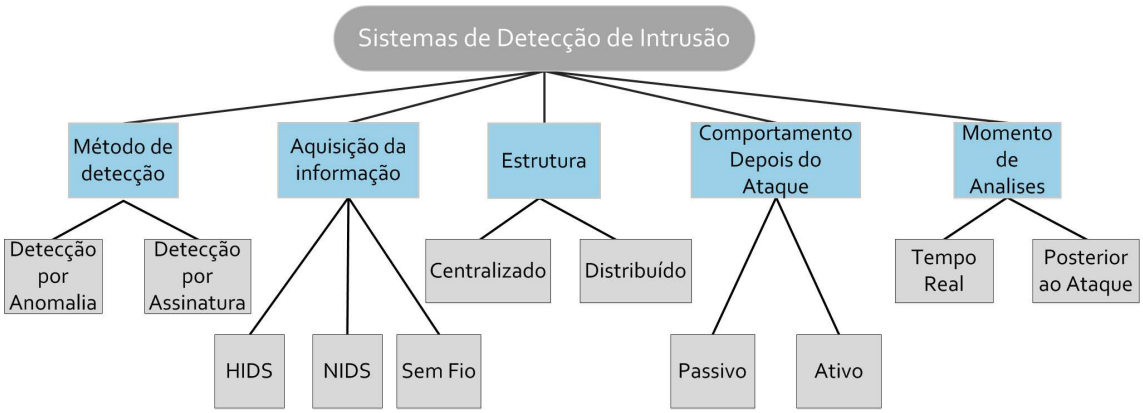


Figura 2.3: Classificação dos Sistemas de Detecção e Prevenção de Intrusão.

Detection System-HIDS), IDS baseado em rede (*Network based Intrusion Detection System*-NIDS), uma combinação de ambos IDS híbrido e os IDS Sem Fio; iii) a estrutura do controle: centralizado ou distribuído; iv) o comportamento após a detecção do ataque: passivo ou ativo e, finalmente, v) o momento da análise: em tempo real ou em análises posterior ao ataque. A seguir são detalhadas estas classificações.

2.2.1 Classificação Segundo o Método de Detecção

Os sistemas de detecção de intrusão devem ser capazes de distinguir entre um comportamento normal ou anormal do sistema. No entanto, traduzir os comportamentos normais ou anormais em decisões de segurança, não é uma tarefa fácil, já que muitos padrões de comportamento são imprevisíveis. Para classificar as diferentes ações, os IDPS aproveitam as abordagens da detecção por anomalia ou da detecção por assinatura. A detecção por anomalia assume que toda a atividade intrusiva, é uma atividade anômala. Por isso, todos os eventos que se desviam em relação ao comportamento normal do sistema, são considerados eventos anômalos. Para isso o IDPS cria perfis com o comportamento normal do sistema. Esses perfis são criados monitorando o ambiente durante um determinado período de tempo. Por exemplo, se um usuário tenta se conectar no sistema durante períodos não laborais, pode ser considerado como uma anomalia. Alternativamente, se um processo utiliza normalmente 10% dos recursos de CPU e subitamente utiliza 90% dos recursos, o processo pode ser considerado anômalo. A grande vantagem do método de detecção por anomalia é de não ser necessário conhecer a priori todas as possíveis ameaças e isto é fundamental para a detecção de novos ataques que exploram novas vulnerabilidades.

Para gerar os perfis de comportamento, o método de detecção por anomalia baseia se em abordagem estatística. Utilizando comparações por limiares ou classificação por inteligência artificial, as atividades de rede são comparadas com os perfis criados anteriormente. O grande inconveniente da detecção por anomalia que

normalmente este procedimento gera uma alta taxa de falsos positivos. O sistema detecta como anomalia, ou seja, detecta e sinaliza como ataques todos os eventos de mudança de comportamento do sistema avaliado, mas que não eram ameaças de segurança. Esses falsos positivos são causados majoritariamente por dois tipos de problemas. O primeiro é a falta de conjunto de dados (*datasets*) no caso da inteligência artificial. Os *datasets* existentes são antigos e os ataques que estes *datasets* contêm foram gerados artificialmente, gerando uma grande limitação na hora de criar o perfil [21]. O segundo problema são possíveis violações dos limiares estabelecidos os quais não sejam realmente ataques, como pode ser o caso dos "Flash Crowd"[1].

Entre as abordagens utilizadas pelo método de detecção por anomalia mais conhecidos existem as baseadas em:

- padrões de comportamento predito, na qual futuros eventos são previstos baseando-se nos eventos já acontecidos, usando perfis para cada um dos usuários. Assim, se nenhum dos possíveis futuros eventos previstos acontecem, o comportamento é considerado anômalo;

- limiares, na qual são definidos limiares para a frequência na qual acontecem os eventos, independentemente dos usuários;

- regras, na qual é definido um conjunto de regras que são usadas para decidir se um determinado comportamento corresponde a uma intrusão.

O outro método de detecção de intrusão é por assinatura. Essa abordagem, também conhecida como detecção por abuso (*misuse*), utiliza a comparação dos dados analisados com uma base de dados previamente definida como caraterísticas típicas, chamadas padrões ou assinaturas, de ataques conhecidos. Essas características, as assinaturas, identificam uma ameaça em particular e são encontradas inspecionando o tráfego e até o conteúdo de cada pacote, chamado de carga útil, utilizando técnicas de Inspeção Profunda de Pacotes (*Deep Packet Inspection* – DPI). Se um padrão ou característica observado no ambiente monitorado correspondente com uma assinatura na base de dados, ele é marcado como um desvio da política de segurança ou como um ataque.

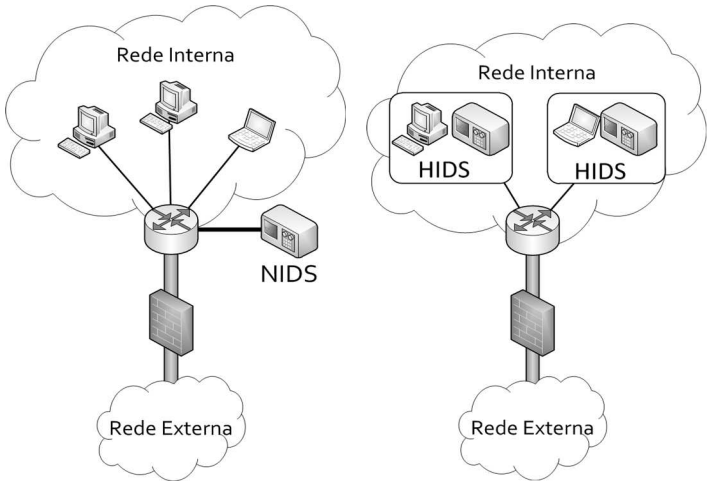
A vantagem principal dessa técnica de detecção é que tem uma alta precisão na detecção de ataques conhecidos e, portanto, uma baixa taxa de falsos positivos. Além disso, o método de detecção é bem mais simples que os métodos utilizados na técnica

[1] Alguns autores [22] consideram os ataques por `bots` de Ataques de Negação de Serviço (*Distributed Denial of Service* - DDoS) como ataques de Flash Crowds. Neste trabalho os flash crowds são considerados como eventos repentinos nos quais milhares de usuários legítimos intentam acessar a um servidor ao mesmo tempo, causando a caída o negação de serviço devido a que o servidor não consegue responder as múltiplas solicitações.

de detecção de anomalia. Por outro lado, como maior desvantagem, não é possível a detecção de ataques que não estejam na base de dados. Os atacantes chegam a criar novos ataques fazendo uma pequena variação que já seja suficiente para não ser considerada uma assinatura conhecida. Portanto, os sistemas IDS baseados unicamente nessa técnica, como é o caso da ferramenta Snort, são vulneráveis a novas ameaças.

2.2.2 Classificação Segundo o Método de Aquisição dos Dados

Os sistemas IDPS podem ser classificados dependendo da forma na qual monitoram os recursos como é mostrado na Figura 2.4. Pode ser feita uma analogia com o monitoramento de um prédio de uma empresa. Se o perímetro do prédio é o que se deseja monitorar, para inspecionar se alguém tenta entrar, é usado um Sistema de Detecção de Intrusão baseado em Rede ou *Network-based Intrusion Detection System (NIDS)*, como ilustrado na Figura 2.4(a). Mais especificamente se deseja monitorar uma sala em especial, por exemplo, a sala do chefe, com os artigos e todas as informações que ela contenha, é usado um Sistema de Detecção de Intrusão baseado de Estação ou *Host-based Intrusion Detection System (HIDS)*, como ilustra a Figura 2.4(b).



(a) Sistema de Detecção de Intrusão em Rede (NIDS).

(b) Sistema de Detecção de Intrusão em Estação (HIDS).

Figura 2.4: Sistemas de Detecção de Intrusão baseados a) em Rede (NIDS), colocado no perímetro da rede, e b) em Estação (HIDS), colocado em cada acesso a estação.

Os Sistemas de Detecção de Intrusão baseados em Estação ou HIDS, monitoram características especificas em um dispositivo específico, tais como arquivos, registros, *logs*, etc. Um HIDS inspeciona a entrada e a saída de pacotes em um dispositivos e ao mesmo tempo são capazes de detectar tentativas repetidas de acesso ou mudanças nos arquivos ou *software* do sistema. Normalmente, os HIDS estão situados nos

servidores ou dispositivos mais fundamentais que se deseja monitorar. Uma grande vantagem dos Sistemas de Detecção baseados em Estação é que eles podem monitorar atividades nas quais foi usada uma comunicação criptografada fim a fim. A análise, nestes casos, tem que ser feita no dispositivo final depois da decriptação da comunicação. Como desvantagem, os HIDS são difíceis de gerenciar em uma rede com muitas estações. Além disso, os HIDS consumem recursos do sistema afetando o desempenho da estação [7]. A maioria dos HIDS tem um *software* de detecção conhecido como "Agente". Cada agente monitora a atividade em um hospedeiro (*host*) único e se tiver capacidade de prevenção, podem reagir ao ataque. Os agentes transmitem a informação detectada ao gerenciador que administra todas as informações. A localização dos agentes na topologia da rede é trivial, pois os agentes devem estar localizados nos hospedeiros (*host*) existentes. Assim, a comunicação entre os agentes e o gerenciador não precisa de uma rede especial de gerenciamento, normalmente se usa a mesma rede não qual são colocados. Os agentes normalmente são colocados em servidores críticos. Normalmente, para não consumir uma grande quantidade de recursos, os HIDS realizam as inspeções periodicamente, com períodos de horas ou dias, o que significa um grande retardo em relação à atividade de ataque que pode estar acontecendo. Por outro lado, se o sistema inspeciona em tempo real, ele adquire a capacidade de prevenção, tais como filtrar do tráfego de rede mal intencionado que procura chegar ao hospedeiro, ou ainda alterando e reparando arquivos críticos do sistema, etc.

Os Sistemas de Detecção de Intrusão baseada em Rede (*Host-based Intrusion Detection System* - HIDS) monitoram tráfego de rede, analisando os protocolos de comunicação para identificar atividade maliciosa. Normalmente, esses sistemas adquirem os pacotes de rede por meio de um *sniffer*, como a biblioteca `libpcap`[2] e examinam pacotes individuais em tempo real ou quase real, que trafegam na rede. Com os NIDS é possível inspecionar o conteúdo ou carga útil dos pacotes.

Os Sistemas de Análise de Comportamento de Rede (*Network Behavior Analysis* - NBA) *systems* examinam o tráfego ou as estatísticas dos sistema na procura de tráfego não usual ou violações das políticas de segurança do sistema. Alguns autores [7] consideram esses sistemas uma outra metodologia de detecção de intrusão, mas nesta dissertação a NBA é incluída dentro do método de aquisição de dados NIDS.

Um sistema NIDS tem dois componentes: servidores, que gerenciam a rede, e sensores, que são os elementos mais importantes dos NIDS, porque monitoram diversos segmentos da rede. Esses sensores são implementados em dois modos diferentes: sensores em linha ou ativos Figura 2.5(a) e sensores passivos, como é mostrado a

[2]A biblioteca `libpcap` fornece funções para captura de pacotes no nível de usuário. http://www.tcpdump.org/

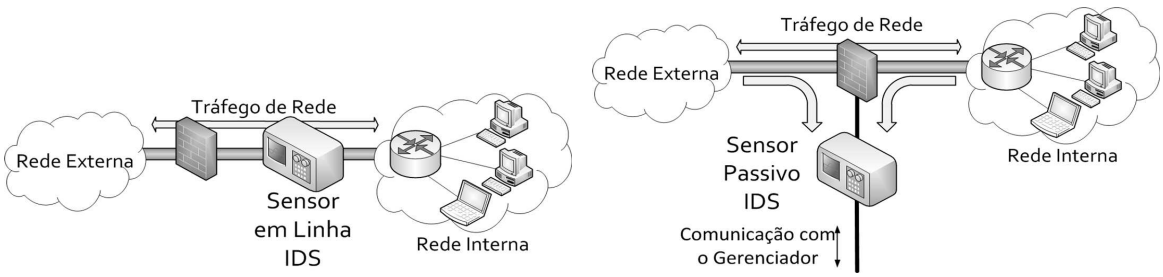
Figura 2.5.

Os sensores em linha ou ativos são colocados em diferentes segmentos da rede e analisam o tráfego que passa através deles. Esses sensores podem atuar conjuntamente com outros elementos de rede como, por exemplo, comutadores ou *firewalls*.

Os sensores passivos são colocados na rede de forma a obter uma cópia do tráfego, ou seja, o tráfego real não precisa atravessar o sensor. A obtenção da cópia do tráfego pode ser feita de três maneiras diferentes: por porta espelhada, por TAP e por balanceador de carga. A cópia por uma porta espelhada é obtida quando o sensor de detecção de intrusão está conectado a um comutador e este comutador usa a facilidade da porta espelho (*Switched Port Analyzer* – SPAN), a qual pode ser configurada para inspecionar e copiar todo o tráfego que circula pelo comutador. Os TAP (*Network TAP*) são dispositivos que estão localizados em um ponto da rede e copiam o tráfego que circula por eles, enviando esse tráfego até os sensores.

Os balanceadores de carga são dispositivos que enviam o tráfego agregado diretamente da rede aos sensores IDPS. Normalmente são estabelecidas regras no dispositivo balanceador para distribuir o tráfego especifico em diferentes sensores.

Em relação ao desempenho da rede, os sensores passivos são os mais eficientes que os sensores em linha, pois a inspeção e análise do pacote não adicionam um passo extra no caminho do pacote, o que significa uma menor contribuição no retardo. No entanto, os sensores em linha são mais efetivos o para prevenir os ataques, se o sistema tem a capacidade de inspecionar, analisar e reagir rapidamente enquanto o tráfego está passando por ele.



(a) Sensores em linha ou ativos.

(b) Sensores passivos.

Figura 2.5: Exemplo de localização dos sensores NIDS a) em linha e b) passivos.

Algumas desvantagens dos NIDS é que em redes de grande porte, é custoso e difícil o processamento de todos os pacotes da rede. Durante períodos de grandes tráfegos esses sistemas podem falhar no reconhecimento de ataques. Além disso, os NIDS não podem analisar informação criptografada.

A Detecção de Intrusão Sem fio monitora o tráfego de uma rede sem fio (*wireless*) e analisam os protocolos sem fio a fim de identificar qualquer atividade suspeita. Não é possível para esses sistemas identificar atividade suspeita nas camadas superiores, de aplicação ou transporte. A maioria dos sistemas utiliza uma combinação dos

diferentes métodos de aquisição de dados, levando em conta todo o espectro possível. Assim os sistemas IDPS que utilizam essas combinações, são chamados de IDS híbridos.

2.2.3 Classificação Segundo a Estrutura de Localização dos Sensores

Os Sistemas de Detecção de Intrusão normalmente são classificados pela forma na qual seus componentes são localizados e atuam. Assim, eles são classificados em Sistemas de Detecção de Intrusão Centralizados ou Distribuídos [23]. Os Sistemas de Detecção de Intrusão Centralizados são aqueles onde a análise dos dados é realizada em uma única posição, independentemente da quantidade de dispositivos monitorados. Portanto, a análise é centralizada, mas a aquisição pode ser distribuída. Por outro lado, os Sistemas de Detecção de Intrusão Distribuídos são aqueles no quais a análise dos dados é realizada em diversos dispositivos, sendo em um número proporcional à quantidade de dispositivos sendo monitorados.

2.2.4 Classificação Segundo o Comportamento Depois do Ataque

O comportamento dos Sistemas de Detecção de Intrusão faz referência à forma de reagir imediatamente depois da detecção do ataque. Assim os Sistemas de Detecção de Intrusão de Modo Ativo tem a característica de atuar em forma de resposta ao ataque. Esses sistemas são também conhecidos como Sistemas de Detecção e Prevenção de Intrusão, já que tomam uma medida de forma de evitar que o ataque continue. Os IDPS normalmente são configurados para reagir automaticamente, como por exemplo, ao bloqueio de um ataque em tempo real. Esses sistemas têm como grande vantagem a resposta ao sistema em tempo real. Portanto, é importante ressaltar, que para reagir a uma ação de ataque em tempo real, os sensores em linha ou ativos têm que ser capazes de controlar (filtrar, bloquear ou descartar) o tráfego que passa por eles. Os Sistemas de Detecção de Intrusão de Modo Passivo são configurados unicamente para funcionar como monitoramento, alertando mediante notificações, geralmente através de interfaces gráficas. Essas notificações são alertas de potenciais vulnerabilidades e ataques. Os IDS de modo passivo não são capazes de realizar nenhuma correção ou contramedida.

2.2.5 Classificação Segundo o Momento das Análises

Quanto ao momento das análises os sistemas IDPS são classificados em dois grandes grupos: em tempo real e em tempo diferenciado. Em tempo real, também

denominado IDSP *on-line*, os quais detectam e reagem à ameaça de maneira imediata, em tempo real ou tempo quase real. Esses sistemas operam continuamente obtendo informação das fontes e analisando os dados enquanto o sistema está em operação. Assim, quando ocorre a detecção de uma ameaça, uma contramedida é tomada interrompendo o processo que foi detectado. Por outro lado, os IDPS de tempo diferenciado efetuam as análises posterior ao ataque ou *off-line*. Dentro deste grupo existe uma metodologia chamada análise forense, na qual mediante diversas técnicas procura-se recompilar informação sobre o ataque, o perfil do atacante e os passos seguidos durante o incidente. Com esta informação é possível recriar o ataque e analisar as vulnerabilidades do sistema, erros de configuração, etc.

2.2.6 Tipos de Ataques Detectados Pelos IDPS

Existe uma ampla gama de ataques que os Sistemas de Detecção e Prevenção de Intrusão conseguem identificar e entre eles podem ser citados:

- exploração da camada de aplicação: análises de protocolos da camada de aplicação entre eles DHCP, DNS, Finger, FTP, HTTP, IMAP, IRC, etc. Entre os ataques se encontram *buffer overflow*, ataques de sequência de caracteres, adivinhação de senha, transmissão de *malware*, etc;

- exploração da camada de transporte: os protocolos analisados são TCP, UDP. Os ataques conhecidos são *scanner* de portas, fragmentação não usual de pacotes, ataques de inundação;

- exploração da camada de rede: análises de IP, ataques conhecidos como IP *spoofing*, homem no meio;

- exploração da camada de enlace: análises dos protocolos ARP, STP, ataque conhecido envenenamento de tabelas ARP, Negação de Serviço.

As maiores quantidades de ataques verificados são na camada de aplicação, mas dependendo da ferramenta é possível analisar todas as camadas. Os Sistemas de Detecção e Prevenção de Intrusão podem monitorar os pacotes de negociação inicial de estabelecimento de conexões criptografadas. Assim, é possível identificar uma vulnerabilidade ou configuração errada. Estas análises podem incluir protocolos de camada de aplicação como *Secure Shell SSH* ou *Secure Sockets Layer SSL* e em uma VPN protocolos tais de camada de rede tais como IPsec. Assim, com ferramentas tais como o Bro é possível programar políticas para detectar ataques, como será visto no capítulo 3, um exemplo é uma falha ou *bug* na ferramenta OpenSSL conhecido como *heartbleed attack* o qual foi detectado com a ferramenta Bro[3].

[3]www.bro.org/heartbleed

2.2.7 Complementação com Outras Ferramentas de Segurança

Normalmente os Sistemas de Detecção e Prevenção de Intrusão não funcionam como único dispositivo de segurança. Os IDPS fazem parte de um conjunto de ferramentas necessárias para prover segurança. A Ferramenta de Software de Análises Forense, também conhecida como *Network Forensic Analysis Tool (NFAT)*, coleta todo o tráfego que circula pela rede. O objetivo desta ferramenta é realizar análises para reconstituir as sessões de usuário anteriores a um ataque. Outro objetivo é criar um perfil do atacante. Essa ferramenta pode se complementar com os IDPS lendo os `logs` criados pelos IDPS. Como os NFAT não possuem capacidades de Prevenção, elas podem interagir com os IDPS no modo de prevenção. Como foi dito anteriormente os IDPS tem a capacidade de implementar regras nos *firewalls* ou comutadores como parte de ação sobre os fluxos detectados como ataques. Além disso, os *firewalls* geram `logs` com as conexões ou tentativas de conexões bloqueadas. Além disso, como é conhecido os *Firewalls* não têm a possibilidade de inspecionar pacotes, já que a maioria dos *firewalls* são baseados nas tuplas porta-endereço origem-destino para bloquear o tráfego. Assim, com a inserção dos IDS é possível para o *firewall* ter maior inteligência no bloqueio de ataques.

Hoje, assiste-se a uma nova tendência de associar e integrar os conceitos de *firewall*, detecção de intrusão e inspeção profunda de pacotes (DPI) em um único produto. Este produto integrado denomina-se *Firewalls* de Geração Futura ou *Next Generation Firewall (NGFW)* que basicamente consiste em integrar todas as técnicas de segurança em um único dispositivo, reduzindo, por exemplo, o alto custo de gerenciamento de cada uma das ferramentas. Os potes de mel (*honeypots*) são *softwares* que são executados em um computador ou em um conjunto deles para atrair atacantes [24]. Normalmente esses dispositivos tentam simular falhas ou vulnerabilidades na segurança dos sistemas. O objetivo básico desta ferramenta é colher as informações do atacante, analisar seu comportamento e técnicas durante e depois de um ataque. A intenção, quando são usados em tempo real, é distrair ao atacante enquanto, se isolam os servidores importantes. Contudo, alguns *honeypots* estão limitados a simular o comportamento de um sistema operacional inexistente, sendo utilizados como uma medida de segurança de baixa intensidade, e são conhecidos por *honeypots* de baixa interação. No entanto, existe outro grupo de *honeypots* os quais funcionam com sistemas operacionais reais, respondendo aos comando do atacante. Esse grupo é conhecido como *honeypots* de alta interação e fundamentalmente são utilizados com fins de pesquisa. Os *honeypots* podem obter ajuda dos IDPS na detecção do intruso na tentativa de penetração no sistema, e assim, desviar o tráfego do atacante ao servidor *honeypots* ou a rede de servidores *honeypots*. Enquanto isso,

depois das análises e da criação do perfil do atacante, o *honeypot* pode enviar esta informação ao IDPS para posteriores detecções sem utilizar recursos de sistema.

2.3 Ataques Contra os Sistemas de Detecção de Intrusão

Os Sistemas de Detecção e Prevenção de Intrusão (IDPS) têm como objetivo detectar e prevenir ataques contra um dispositivo ou um conjunto deles. Por outro lado, existem ataques aos sistemas de defesa, por exemplo, ataques com a intenção de destruir os sistemas IDPS, ou seja, com a intenção de pará-los. Os ataques ao IDPS são conhecidos como ataques de falha (*crash attacks*), e tentam desabilitar o IDPS causando uma falha nele [25]. Uma classe de ataques contra os IDPS, conhecido como ataques de inserção (*insertion attacks*), tem como objetivo confundir ao IDPS. O atacante força ao IDPS a ler pacotes inválidos, com campos do pacote alterados como IP origem e destino, fazendo-o perder tempo e recursos, enquanto o agressor ataca outros dispositivos. Os ataques de evasão consideram que os dispositivos finais podem aceitar pacotes que os IDPS consideram legítimos. Essa condição pode ser explorada por um atacante, no qual a informação é passada através de pacotes com uma sequência de caracteres muito semelhante. Assim se o atacante envia uma sequência de caracteres com uma mínima diferença do que o IDPS tem na sua base de dados, o IDPS deixará passar o pacote até o seu destino final. Outra forma de realizar um ataque de evasão é o atacante enviar um tráfego encriptado, já que para o IDPS é impossível inspecionar este tipo de tráfego.

Entre os ataques de evasão[26] contra os IDS baseados em redes, NIDS, se encontram a ofuscação que é o processo de manipular os dados de tal maneira que o detector por assinatura não faça combinação com o pacote manipulado. Um exemplo é enviar um pacote inserindo caracteres nulos. O texto original seria `"./../c:\winnt\system32\netstat.exe"`, mas na ofuscação o texto enviado seria `"%2e%2e%2f%2e%2e%2fc:\winnt\system32\netstat.exe"` o qual não seria identificado pelo NIDS, mas seria identificado e processado da mesma maneira por um servidor HTTP. Outro ataque conhecido é a fragmentação, que simplesmente divide um pacote em múltiplos pacotes. Os NIDS mais comuns não analisam pacotes fragmentados. Assim, se é enviado um pacote com as partes fragmentadas, o NIDS só analisa pacote por pacote e não consegue analisar o conteúdo da mensagem toda. Portanto, embora cada pacote individualmente não representa nenhum risco, mas o conteúdo total faz parte da assinatura de ataque que não vai ser detectada. Existem alguns NIDS atuais conseguem fazer a remontagem dos pacotes fragmentados e evitar este tipo de ataque. No entanto, este ataque tem uma variação no qual o

atacante espera encher o buffer do NIDS enviado o último pacote em um intervalo de tempo posterior. Os primeiros pacotes do ataque serão enviados ao destinatário no momento que o último pacote está chegando ao NIDS, em um intervalo de tempo posterior, o destinatário receberá todos os pacotes e na remontagem será vítima de um ataque. Os NIDS tem que analisar pacote a pacote para ser efetivos contra um ataque. Quando são estabelecidos túneis com protocolos como SSL, SSH e IPSec, é impossível para o NIDS inspecionar o conteúdo dos pacotes. Os atuais NIDS alertam ao gerenciador da rede quando vem uma conexão encriptada. Além disso, os NIDS conseguem verificar o estado dos certificados trocados durante a conexão criptografada. Outra das formas de ataques contra os IDPS é o ataque de Negação de Serviço (*Denial of Service - DoS*). O atacante lança uma grande quantidade de pacotes com a intenção de estourar os recursos do IDPS, comprometendo a habilidade do IDPS de monitorar todo o tráfego ou quebrando-o definitivamente. Este é um grande problema para os NIDS e será tratado com maior profundidade no Capítulo 4.

2.4 Sistemas de Detecção e Prevenção Populares

Os IDPS têm atraído a atenção de diversas empresas e pesquisadores. Devido a isso, os Sistemas de Detecção e Prevenção de Intrusão podem ser divididos em duas seções: IDPS comerciais e IDPS de código aberto (*Open Source*).

2.4.1 IDPS Comerciais

Os IDPS comerciais podem ser uma ferramenta *hardware* ou unicamente baseados em ısoftware. Existem diversas companhia desenvolvendo produtos nesta área e algumas estão citadas a seguir.

IBM RealSecure [27] monitora e detecta intrusão num servidor, além disso, tem um módulos sensores extras para a prevenção de intrusões e bloqueio de pacotes. IBM System Virtual Appliance [28] é uma ferramenta que pode ser usada com a anterior para ambientes virtuais, sendo possível detectar ataques dentro de uma nuvem virtual assim como a infraestrutura física. Uma importante característica deste equipamento é que só pode ser usado de maneira centralizada num ambiente virtual.

A Cisco [29] tem um módulo de detecção de anomalias para ataques de DDoS massivos. Essa ferramenta emprega um modelo de implantação "on-demand", desviando e depurando apenas o tráfego dirigido a dispositivos específicos sem afetar outros tipos de tráfego. As múltiplas camadas integradas de defesa dentro do módulo *Anomaly Guard* habilita a identificação e o bloqueio do tráfego malicioso do

ataque, permitindo as comunicações legítimas aos seus destinos originais.

A empresa Radware comercializa um equipamento IDPS [30] para a prevenção de ataques de DoS em Redes Definidas por Software. O equipamento implementa um controlador de código proprietário que gerencia uma rede de comutadores OpenFlow e utiliza lógica nebulosa para determinar o grau do ataque, baseado no comportamento da taxas de pacotes, transmissão e abertura de conexões, além do comportamento dos protocolos, de acordo com padrões previamente obtidos. Com esses valores é possível diminuir a alta taxa de falsos positivos obtidos pela lógica nebulosa. A ferramenta precisa de um período de aprendizagem do comportamento benigno para obter a base de dados para o treinamento.

A empresa StoneSoft [31] do grupo McAffe, tem desenvolvido uma ferramenta de IDPS por *software*, a qual pode ser usada em ambientes virtuais. Entre algumas das suas características o produto é otimizado para evitar ataques de evasão e se integra com outras ferramentas de segurança baseadas em *hardware* ou *software* de distintos fornecedores.

2.4.2 IDPS de Código Aberto ou *Open Source*

Os IDPS de código aberto mais conhecidos são o Snort [32], o Suricata[4], e o Bro [33]. Os dois primeiros, Snort e Suricata, são ferramentas orientadas para o uso comercial, e o sistema Bro é orientado para pesquisa, sendo que os desenvolvedores do Bro, recentemente criaram uma companhia para dar suporte ao sistema [5]. Todos estes sistemas são ferramentas de Detecção de Intrusão baseada em rede (NIDS) de código aberto, mas todos variam suas características de monitoramento, detecção, etc. Nesta seção serão analisados alguns dos Sistemas de Detecção de Intrusão de código abertos.

O Sistema Snort IDS

Snort é uma ferramenta de Detecção e Prevenção de Intrusão baseada em rede. Realiza o monitoramento e a inspeção no nível de pacotes. Esses pacotes são examinados em busca de assinaturas de ataques ou anomalias nos protocolos. Uma das principais vantagens do Snort é a capacidade de fazer busca de sequência de caracteres e um código simples para especificar e escrever as sequências. Todas as assinaturas escritas, chamadas no Snort de regras, são avaliadas por uma grande comunidade. O Snort pode ser configurado para funcionar em três modos diferentes:

1. Modo *Sniffer*: no qual simplesmente lê os pacotes da rede e os mostra como um fluxo contínuo no terminal de saída. O modo *Sniffer* do Snort é semelhante

[4]http://www.openinfosecfoundation.org/
[5]http://www.broala.com/

à ferramenta de análise de tráfego `tcpdump`;

2. Modo Armazenado de pacotes: neste modo o Snort lê os pacotes da rede e armazena eles num arquivo, com formato `pcap` no disco;

3. Modo Sistema de Detecção de Intrusão de Rede: esse modo permite ao Snort analisar os pacotes buscando assinaturas por sequência de caracteres.

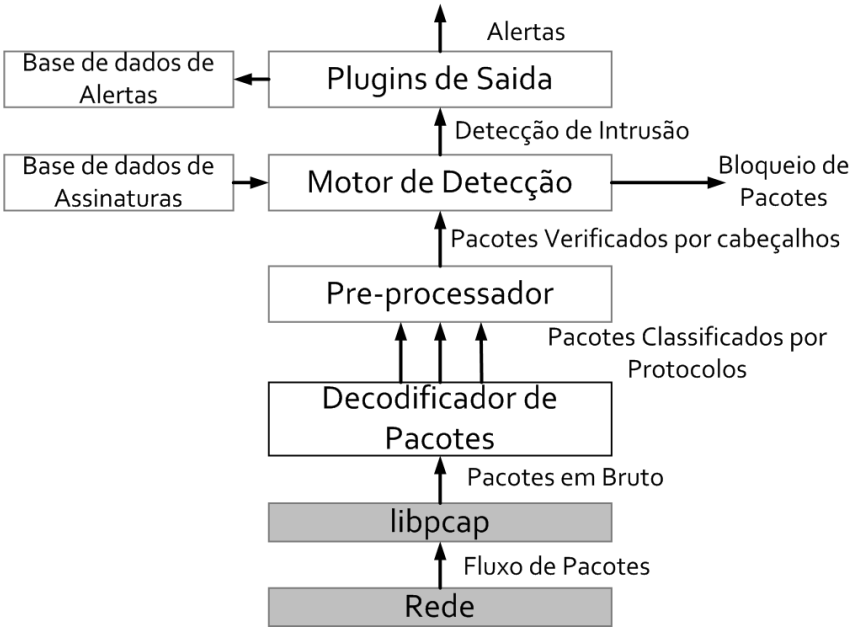


Figura 2.6: Arquitetura do sistema Snort.

A arquitetura Snort é modular e consiste em um decodificador ou classificador de pacotes, um pré-processador, um motor de detecção e um gerador de alerta também denominado *plug-ins* de saída, como mostrado na Figura 2.6. O Snort não possui um *Sniffer* nativo, por esta razão utiliza a `libpcap` para capturar os pacotes nos dispositivos de rede. O decodificador, depois de receber os pacotes em bruto da `libpcap`, decodifica-os e classifica-os, dependo do protocolo, em elementos da camada de enlace, camada de rede e camada de transporte. Esses pacotes são armazenados numa estrutura de dados sendo pronto para o processamento. No pré-processador são chamadas distintas funções para verificar se existe alguma mal formação no pacote ou se ele é normal. Se o sensor Snort é instalado como ativo e se o pacote estiver mal formado, esse módulo pode ser configurado para descartar o pacote. No pré-processador são feitos outras ações como montagem de fluxos TCP, fragmentação de pacotes, etc. O motor de detecção é o núcleo do Snort, nele são aplicadas as regras das assinaturas dos ataques aos pacotes recebidos do pré-processador e as saída desde módulo são enviadas no formato de alertas aos plug-ins. Os plug-ins podem ter diferentes formatos de saída como XML, CSV, etc.

Uma das vantagens do Snort é a facilidade na criação de regras. O Snort utiliza uma linguagem de descrição simples e fácil de usar. As regras são divididas em duas seções lógicas, cabeçalho e opções. O cabeçalho contém a ação da regra, por exemplo, alerta, *drop*, gravar, o protocolo, IP e portas de origem e destino e máscara de rede, no formato CIDR. As opções das regras contêm a mensagem de alerta e a informação em qual parte do pacote deve ser inspecionado, por exemplo, no conteúdo do pacote, a carga, os *flags* ou cabeçalho, para determinar que ação a tomar.

A regra que se segue `alert tcp any any -> 192.168.1.0/24 111 (content:"|00 01 86 a5|"; msg:"mountd access";flags:A+;)`
é uma regra gerada pelo Snort quando pega um pacote com os seguintes atributos.

Cabeçalho da Regra:

- Protocolo: TCP.
- Fonte: qualquer endereço IP com qualquer porta.
- Destino: qualquer endereço IP dentro da rede 192.168.1.0/24 porta 111.

Opções da Regra:

- Conteúdo: sequência de caracteres no pacote, podem existir mais de um *content* significando que existe mais de uma sequência de caractere a encontrar.
- Mensagem: a mensagem que será impressa na alerta.
- Flags: ACK flag está estabelecido.

Outra vantagem do Snort é o seu funcionamento "ligar e usar" ou *Plug and Play* no qual a ferramenta começa a funcionar sem ter que estabelecer muitas configurações pelo usuário. No entanto, como desvantagem da ferramenta Snort é a utilização da detecção por assinatura. É dizer que só é possível identificar os ataques que estejam estabelecidos no conjunto de regras, sendo impossível detectar novos ataques ou violações das politicas de segurança.

O Sistema Suricata IDS

Suricata é um NIDS de código aberto, que foi oficialmente lançado em julho de 2010 e foi desenvolvido e atualmente é mantido pela *Open Information Security Foundation*. O objetivo do Suricata é ter tecnologia avançada no IDS, por isso, a diferença dos demais IDS de código aberto, utiliza uma tecnologia de processamento *multithread* para ter benefício dos múltiplos núcleos de um computador. Além disso, para ter um melhor desempenho, utiliza *hardware* de aceleração. A empresa NVIDIA tem uma parceria com Suricata, e estão desenvolvendo o CUDA,

uma biblioteca de computação paralela para a utilização de computação em Unidades de Processamento Gráfico ou *Graphic Processor Unit* (GPU). O Suricata realiza detecção de intrusão baseadas em anomalia e em assinaturas. As assinaturas desenvolvidas para o Snort podem ser utilizadas diretamente ou otimizadas para o uso do motor de detecção do Suricata. As anomalias dos protocolos são fornecidas pelos pré-processadores do Suricata, e quando é implementado em modo ativo, ele pode ser usado com a modalidade de Prevenção.

Apesar de que todo o código de Suricata ser original, os desenvolvedores não hesitam em afirmar que a arquitetura foi "copiada" do Snort. Inclusive, na página web do Suricata eles reconhecem que o Snort são as "raízes coletivas" para o Suricata. Por isso o esquema da Figura 2.7 representa a mesma arquitetura do Snort, mas utilizando a vantagem da arquitetura *multi-thread* do Suricata. Da mesma maneira do que o Snort, o Suricata não possui um *Sniffer* nativo, assim, é utilizado a `libpcap` para a captura dos pacotes na rede.

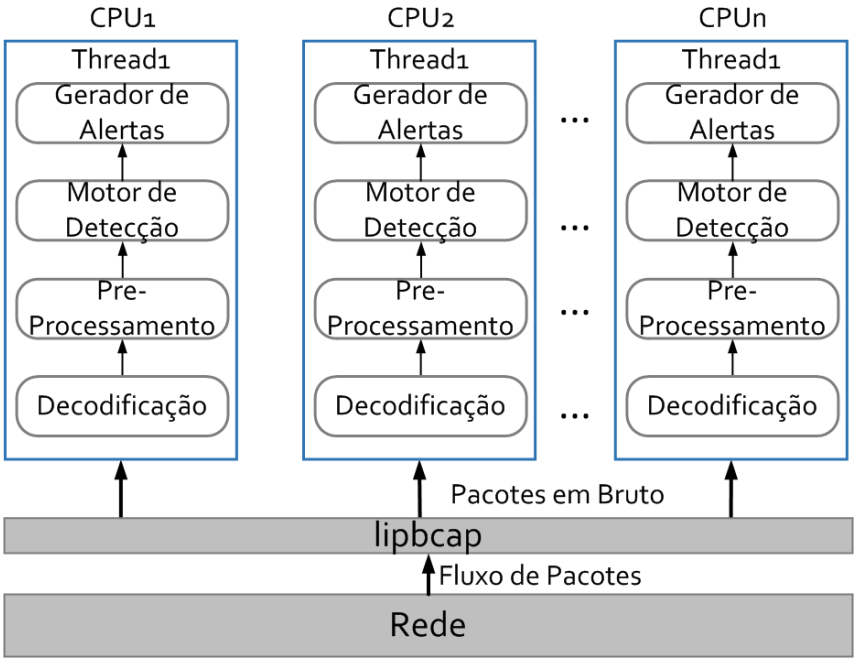


Figura 2.7: Arquitetura *Multi Thread* do Sistema Suricata.

Como vantagem da ferramenta Suricata é o processamento de informação mediante aceleração por *hardware* e a utilização do paralelismo entre núcleos de um computador chamado *multi-thread*. Como desvantagem presenta a detecção por assinatura ao igual que a ferramenta Snort como foi dito anteriormente.

O Sistema Bro

O Vern Paxson descreveu o Bro [33] como um sistema monitor de rede independente ou *standalone* para inspecionar tráfego em tempo real em um enlace de

rede. Atualmente na Versão 2.2, o Bro tem sido desenvolvido por mais de 15 anos por Paxson que ainda segue liderando o grupo de pesquisa no *International Computer Science Institute* (ICSI), na Universidade de Berkeley e no *National Center for Supercomputing Applications* na universidade Urbana-Champaign.

O Bro pode ser configurado para funcionar como NIDS, inspecionando pacote por pacote que trafega na rede. Ele vem configurado com algumas políticas pré-escritas, semelhante às regras do Snort, com licença BSD, permitindo o seu uso livre ainda com menores restrições que as regras com licença GPL v.2 do Snort e Suricata. É importante destacar que as políticas do Bro são escritas na sua própria linguagem, a qual é muito efetiva na descrição de regras de segurança, mas com uma complexidade maior que no Snort. Até a versão 1.5 existia um módulo, `snort2bro` escrito na linguagem `Bro`, que adaptava as regras ou assinaturas do Snort para ser usadas como políticas no Bro, contudo este módulo foi tirado da última versão. O Bro foi projetado para ser usado como uma arquitetura de único ou *Single-Thread*, mas já existe uma versão para utilização com múltiplos núcleos ou *Multi-Thread*.

O Bro foi escolhido neste trabalho pela sua flexibilidade na descrição das políticas de segurança. No Capítulo 3 é mostrada a linguagem descritiva das politicas, as quais permitem a implementação de diferentes algoritmos ou padrões conhecidos de ataques, chamados de assinaturas, para realizar a detecção de diferentes tipos de ataques. Além disso, o Bro presenta de monitoramento distribuído, o qual mediante a comunicação dos eventos dos distintos sensores é possível fazer uma detecção distribuída.

Comparação das Ferramentas IDPS de Código Aberto

A Tabela 2.1 compara as características mais importantes das três ferramentas mencionadas nesta seção, Snort, Suricata e Bro. Entre as comparações mais importantes, por exemplo, é que o Snort não possui identificador automático de protocolo, essa já característica foi resolvida no Suricata, e no Bro é nativo já que ele possui diferentes analisadores de protocolos como será visto na próxima seção. Também é importante destacar que a única ferramenta com aceleração por hardware nativa é o Suricata. Como características mais importante do Bro é destacado a sua linguagem para escrever políticas algo que o diferença de todos os outros e que ele é otimizado para o funcionamento em redes de alta velocidade, mediante a configuração de *cluster*, o que permite a análises em paralelo. O Snort possui como vantagem as análises de múltiplos arquivos pcap em simultâneo e a possibilidade de trabalhar nativamente como IPS descartando pacotes. O sistema de Monitoramento do Bro, não possui a caraterística "ligar e usar" como é o casso do Snort. Os mesmos autores afirmam que depois da instalação o Bro tem um comportamento de políticas neutrais [13], é dizer que ele não gerará alertas por a sua conta. Todas as alertas serão geradas de-

pois da descrição das políticas pelo usuário, as quais representam o comportamento bom ou malicioso do sistema. Isto pode representar uma desvantagem no momento da implementação da ferramenta, mas representa uma grande vantagem na flexibilidade de detecção, já que podem ser aplicados diversos algoritmos de anomalias ou assinatura.

Tabela 2.1: Comparação dos IDS de código aberto. Fonte [34].

Características	**Snort**	**Suricata**	**Bro**
Processamento Multi-Thread	Não	Sim	Não
Detecção Automática do Protocolo	Não	Sim	Sim
Aceleração por GPU	Não	Sim	Não
IPS Nativo	Snort_inline ou compilado com opção -Q	Opcional mediante nfqueue	Execução de Programas Externos
Linguagem Própria	Não	Não	Sim
Licença	GNU GPL v.2	GNU GPL v.2	BSD
Análises Offline (Arquivos pcap)	Sim, múltiplos arquivos simultaneamente	Sim, um único arquivo	Sim, um único arquivo
Compatibilidade de Sistemas Operativos	Qualquer	Qualquer	Unix
Monitoramento em redes de Alta velocidade	Não	Não	Até 10 Gb/s

Capítulo 3

A Ferramenta Bro de Análise de Tráfego

O Bro[1] [33] é uma ferramenta de código aberto para o monitoramento e análise em tempo real do tráfego de rede. Essa ferramenta é baseada em um monitor de pacotes que escuta o tráfego em um ponto central da rede. O Bro permite seguir a comunicação entre dois pontos da rede, reconstruindo a semântica das conexões e armazenando os estados para cada uma delas.

3.1 Arquitetura do Bro

A ferramenta Bro foi desenvolvida para funcionar como um sistema autônomo de monitoramento de tráfego em tempo real com finalidade de segurança. O Bro permite análises de uma ou múltiplas interfaces ou também pode ser executado para analisar arquivos `pcap`. Entre os requisitos de projeto foram visados pelo Bro podem ser citados:

- eficiência em redes de alta velocidade, com grandes volumes de dados: as redes atuais conseguem atingir largura de banda praticamente a níveis de Gb/s e, portanto, um sistema de detecção de intrusão de rede (NIDS) deve ter a capacidade de inspecionar tráfego a essas taxas de transmissão em tempo real;

- perda de pacotes baixa ou nula: este requisito é importante do ponto de vista da segurança, pois a perda de pacotes é considerada gravíssima. Todo pacote pode ser um ataque potencial, ou uma identificação de uma intrusão;

- notificações em tempo real: um sistema de monitoramento em tempo real focado na segurança deve ser capaz de notificar o usuário de possível ataque

[1] O nome Bro tem origem na conotação de "Big Brother" uma vez que a sua finalidade é "inspecionar" o tráfego.

no momento que os ataques são efetuados, para que reações seja possível, além de armazenar registros ou *logs* para posterior análises;

- alta resistência a ataques ao sistema: no caso de um atacante descobrir que o sistema está sendo monitorado por algum tipo de sistema de detecção de intrusão (IDS), o próprio sistema de detecção de intrusão poderia passar a ser o alvo do atacante para comprometer ou desabilitar o monitoramento. Assim, o Bro tem que ser capaz de resistir ataques;

A arquitetura do Bro, representada na Figura 3.1, é dividida em três camadas principais: a captura dos pacotes; a abstração da atividade da rede em eventos; e a interpretação das políticas, na qual o usuário define os seus próprios *scripts*. O Bro possui mais de 30 analisadores de diferentes protocolos, tais como FTP, HTTP, ICMP, UDP, TCP, DNP3, entre outros, e as políticas de segurança estabelecem as ações que devem ser executadas pelo mecanismo de segurança. Logo, se uma atividade suspeita é observada, os analisadores disparam eventos de forma assíncrona que são enviados à camada superior, na qual são interpretados baseados nas políticas descritas pelos usuários numa linguagem específica, a linguagem `Bro`. Portanto, as políticas de segurança estão separadas da detecção de cada ameaça o que confere à ferramenta Bro uma enorme flexibilidade.

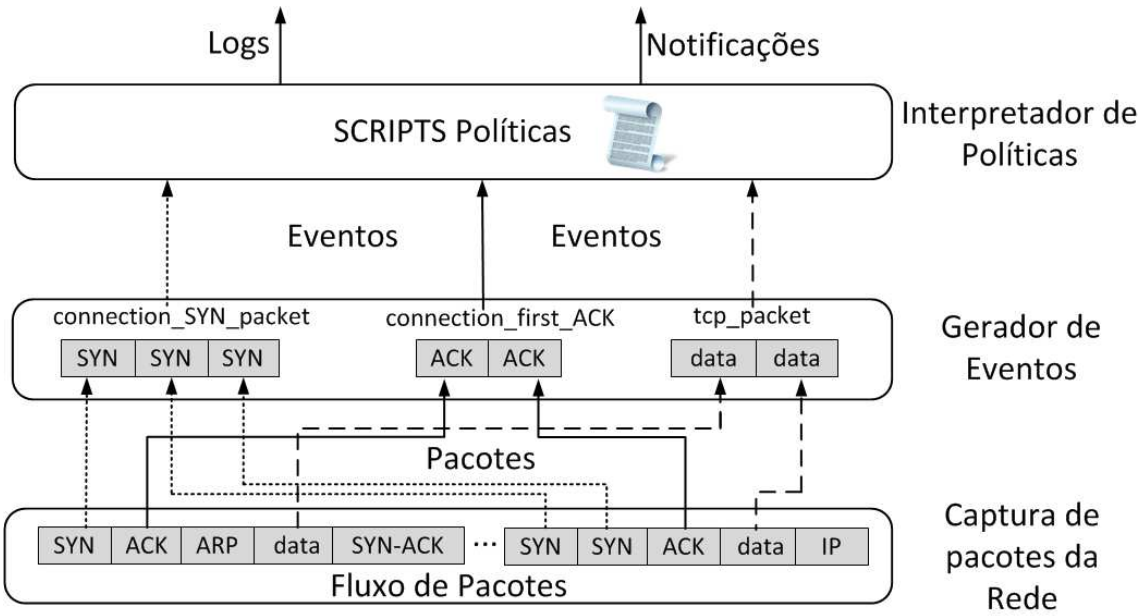


Figura 3.1: A arquitetura Bro de três camadas: captura dos pacotes na rede; gerador de eventos, onde é analisada a semântica dos pacotes; interpretação de políticas, na qual os eventos são tratados pelos *scripts* do usuário.

A captura dos pacotes no Bro é igual a qualquer outro analisador de tráfego como Snort, tcpdump, entre outros. A captura é feita pela biblioteca `libpcap` o que

permite ao Bro ficar independente das tecnologias das camadas de enlace. Assim, o *hardware* para a aquisição dos dados pode ser trocado independentemente do Bro. Outra vantagem na utilização da `libpcap` é no uso de expressões lógicas para filtrar tráfego, como por exemplo no tcpdumo para filtrar o trafego da fonte 10.0.2.4 que sejam com destino a porta 3389 ou 22 se usaria a expressão `tcpdump 'src 10.0.2.4 and (dst port 3389 or 22)'`. Assim, mediante as expressões e possível reduzir a quantidade de pacotes analisados. Esses filtros mediante expressões são modificados pelo Bro para a captura seletiva de pacotes.

3.1.1 Gerador de Eventos do Bro

O gerador de eventos do Bro define eventos para as atividades da rede, o que representa uma abstração dos pacotes ou de uma sequência de pacotes em um nível de informação mais elevado. As análises para o estabelecimento dos eventos é realizada pacote a pacote. Depois que o fluxo de pacotes é capturado, o gerador de eventos realiza a verificação de *checksum* para assegurar a integridade do pacote e checar se os cabeçalhos estão bem formados. No caso das verificações falharem, os pacotes são descartados e são gerados eventos indicando o problema encontrado. No caso de sucesso das verificações de integridade, o gerador de eventos procura o estado associado aos endereços IP e às portas de origem e destino do pacote capturado e, se não encontrar nenhum estado associado, cria um novo estado. Essa associação é feita mediante o gerenciador das conexões. Em seguida, os pacotes são ordenados por conexões, reagrupados por fluxos de dados TCP/UDP/ICMP e decodificados pelos protocolos da camada de aplicação.

O Bro mantém um arquivo *pcap* associado ao tráfego que está sendo inspecionado. O gerenciador das conexões indica ao gerador de eventos o que deve ser salvo no arquivo, se o pacote inteiro, quando o conteúdo do pacote está sendo examinado, ou se apenas o cabeçalho se o pacote trata os campos SYN/FIN/RST do TCP ou ainda se não armazena nada se não tiver nenhum processamento.

O Gerador de Eventos foi projetado em C++ para ter um melhor desempenho, já que deve analisar até milhões de pacotes por segundo. Assim, o gerador de evento é composto por diversos analisadores[2], cada um responsável por uma tarefa definida, como por exemplo a decodificação de diferentes protocolos ou a correspondência de assinaturas, etc. Na Figura 3.2 são mostrados os quatro componentes principais do gerador de eventos no Bro: o verificador, os analisadores de camada de transporte e de aplicação e a criação de eventos.

- O verificador: Neste componente são recebidos os pacotes e são feitas diferentes

[2]O termo analisadores não é só aplicado a aqueles componentes conteúdos no gerador de eventos, mas se refere a qualquer parte do Bro que realiza uma tarefa específica. Uma descrição mais detalhada dos analisadores se encontra em[13].

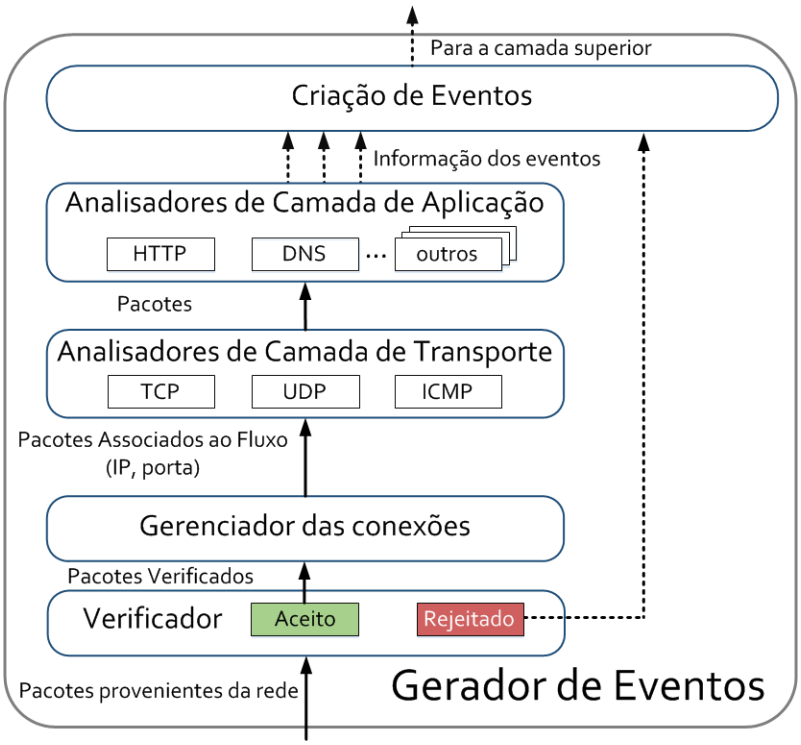


Figura 3.2: Os quatro componentes principais do gerador de eventos do Bro: a verificação, os analisadores de camada de transporte e de aplicação e a criação de eventos.

verificações para verificar se o pacote é bem formado, no caso que o pacote analisado não passe nas verificações, o pacote é descartado e a informação é envaida para a geração do evento;

- Gerenciador de conexões: o tipo de dado mais importante no Bro é denominado *connection*, que vai ser traduzido por conexão. Todo pacote pertence a uma conexão, e no caso dos pacotes dos protocolos que não são orientados a conexão, como UDP e ICMP, é usada uma abstração de fluxo, IP e porta origem e destino. Assim, neste módulo é estabelecido um estado associado a uma conexão para cada pacote que o Bro inspeciona;

- Analisador de camada de transporte: aqui são analisados os protocolos TCP, ICMP e UDP. No caso do TCP, como o analisador de aplicação necessita da sequencia dos conteúdos dos pacotes dos dois extremos, o analisador do TCP segue quase de maneira completa a máquina de estados do protocolo, seguindo as trilhas dos ACK, lidando com as retransmissões, etc. O resultado é um fluxo de bytes real de carga útil. Esta é uma diferença importante do Bro quando comparado com o Snort, já que este últimos não reconstitui um fluxo de bytes completo, utilizando heurísticas para combinar diversos pacotes em um "fluxo virtual" [13], é dizer que o Snort não é capaz de seguir a máquina de estados

dos protocolos nem reconstruir os fluxos pacote a pacote das conexões;

- Analisador de camada de aplicação: as análises realizadas dependem de cada serviço. Existem mais de 32 analisadores de protocolos de camada de aplicação, tais como, HTTP, DNS, SMTP, DNP3, FTP, entre outros. Além destes 32 protocolos também é possível criar os próprios analisadores de aplicação para um protocolo especifico.

Um exemplo da geração de eventos são as três trocas de mensagens da conexão do TCP, denominadas *three way handshake*, correspondente a um evento de estabelecimento de conexão bem sucedido (`connection_established`) com informações tais como o hospedeiro (*host*), IP, porta, tempo de início, entre outros, que são enviados à camada de políticas. A geração de um evento não implica nenhum tipo de "julgamento", se é um evento legítimo ou malicioso. Logo, é na camada de interpretação de políticas que vão ser criados os algoritmos, os limiares etc para decidir o evento é malicioso ou não.

3.1.2 A Camada de Interpretação de Políticas

Depois que o gerador de eventos termina de processar os pacotes, e gerar os eventos correspondentes, envia os eventos até a camada de interpretação de políticas. Os eventos são enviados de maneira assíncrona e são armazenados numa fila FIFO, denominada pilha de eventos. A camada de interpretação de políticas vai gerenciando estes eventos até esvaziar a fila de eventos. O processo completo pode ser apreciado na Figura 3.3, no qual é um exemplo de uma conexão web de um dispositivo até um servidor. O Detector Dinâmico de Protocolo é um mecanismo desenvolvido no Bro para detectar automaticamente o protocolo e as suas derivações com a chegada do primeiro pacote, criando uma árvore de analisadores, isto permite que a análises dos protocolos seja feita independente da porta na qual o pacote chega. Assim, após os eventos passarem pelos analisadores, eles passam a serem analisados como se fizessem parte de uma pilha de protocolos. Os gerenciadores de eventos, situados nos *scripts* de políticas Bro, vão usando estes eventos a medida que eles chegam, esvaziando a fila de eventos. Múltiplos eventos podem ser analisados ao mesmo tempo, e um mesmo evento pode ser executado mais de uma vez, como mostrado na Figura 3.3.

A camada de interpretação de políticas do Bro fornece flexibilidade ao sistema, pois as políticas de segurança são especificadas mediante *scripts* descritos na linguagem `Bro`, que permite a definição da forma de gerenciar os eventos gerados previamente. Portanto, fica clara a separação entre os mecanismos e as políticas, já que os *scripts*, também denominados gerenciadores de eventos ou *event handler*, permitem especificar a forma de reação aos eventos gerados previamente.

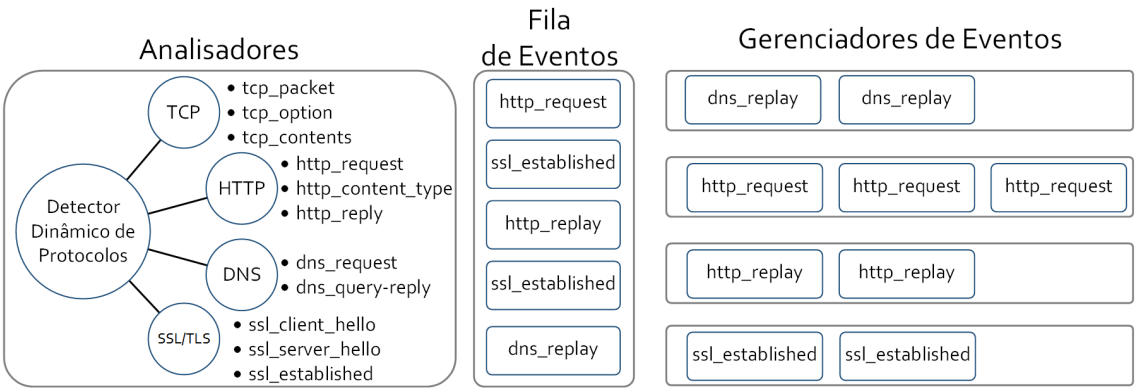


Figura 3.3: Exemplo do processo de criação e gerenciamento de eventos no Bro. Fonte: www.bro.org/bro-scripts

A definição de políticas depende do ambiente de segurança que, para o mesmo evento, pode exigir ações mais frouxas e liberais, como requer um ambiente universitário, ou mais rígido, quando se trata de setores sensíveis de uma empresa. Outra vantagem é a inspeção e análise posterior, pois toda vez que uma violação das políticas é detectada, um alerta é disparado, gerando notificações em tempo real ou `logs` para análise posterior. É interessante a observação de que a medida que a arquitetura é escalada, o fluxo de dados vai diminuindo. Assim é possível realizar um melhor processamento refinado aos pacotes.

Um ou mais *scripts* podem ser associados a um evento, e cada gerenciador de eventos pode ser chamado tantas vezes quantas o evento o evento é detectado na rede.

3.2 Programação de Políticas no Bro

O Bro tem a possibilidade de descrever e escrever políticas de segurança, como já ressaltado. Esta característica permite ao Bro ser um sistema muito eficiente e flexível, que se destaca quando é comparado com seus competidores, pois dependendo do ambiente é possível criar diferentes políticas. Apesar de ser muito utilizada, a documentação da ferramenta Bro não é completa e não está atualizada e ainda não existe uma documentação oficial de uso do Bro. Segue uma descrição da linguagem `Bro` e alguns exemplos de programação da ferramenta.

A Linguagem `Bro`.

O que diferencia o Bro dos demais analisadores de tráfego, como Snort, `tcpdump`, netFlow, entre outros, é a descrição de políticas na sua própria linguagem. A linguagem `Bro`, similar à linguagem C, foi projetada levando em conta o uso de redes.

Assim, existem tipos de dados específicos para endereços IP, portas, intervalos de tempo, entre outros tipos característicos do ambiente de redes. Quando comparado com outros monitores de redes, tais como o `tcpdump`, baseado em caracteres ASCII, os tipos de dados trazidos pela linguagem `Bro` facilitam o entendimento e evitam "erros simples" [33]. Ao fazer esses tipos de dados variáveis em Bro, aumenta a facilidade de expressão oferecida pela linguagem e assim passa a detectar erros, que poderiam ser facilmente feitos como, por exemplo, comparar uma porta do TCP usando um endereço IP. Dessa forma, com a linguagem específica do Bro, é possível escrever de maneira fácil e intuitiva, *scripts*, denominados políticas. Logo, com as políticas é possível programar detecções de intrusão por anomalias ou assinaturas. Na Tabela 3.1 são mostrados os tipos de dados que podem ser usados na linguagem `Bro`.

Os operadores da linguagem `Bro` são semelhantes aos operadores utilizados na linguagem C, com a diferença que eles são orientados para gerenciar sequência de caracteres (*strings*) ou para fazer combinação de sequência de caracteres, a Tabela 3.2 resume os operadores utilizados na linguagem `Bro`. Por outro lado, nesta linguagem existem laços ou *loops* com os quais é possível recorrer os diferentes vetores, tabelas ou *sets* criados. É importante destacar que o Bro funciona mediante a geração de eventos, deste modo, não é possível ter laços `while`, devido ao alto consumo nos recursos de processamento e no qual seria possível perder pacotes correspondentes a eventos importantes. Assim, é possível utilizar iterações com laços `for`, por exemplo `for (i in x)`, ou fazer ramificações do estilo `if (expr); else ;`. Também de um modo assíncrono é possível esperar que se ocorra um evento com `when (expr) ;`.

As variáveis na linguagem `Bro` podem ser de dois tipos: locais ou globais. As variáveis `locais` são usadas unicamente dentro de uma função enquanto as variáveis `globais` podem ser acessadas por qualquer outro *script*.

Políticas com Notificação de Alarmes

O Bro funciona para uma grande quantidade de *scripts*, que podem usar os analisadores disponibilizados pela própria ferramenta, como foi explicado na Seção 3.1.1, ou o usuário pode definir e programar os seus próprios *scripts*. O objetivos destes *scripts* é alertar, notificar ou registrar comportamentos suspeitos. Na terminologia do Bro estas ações são conhecidas como notificação ou no inglês *raise a notice*. Para isto o Bro possui um arcabouço de notificações chamado `NOTICE`. Nestas notificações é possível incluir o tipo de informação vai ser enviado ao usuário. Informações como tempo de conexão, endereços IP origem e destino, protocolos, ou até gerar uma mensagem definida pelo próprio usuário com as características que ele especificar. Essas notificações são agrupadas por as suas características, propriedades, protocolos, ambiente, ataque, entre outros, e mapeadas em ações predefinidas. Estas ações definidas são mostradas na Tabela 3.3. O exemplo a seguir é uma notificação

Tabela 3.1: Tipos de dados suportados na linguagem `Bro`. Fonte: www.bro.org/data-types-and-data-structures

Tipos de dados	**Descrição**	**Exemplo**
`bool`	Booleano	T/F
`int`	Inteiro com sinal de 64-bit	-12
`count`	Inteiro sem sinal de 64-bit (Não negativo)	85
`double`	Dupla precisão de ponto flutuante	104.69
`string`	Sequência de bytes	"hello"
`time`	Tempo absoluto formato UNIX	1320977325 (*epoch time*)
`interval`	Diferenças em intervalos de tempo	6 seg/min/hr/dias
`port`	Número de porta camada de transporte	22/tcp
`addr`	Endereço IP	v4: 127.0.0.1, v6: [fe80::db15]
`subnet`	Máscara de rede formato CIDR	10.0.0.0/8
`hostname`	Nome do dispositivo em DNS	guanabara
`file`	Gerenciamento de arquivos	open(f: string): file
`pattern`	Expressões Regulares	test_pattern = /quick/
`record`	Coleção de elementos de tipos arbitrários	record { a: addr, b: int}
Estruturas de dados		
`vector`	Coleção de objetos de um mesmo tipo de dados	v2 = vector(1, 2, 3, 4)
`table`	Tabelas, mapeamento de um valor ou um campo	X: table[string] of port
`set`	Conjunto de elementos de um mesmo tipo de dados	Ports = set(23/tcp, 80/tcp, 143/tcp, 25/tcp)

gerada por uma política, na qual é inserida a ação de enviar email na notificação, `Notice::ACTION_EMAIL`, cujo endereço(s) pode(m) ser definido(s) previamente. A ação será inserida mediante o comando `add n$actions[Notice::ACTION_EMAIL];`. Assim, é possível utilizar as ações apresentadas na Tabela 3.3, o `Notice::ACTION_LOG` armazenará a notificação em um `log` por padrão criado na rota de instalação do Bro, o `notice.log`. A última ação `Notice::ACTION_ALARM` gera um `log` como foi explicado previamente e além disso, gera uma alerta na qual por padrão é enviado no e-mail de estado uma vez por hora, ou pode ser envaido a um programa externo para gerar algum alerta visual, sonoro, etc.

Neste exemplo duas variáveis globais são definidas: `attempts`, que é uma tabela

Tabela 3.2: Operadores na linguagem `Bro`.

Operador	**Descrição**	**Exemplo**
+, -, *, /,%	Operadores aritméticos	a+b;
<,<=,>=,>	Operadores de comparações	a<b;
&&, ||,==, !=	Operadores de conjunção, disjunção, igualdade, desigualdade	if (a==4 && b!=3);
++, –	Incremento, Decremento unitário	++a
in, !in	correspondência de padrão	`[src addr, dst addr,serv] in RPC okay`

```
hook Notice::policy(n: Notice::Info)
  {
  if ( n$note == SSH::Hostname_Login )
      add n$actions[Notice::ACTION_EMAIL];
  }
```

Tabela 3.3: Ações nas notificações no Bro

Operador	**Descrição**
`ACTION_NONE`	Nenhuma ação
`ACTION_LOG`	Cria um `log` especifico com a notificação
`ACTION_EMAIL`	Envia um email com a notificação no corpo da mensagem
`ACTION_ALARM`	Gera um alarme e cria um `log` com a notificação

de endereços IP, e *threshold*, que é um inteiro o qual é atribuído o valor 50. O *script* incrementa o contador cada vez que o evento de rejeição de conexão ocorre. O evento de rejeição de conexão cada vez que um servidor TCP rejeita uma conexão de um cliente. Cada vez que este evento ocorre, o contador `attempts` que é para esse cliente, nesse endereço origem, é incrementado. O endereço se define como `orig_h` da origem, e como `resp_h` do destino ambos são indicados da conexão (`c`). O contador é assinado numa variável local (`n`) a qual é comparada com o limiar, *threshold* e, se supera o valor o limiar é realizada uma notificação chamada de `Notify` através do comando `NOTICE`, com a informação do endereço de origem. No Bro o caracter especial $ é utilizado como o (.) em outras linguagens para acessar a um elemento dentro de uma estrutura de dados.

```
global attempts: table[addr] of int &default=0;
global threshold = 50;
event connection_rejected(c: connection)
{ #Cada vez que uma conexão é rejeitada
local source = c$orig_h; #Pego o endereço de origem
local n = ++attempts[source]; #Incremento o contador.
if ( n == threshold ) #O contador chegou ao limiar?
NOTICE(Notify, source); #Envio a fonte para notificação
}
```

3.2.1 Arcabouço de Assinatura

O Bro não se destina especificamente a detecção por assinatura, como é o caso, por exemplo, do Snort. Portanto, o Bro tem um espectro maior na detecção de ataques. Não obstante, dentro da linguagem `Bro`, existe um arcabouço para a especificação de assinaturas. Até a versão 1.5 do Bro, existia um módulo para fazer a tradução direta das assinaturas do Snort para a sintaxes das assinaturas do Bro, o módulo era chamado de *Snort2Bro*. No entanto, segundo os desenvolvedores do Bro, este módulo não conseguia fazer um uso ótimo das capacidades adicionais que o Bro possui, marcando os enfoques das duas abordagens. Assim, esse módulo sofreu uma descontinuidade para as atuais versões do Bro.

Uma assinatura em Bro tem o seguinte formato:

assinatura <id> <atributos>. <id> é uma identificação única e individual para cada assinatura. <atributos> podem ser de dois tipos: condições ou ações. As condições definem as combinações das assinaturas, as ações definem que fazer em casso de uma combinação. Dentro das condições encontram-se quatro tipos: cabeçalho, conteúdo, dependência e contexto.

- cabeçalho: limita a aplicação da assinatura a um subconjunto de tráfego que contém os cabeçalhos dos pacotes combinados. Esse tipo de combinação é realizada unicamente pelo primeiro pacote da conexão. Exemplo: `dst-ip == 1.2.3.4/16`

- conteúdo: é definido para combinar expressões regulares. São definidos dois tipos de condições de conteúdo: declarando a expressão no conteúdo para pacotes em cru, por exemplo, `payload /<regular expression>/`, ou declarando a expressão com um analisador da camada de aplicação, por exemplo `http-request /<regular expression>/`.

- dependência: utilizada para definir dependência com outras assinaturas. Por exemplo `requires-signature [!] <id>`.

- contexto: transfere a decisão a outro componente do Bro, essa condição só é avaliada quando todas as anteriores foram combinadas. Exemplo `payload-size <cmp> <integer>` o qual compara o tamanho do conteúdo de um pacote com um número inteiro.

Cada vez que uma assinatura é combinada, um evento no Bro é gerado, por exemplo:

```
event signature_match(state:signature_state, msg:string, data:string)
```

A geração de um evento por assinatura diferença o Bro do resto dos sistemas, já que oferece uma grande precisão na detecção por assinatura. Com a utilização das condições de contexto dentro de uma assinatura, é possível analisar eventos cruzados, é dizer se foi cumprida a assinatura A no lado do cliente, logo de que a assinatura B se cumpra no servidor durante a mesma conexão permite evitar os falsos positivos. Mais informação sobre as assinaturas com contexto pode ser encontradas em [35].

A Figura 3.4 mostra a comparação das assinaturas no Bro e no Snort.

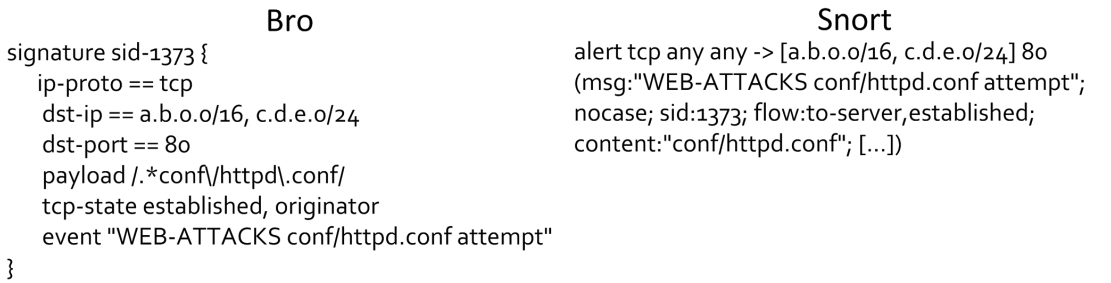


Figura 3.4: Comparação da elaboração de assinaturas no Bro e no Snort. Fonte [35]

3.3 Bro na Configuração de *Cluster*

Como foi dito anteriormente, o Bro não possui uma tecnologia *multithreaded*. Por isso existem limitações no momento de inspecionar o tráfego em redes de alta velocidade. Essa limitação em um IDS pode ser crítica no caso de pacotes serem perdidos, já que um único pacote pode ser o indício de uma ameaça ou ataque. Uma das opções para tentar diminuir ou evitar as perdas é utilizar o paralelismo existentes nos computadores atuais multi-núcleo [36]. Nesta proposta a carga do tráfego é distribuída para os diferentes núcleos de um processador, ou inclusive é possível distribuir entre diferentes computadores físicos. No Bro esta solução é feita através de *Bro Cluster*. Essa configuração, mostrada na Figura 3.5, funciona da mesma maneira que a configuração única ou *standalone*. Portanto, o Bro fornece

uma estrutura de gerenciamento de diversos dispositivos únicos, funcionando como *cluster*, inspecionando pacotes e fazendo correlação de atividades, mas atuando como entidades singulares ou únicas.

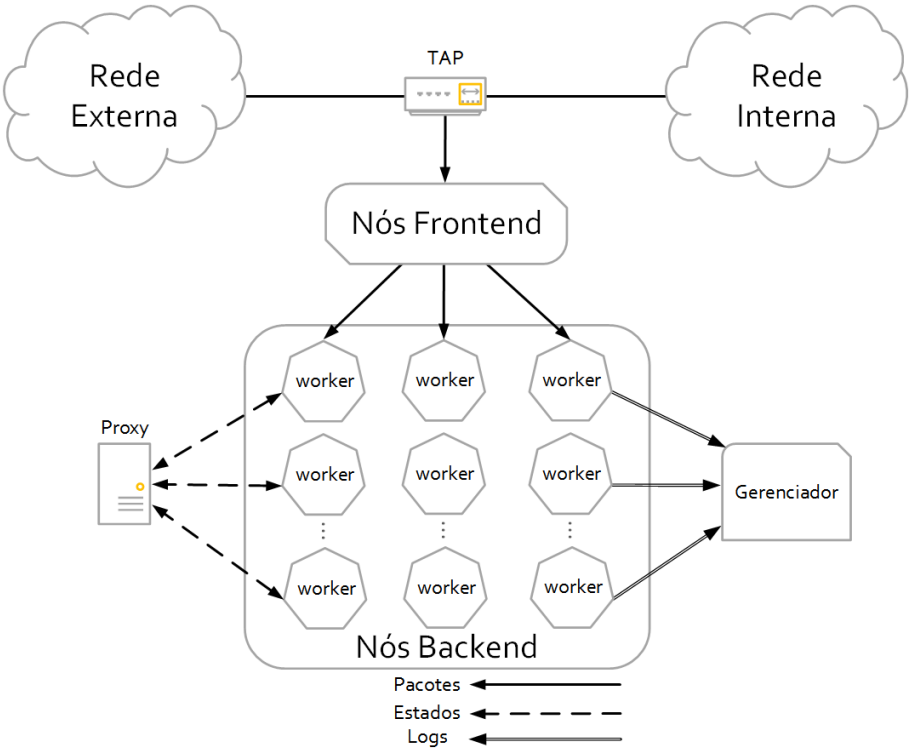


Figura 3.5: Arquitetura do *cluster Bro*.

A Figura 3.5 mostra os componentes principais de uma arquitetura de *cluster* em um Sistema de Detecção de Intrusão (IDS). Os principais componentes deste sistema são:

- Tap: que é um mecanismo que divide o fluxo de pacotes a fim de realizar uma cópia disponível para inspeção;
- Nós *Frontend*: sáo dispositivos de *hardware* que dividem o tráfego em muitos fluxos;
- Gerenciador: é o processo que tem duas tarefas principais. Primeiro, receber as mensagens e notificações *logs* do outros nós do *cluster*, resultando em unicamente um arquivo *log* em vez de ter tantos quantos forem os nós no *cluster*. Segundo realizar uma correlação dos eventos obtidos nos *logs*, para não duplicar as mensagens de alerta;
- Proxy: é o processo que administra a sincronização de estados. Assim, todas as variáveis dentro do Bro podem ser sincronizadas automaticamente. Portanto, elimina a necessidade de que todos os trabalhadores se comuniquem entre

eles. Por exemplo, se o trabalhador A detecta o hospedeiro 1.2.3.4 como um endereço ativo é importante que todos os outros trabalhadores conheçam esta informação. Assim, o trabalhador A envia esta informação ao *proxy*, e é ele que administra o envio de mensagem para os demais trabalhadores;

- Trabalhador: é o processo no *cluster* que inspeciona (*sniffer*) o tráfego fazendo o agrupamento dos fluxos do tráfego. A maior parte do trabalho pesado no IDS é feito nos trabalhadores.

Utilização da Biblioteca PF_RING

A biblioteca PF_RING [37] é um software para o Sistema Operacional Linux, o qual tem uma característica de *cluster* e permite fazer balanceamento de carga baseados em fluxos, através de diversos processos que inspecionam *sniffer*) a mesma interface. Esta característica permite fazer uso dos múltiplos núcleos em um hospedeiro físico utilizando a característica do *Bro single - threaded*, a qual nativamente só faz uso de um núcleo simples. A forma de fazer o balanceamento dentro do cluster é utilizando políticas baseadas em fluxos, tupla endereço IP porta origem-destino, com a técnica por padrão *Round-Robin*. Isso significa que todos os pacotes pertencentes ao mesmo fluxo vão para uma aplicação.

Outra das principais vantagens da PF_RING é que possui características de aquisição de pacotes a alta velocidade. Basicamente o funcionamento da PF_RING substitui a biblioteca `libpcap` utilizada por tcpdump, e Bro, entre outros. Com a vantagem de que esta biblioteca cria um *buffer* circular, chamado de anel, no qual os pacotes são armazenados, e utilizando um *socket* direto com as aplicações consegue ler pacotes em uma taxa maior, ou seja, com o socket criado a PF_RING consegue pular o `kernel` do linux. Seu funcionamento, ilustrado na Figura 3.6, é o seguinte, uma vez que o pacote é lido pela aplicação, a PF_RING descarta o pacote para dar espaço a um novo. A biblioteca pode criar tantos anais como aplicações existam, assim o desempenho é muito melhor que o obtido pela `libpcap`. A PF_RING é totalmente transparente ao sistema operativo e as placas de rede ou *Network Interface Controller - NIC*. Além disso, a biblioteca tem um módulo especial para a utilização das aplicações dependentes da biblioteca `libpcap`.

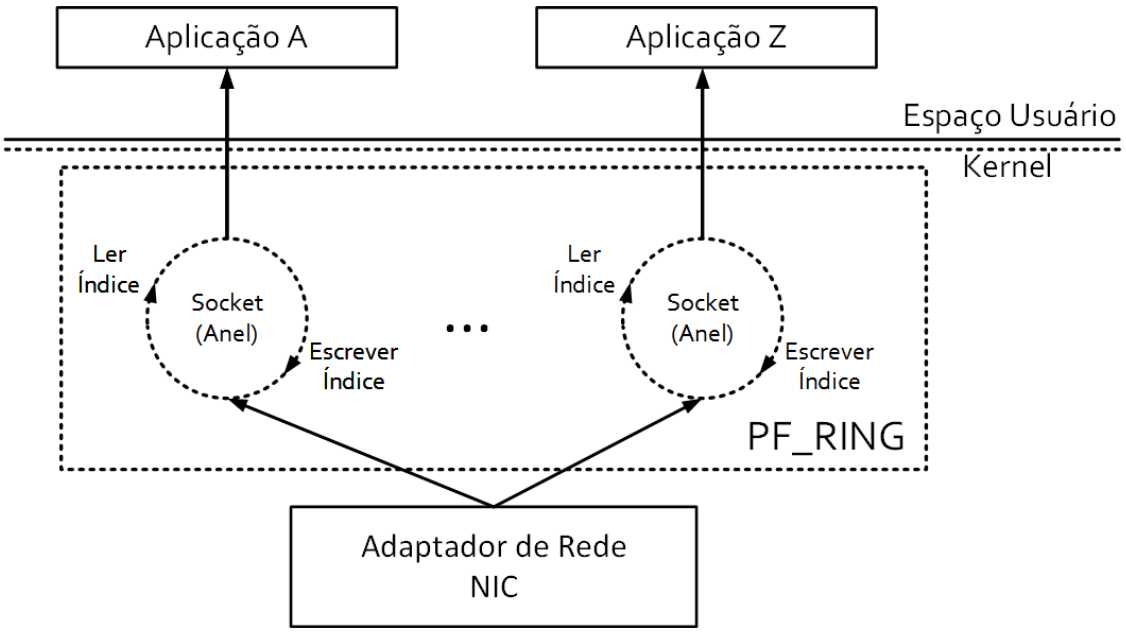


Figura 3.6: Esquema da biblioteca PF_RING.

3.4 Aplicações Complementares do Bro

Existe componentes adicionais a ferramenta Bro cujas características podem ser usadas para complementar à ferramenta.

3.4.1 BroControl

A ferramenta Bro pode ser executada de dois modos: analisando uma ou várias interfaces de redes em tempo real ou lendo arquivos `pcap`. O modo de funcionamento é através do comando **bro**. Assim o comando para analisar uma interface é `bro -i eth0 <scripts>` carregando os scripts ou políticas que vão ser aplicadas. De maneira semelhante para analisar um arquivo `pcap` o comando é `bro -r <nome do arquivo> <scripts>`. Também é possível escrever um arquivo `pcap` com os dados obtidos de uma interface mediante `bro -i eth0 -w <nome do arquivo> <scripts>`. O funcionamento bem semelhante ao `tcpdump` devido ao uso da `libpcap`. Além disso, também é possível utilizar a configuração *cluster* do Bro para fazer uma otimização de desempenho usando a tecnologia *multi-thread*. O BroControl é uma (*Application Programming Interface* - API) desenvolvida em python. Com ela é possível gerenciar os trabalhadores mediante comandos como iniciar, parar, instalar novas políticas, ver os estados dos nós, etc. Os comandos mencionados acima são executados automaticamente mediante esta API. Outra característica é que o `BroControl` armazena os *logs* dos trabalhadores, gerenciador, *proxys* em uma pasta única para maior facilidade de leitura. Do mesmo modo, é possível utilizar

os comandos que fornece a API para criar scripts, chamados de *plugins* de controle em python. Assim, por exemplo, é possível gerenciar nós separadamente ou obter status, *logs*, estatísticas, de maneira discriminada por nós.

3.4.2 A Biblioteca Broccoli

O Broccoli é uma biblioteca de comunicação entre clientes Bro ou em inglês *Bro Client Communications LIbrary* - BROCCOLI ou ainda de comunicação entre diferentes aplicações externas no Bro. Permite criar aplicações para a comunicação mediante o protocolo Bro. A capacidade importante desta biblioteca é o poder enviar ou receber eventos Bro e os estados dos diferentes nós. É programada em python ou C, e com ela é possível enviar e receber eventos Bro, status dos demais sensores. Além disso, é possível enviar e receber atributos dentro das funções nos *scripts* como inteiros, contadores, endereços IP, número de portas, *strings*, etc. Neste exemplo, é criado um *script* em python para a comunicação entre dois nós através do Broccoli. No exemplo, um nó se conecta com um servidor cujo endereço IP 127.0.0.1 mediante a porta 47758, e envia um evento, `event-create` e no corpo da mensagem são enviados dois atributos, um endereço `addr` e um contador de valor `42`.

```
from broccoli import *
bc = Connection("127.0.0.1:47758")
bc.send("event-create", addr("192.168.1.1"), count(42))
```

A recepção da mensagem pode ser escrita na linguagem python da como no exemplo seguinte, o qual se corresponde com o exemplo anterior. É criado um evento com dois argumentos é são mostrados por tela. Como é mostrada, a definição dos argumentos é são na linguagem Bro, mas o script é feito em python.

```
from broccoli import *
@event(addr,count)
def event-create(arg1, arg2):
     print arg1, arg2
```

A comunicação entre os nós não possui nenhuma hierarquia, o que significa que todos os nós podem se comunicar entre eles. Um nó pode criar um pedido de um evento e quando em qualquer dos demais nós esse evento seja executado, é informado ao nó que espera o pedido. Broccoli opera de maneira assíncrona. Por exemplo, um evento previsto para ser enviado através do comando `send()`, não sempre é enviado automaticamente, ele pode entrar em uma fila para ser enviado em uma transmissão

posterior. Se existe algum tipo de urgência na transmissão do evento, isto pode ser resolvido atribuindo uma prioridade a ele.

Outras das grandes vantagens do Broccoli, é que devido a que ele é programado em python ou C, existem duas versões funcionando paralelamente, pode se comunicar com aplicações em outros sistemas operativos, como Windows.

Este capítulo procurou mostrar a grande flexibilidade de definição de políticas de segurança provida pela ferramenta Bro e também a característica de alto desempenho que o Bro possui. Estas duas características e mais o fato de ser gratuita e de acesso fácil foram os principais fatores que influenciaram na escolha do Bro como ferramenta de análise de tráfego para o sistema desenvolvido que é apresentado no capítulo que se segue.

Capítulo 4

O Sistema BroFlow

Esta dissertação propõe o Sistema BroFlow [12] que é um Sistema de Detecção e Prevenção de Intrusão (IDPS) distribuído em Redes Definidas por Software (*Software Defined Networking* - SDN) para a defesa de ataques de Negação de Serviço (DoS). O sistema BroFlow utiliza a ferramenta Bro para analisar e monitorar o tráfego de rede inspecionando com eficiência todos os pacotes de um fluxo. O Bro provê uma sinalização de eventos em alto nível e uma linguagem própria para estabelecer políticas de segurança de uma maneira fácil. Logo, o Bro permite a implementação de diferentes algoritmos para a detecção por anomalias de ataques de negação de serviço por inundação. Na proposta, a ferramenta Bro de análise de tráfego é utilizada como sensores que geram alarmes quando uma anomalia é detectada. Além disso, esses alarmes são enviados a um controlador de rede OpenFlow que aciona uma contramedida para bloquear o ataque de maneira global. O BroFlow utiliza a visão global fornecida pelo OpenFlow para bloquear os ataques de DoS em todos os enlaces desde o destino até a origem. Também foi desenvolvido com os colegas Figueiredo e Lobato [14], uma facilidade para replicação do sistema de análise de tráfego. Logo, se o sistema de análise de tráfego estiver sobrecarregado, ele envia uma mensagem ao sistema de gerência da rede, que disponibiliza mais recursos físicos. Isso é obtido com a criação de máquinas virtuais específicas que realizam a análise de tráfego, que podem ser rapidamente replicadas e desligadas para atender às demandas variáveis. O controlador OpenFlow é então informado para que seja instaurada uma nova regra para a distribuição dos fluxos espelhados. Caso após algum tempo o tráfego diminua e não sejam mais necessárias tantas máquinas de análise, outro alerta é enviado para que estas máquinas sejam desativadas, garantindo assim um melhor aproveitamento de recursos.

4.1 Trabalhos Relacionados

As Redes Definidas por Software (*Software Defined Networking* - SDN) têm o controle centralizado e a visão global da rede, o que faz com que elas sejam eficazes para a detecção e reação a ataques de segurança, em particular aos ataques de negação de serviço por inundação. A programabilidade do protocolo OpenFlow permite o gerenciamento dinâmico dos fluxos nos comutadores da rede. Essa característica permite o seu uso na área de segurança de redes e vem sendo abordada por diferentes pesquisadores [38–40].

O sistema QFlow [3] propõe o isolamento de recursos em uma rede OpenFlow. Essa proposta é efetiva para a prevenção de ataques de negação de serviços entre múltiplos inquilinos que compartilham uma mesma infraestrutura física da rede. Mattos *et al.* propõem o isolamento da comunicação entre redes virtuais que compartilham mesma uma infraestrutura física [41]. Esta proposta visa assegurar a confidencialidade da rede virtual de cada inquilino sobre a infraestrutura física através do isolamento de recursos e de tráfego, prevenindo ataques de bisbilhotamento (*eavesdroping*). No entanto, garantir o isolamento entre redes virtuais previne o ataque de uma rede virtual sobre outra rede virtual no mesmo roteador físico, mas não detecta a ocorrência de ataques de negação de serviço internos a uma rede virtual. Ou seja, o sistema QFlow isola redes virtuais em uma mesma máquina física, evitando ataques que uma rede virtual possa promover sobre outra rede virtual, mas não evita todo tipo de ataque em uma rede virtual, como, por exemplo, ataques de negação de serviço que ocorrem em uma rede virtual. A proposta de sistema de detecção de intrusão que detectem este tipo de ataque é o foco principal desta dissertação.

O AVANT-GUARD [42] aborda dois problemas em redes SDN. O primeiro é o do gargalo da comunicação entre o plano de dados e o plano de controle que é uma vulnerabilidade que pode ser explorada com ataques de negação de serviço. Para evitar o excesso de comunicação entre os dois planos, cria-se uma extensão do plano de dados que lida com mensagens que podem ser ameaças de inundação. Essas mensagens são analisadas e a conexão só é aceita se for considerada legítima. Assim, a conexão considerada legítima é aceita e então reportada ao plano de controle, enquanto as que são consideradas malignas ou ilegítimas são rejeitadas ainda no plano de dados, evitando a sobrecarga do canal de comunicação entre o plano de dados e o plano de controle. O segundo problema abordado pelo AVANT-GUARD é a caída no desempenho do plano de controle frente a uma sobrecarga de mudanças dos fluxos dentro do plano de dados. Com pequenas mudanças dos fluxos no plano de dados são enviadas estatísticas ao plano de controle, ou plano de controle pode pedir essas estatísticas mediante *pooling* ao plano de dados. Se o envio o essas estatísticas

sofreram atrasos não seria possível para o plano de controle agir rapidamente sob ameaças. Também se as estatísticas são enviadas com muita frequência, existiria uma sobrecarga. Como solução, no AVANT-GUARD são introduzidos gatilhos no plano de dados que reportam informação de fluxos, assincronamente, ao plano de controle, tendo a possibilidade de atuar em caso de ameaças. Desta forma é evitado o atraso de ação do plano de controle.

Porras *et al.* abordam o problema de potenciais conflitos no estabelecimento de regras de fluxos entre diferentes aplicações nos controladores OpenFlow. Como solução, propõem uma extensão de segurança nos controladores NOX-OpenFlow [38]. Nessa extensão, cada aplicação tem políticas com diferentes prioridades atribuídas, dependendo da sua função. Ao serem aplicadas nos fluxos do controlador, as políticas voltadas para as aplicações de segurança OpenFlow são prioritárias. Shin *et al.* propõem um arcabouço de segurança para o controlador NOX-OpenFlow, o sistema FRESCO [39], o qual provê uma linguagem de programação e módulos de software para desenvolver aplicações de segurança. Esses módulos permitem a integração de serviços de segurança sobre uma nova arquitetura de controladores NOX-OpenFlow. De maneira semelhante, Kim *et al* apresentam uma abordagem baseada nos estados da rede, que usa a alta programabilidade oferecida por SDN para decidir que aplicações executar dependendo do evento [40], nele a linguagem Pyretic, que modulariza funções de rede, é utilizada.

OpenSAFE [43] propõe um sistema para redirecionamento de fluxos para aplicações de monitoramento e segurança. OpenSAFE também especifica uma linguagem para um gerenciamento simplificado de fluxos em aplicações de segurança. O OpenSAFE se baseia no protocolo OpenFlow para implementar funções básicas para manipulação de fluxos, como por exemplo filtros, que selecionam fluxos específicos e também desvios, para encaminhar o tráfego para aplicações de monitoramento e segurança. Há ainda um componente OpenFlow que interpreta as políticas definidas pela linguagem de gerenciamento de fluxo para que as ações sejam realizadas.

Os ataques de negação de serviço em redes SDN são estudados por Braga *et al.*, que propõem um método de detecção de ataques de negação de serviço distribuídos (*Distributed Denial of Service* -DDoS) usando um controlador NOX-OpenFlow [44]. Esse trabalho consiste em monitorar os comutadores de uma rede durante determinado período, extraindo características dos fluxos que entram nas tabelas do comutador, tais como média de pacotes por fluxo, média de bytes por fluxo, entre outras. Esses atributos são enviados a um classificador dentro do controlador NOX, baseado em redes neurais, que determina se o tráfego é normal ou se corresponde a um ataque. A vantagem desta proposta é a possibilidade de detecção de Ataques de Negação de Serviço Distribuídos diretamente no controlador. Como desvantagem, as redes neurais requerem um treinamento prévio com conjuntos de dados artificiais,

o que é uma limitação importante na área de IDS [21]. Além disso, o processamento do controlador da rede é sobrecarregado pelo processo de detecção.

Mehdi *et al.* implementam algoritmos de detecção por anomalia diretamente sobre o controlador NOX-OpenFlow [45]. A proposta consiste em inspecionar somente os primeiros pacotes das conexões, sendo efetivo em ataques de varredura de portas, onde só é analisado o cabeçalho do pacote. Contudo, essa abordagem apresenta deficiências na detecção de ataques mais sofisticados como vermes (*worms*) ou propagação de vírus. Além disso, a leitura dos primeiros pacotes das conexões no controlador gera entradas nas tabelas de fluxos, as quais durante um ataque de DDoS crescem em uma alta velocidade, criando um gargalo que aumenta o tempo de entrega de pacotes.

Existem propostas de Sistemas de Detecção e Prevenção de Intrusão (IDPS) em redes virtuais. NICE [46] implementa um IDPS em ambientes de nuvem, criando um modelo analítico baseado em grafos dos ataques que, dependendo do estado do ataque, realiza, como contramedida, a adaptação da topologia da rede para evitar o ataque. Esta proposta é baseada no protocolo OpenFlow sobre comutadores Open vSwitch (OVS). NICE evoluiu para o sistema denominado SnortFlow [47], que consiste em um IDPS baseado no Snort, um IDS *open source* de detecção por assinatura, e no OpenFlow para o encaminhamento dos pacotes. O analisador Snort é instalado no domino de gerência do hipervisor XEN, que é conectado em um comutador ligado às máquinas virtuais de uma máquina física para inspecionar o tráfego das máquinas virtuais. Como vantagem tem a propriedade de funcionamento da ferramenta Snort, a qual funciona imediatamente depois da instalação. No entanto, essa proposta carece do sincronismo entre o agente Snort e controlador para ter uma visão global da rede.

O IPSFlow [48] é outra proposta baseada em SDN e propõe um mecanismo de bloqueio automático de tráfego malicioso utilizando o protocolo OpenFlow. Na proposta, o tráfego é duplicado para a análise no IDS, gerando novos fluxos na rede. Além disso, a análise dos fluxos é feita de maneira seletiva, porém existe uma grande possibilidade de não inspecionar fluxos maliciosos durante a seleção, já que durante um ataque de DoS, todos os campos dos pacotes são similares aos legítimos.

A empresa Radware comercializa um equipamento [30] para a prevenção de ataques de Negação de Serviço em Redes Definidas por Software SDN. O equipamento implementa um controlador de código proprietário que gerencia uma rede de comutadores OpenFlow e utiliza lógica nebulosa para determinar o grau do ataque, baseado no comportamento da taxas de pacotes, transmissão e abertura de conexões, além do comportamento dos protocolos, de acordo com padrões previamente obtidos. Com esses valores é possível diminuir a alta taxa de falsos positivos obtidos pela lógica nebulosa. A ferramenta precisa de um período de aprendizagem do

comportamento legítimo para obter a base de dados para o treinamento.

O sistema BroFlow proposto tem a vantagem de funcionar na configuração fora da rota ou passivo. Mediante a comunicação com o controlador, é feito o bloqueio dos fluxos atacantes de maneira semelhante à configuração em linha, sem ter os atrasos que esta configuração tem. A configuração fora da rota gera uma duplicação nos fluxos analisados, no entanto, o fluxo duplicado pode ser enviado a máquina de análises dedicada ou para um *cluster* de máquinas para a detecção de ataques. Este funcionamento permite também ao sistema fornecer recursos sob demanda frente a uma sobrecarga da máquina de análises, o qual faz o sistema altamente escalável e robusto contra Ataques de Negação de Serviço. O sistema proposto BroFlow implementa diferentes algoritmos de detecção diretamente nos sensores tanto de anomalia como de assinatura, o qual se diferença do SnortFlow que só consegue detectar ataques por assinatura e em relação com a proposta de Braga *et al.*, alivia o controlador do processo de detecção de ataques. Além disso, o BroFlow implementa sensores tanto no domínio privilegiado ou domínio 0 como em cada um dos roteadores virtuais e nas máquinas físicas, em comparação com o SnortFlow que implementa sensores só no domínio 0 e com a proposta do Braga *et al.* que só detecta no controlador. Esta localização específica dos sensores dá ao sistema uma visão global da rede, uma granularidade de cada uma das redes virtuais, e uma proteção da infraestrutura.

4.2 Modelos de Virtualização

A proposta desta dissertação considera um ambiente híbrido de virtualização. Para criar esse ambiente virtual foram utilizadas a virtualização de máquinas provida pelo Xen e a comutação de fluxos programável provida pelo OpenFlow. A vantagem da utilização do Xen é a criação de máquinas virtuais isoladas entre si e com um controle distribuído. Essa vantagem possibilita que sejam desenvolvidas novas aplicações para o gerenciamento da infraestrutura virtual. A vantagem da utilização do protocolo OpenFlow em um comutador programável é que a migração de fluxos é feita de forma simplificada e, consequentemente, também é possível a migração de enlaces virtuais, o que permite o remapeamento de topologias lógicas sobre a topologia física [3].

4.2.1 A Plataforma Xen de Virtualização de Máquinas

O Xen [49] é uma plataforma de virtualização de computadores pessoais. Essa plataforma de virtualização é usada em servidores de centro de dados (*data-centers*) em empresas como a *Amazon*, onde a virtualização do Xen teve seu desempenho

avaliado [50]. A arquitetura do Xen se baseia em uma camada hipervisora de virtualização logo acima do *hardware*, que monitora o comportamento das máquinas virtuais, como mostrada na Figura 4.1. As máquinas virtuais são executadas sobre o hipervisor, que garante um acesso independente a recursos como CPU, memória, rede e disco.

No Xen, cada ambiente virtual está isolado dos demais, ou seja, o funcionamento de uma máquina virtual não afeta o funcionamento das outras. Além disso, permite que essas máquinas virtuais possuam sistemas operacionais distintos. O Xen possui um ambiente virtual privilegiado, o Domínio 0, que executa as operações de gerência do hipervisor e detém a exclusividade no acesso aos dispositivos físicos, como os dispositivos de Entrada/Saída (E/S). Desta forma, as máquinas virtuais só conseguem acesso aos dispositivos físicos através de *drivers* virtuais que se comunicam com o Domínio 0 ou através de uma porta da máquina física dedicada à máquina virtual. Outra solução seria utilizar um *hardware* específico, porém de alto custo.

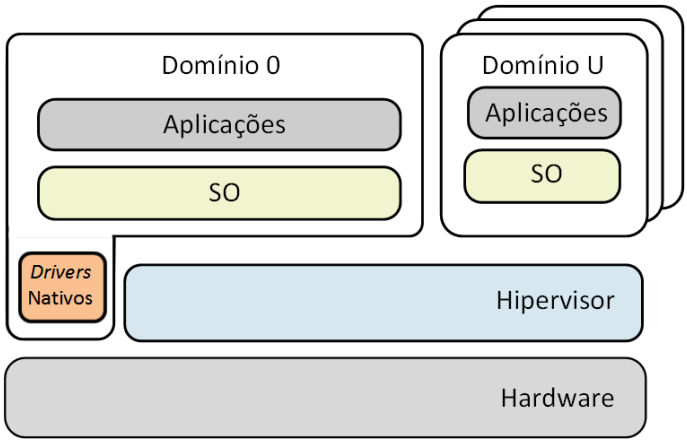


Figura 4.1: Arquitetura Xen com a camada hipervisora e a separação entre o domínio privilegiado e os demais domínios.

4.2.2 O Comutador de Fluxos OpenFlow

A abordagem das Redes Definidas por Software (*Software Defined Networking* – SDN) permite o gerenciamento do tráfego de redes. Elas permitem a definição de fluxos de pacotes de forma dinâmica. O paradigma de SDN é o da separação de planos, em que a rede é dividida em um plano de dados e um plano de controle. O plano de controle é responsável pela definição da próxima rota de um fluxo e, para isso, executa algoritmos de controle da rede. Enquanto o plano de dados faz o encaminhamento dos pacotes de acordo com políticas definidas pelo controlador. Essa separação fornece a facilidade de que vários comutadores podem compartilhar o mesmo plano de controle, que possui a visão global da rede. Desta forma, é possível

redefinir as rotas de fluxos de múltiplo comutadores apenas mudando a aplicação no controlador.

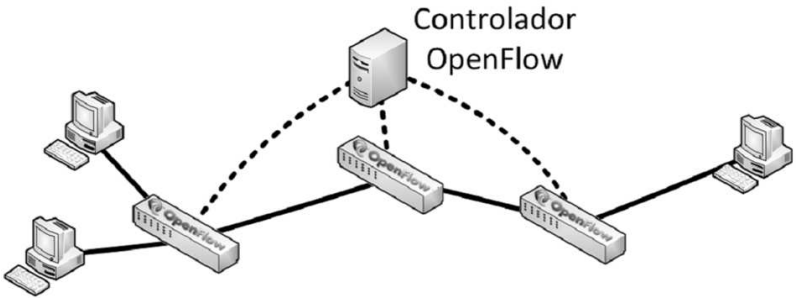


Figura 4.2: Arquitetura OpenFlow na qual os comutadores se comunicam com o controlador para definição da tabela de comutação de fluxos.

O protocolo OpenFlow [10] define regras e padrões para SDN. Na arquitetura OpenFlow, existe uma tabela de fluxos nos comutadores que é atualizada pelo plano de controle. Quando chega um pacote de um fluxo que já está definido nessa tabela, ele é simplesmente comutado. Caso chegue um pacote de um fluxo não definido, ele é enviado para o controlador, onde é definido qual será o próximo destino desse pacote. O destino é escolhido de acordo com as aplicações sendo executadas e também de acordo com o cabeçalho do pacote. O controlador então adiciona essa regra na tabela de fluxo do elemento encaminhador. É importante ressaltar que só o primeiro pacote de um fluxo ainda não definido é encaminhado ao controlador, sendo a partir deste primeiro pacote criada a regra na tabela de fluxo, o que faz que os demais pacotes desse fluxo sejam somente comutados, afetando o desempenho da comunicação apenas no primeiro pacote de um fluxo. A arquitetura OpenFlow está ilustrada na Figura 4.2.

Na proposta desta dissertação, o comutador programável por *software* denominado Open vSwitch (OVS) [51] é utilizado como comutador OpenFlow. Além de apresentar uma tabela de encaminhamento que pode ser atualizada por um controlador OpenFlow, o Open vSwitch oferece as funcionalidades de descartar pacotes, alterar campos do cabeçalho, entre outras. Nesse ambiente de virtualização, as máquinas virtuais se interconectam através de comutadores OpenFlow, implementados pelo comutador programável Open vSwitch, instanciados nas máquinas físicas. A configuração e o controle dos comutadores OpenFlow é realizada pelo controlador POX[1]. O POX foi escolhido entre outros controladores, pela simplicidade na programação mantendo o seu alto desempenho [52].

[1]O controlador POX escrito em Phyton pode ser obtido em http://www.noxrepo.org/pox/about-pox/.

4.3 O Sistema BroFlow

A arquitetura proposta consiste basicamente em um esquema modular. Cada um desses módulos tem uma função específica, entre eles estão os sensores BroFlow, os quais realizam a inspeção dos pacotes, o módulo de políticas, no qual são definidos os algoritmos de detecção por anomalia e a aplicação BroFlow POX rodando no controlador a qual administra a comunicação entre os sensores e o gerenciamento dos fluxos. Na arquitetura proposta as máquinas físicas podem alocar sensores BroFlow e executar aplicações em paralelo. Aliás, a arquitetura provê um mecanismo de redefinição dos fluxos para balancear o tráfego direcionado às máquinas que realizam a detecção de intrusão e a tomada de ações em caso de alguma atividade maliciosa. Nesta seção é explicado cada um esses módulos e mecanismos.

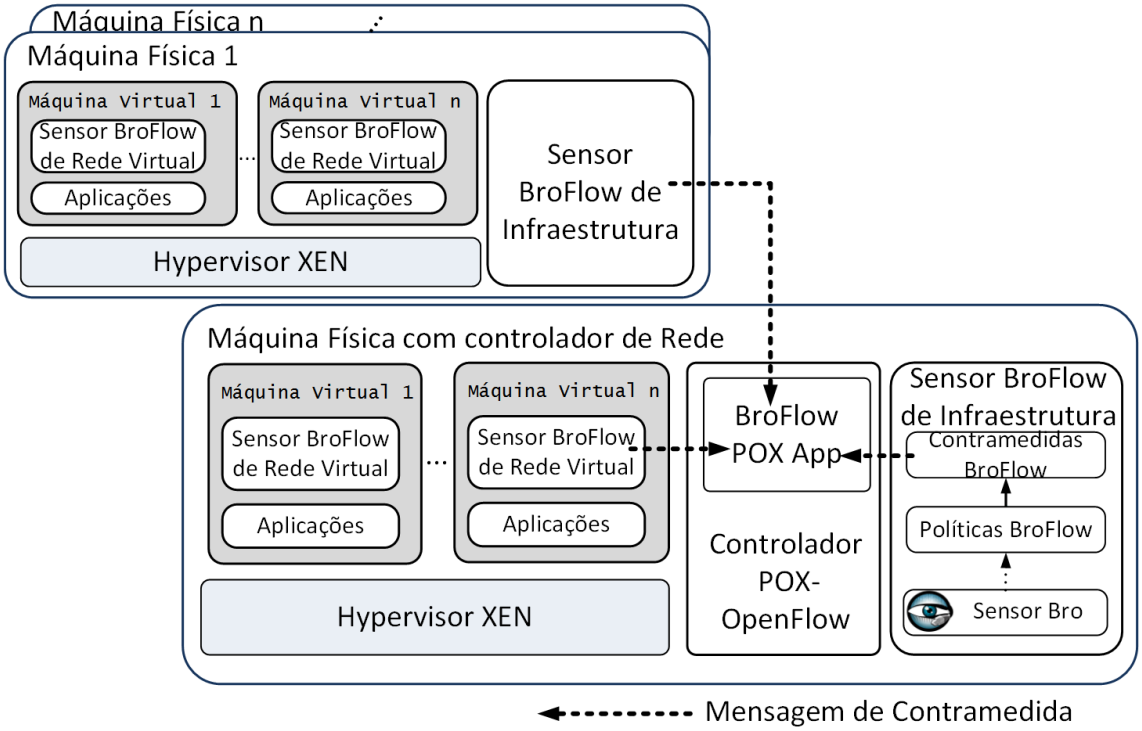


Figura 4.3: A arquitetura do sistema BroFlow. Módulos de inspeção e análise de tráfego da infraestrutura e das redes virtuais para detecção de ataques à infraestrutura e a uma rede virtual em particular.

4.3.1 Os Sensores BroFlow

No sistema BroFlow existem dois tipos de sensores. Os sensores BroFlow de Rede Virtual e o sensor BroFlow da Infraestrutura, como ilustrado na Figura 4.3. Os sensores de BroFlow de Rede Virtual devem estar distribuídos em todos os roteadores virtuais ou localizados em pontos estratégicos da rede virtual a ser inspecionada e

analisada. Os sensores da rede virtual podem monitorar tanto os roteadores virtuais como o tráfego de uma estação específica, tal como um servidor de aplicação devido à sua importância na rede. Em cada um dos sensores BroFlow da rede virtual são estabelecidas políticas específicas e independentes para cada rede virtual. Esta facilidade provida pelo sistema BroFlow é importante dentro de um ambiente de nuvem, pois a política é definida para a rede virtual independentemente na máquina física que está hospedada. Assim, a política de segurança persiste mesmo quando ocorre a migração de um roteador virtual de uma máquina física para outra, uma vez que o sensor BroFlow também migra junto com o roteador virtual. Nesta arquitetura, as máquinas físicas podem alocar diversos sensores de rede virtual BroFlow e executar uma análises de eventos e emitir alarmes mediante uma comunicação segura até o controlador, como mostrada na Figura 4.4. Além disso, apresentam um *daemon* para monitoramento dos recursos consumidos tanto pela máquina física quanto pelas máquinas virtuais.

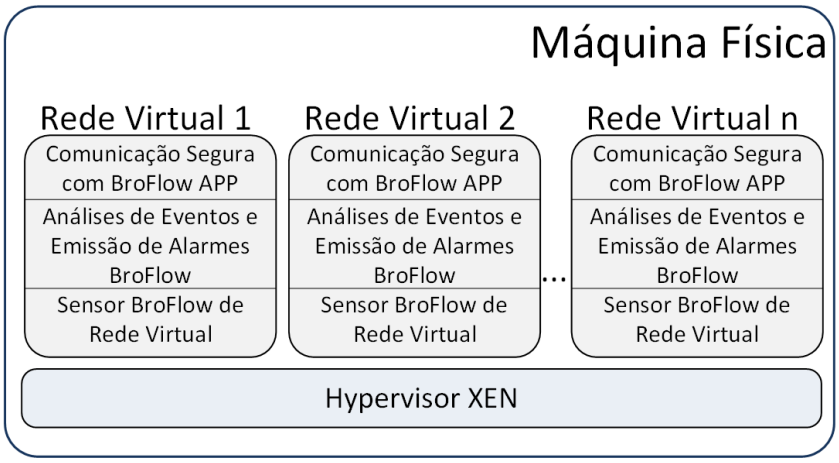


Figura 4.4: Máquina física com sensores de redes virtuais Bro.

Para total controle do tráfego de uma rede há necessidade de inspecionar e analisar todos os enlaces com sensores, mas esta tarefa é de muito alto custo. Logo, um primeiro desafio para a detecção de intrusão é a escolha dos locais nos quais se devem colocar os sensores para que resulte na melhor detecção de intrusão. Um segundo desafio também importante é de como correlacionar as informações dos sensores que são colocados de forma distribuída na rede. Por exemplo, os ataques de negação de serviço distribuídos usam o fato de se originarem de várias fontes, ou seja, de serem distribuídos, para dificultar a sua detecção. Com sensores distribuídos aumenta a possibilidade de se correlacionar as informações de sensoriadas para detectar estes ataques. A proposta BroFlow tem como objetivo obter essa vantagem com a combinação da ferramenta Bro com a visão global fornecida pelo OpenFlow. Os sensores distribuídos enviam informação para uma aplicação centralizada que executa no controlador POX do OpenFlow. Assim, é possível ter um controle global

da rede mediante o monitoramento distribuído de sensores BroFlow. Logo, critério para definir a melhor localização específica mediante análise matemática dos sensores em uma rede é um problema em aberto que está fora do escopo desta dissertação e está previsto como trabalhos futuros. É fato que um ataque de negação de serviço distribuído por inundação se faz mais presente nos enlaces perto da vítima pela convergência dos pacotes de ataque. No entanto, a eliminação dos pacotes de ataque perto da vítima não reduz o tráfego de ataque desde a origem do ataque até o ponto no qual foi detectado. A proposta desta dissertação, o sistema BroFlow, possui a visão global da rede devido à centralização. Assim, o sistema BroFlow tem a vantagem, de uma vez detectados os "fluxos atacantes", poder acionar uma contramedida que pode ser enviada a todos os comutadores da rede, eliminando desde a sua origem o tráfego relacionado ao fluxo mal intencionado.

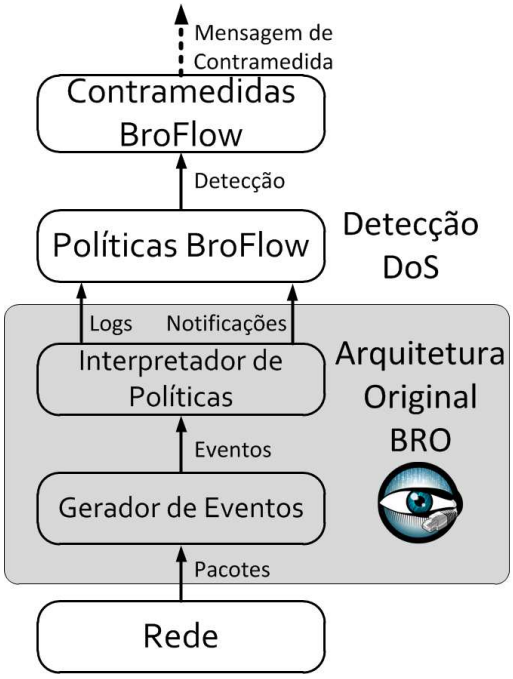


Figura 4.5: Sequência de atividades do sensor BroFlow da inspeção do pacote na rede até o envio de mensagem de contramedida para a aplicação BroFlow POX no controlador.

O Sensor BroFlow de Infraestrutura tem a finalidade de analisar o tráfego nas máquinas físicas enquanto o Sensor BroFlow de Redes Virtual analisa apenas o tráfego de uma redes virtual específica. Em redes virtuais, podem ocorrer ataques a máquinas físicas que consiste, por exemplo, de uma negação de serviço por inundação que seja difícil de detectar em roteadores virtuais. O sucesso do ataque seria obtido pela soma de todos os pacotes dos roteadores virtuais que podem ocasionar interrupção de serviço de um roteador físico. Logo, os Sensores BroFlow de Infraestrutura são necessários para proteger a infraestrutura física da rede, ou seja, a

rede física subjacente às redes virtuais monitoradas pelo BroFlow. Outro exemplo de ataque à infraestrutura é o ataque de inundação por ARP, também conhecido como envenenamento ARP. Esse ataque consiste em atacar comutadores com pacotes ARP para saturar as tabelas de encaminhamento com endereços MAC falsos. Isso provoca a negação de serviço na memória dos comutadores da infraestrutura. O Sensor da Infraestrutura BroFlow que fica na máquina física que tem o controlador OpenFlow detecta e impede esse tipo de ataque, protegendo a rede física e as redes virtuais que ela hospeda.

Cada um dos sensores BroFlow contém um *daemon* da ferramenta Bro, o qual executa em paralelo às demais aplicações, com um consumo mínimo de recursos do sistema. Cada Sensor BroFlow tem aplicações de detecção de ataques e envio de alarmes definidos por dois módulos: o Módulo de Políticas BroFlow e o Módulo de Contramedidas. O módulo de Políticas BroFlow é implementado diretamente na ferramenta Bro, sendo abstraídos os algoritmos de detecção em políticas descritas na linguagem Bro.

Além disso, a aplicação BroFlow POX, que executa no controlador, gerencia os alarmes e as contramedidas recebidas. A Figura 4.5 detalha os módulos dos sensores BroFlow. Os pacotes capturados são enviados ao gerador de eventos, que os verifica, ordena e converte em eventos, que por sua vez são enviados em seguida ao interpretador de políticas. O gerador de eventos e o interpretador de políticas fazem parte da arquitetura original da ferramenta Bro. O módulo de políticas BroFlow possui a inteligência para decidir se os eventos gerados pelo Bro constituem realmente um ataque e, em caso positivo, qual ação a aplicação BroFlow POX deve realizar.

As Políticas BroFlow

A arquitetura de políticas de segurança do sistema BroFlow, composta de três módulos principais: inspeção de eventos de rede, detecção de ataques e contramedidas, é apresentada na Figura 4.6. O módulo de inspeção de eventos de rede analisa os eventos de rede, fornecidos em tempo real pelo sistema Bro com as informações pertinentes aos fluxos estabelecidos durante a recepção de pacotes. As políticas são descritas na linguagem `Bro` e o sistema BroFlow oferece três políticas para detecção de ataques de negação de serviço por inundação de pacotes: TCP-SYN, ICMP, e UDP. Assim, toda vez que um pacote relativo a esses eventos é detectado pelo módulo de inspeção de eventos de rede, o módulo de detecção de ataques é invocado. Nesse último módulo, são implementados os diversos algoritmos, abstraídas as políticas na linguagem `Bro`, que decidem se existe ou não um ataque para que, se afirmativo, um alarme seja emitido. Nos exemplos implementados, a detecção é feita por estabelecimento de limiares de rampa ou adaptativo. No caso de desejar

adicionar ou modificar o algoritmo de detecção, a sua implementação é facilitada pela programação de alto nível da linguagem `Bro`. Quando um ataque é detectado, o módulo de contramedida é invocado. Nesse módulo qualquer programa externo pode ser chamado para realizar as contramedidas necessárias para evitar o ataque.

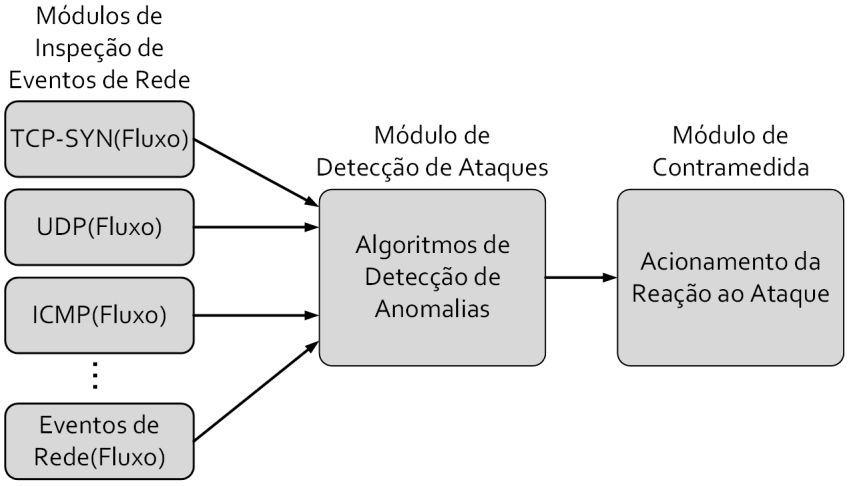


Figura 4.6: Arquitetura de políticas de segurança do sistema proposto BroFlow. Composta por três módulos sequenciais: inspeção de eventos de rede, detecção de ataques e contramedidas.

O protótipo do BroFlow implementa dois tipos de algoritmos de detecção de ataques por inundação: por rampa e por limiar adaptativo [8]. O Algoritmo 1 executa a detecção por rampa ao efetuar o somatório de pacotes durante um determinado período de tempo e emite um alarme se um limiar especificado é ultrapassado. O Algoritmo 2 usa a técnica de limiar adaptativo e tem como objetivo diminuir os falsos positivos ou "flash crowds", que excedem rapidamente os valores médios. O algoritmo detecta mudanças nas estatísticas do tráfego, baseado em medições de tráfego de rede em intervalos consecutivos de tempo T.

Toda vez que o limiar é ultrapassado, um contador k é incrementado. Se o contador for maior que um, indica que houve violação do limiar por períodos consecutivos de tempo, o que caracteriza uma anomalia considerada ataque. Os parâmetros de ajuste do algoritmo são a amplitude do fator α, para calcular o limiar, o número de violações do limiar consecutivas k, o fator média móvel exponencial ponderada (*Exponential Weighted Moving Average* - EWMA) β e o tamanho em segundos do intervalo de tempo T.

4.3.2 As Contramedidas BroFlow

O módulo de contramedida realiza a comunicação com a aplicação BroFlow POX no controlador de rede OpenFlow. O módulo de contramedidas, escrito na linguagem *Python*, traduz as informações geradas pelos sensores BroFlow e encaminha as

```
NumPac = Número de pacotes SYN
lim = limiar estabelecido por heurística
tempo = instante de tempo da chegada de um pacote
tempoMax = tempo de observação da rampa

while chega Pacote SYN do
    Soma dos pacotes;
    if soma > lim && tempo < tempoMax then
        ativa alerta;
        mensagem para o controlador;
    end
end
```

Algoritmo 1: Detecção de ataque de negação de serviço por inundação pelo método da rampa.

```
α = porcentagem acima da média que é considerada anômala
β = valor Média Móvel Exponencial Ponderada (EWMA)
μ_n = média estimada de pacotes no intervalo n
λ_n = quantidade de pacotes no intervalo n
T = duração do intervalo em segundos
k = número de intervalos consecutivos em que o limiar foi violado
while período < T segundos do
    λ_n = λ_n + 1
end
μ_n = βμ_{n-1} + (1 − β)λ_n
if μ_n>(α + 1)λ_n then
    incremento de k;
    if k> 1 then
        ativa alerta;
        mensagem para controlador;
    end
end
```

Algoritmo 2: Detecção de ataques de negação de serviço pelo método de limiar adaptativo.

mensagens de alarme para a aplicação BroFlow POX App. Quando um ataque é detectado pelo módulo de políticas nos sensores BroFlow, uma mensagem de alarme é enviada para o controlador POX-OpenFlow. Esses módulos de comunicação usam o protocolo *Secure Socket Layer (SSL)* em interfaces de redes dedicadas, garantindo comunicações encriptadas e autenticadas. As mensagens são enviadas no formato JSON e contêm as informações do fluxo, como endereços IP e portas, de origem e de destino, obtidas através da ferramenta Bro, o endereço MAC de destino do controlador e a contramedida a ser tomada.

Os campos nas mensagens enviadas são as informações que o sistema Bro consegue monitorar de um pacote. Como esses campos não exaurem as possibilidades de

campos de um fluxo completo OpenFlow, os demais campos são completados com valores coringas (*wildcard*). Embora essa definição instale fluxos gerais nos comutadores, o uso de campos coringas não gera ambiguidade, pois uma conexão TCP é normalmente definida por quatro campos do pacote somente: IP e portas de origem e destino. Assim, como os quatro campos que identificam uma conexão TCP são bem definidos, não há ambiguidade nos fluxos considerados maliciosos.

4.3.3 A Aplicação BroFlow POX App

A aplicação BroFlow POX é uma aplicação do controlador POX-OpenFlow que recebe as mensagens de alarme provenientes dos diferentes sensores BroFlow e executa as contramedidas necessárias para respondê-los. Assim, ao receber a mensagem de alarme proveniente do modulo de contramedidas, a aplicação BroFlow POX verifica em sua tabela de fluxos a qual o fluxo corresponde ao conteúdo da mensagem de alarme. Em seguida, a aplicação indica ao controlador POX-OpenFlow a ação de contramedida a ser executada em todos os comutadores da rede. No protótipo BroFlow, as contramedidas possíveis, no caso de uma ameaça, correspondem às ações OpenFlow especificas: *drop*, de descarte do pacote e *output*, de encaminhamento do pacote para uma porta definida. Portanto, com essas duas ações do protocolo OpenFlow sobre os comutadores da rede, as contramedidas definidas pelo BroFlow podem ser, o *bloqueio* de um fluxo específico ou *desvio* de um fluxo para uma outra estação. É importante listar os fluxos já instalados, pois no protocolo OpenFlow regras com correspondência mais completa tem prioridade sobre as demais. Caso contrário, a regra de apenas bloqueio da origem não surtiria efeito sobre os fluxos já existentes.

Assim, no caso de um ataque de negação de serviço (DoS), os sensores BroFlow detectam o ataque de acordo com a política de segurança enviando uma mensagem de alarme indicando a contramedida a ser realizada. Vale lembrar, que a comunicação entre sensores e o controlador é executada em uma rede isolada da rede de comunicação compartilhada pelos inquilinos. Além disso, se a política de segurança do BroFlow detectar um ataque de varredura de portas, o fluxo malicioso pode ser redirecionado para um servidor de pote de mel (*honeypot*), para estudar as características do ataque. Todas as contramedidas são aplicadas sob-regime de quarentena, ou seja, cada vez que uma contramedida é aplicada nos comutadores, um temporizador é ativado. Quando o temporizador finaliza, a contramedida é apagada e todas as análises de ataques são realizadas novamente. Assim, é possível detectar se o ataque foi encerrado, deixando de utilizar recursos nas contramedidas.

Módulo de Gerenciamento de Fluxo

Existe a possibilidade de que a máquina de análise de tráfego fique sobrecarregada com uma taxa alta de pacotes, o que pode ser uma tentativa de ataque de evasão no sistema de detecção de intrusão ou um ataque de Negação de Serviço por inundação. Nestes casos, o sistema pode desviar os fluxos atacantes para que sejam analisados em diversas máquinas em paralelo. Para isto, em colaboração com os colegas Figueiredo e Lobato, foi criado um módulo no controlador que é responsável pela distribuição dos fluxos entre as máquinas com o Bro. O módulo é parte da aplicação BroFlow POX do controlador OpenFlow POX, que se comunica tanto com os sensores BroFlow, quanto com o Módulo de Gerenciamento de Recursos. O espelhamento de pacotes da rede para as máquinas de análise é feita através de um túnel GRE (*Generic Routing Encapsulation*) e a análise dos pacotes é feita após o desencapsulamento, para garantir que estes pacotes não tenham sido alterados. Essa é uma das formas de se fazer o espelhamento e apresenta a vantagem que as máquinas virtuais podem estar em redes diferentes. A distribuição dos fluxos é feita levando em conta dois aspectos: a quantidade de recursos para análise de pacotes disponíveis em cada máquina virtual de análises e a origem dos pacotes. Um fluxo de uma origem nova é alocado na máquina de análises que possuir o menor processamento no momento. Já os fluxos de mesma origem têm prioridade para serem alocados na mesma máquina de análises, para evitar que um ataque desta origem passe despercebido.

Módulo de Gerenciamento de Recursos

O módulo de gerenciamento dos recursos se encontra no domínio privilegiado, Domínio 0, de todas as máquinas físicas da rede, como foi mostrado na Figura 4.4. Nesse módulo são monitorados os consumos de banda, de processamento e de memória da máquina física e também o quanto desses recursos é consumido por cada máquina virtual. Este monitoramento é feito através da coleta de dados do Xen realizadas pela *Libvirt*. Cada máquina física executa um *daemon* para coleta de dados e das estatísticas de todas as máquinas físicas da rede que são agregadas no controlador. Desta forma, o controlador tem a informação da quantidade de máquinas de análise da rede e o consumo de recursos de cada uma.

Para evitar sobrecarga ou a presença de recursos ociosos, o sistema analisa principalmente as máquinas do Módulo de Detecção de Intrusão. Em caso de sobrecarga, este módulo analisa os recursos disponíveis nas máquinas físicas e decide então onde instanciar uma nova máquina de analises. De maneira análoga, também é feito o monitoramento conjunto de todas as máquinas de analises, para detectar quando for possível uma redistribuição de fluxos que permita a desativação de uma dessas

máquinas, garantindo a elasticidade da proposta. A arquitetura e cada um de seus elementos é ilustrada na Figura 4.7. A comunicação entre os diversos elementos é representada pelas linhas tracejadas.

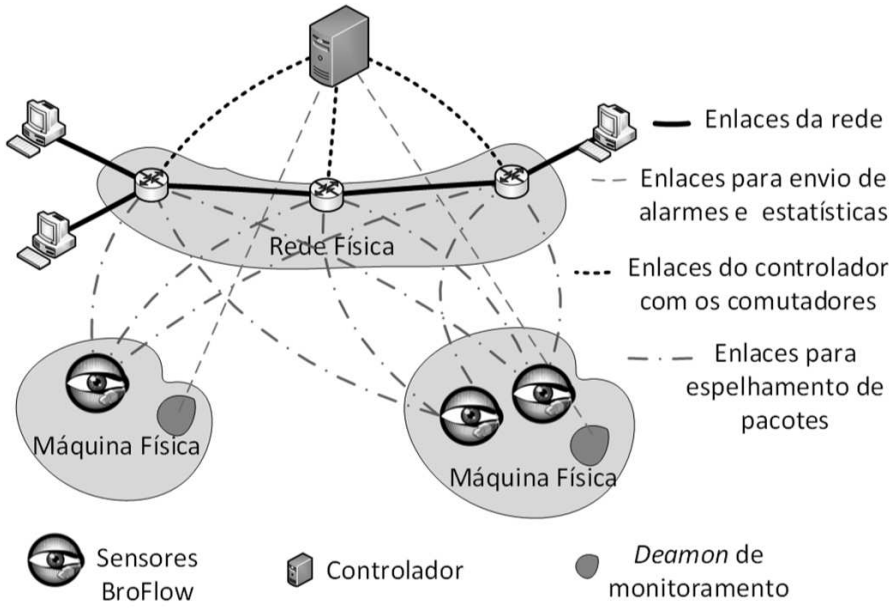


Figura 4.7: Exemplo da arquitetura proposta e a comunicação entre seus elementos.

4.4 Resultados

Para validar a proposta de arquitetura de detecção de intrusão, foi desenvolvido um protótipo na plataforma FITS[2] (*Future Internet Testbed with Security*) [53–55] que é uma rede de testes interuniversitária para experimentação de propostas para a Internet do Futuro. O FITS possui nós distribuídos geograficamente em universidades brasileiras e europeias para o desenvolvimento de experimentações em redes de nova geração. Esta plataforma permite a criação e o gerenciamento de diferentes redes virtuais. As principais características da plataforma FITS são o isolamento de redes virtuais, o acesso seguro do gerenciamento da rede e a diferenciação da Qualidade de Serviço entre redes virtuais. O FITS é baseado nos mecanismos Xen e OpenFlow para prover uma arquitetura pluralista, permitindo a coexistência de múltiplas redes em paralelo executando aplicações diferentes. Na plataforma FITS o plano de controle executa nas máquinas virtuais Xen e o encaminhamento de pacotes é desempenhado pelo OpenFlow. A arquitetura está ilustrada na Figura 4.8, na qual os nós das redes são compostos por máquinas virtuais Xen isoladas agindo como comutadores OpenFlow.

[2]A plataforma de testes FITS para Internet do Futuro http://www.gta.ufrj.br/fits.

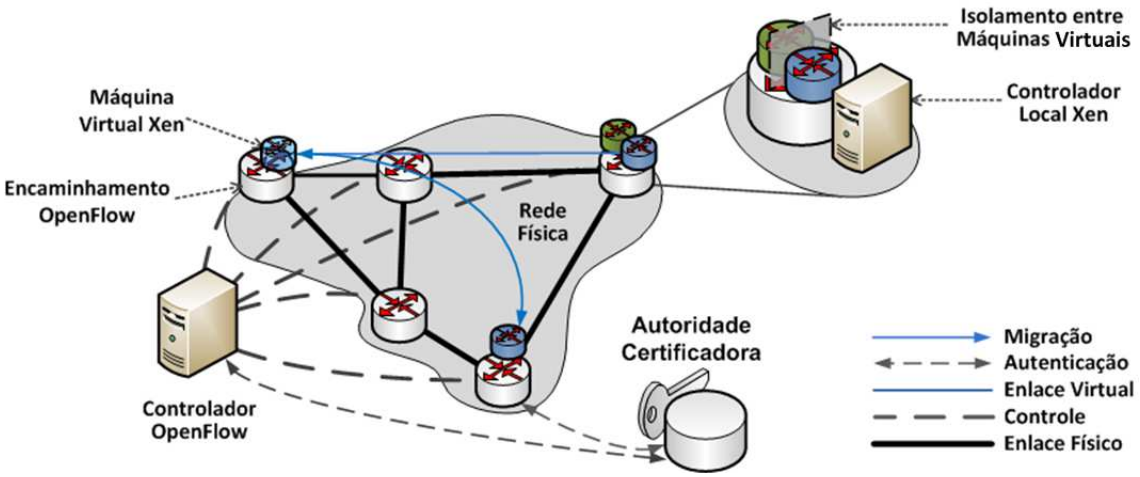


Figura 4.8: A arquitetura da plataforma FITS: roteadores virtuais com isolamento executam nos roteadores físicos usando a plataforma de virtualização Xen e encaminhamento de dados usando OpenFlow.

Foram realizados os seguintes experimentos para avaliar o correto funcionamento e o bom desempenho da arquitetura de detecção de intrusão proposta: avaliação das contramedidas da ferramenta, tanto no bloqueio de fluxos como no desvio para o pote de mel; avaliação dos recursos consumidos para análise de pacotes; e elasticidade de sobrecarga e descarga no Módulo de Detecção de Intrusão.

4.4.1 Avaliação das Contramedidas

O primeiro experimento analisa ataques de negação de serviço (DoS) por inundação de pacotes SYN em redes virtuais que são hospedadas em roteadores físicos. No ambiente virtualizado, também existem ameaças de segurança às redes virtuais, percebida por ataques aos roteadores virtuais, e ameaça à infraestrutura física, percebida por ataques ao roteador físico. O cenário considerado é constituído de um roteador físico hospedando quatro roteadores virtuais pertencentes a quatro redes virtuais diferentes. Em cada uma das máquinas virtuais correspondentes aos roteadores virtuais é instalado um sensor BroFlow, que analisa o tráfego de cada rede virtual separadamente. Além disso, um sensor BroFlow de infraestrutura é instalado no roteador físico.

Para realizar o ataque de inundação de SYN foi desenvolvido um *script*, no qual é possível regular uma taxa constante de pacotes SYN do ataque em 45, 50 ou 55 pacotes por segundo. Define-se, a critério de teste, o valor do limiar como 100 pacotes SYN por segundo. Esse valor representa a taxa de pacotes SYN permitida em cada uma das redes. Logo, o valor limiar das redes não é superado. No entanto, como os inquilinos legítimos agem em conluio, o limiar estabelecido é superado no roteador físico da infraestrutura em mais de 50%, pois a taxa agregada de conexões

por segundo das redes virtuais é totalmente encaminhada pelo roteador físico que as hospeda. Portanto, nesse experimento o limiar das redes virtuais não é extrapolado individualmente, mas o limiar agregado é considerado um ataque pelo sensor da infraestrutura. A cada instante em que o limiar é superado, um contador é incrementado para uso no algoritmo de limiar adaptativo. Sendo assim, do ponto de vista do sensor de infraestrutura, há a ocorrência de um ataque de negação de serviço à infraestrutura física de redes virtuais.

A Figura 4.9 mostra o experimento com os três atacantes enviando pacotes SYN com três taxas constantes diferentes. No momento da detecção, aproximadamente aos 40 segundos, uma mensagem é gerada pelo sensor do BroFlow presente no roteador físico do roteador atacado. Esse valor de 40 segundos é devido à implementação do algoritmo de detecção por limiar adaptativo. Como foi mencionado na Seção 4.3, esse algoritmo incrementa um contador k, quando a taxa média do período anterior for violado. Assim, se a taxa média de pacotes SYN for ultrapassada só uma vez, a contramedida não é disparada assumindo que é um falso positivo, mas se a taxa for violada quatro vezes consecutivas, representado pelo contador k, a contramedida é enviada à aplicação BroFlow POX.

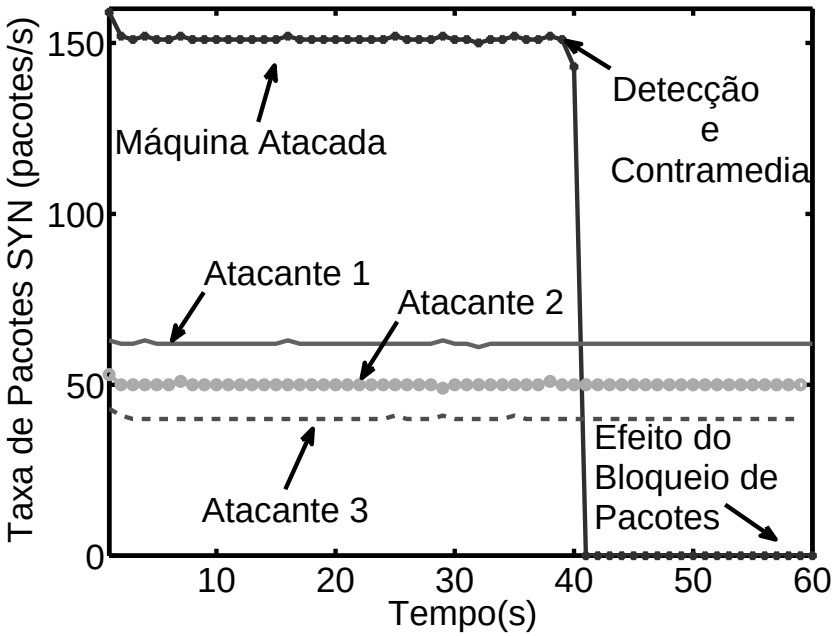


Figura 4.9: Ataques de inundação de pacotes à taxa constante e efeito da ação de bloqueio. Três ataques de inundação de SYN com uma taxa constante, detecção do ataque e efeito da ação de contramedida de bloqueio em 40 segundos.

O segundo experimento, ilustrado na Figura 4.10, avalia o algoritmo de detecção de anomalias por limiar adaptativo no mesmo cenário anterior, mas com um ataque de inundação UDP, utilizando a ferramenta de ataques DoS `Trin00`. Esse método divide a recepção de pacotes em intervalos T de 10 segundos. A taxa média de

pacotes UDP recebidos é estabelecida em 1000 pacotes por segundo. A cada vez que o valor da média do intervalo anterior é superado, um valor k é incrementado. Nesse experimento, o valor limiar de $k = 4$ foi estabelecido. Assim, sempre que os fluxos de pacotes UDP recebidos superarem a média de pacotes do intervalo anterior, o valor k é incrementado. Quando esse valor supera 4, uma mensagem de alarme é enviada ao controlador, indicando que a rede está sofrendo um ataque de inundação UDP. O controlador então realiza o bloqueio do fluxo malicioso com um tempo mínimo de reação de 40 segundos. No experimento, os valores do algoritmo por limiar adaptativo adotados foram $\mu_0 = 1000, \beta = 0.75, \alpha = 0.2$, e $T = 10$ segundos, sendo os mesmos valores utilizados por Siris e Papagalou [8] durante a avaliação dos seus algoritmos.

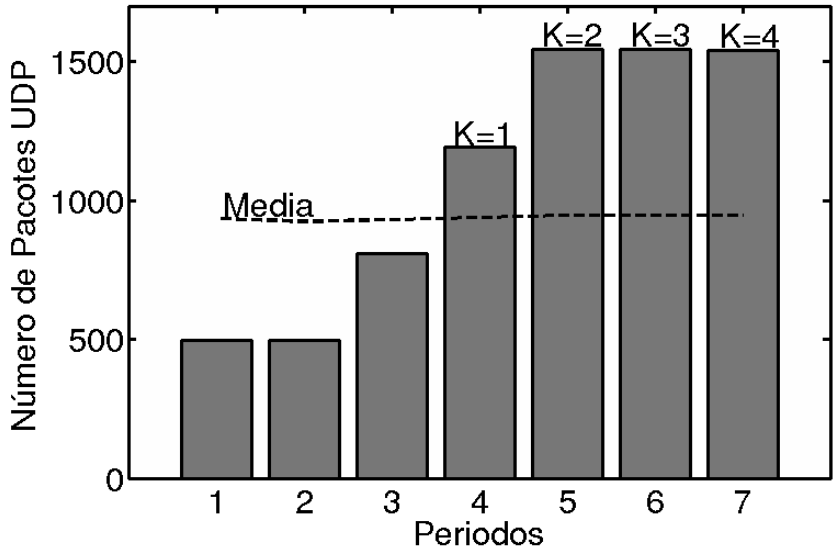


Figura 4.10: Número de pacotes UDP por período T e incremento do valor k. Comportamento da variável k do algoritmo de detecção por limiar adaptativo, durante um ataque de inundação UDP.

Este experimento analisa o efeito do bloqueio de um fluxo considerado malicioso sobre os demais fluxos da rede. O resultado das medidas está ilustrado na Figura 4.11 e mostra que quando o bloqueio do ataque acontece, o fluxo considerado legítimo não é afetado. Quando uma intrusão é detectada, o Módulo de Detecção de Intrusão manda um alerta para o controlador que atualiza a tabela de fluxos, bloqueando o fluxo malicioso em todos os comutadores OpenFlow da rede. Esta visão global da rede, característica das redes definidas por software, resulta em que o ataque seja bloqueado o mais perto possível de sua origem.

Finalmente, foi realizado uma experimentação na avaliação das contramedidas BroFlow. Para isso, foi realizado um experimento no qual o controlador desvia um

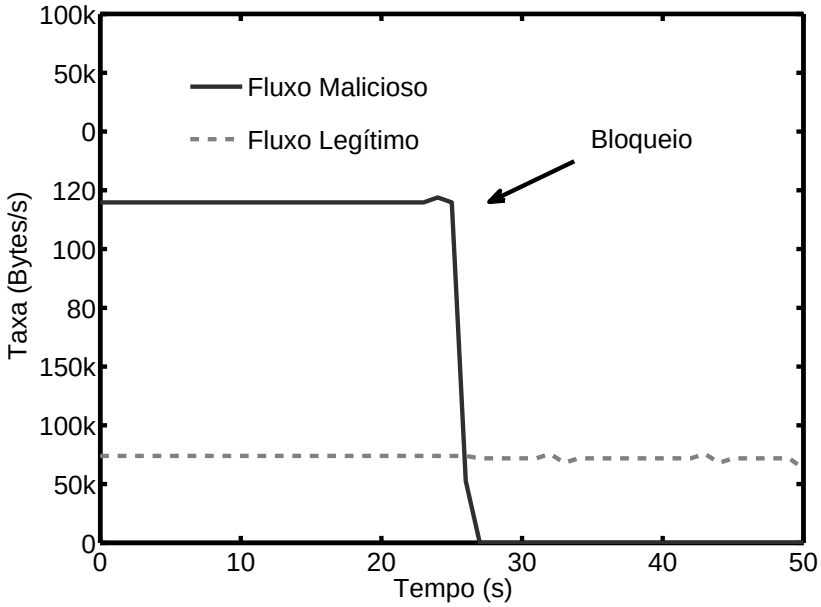


Figura 4.11: Bloqueio seletivo de fluxos. Ação eficaz de bloqueio de um fluxo malicioso de negação de serviço por inundação sem afetar o tráfego legítimo.

fluxo atacante até um pote de mel ou (*honeypot*). Como pode ser observado na Figura 4.12 o atacante matém um fluxo constante de 500 pacotes por segundo, e no momento da detecção, aproximadamente aos 40 segundos, o sensor BroFlow detecta o ataque e envia a mensagem à aplicação BroFlow POX com a contramedida de desviar o fluxo. A ferramenta utilizada como pote de mel é o *honeyd*[3] o qual é um pote de mel de alta interação *open source*. Esta ferramenta simula um sistema operativo Windows XP ou Linux Debian. Esta contramedida não é a melhor no casso de Ataques de Negação de Serviço, devido a que o desvio até outro computador, como no casso do pote de mel, o ataque segue consumindo recursos dos sistema. No entanto, a avaliação desta medida foi feita para demostrar a variabilidade das contramedidas aplicadas no sistema.

Na Figura 4.13(a) mostra a avaliação de desempenho da ferramenta sob ataque de inundação. A Figura 4.13 mostra tanto a sobrecarga introduzida pelo BroFlow, quanto a sua eficácia em bloquear um ataque de negação de serviço. A Figura 4.13(a) compara os atrasos médios de comutação de pacotes do sistema sem ataque e durante o ataque. Observa-se que o atraso inserido pelo sistema BroFlow é insignificante quando não há ataques. Já na presença de um ataque, o sistema BroFlow diminui o atraso médio devido aos descartes dos pacotes de ataque. Na Figura 4.13(b) é mostrada a taxa de transferência de pacotes. Observa-se que o BroFlow praticamente não sobrecarrega o sistema na condição sem ataque de negação de serviço atingindo a taxa máxima de 100 Mb/s. Durante o ataque de DoS, a taxa de transferência cai

[3]http://www.honeyd.org/

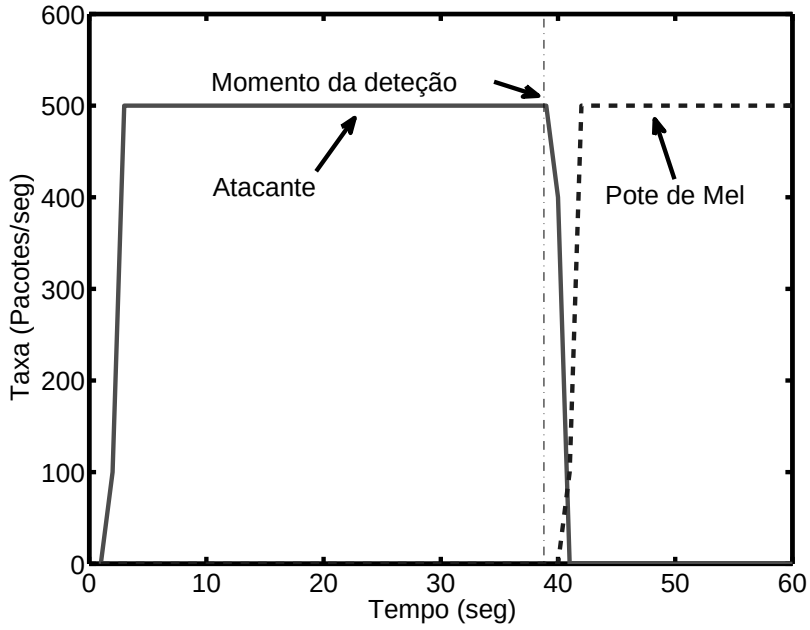
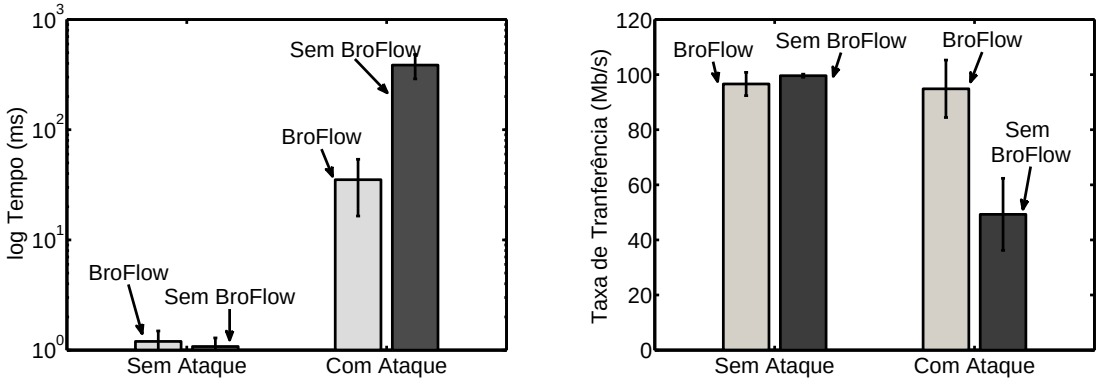


Figura 4.12: Desvio de um fluxo atacante até um pote de mel.

de 50% da taxa máxima, enquanto a taxa máxima permanece praticamente inalterada com o sistema BroFlow. Os resultados mostrados são a média durante 5 execuções com um intervalo de confiança de 95%.



(a) Atraso médio de comutação de pacotes.

(b) Comparação da taxa de transferência.

Figura 4.13: Avaliação do desempenho do atraso e da taxa de transferência do sistema BroFlow com e sem ataque. O atraso na rede é reduzido em até 10 vezes em um cenário com ataques.

4.4.2 Avaliação dos Recursos Consumidos pelo Bro em uma Máquina Virtual

O Sistema de Detecção de Intrusão utilizado captura e analisa em tempo real os pacotes espelhados e, portanto, há uma sobrecarga associada a esta função. Assim, avaliou-se o consumo de recursos da análise de pacotes pelo Sistema de Detecção de Intrusão para se determinar qual dos aspectos, banda ou processamento, é o mais crítico para a detecção deste ataque de negação de serviço por inundação. É importante ressaltar que o processamento exigido pela análise dos pacotes depende da política de segurança e de que tipos de ameaças são analisados. A técnica de Inspeção Profunda de Pacotes (*Deep Packet Inspection*–DPI) requer uma quantidade considerável de processamento. Neste experimento foram geradas taxas crescentes de pacotes e foi analisado tanto o processamento gasto pela máquina de DPI, quanto o percentual de pacotes analisados pelo sistema Bro.

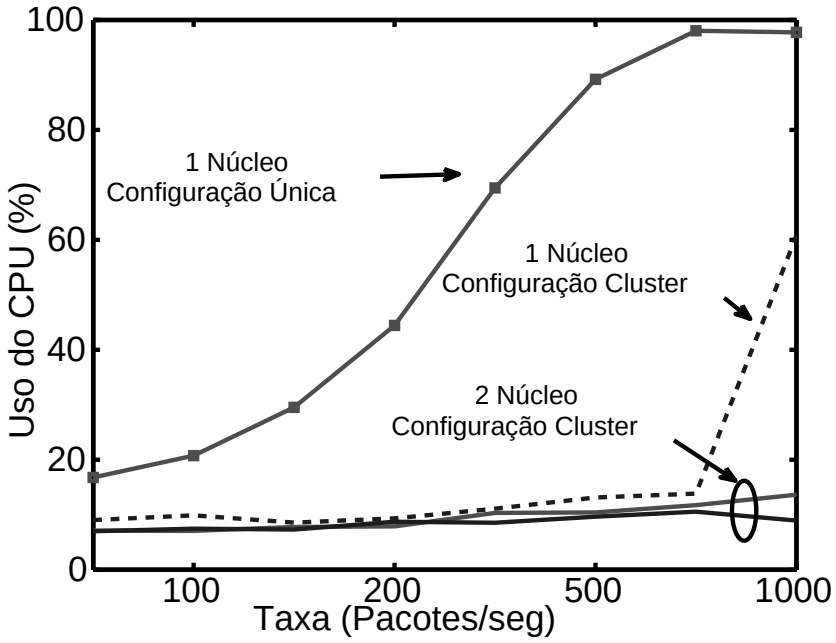


Figura 4.14: Comparação do Processamento no Bro. Sistema Bro na configuração única e *cluster* com um e dois núcleos.

Como foi explicado no Capitulo 3, o Bro não pode executar em *multi-threading*, ele só usa um único núcleo da CPU, ou pode fazer uso da técnica *multi-threading* mediante a utilização da tecnologia de *cluster* com a melhora no desempenho da biblioteca `PF_RING` em vez da `libpcap`. Assim neste experimento foi avaliado o uso de ambas tecnologias durante um ataque de DoS diretamente sobre a máquina de análises

Na configuração convencional a máquina virtual foi configurada de forma a ter acesso a somente um núcleo, para evitar desperdício de recursos. Já na configura-

ção em *cluster* foram avaliadas o uso de só um núcleo e da mesma maneira com dois núcleos. Como mostra a Figura 4.14 no casso da configuração *cluster* com dois núcleos ele gera dois traços, isto é devido a que na configuração *cluster* o Bro utiliza um núcleo do processador durante a execução. Também é importante ressaltar que nesta técnica unicamente foram analisados os consumos dos processos de CPU para os trabalhadores, que são os que inspecionam o tráfego. Não foram analisados os processos do *proxy* e o gerenciador dado que eles podem ser executados em computadores distintos.

Na experimentação foram utilizadas as duas técnicas de modo de uso do Bro tanto a configuração unica ou *standalone* como a configuração por *cluster*.

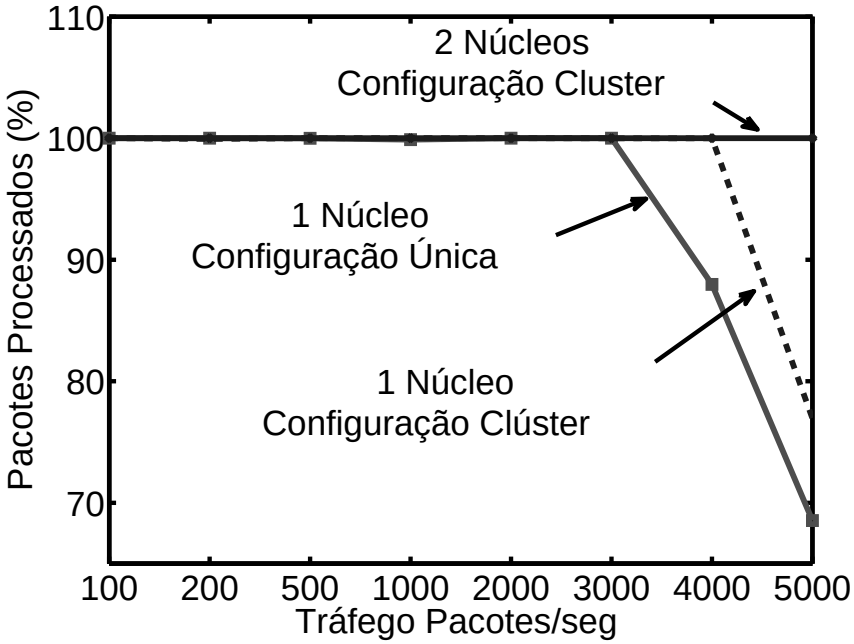


Figura 4.15: Comparação da quantidade de pacotes analisados pelo Bro. Sistema Bro na configuração única e *cluster* com um e dois núcleos.

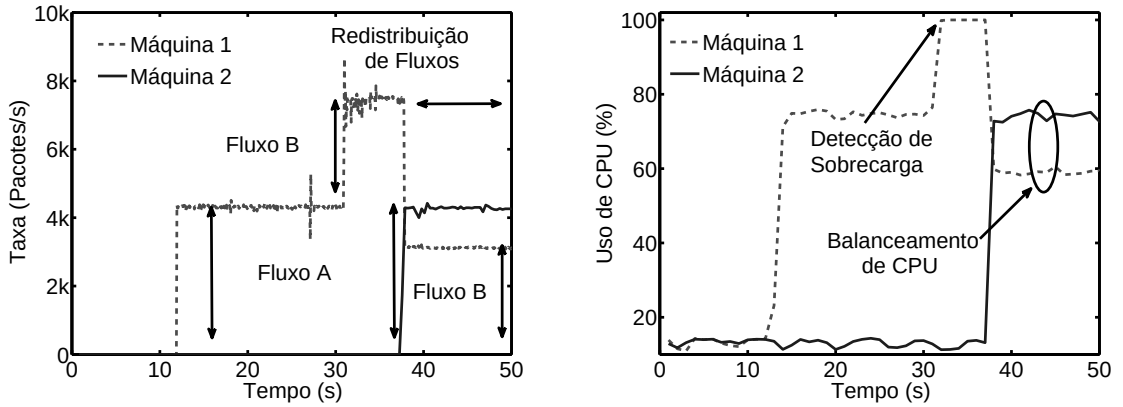
Na Figura 4.14 mostra o consumo de CPU pelos processos Bro dentro de um máquina virtual. Aqui pode ser visto o sistema tendendo a saturação utilizando todos os recursos de CPU da máquina, na configuração com um único núcleo. Assim, depois de aproximadamente 3600 pacotes por segundo o consumo de CPU chega até o 100%. No entanto, para essa mesma quantidade de pacotes, na configuração em *cluster* com um núcleo não consegue saturar em 100% e na configuração com dois núcleos o desempenho é muito melhor quase sendo desprezível o aumento de CPU na taxa máxima de 5000 pacotes por segundo. Na Figura 4.15 é mostrado a porcentagem de pacotes analisados pelo Bro, com pacotes analisados entenda-se a diferença entre pacotes enviados e pacotes que são analisados no Bro. Comparando esses valores com os da Figura 4.14, se observa que a quantidade de pacotes ana-

lisados sofre uma queda quando o processamento é máximo, este efeito é notório com a configuração de um núcleo único chegando a analisar, no máximo, apenas 70% dos pacotes na taxa máxima de 5000 pacotes por segundos. Os resultados são bem melhores na configuração em *cluster*, já que com um núcleo só o análises na taxa de 500 pacotes por segundo é do 80%, no entanto na configuração de com dois núcleos, nesta taxa não existe perdas. Como pode se observar, o sistema Bro é uma excelente escolha como analisador de tráfego, já que mesmo durante um ataque ao analisador consegue de maneira robusta analisar todos os pacotes enviados. Estes resultados foram semelhantes aos obtidos por Weaver e Sommer no seu artigo [56]. Concluindo, a ferramenta Bro tem evoluído no seu desempenho fazendo uma otimização dos recursos utilizados na configuração *cluster* mediante o uso da biblioteca `PF_RING`. No entanto, na proposta desta dissertação é feita a detecção utilizando unicamente uma máquina Bro com um único processador na configuração única ou *standalone*.

4.4.3 Sobrecarga na Detecção de Intrusão

Este experimento visa avaliar o desempenho da arquitetura em caso de sobrecarga nas máquinas analisadoras de tráfego como foi explicado anteriormente. Para este experimento, foram geradas taxas de pacotes constantes que eram inspecionados pela única máquina de análise ativa no momento devido à baixa demanda. Em seguida, um novo fluxo foi criado, de forma a sobrecarregar esta máquina. Com isso, a máquina física onde a máquina de análise estava localizada enviou uma mensagem de sobrecarga para o controlador. Neste experimento, a sobrecarga foi avaliada pelo processamento da máquina com o Bro, pois, como visto na Seção 4.4.2, o processamento satura antes da banda passante saturar. Quando o controlador recebe a mensagem de sobrecarga, ele executa o algoritmo de balanceamento para decidir em qual máquina física deve ser instanciada a nova máquina de detecção de intrusão. Após a ativação da nova máquina, os fluxos são redistribuídos levando em conta sua origem e os recursos de cada máquina de análise de pacotes.

Os resultados ilustrados na Figura 4.16(a) mostram que a partir do segundo fluxo ocorre uma sobrecarga na máquina de análises. Quando a sobrecarga é detectada, existe um intervalo de tempo, devido à instanciação da nova máquina, até que os fluxos sejam redistribuídos. Após o balanceamento dos fluxos, todos os pacotes passam a ser analisados sem saturar a máquina de análises. Neste experimento é avaliada uma análise temporal das máquinas de análises Bro em caso de sobrecarga. São iniciados dois fluxos na máquina 1, Figura 4.16(a), causando sobrecarga de CPU, Figura 4.16(b). Para evitar a saturação da máquina 1, a máquina 2 é ativada e os fluxos são redistribuídos, balanceando o consumo de CPU.

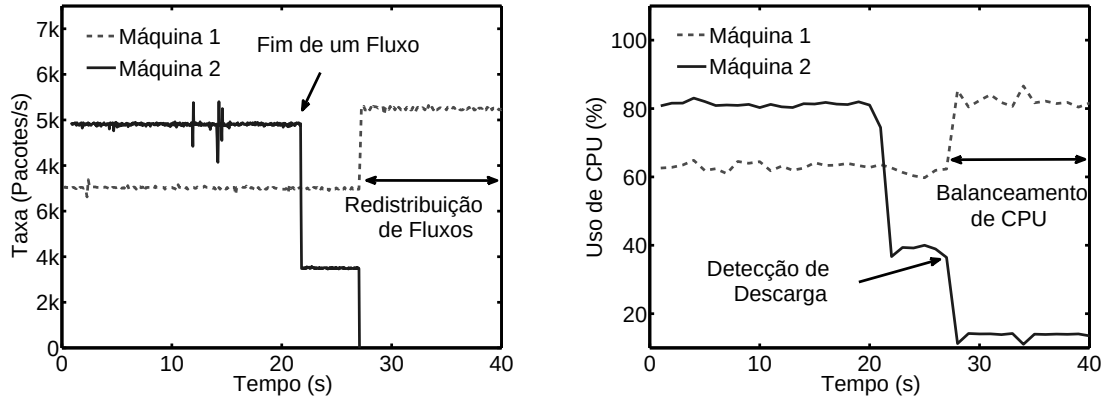


(a) Taxa de pacotes recebida pelas máquinas Bro. (b) Consumo de CPU de cada máquina Bro.

Figura 4.16: Análise do processamento e fluxos em cenário de sobrecarga da máquina de análise.

Descarga na Detecção de Intrusão

Este experimento foi feito para medir a elasticidade da arquitetura. Caso haja recursos ociosos, a proposta visa redistribuir os fluxos de forma que uma máquina possa ser desativada. Os recursos consumidos são constantemente analisados e informados ao controlador para que ele possa detectar quando um remanejamento de fluxos irá permitir a desativação de uma das máquinas do Módulo de Detecção de Intrusão.



(a) Taxa de pacotes recebida pelas máquinas Bro. (b) Consumo de CPU de cada máquina Bro.

Figura 4.17: Análise do processamento e fluxos em cenário de descarga da máquina de análise.

O teste começa com duas máquinas virtuais de analises recebendo taxas constantes de pacotes, como mostrada na Figura 4.17(a). Após um período de tempo, um desses fluxos foi cancelado, causando uma queda perceptível no processamento de uma das máquinas. Ao detectar esta descarga, o controlador redistribui os fluxos

de forma que a máquina com menor processamento passe a receber nenhum fluxo e, portanto, possa ser desativada. Nas figuras é mostrada a análise temporal do Módulo de Detecção de Intrusão em caso de descarga. Duas máquinas de analises Bro estão operando quando um dos fluxos termina, Figura 4.17(a). Então, o controlador analisa o consumo de CPU das duas máquinas, Figura 4.17(b), e redistribui os fluxos de forma a poder desativar uma das máquinas.

Capítulo 5

Conclusões

As vulnerabilidades representadas pelo grande crescimento e complexidade da Internet são exploradas por usuários mal intencionados. Tanto usuários legítimos, autenticados e autorizados, como usuários ilegítimos fazem uso de diversas técnicas para explorar essas vulnerabilidades mediante ataques. Em especial, os ataques de negação de serviço por inundação representam um grande problema na internet. Durante o ano de 2013 foi realizado o maior cyber ataque da história, com taxas de até 300 Gb/s, causando perdas milionárias e negações completas de serviços ou recursos aos usuários legítimos. Os métodos convencionais de segurança que se baseiam em barreiras de proteção tais como *firewall* e mecanismos de controle de acesso acabam ficando obsoletos ante o rápido avanço das ameaças.

Para proteger aos usuários dos possíveis danos causados pelos ataques e intrusões, os sistemas de Detecção e Prevenção de Intrusão se servem de análises detalhadas de tráfego de rede o qual é comparado com as assinaturas de ataques conhecidos ou comportamento suspeito, alertando ao operador de rede ou realizando ações ante o ataque. Esses sistemas permanecem vulneráveis ante a sobrecarga de pacotes, sendo um ponto crítico nas análises, já que pacotes de potenciais ataques deixam de ser analisados.

Nesta dissertação foi apresentado o BroFlow, um sistema de detecção e prevenção intrusão, em especial para ataques de negação de serviço por inundação de pacotes, para redes definidas por software. O BroFlow une a eficiência, a flexibilidade e a simplicidade de elaboração de políticas de segurança, providas pela ferramenta Bro de análise de tráfego, com as características da visão global e da agilidade de ação em toda a rede oferecidas pelo OpenFlow. As contribuições da dissertação são evidenciadas através do desenvolvimento de um protótipo e as avaliações realizadas. O protótipo do sistema demonstra o funcionamento dos diferentes algoritmos de detecção de intrusão por anomalia. O Sistema BroFlow implementa uma arquitetura de políticas modular na qual podem ser implementados diversos algoritmos de detecção por anomalia. Ao tomar ações imediatas para interromper o ataque, o pro-

tótipo mostrou ser altamente efetivo na detecção de ataques de negação de serviço por inundação e também eficaz na ação de bloqueio contra os ataques, tendo como vantagem o bloqueio e a possibilidade de desvio até máquinas de analises ou potes de mel de fluxos seletivos, considerados anômalos. O sistema descarta os pacotes de ataque ainda na sua origem o que permite aumentar a disponibilidade da rede. O sistema mostrou as diferentes implementações do monitor, mostrando que ante ataques de negação de serviço contra os próprios sensores o sistema não apresenta sobrecarga de recursos nem perda de pacotes. O posicionamento estratégico dos sensores de ataques permite ao sistema reduzir em até dez vezes o atraso na rede sob ataque e garante o encaminhamento de pacotes úteis na taxa máxima do enlace, ao custo de acrescentar um atraso praticamente desprezível na rede em um cenário sem ataques. Vale ressaltar, que os experimentos com o protótipo evidenciam que o posicionamento estratégico de sensores de ataques na infraestrutura física assegura a proteção da infraestrutura física contra ataques provenientes de inquilinos legítimos.

Além disso, mediante técnicas de elasticidades, máquinas de análise de pacotes são instanciadas e desativadas de maneira dinâmica em relação aos consumos de CPU, evitando que pacotes deixem de ser inspecionados fazendo uma otimização dos recursos. O comportamento da arquitetura proposta foi analisado para os casos de sobrecarga e descarga das máquinas de Detecção de Intrusão. No caso da sobrecarga, a distribuição dos fluxos entre duas máquinas, permitiu que todos os pacotes fossem analisados, já que ouve um balanceamento de CPU. No caso de descarga, o fluxo foi redistribuído de forma que uma máquina parasse de receber fluxos, o que permite a desativação dessa máquina de analises. Portanto, a arquitetura fornece recursos de acordo com a demanda, ou seja, realiza a prevenção de intrusão de forma elástica.

Como trabalhos futuros, está prevista a proposta de novos algoritmos para a detecção de um maior espectro de ataques e introduzir novas contramedidas que se sirvam da característica peculiar de controle global para correlacionar os alarmes de diferentes sensores BroFlow. Para isso, se explorará os Sistemas de IDS Colaborativos contra ataques distribuídos como o realizado por Fung e Boutaba em [20]. Além disso, como foi estabelecido previamente, se abordará o problema de otimização matemática na localização específica dos sensores BroFlow. Além disso, pretende-se integrar a arquitetura ao FITS, para fornecer uma ferramenta de prevenção de intrusão elástica como serviço para novas propostas de experimentação para a Internet do Futuro.

Como resultado deste trabalho foram publicados um artigo nacional [14], um artigo a um congresso nacional [12] e uma publicação em um *workshop* internacional [57].

Referências Bibliográficas

[1] LANGNER, R. "Stuxnet: Dissecting a Cyberwarfare Weapon", *IEEE Security and Privacy*, v. 9, n. 3, pp. 49–51, maio 2011.

[2] FERNANDES, N., MOREIRA, M., MORAES, I., et al. "Virtual networks: Isolation, performance, and trends", *Annals of Telecommunications*, pp. 1–17, 2010.

[3] MATTOS, D. M. F., DUARTE, O. C. M. B. "QFlow: Um Sistema com Garantia de Isolamento e Oferta de Qualidade de Serviço para Redes Virtualizadas". In: *XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos SBRC'12*, abr. 2012.

[4] GUIMARÃES, P. H. V., MURILLO P., A. F., ANDREONI L., M. E., et al. "Comunicação em Redes Elétricas Inteligentes: Eficiência, Confiabilidade, Segurança e Escalabilidade". In: *Minicursos do XXXI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos SBRC'13*, pp. 101–164, maio 2013.

[5] MCHUGH, J., CHRISTIE, A., ALLEN, J. "The role of intrusion detection systems", *Washington Post*, 2000.

[6] LYNCH, D. M. "Securing against insider attacks", *Information Systems Security*, v. 15, n. 5, pp. 39–47, 2006.

[7] SCARFONE, K., MELL, P. "Guide to Intrusion Detection and Prevention Systems", *NIST Special Publication*, v. 800, pp. 94, 2007.

[8] SIRIS, V. A., PAPAGALOU, F. "Application of anomaly detection algorithms for detecting SYN flooding attacks", *Computer communications*, v. 29, n. 9, pp. 1433–1442, 2006.

[9] CAMPISTA, M., RUBINSTEIN, M., MORAES, I., et al. "Challenges and Research Directions for the Future Internetworking", *IEEE Communications Surveys Tutorials*, v. 16, n. 2, pp. 1050–1079, fev. 2014.

[10] MCKEOWN, N., ANDERSON, T., BALAKRISHNAN, H., et al. "OpenFlow: enabling innovation in campus networks", *SIGCOMM Computer Communication*, mar. 2008.

[11] LANTZ, B., HELLER, B., MCKEOWN, N. "A network in a laptop: rapid prototyping for software-defined networks". In: *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, p. 19. ACM, 2010.

[12] ANDREONI LOPEZ, M., DA ROCHA FIGUEIREDO, U., LOBATO, A. G. P., et al. *BroFlow: Um Sistema Eficiente de Detecção e Prevenção de Intrusão em Redes Definidas por Software*. Relatório Técnico 14-13, Grupo de Teleinformática e Automação-GTA, Programa de Engenharia Elétrica, COPPE/UFRJ, abr. 2014.

[13] SOMMER, R. "Bro: An Open Source Network Intrusion Detection System." In: *DFN-Arbeitstagung über Kommunikationsnetze*, pp. 273–288, 2003.

[14] LOBATO, A. G. P., DA ROCHA FIGUEIREDO, U., ANDREONI LOPEZ, M., et al. "Uma Arquitetura Elástica para Prevenção de Intrusão em Redes Virtuais usando Redes Definidas por Software". In: *XXXII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos- SBRC 2014*, pp. 1–14, maio 2014.

[15] AXELSSON, S. *Intrusion detection systems: A survey and taxonomy*. Relatório Técnico 99-15, Department of Computer Engineering, Chalmers University, mar. 2000.

[16] SOMMER, R., PAXSON, V. "Enhancing Byte-level Network Intrusion Detection Signatures with Context". In: *Proceedings of the 10th ACM Conference on Computer and Communications Security*, CCS'03, pp. 262–271. ACM, 2003.

[17] KUMAR, S., DHARMAPURIKAR, S., YU, F., et al. "Algorithms to accelerate multiple regular expressions matching for deep packet inspection". SIGCOMM'06, pp. 339–350, 2006.

[18] MCHUGH, J. "Intrusion and intrusion detection", *International Journal of Information Security*, v. 1, n. 1, pp. 14–35, 2001.

[19] LAZAREVIC, A., KUMAR, V., SRIVASTAVA, J. "Intrusion detection: A survey". In: *Managing Cyber Threats*, Springer, pp. 19–78, 2005.

[20] FUNG, C., BOUTABA, R. "Design and Management of Collaborative Intrusion Detection Networks". In: *Proceedings of the 13th IFIP/IEEE Integrated Network Management Symposium (IM 2013)*, Ghent, Belgium, maio 2013.

[21] SOMMER, R., PAXSON, V. "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection". In: *IEEE Symposium on Security and Privacy*, pp. 305–316, 2010.

[22] OIKONOMOU, G., MIRKOVIC, J. "Modeling human behavior for defense against flash-crowd attacks". In: *IEEE International Conference on Communications ICC'09*, pp. 1–6. IEEE, 2009.

[23] JONES, A. K., SIELKEN, R. S. "Computer system intrusion detection: A survey", *Computer Science Technical Report*, pp. 1–25, 2000.

[24] PETER, E., SCHILLER, T. "A Practical Guide to Honeypots", *Washington Univerity*, 2011.

[25] PTACEK, T. H., NEWSHAM, T. N. *Insertion, evasion, and denial of service: Eluding network intrusion detection*. Relatório Técnico 074-0188, Secure Networks Inc., 1998.

[26] DEL CARLO, C. "Intrusion detection evasion", *SANS Institute InfoSec Reading Room*, maio 2003.

[27] IBM. *Real Secure Network*, acessado em maio de 2014. `http://www.ibm.com/realsecure.pdf/`.

[28] IBM. *Intrusion Prevention System Virtual Appliance*, acessado em maio de 2014. `http://www.ibm.com/IPSVA.pdf/`.

[29] CISCO. *Cisco Anomaly Guard Module*, acessado em maio de 2014. `http://www.cisco.com/anomaly/`.

[30] RADWARE. *DefenseFlow Resources*. Radware, acessado em maio de 2014. `http://www.radware.com/Products/DefenseFlow/`.

[31] MCAFFE. *Stonesoft IPS 5.6*, acessado em maio de 2014. `http://www.stonesoft.com/ips.pdf/`.

[32] ROESCH, M. "Snort: Lightweight Intrusion Detection for Networks." In: *Proceedings of LISA: 13th Systems Administration Conference*, v. 99, pp. 229–238, 1999.

[33] PAXSON, V. “Bro: a System for Detecting Network Intruders in Real-Time”, *Computer Networks*, v. 31, n. 23-24, pp. 2435–2463, 1999.

[34] PIHELGAS, M. *A Comparative Analysis of Open-Source Intrusion Detection Systems*. Tese de Mestrado, Tallinn University of Technology, 2012.

[35] SOMMER, R., PAXSON, V. “Enhancing Byte-level Network Intrusion Detection Signatures with Context”. In: *Proceedings of the 10th ACM Conference on Computer and Communications Security*, CCS ’03, pp. 262–271. ACM, 2003.

[36] PAXSON, V., SOMMER, R., WEAVER, N. “An architecture for exploiting multi-core processors to parallelize network intrusion prevention”. In: *Sarnoff Symposium, 2007 IEEE*, pp. 1–7. IEEE, 2007.

[37] FUSCO, F., DERI, L. “High Speed Network Traffic Analysis with Commodity Multi-core Systems”. In: *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC ’10, pp. 218–224. ACM, 2010.

[38] PORRAS, P., SHIN, S., YEGNESWARAN, V., et al. “A security enforcement kernel for OpenFlow networks”. In: *Proceedings of the first workshop on Hot topics in software defined networks*, pp. 121–126. ACM, ago. 2012.

[39] SHIN, S., PORRAS, P., YEGNESWARAN, V., et al. “FRESCO: Modular composable security services for software-defined networks”. In: *Proceedings of Network and Distributed Security Symposium*, fev. 2013.

[40] KIM, H., GUPTA, A., SHAHBAZ, M., et al. “Simpler Network Configuration with State-Based Network Policies”, *Georgia Institute of Technology, Atlanta, USA*, 2013.

[41] MATTOS, D. M. F., FERRAZ, L. H. G., DUARTE, O. C. M. B. “Um Mecanismo para Isolamento Seguro de Redes Virtuais Usando a Abordagem Híbrida Xen e OpenFlow”. In: *XIII SBSeg’13*, pp. 128–141, nov. 2013.

[42] SHIN, S., YEGNESWARAN, V., PORRAS, P., et al. “AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks”. In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 413–424. ACM, abr. 2013.

[43] BALLARD, J. R., RAE, I., AKELLA, A. “Extensible and scalable network monitoring using opensafe”, *Proc. INM/WREN*, 2010.

[44] BRAGA, R., MOTA, E., PASSITO, A. "Lightweight DDoS flooding attack detection using NOX/OpenFlow". In: *IEEE 35th Conference on Local Computer Networks*, pp. 408–415, out. 2010.

[45] MEHDI, S. A., KHALID, J., KHAYAM, S. A. "Revisiting traffic anomaly detection using software defined networking". In: *Recent Advances in Intrusion Detection*, pp. 161–180. Springer, 2011.

[46] CHUNG, C., KHATKAR, P., XING, T., et al. "NICE: Network Intrusion Detection and Countermeasure Selection in Virtual Network Systems", *IEEE Transactions on Dependable and Secure Computing*, v. 10, n. 4, pp. 198–211, ago. 2013.

[47] XING, T., HUANG, D., XU, L., et al. "SnortFlow: A OpenFlow-Based Intrusion Prevention System in Cloud Environment". In: *2nd GENI Research and Educational Experiment Workshop*, pp. 89–92, out. 2013.

[48] NAGAHAMA, F. Y., FARIAS, F., AGUIAR, E., et al. "IPSFlow Uma Proposta de Sistema de Prevenção de Intrusão Baseado no Framework OpenFlow". In: *III Workshop de Pesquisa Experimental da Internet do Futuro WPEIF-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos SBRC'12*, pp. 42–47, maio 2012.

[49] EGI, N., GREENHALGH, A., HANDLEY, M., et al. "Towards high performance virtual routers on commodity hardware". In: *Proceedings of the 2008 ACM CoNEXT Conference*, pp. 1–12. ACM, 2008.

[50] WANG, G., NG, T. E. "The impact of virtualization on network performance of amazon ec2 data center". In: *INFOCOM, 2010 Proceedings IEEE*, pp. 1–9. IEEE, 2010.

[51] PFAFF, B., PETTIT, J., KOPONEN, T., et al. "Extending networking into the virtualization layer". In: *8th ACM Workshop on Hot Topics in Networks-HotNets*. Citeseer, 2009.

[52] SHALIMOV, A., ZUIKOV, D., ZIMARINA, D., et al. "Advanced Study of SDN/OpenFlow Controllers". In: *Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia*, CEE-SECR '13, pp. 1–6. ACM, 2013.

[53] GUIMARAES, P. H. V., FERRAZ, L. H. G., TORRES, J. V., et al. "Experimenting Content-Centric Networks in the future internet testbed environment". In: *IEEE ICC Workshops*, pp. 1383–1387, 2013.

[54] MATTOS, D. M. F., MAURICIO, L. H., CARDOSO, L. P., et al. "Uma Rede de Testes Interuniversitária com Técnicas de Virtualização Híbridas". In: *XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos SBRC'12*, maio 2012.

[55] MORAES, I. M., MATTOS, D. M., FERRAZ, L. H. G., et al. "FITS: A Flexible Virtual Network Testbed Architecture", *Computer Networks*, 2014.

[56] WEAVER, N., SOMMER, R. "Stress testing cluster Bro". In: *DETER workshop*, pp. 1–4, 2007.

[57] ANDREONI LOPEZ, M., DUARTE, O. C. M. B. "Detecting and Preventing Intruders for Cyber-Security". In: *Proceedings of the Second Workshop on Network Virtualization and Intelligence for the Future Internet-WNetVirt'13*, out. 2013.