Abordagem autonômica para mitigar ciberataques em LANs

Luiz Arthur Feitosa dos Santos[1,2], **Rodrigo Campiolo**[1,2],*
Wagner Aparecido Monteverde[1], **Daniel Macêdo Batista**[2]

[1]Dept. Acad. Ciência da Computação – Universidade Tecnológica Federal do Paraná
[2]Instituto de Matemática e Estatística – Universidade de São Paulo

luizsantos@utfpr.edu.br,rcampiolo@utfpr.edu.br,
wagner.ap.monteverde@gmail.com e batista@ime.usp.br

Abstract. *It is a hard task for a human to deal with cyber attacks, mainly due the increasing number of users and heterogeneous devices in LAN encouraged by practices such as BYOD, that constantly brings new threats to these environments. Therefore, we propose an autonomic approach that uses machine learning to process, in a similar way of human memory, the historical of network usage and security alerts, to generate security rules, that are imposed to network using SDN resources, to mitigate cyber attacks. Experiments have shown that the approach was able to avoid 97.3% of malicious packets sent to a victim in DDoS TCP-SYN attacks and to mitigate 44 types of cyber threats, in a real network.*

Resumo. *Lidar com ciberataques em LANs tem se tornado uma árdua tarefa para o ser humano, principalmente devido a agregação crescente de usuários e dispositivos heterogêneos incentivada por práticas como BYOD, que trazem constantemente novas ameaças à segurança das LANs. Assim, propomos uma abordagem que emprega aprendizagem de máquina para processar, similarmente à memória humana, históricos de uso da rede e alertas de segurança, para extrair regras de segurança que são aplicadas autonomicamente na rede, por intermédio da tecnologia OpenFlow. Em ambiente simulado, a abordagem proposta foi capaz de mitigar 97,3% dos pacotes maliciosos em ataques DDoS TCP-SYN, já em uma rede real ajudou a mitigar 44 tipos de ciberameaças.*

1. Introdução

Práticas como *Bring Your Own Device* (BYOD) tornam as redes de computadores locais mais dinâmicas e amigáveis para o usuário final mas, em contrapartida, adicionam novas vulnerabilidades que são utilizadas frequentemente contra a segurança da rede [Symantec 2015]. O uso de mecanismos de segurança tradicionais, tais como *firewall* e *Intrusion Detection System* (IDS), normalmente exigem dedicação e perspicácia do administrador, que precisa constantemente analisar o histórico de uso da rede e alertas de segurança para manter um nível de segurança condizente com as ameaças. Todavia, muitas vezes o responsável pela rede local (*Local Area Network* – LAN) acumula cargos e por isto não tem tempo hábil para gerenciar a segurança de maneira apropriada. Em casos mais extremos, algumas LANs não possuem um responsável direto e/ou competente pela segurança do ambiente. Como consequência, a detecção e o tempo de reação às ameaças é

*Agradecimentos à Fundação Araucária, Secretaria de Estado da Ciência, SETI-PR, ao Governo do Estado do Paraná e a Rede Nacional de Pesquisa (RNP), pelo apoio a esta pesquisa e participação no evento. Agradecimento especial ao Romildo Martins da Silva (*In Memoriam*), pela contribuição.

muitas vezes elevado, o que pode comprometer o bom funcionamento da rede por longos períodos de tempo.

Há várias abordagens para facilitar o gerenciamento das redes e sua segurança. Por exemplo, tecnologias como Redes Definidas por Software (*Software-Defined Networks* – SDN) viabilizam o gerenciamento e acesso a novos fluxos de informação das redes. Com o OpenFlow [McKeown et al. 2008], um protocolo para SDN, é possível programar dinamicamente a rede para que *switches* insolem fluxos de redes, bloqueiem conexões indesejadas e implementem qualidade de serviço. Já a Computação Autonômica (CA) possibilita que sistemas se tornem auto-gerenciáveis, o que é possível através de ciclos autonômicos que monitoram, e caso necessário, adaptam o ambiente visando a proteção e a normalidade dos recursos sem a necessidade de interação humana [Hariri et al. 2006].

Neste contexto, este artigo apresenta uma abordagem autonômica que tem por objetivo retirar parte do ônus do administrador em manter a segurança de LANs. O presente trabalho inova pois propõe métodos que empregam algoritmos de análise de associação para processar históricos de uso da rede e alertas de segurança, extraindo então regras de segurança que são aplicadas autonomicamente na rede, por intermédio de SDN. Outra contribuição é a criação de um método inspirado na estrutura da memória humana, que permite gerar autonomicamente regras de segurança que: (a) mitigam problemas de segurança na rede; (b) previnem contra ameaças futuras; e (c) evitam que *hosts* e/ou serviços da rede sejam influenciados por ataques e pela própria defesa autonômica, em caso de falsos positivos.

Apresenta-se na Seção 2 os trabalhos relacionados, na Seção 3 a abordagem e métodos usados para implementá-la, na Seção 4 os experimentos usados para avaliar o desempenho da abordagem e os resultados obtidos, e na Seção 5 as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

O conceito de SDN tem possibilitado novos tipos de aplicações para redes de computadores e segurança [Lara et al. 2013]. A reação autonômica contra ameaças aos recursos e serviços das redes também vem sendo abordada por diversos trabalhos [Yuan et al. 2014].

Dentre os trabalhos que utilizam OpenFlow para proteger redes, [Chung et al. 2013] propõem NICE, uma abordagem baseada em SDN para detecção e contenção de tentativas de comprometimento de máquinas virtuais que podem se tornar zumbis. NICE possui agentes baseados no IDS Snort que analisam o tráfego de cada servidor e realizam varreduras de vulnerabilidades; [Wang et al. 2013] propõem o NetFuse, um mecanismo para detectar e mitigar fluxos suspeitos que sobrecarregam redes que usam OpenFlow. Os fluxos identificados são roteados para um mecanismo de contenção, que pode, por exemplo, causar atrasos ou remoção de pacotes. Além disso, NetFuse possui um controle adaptativo inspirado em sistemas biológicos toxina-antitoxina, principalmente para lidar com falsos positivos; [Lara and Ramamurthy 2014] desenvolveram o OpenSec, que permite ao administrador especificar políticas de segurança reagindo a alertas de segurança usando OpenFlow. No OpenSec, o controlador encaminha os fluxos definidos nas políticas para unidades de processamento externas que realizam inspeção e devolvem alertas de segurança para o controlador selecionar uma reação.

Quanto aos trabalhos de CA que se destacam dentro do nosso contexto,

[Barrère et al. 2011, Barrère et al. 2012] afirmam que as redes autonômicas podem gerar novas vulnerabilidades e desta forma propõem um modelo para traduzir descrições de vulnerabilidades de elementos autonômicos em regras para políticas de segurança. Tais regras são interpretadas por um sistema autonômico que deve detectar vulnerabilidades e executar operações para sanar o problema; [Sibai and Menasce 2011, Sibai and Menasce 2012] propõem um arcabouço chamado AVPS, que tem por objetivo a auto-proteção contra ameaças internas que violem a política de segurança descrita pelo administrador da rede. O AVPS utiliza o Snort com políticas personalizadas para proteger sistemas contra abusos cometidos por usuários com privilégios ou contra configurações incorretas.

Já em relação a trabalhos que utilizam conjuntamente CA e SDN, [Chu et al. 2010] propuseram um método autonômico para mitigar ataques distribuídos de negação de serviço (*Distributed Denial of Service* – DDoS) por meio da contagem de volume de tráfego em *switches* OpenFlow; [Chen et al. 2014] introduziram um modelo de gerenciamento de segurança que emprega CA e OpenFlow para estimar, detectar e reagir prematuramente contra ataques à segurança em ambientes de Internet das Coisas (Internet *of Things* – IoT).

O presente trabalho se diferencia dos demais por propor o uso de CA e SDN para proteger LANs contra ciberataques. Este trabalho inova ao criar métodos que empregam aprendizagem de máquina não supervisionada para identificar padrões de uso e ataques à rede, para então gerar regras de segurança que mitigam problemas de segurança em LANs, sem intervenção humana. Não faz parte do escopo deste trabalho propor/melhorar mecanismos de detecção de intrusão. Trabalha-se com a hipótese de que os mecanismos de detecção atuais são eficientes e possuem uma taxa aceitável de falsos positivos.

2.1. Arquitetura Base

Em [Santos et al. 2014], propomos uma arquitetura autonômica para proteger LANs contra ciberataques (Figura 1). Tal arquitetura serve de base para o desenvolvimento deste trabalho e é implementada na ferramenta chamada *OpenFlow Intrusion Detection and Prevention System* (Of-IDPS) – arquitetura e o nome da ferramenta são usadas como sinônimos.

O processamento proposto na arquitetura, inicia com os sensores coletando dados a respeito do uso da rede e seus distúrbios (passo 1 da Figura 1). Estes dados vem do IDS Snort e dos *switches* OpenFlow. Na sequência o módulo Monitor normaliza os dados coletados pelos sensores (passo 2) e ajuda na identificação de alertas de segurança (passo 3), que podem ser gerados a partir da análise dos dados vindos do OpenFlow a respeito do uso da rede – os dados fornecidos pelo Snort já são considerados alertas de segurança e por isto só requerem normalização. Os alertas de segurança gerados são processados pelo módulo Análise e Planejamento, que cria planos de ações para remediar os problemas identificados (passo 4). O módulo Execução converte o plano de ações em regras de segurança, que são implantadas na rede com dois propósitos: mitigar fluxos de redes maliciosos, que já estão instalados e ativos na rede (passo 5.1); repudiar ou restringir a comutação de novos fluxos, quando estes estiverem relacionados com ataques à segurança (passo 5.2). O módulo Execução aplica as regras de segurança por intermédio dos atuadores (passo 6), que na prática são os *switches* e o controlador OpenFlow. Assim, os

fluxos legítimos são instalados normalmente na rede e fluxos maliciosos ou suspeitos são submetidos a contramedidas que visam resolver desequilíbrios na rede. Por fim, sensores retroalimentam a arquitetura (passo 1) fechando o ciclo autonômico infinito. Todas ações e informações geradas pela arquitetura são registradas no módulo Base de Conhecimento para auxiliar em tomadas de decisões futuras. Na Figura 1, a Base de Conhecimento está espalhada e representada pelos cilindros: Fluxos Normais, IDS, Regras Memória Sensorial, etc. Embora tenha-se usado Snort e o OpenFlow, é possível empregar outras tecnologias implementado novos sensores e atuadores.

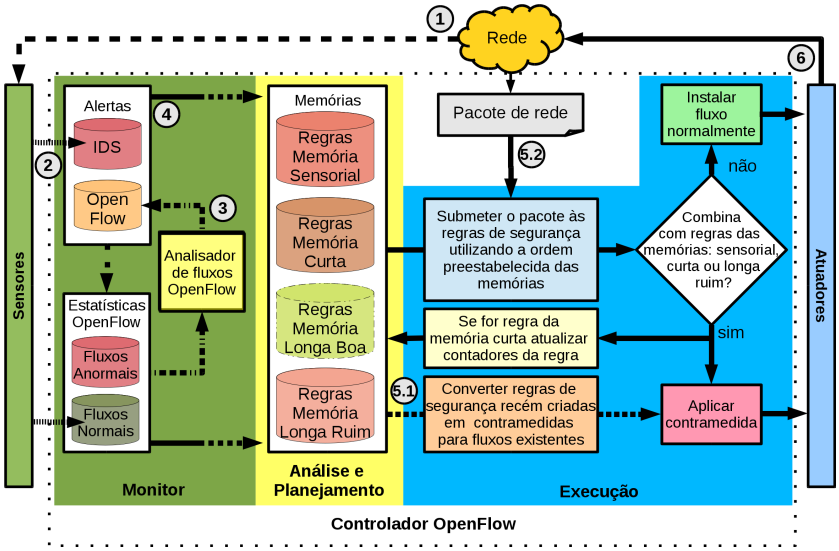

Figura 1. Arquitetura do Of-IDPS.

Em [Santos et al. 2014], o principal objetivo foi analisar e desenvolver ferramentas que permitissem monitorar e reagir contra ameaças que estivessem afetando a rede, ou seja, sensores, atuadores e os módulos Monitor e Execução. Já o objetivo do presente trabalho é explorar o módulo Análise e Planejamento adicionando o submódulo Memórias (Figura 1) que dá à arquitetura a capacidade de analisar o comportamento da LAN, aprendendo e distinguindo o tráfego de rede normal do malicioso, para autonomicamente privilegiar *hosts*/serviços legítimos e mitigar ciberameaças.

3. Metodologia

O principal objetivo deste trabalho é propor uma abordagem inspirada na estrutura da memória humana que possibilite autonomicamente criar e manter regras de segurança que levam em consideração o histórico de tráfego e alertas da rede.

3.1. Aprendizagem e Reação Autonômica

O módulo de Análise e Planejamento que faz parte da arquitetura apresentada na Figura 1, tem por objetivo analisar os dados do módulo Monitor, aprendendo com os padrões de uso da rede e ataques contra a LAN que fundamentarão a criação e manutenção de regras de segurança autonômicas que: (i) favoreçam fluxos de rede legítimos; e (ii) mitiguem fluxos de redes relacionados com ciberameaças.

Para que o Of-IDPS possa aprender com o histórico da rede, foi utilizado aprendizado de máquina não supervisionado por meio do algoritmo FP-Growth [Han et al. 2004], que permite descobrir/evidenciar relacionamentos entre informações em grandes volumes

de dados. Esta técnica é conhecida como análise de associação ou regras de associação, e uma das abordagens é minerar os conjuntos de itens frequentes (*frequent itemsets*) [Harrington 2012]. A escolha do FP-Growth é devido ao seu desempenho superior em relação a outros algoritmos similares [Han et al. 2004].

O FP-Growth tem como entrada um conjunto de itens referentes a transações de banco de dados e como saída os itens que mais aparecem juntos nestas transações. Além desses itens, também é devolvida outra informação chamada *suporte*, que é um valor numérico que representa a frequência/porcentagem de ocorrência de um conjunto de itens [Harrington 2012]. Assim, no contexto deste trabalho, é proposta a submissão do histórico da rede como transações a serem analisadas pelo FP-Growth, que então devolverá como saída: (i) padrões de ataques, fundamentados nos alertas de segurança; e (ii) padrões de uso da rede, embasado em dados que representam os fluxos de rede, obtidos dos *switches* OpenFlow.

Cada alerta de segurança a ser submetido no FP-Growth é composto pelos campos IP de origem, IP de destino, protocolo (TCP, UDP, ICMP, etc), porta de origem, porta de destino, identificação do alerta/problema de segurança e prioridade/risco do alerta. Os alertas, bem como o suporte mínimo esperado, são submetidos ao FP-Growth, que devolve conjuntos de itens que atendem ao suporte. Cada conjunto de itens obtido a partir dos alertas gerará uma regra de segurança. As regras de segurança possuem os mesmos campos dos alertas. Desta forma, para criar uma regra basta copiar os campos/itens do conjunto retornado para a regra de segurança. Todavia, em alguns casos, um conjunto pode não retornar todos os campos de um alerta, por não atender ao suporte. Caso isto ocorra, os campos faltantes são preenchidos com o valor especial *, que simboliza todo/qualquer, similar com o que acontece na especificação de regras de *firewalls*.

Por exemplo, se os alertas de segurança (fictícios) da Tabela 1 forem submetidos ao FP-Growth, com o suporte mínimo de 60%, o resultado será as regras de segurança da Tabela 2. A regra apresentada na linha 1 da Tabela 2, por exemplo, especifica que pacotes originados do endereço IP 20.0.0.1, destinados a qualquer endereço IP (*), utilizando o protocolo de rede TCP, originados de qualquer porta (*), destinados à porta 22, serão processados considerando o nível de prioridade de segurança médio.

Tabela 1. Exemplo de alertas a serem processados pelo FP-Growth.

N°	Endereço IP		Protocolo	Portas de rede		Alerta de segurança	
	Origem	Destino		Origem	Destino	Identificação	Prioridade
1	20.0.0.1	30.0.0.10	TCP	50000	22	SSH01	Média
2	20.0.0.1	30.0.0.20	TCP	50001	22	SSH02	Média
3	20.0.0.1	30.0.0.30	TCP	50002	22	SSH03	Média
4	20.0.0.1	30.0.0.40	TCP	50003	22	SSH01	Média
5	20.0.0.1	30.0.0.50	TCP	50004	22	SSH01	Média
6	20.0.0.1	30.0.0.60	TCP	50005	22	SSH01	Média
7	20.0.0.2	30.0.0.70	UDP	50006	53	DNS01	Alta
8	20.0.0.2	30.0.0.80	UDP	50007	53	DNS01	Alta
9	20.0.0.2	30.0.0.90	UDP	50008	53	DNS01	Alta
10	20.0.0.1	30.0.0.10	ICMP	0	1	ICMP01	Baixa

Tabela 2. Exemplo de regras criadas a partir dos alertas de segurança

N°	Endereço IP		Protocolo	Portas de rede		Alerta de segurança		Suporte (%)
	Origem	Destino		Origem	Destino	Identificação	Prioridade	
1	20.0.0.1	*	TCP	*	22	*	Média	60
2	20.0.0.1	*	TCP	*	*	*	Média	60
3	20.0.0.1	*	*	*	*	*	Média	70
4	*	*	TCP	*	22	*	Média	60
5	*	*	TCP	*	*	*	Média	60

Analisando as regras de segurança da Tabela 2, constata-se que existem alguns problemas com as regras de segurança geradas pelo FP-Growth. Assim, estes problemas e as respectivas soluções (representadas no Algoritmo 1), são apresentadas a seguir:

- Há regras conflitantes. Por exemplo, quando o controlador OpenFlow analisar um pacote (passo 5.2 da Figura 1) originado do *host* 20.0.0.1 destinado à porta TCP 22, haverá ambiguidade na escolha da regra a ser aplicada, pois todas combinam com o pacote. Para resolver este problema, aplica-se a regra que combina com a maior quantidade de campos do pacote e represente maior ameaça. Caso o impasse continue, aplica-se a primeira regra dentre as restantes;
- Algumas combinações de campos não são úteis. Por exemplo, não é possível formar regras somente com os campos: prioridade e/ou identificação. Também, regras contendo portas só devem ser utilizadas se estiverem acompanhadas do campo protocolo. O método *removerRegrasQuePossuemApenasCamposInvalidosParaFormacaoDeRegras()* do Algoritmo 1, remove regras compostas somente pelos campos identificação, prioridade, portas ou um combinado destes;
- O uso do FP-Growth pode gerar regras muito genéricas e afetar negativamente a rede inteira. Um exemplo é a regra 5 da Tabela 2, que se aplica a todos os fluxos TCP. Para resolver este problema propõem-se o uso de um suporte mínimo que varia em relação ao número de itens retornado pelo conjunto de itens (linhas 5-10 do Algoritmo 1). Então, exige-se que conjuntos com poucos itens tenham um suporte maior do que conjuntos com muitos itens. Um caso especial é quando a regra tem apenas o campo protocolo, pois como este tipo de regra é muito abrangente, então também são descartadas no método *removerRegrasQuePossuemApenasCamposInvalidosParaFormacaoDeRegras()* do Algoritmo 1. Desta forma, submetendo os alertas da Tabela 1 ao Algoritmo 1, a saída será apenas a regra 1 da Tabela 2, pois esta é a única que atende aos requisitos exigidos.

Portanto, o uso do Algoritmo 1 propicia a criação autonômica de regras de segurança no Of-IDPS. Entretanto, ainda há um grande problema, pois conforme o tempo passa e os alertas de segurança forem surgindo, a base de dados de alertas crescerá gradativamente, o que pode causar *overfitting* [Harrington 2012]. Ou seja, surge a possibilidade de que o grande número de alertas antigos ofusque os alertas novos, ao ponto de influenciar negativamente a criação de regras para mitigar distúrbios/ameaças recentes. A solução para este problema é apresentada na próxima subseção.

Algoritmo 1: gerarRegrasDeSegurançaAutonômicas()

```
   Entrada: listaAlertasSegurança - Lista de alertas gerados pelo Of-IDPS
   Saída: listaRegrasSegurança - Lista com regras de segurança
 1 listaConjuntoRegras ← φ
 2 listaRegrasSegurança ← φ
 3 listaConjuntoRegras ← processaAlertasComAlgoritmoFPGrowth(listaAlertasSegurança)
 4 listaConjuntoRegras ← removerRegrasQuePossuemApenasCamposInvalidosParaFormacaoDeRegras(listaConjuntoRegras)
 5 para cada regra em listaConjuntoRegras faça
 6     suporte ← regra.obterSuporte()
 7     numeroItens ← regra.obterNumeroItensPresenteNaRegra()
 8     percentualSuporte = 110 − 20 ∗ numeroItens
 9     se suporte ≥ percentualSuporte então
10         listaRegrasSegurança.adiciona(regra)
11 retorna listaRegrasSegurança
```

3.1.1. Reação Autonômica Inspirada na Memória Humana

Para que o Of-IDPS utilize informações de ataques do passado ou ocorrendo no presente, visando remediar problemas em andamento e prevenir novos ataques, foi desenvolvido um método inspirado na estrutura da memória humana. Isto porque, a memória humana é processada/dividida fundamentalmente em três partes: (1) sensorial, trata de informações ligadas aos sentidos do corpo (visual, tátil, etc) e que estejam ocorrendo no presente; (2) curta, armazena informações passadas ocorridas recentemente. (3) longa, armazena acontecimentos marcantes que ocorreram em um passado mais distante, se comparado com a memória curta [Baddeley 1997]. Então, a ideia é viabilizar ao Of-IDPS uma forma de processar o histórico da rede segundo a mesma política.

No Of-IDPS, a **memória sensorial** é responsável por processar os alertas de segurança extremamente recentes e que representam o que a rede está "sentindo" naquele exato momento. Dentre as memórias do Of-IDPS, esta é a única que não emprega o Algoritmo 1 para gerar as regras de segurança, pois deve atacar pontualmente o problema/alerta em curso. Assim, o processo de geração de regras na memória sensorial consiste em obter os alertas recém identificados pelo módulo Monitor, sendo que cada alerta obtido origina uma regra de segurança. A criação da regra consiste em copiar todos os campos do alerta para a regra de segurança, exceto pela porta do cliente que é generalizada (símbolo *). Isso é feito porque a porta do cliente é aleatória [Tanenbaum and Wetherall 2013], portanto é necessário generalizá-la com o intuito de mitigar também as novas investidas do *host* atacante. A análise e criação de regras na memória sensorial é executada continuamente, em um intervalo configurável de alguns poucos segundos. As regras de segurança geradas são armazenadas na base de dados Regras Memória Sensorial (Figura 1).

A **memória curta** do Of-IDPS retém alertas por pouco tempo. O processamento desta memória inicia com a obtenção de alertas ocorridos recentemente e registrados pelo módulo Monitor. Tais alertas são submetidos ao Algoritmo 1 para gerar regras de segurança extraídas dos padrões de ataques presentes nestes alertas. A cada ciclo autonômico das memórias sensorial e longa, as regras antigas são removidas e as novas incluídas. Todavia, o mesmo não pode ser feito na memória curta, pois as regras recém criadas mitigarão os fluxos maliciosos que geravam os alertas. Naturalmente, sem alertas o Of-IDPS presumirá que não há mais risco e destituirá as regras que policiavam o ataque, propiciando o ressurgimento do mesmo. Isto ocorre porque a memória curta analisa a situação de risco fundamentada em um histórico recente e breve, para tratar de problemas em andamento. Já a memória sensorial trata de situações imediatistas e pontuais e a memória longa é responsável por analisar todo o histórico de rede, que por sua vez não tende a mudar constantemente e nem rapidamente. Desta forma, a memória curta é a única que necessita de uma política de manutenção de regras de segurança mais elaborada. Assim, toda regra na base de dados Regras Memória Curta (Figura 1) tem um contador, que é incrementado caso a regra esteja em uso na rede ou decrementado se ociosa. A regra só é removida quando este contador atinge um valor mínimo, indicando que a regra não é mais útil. Este gerenciamento é feito na própria memória curta e com auxílio do módulo Execução, que fornece estatísticas a respeito do uso das regras (Figura 1).

A **memória longa** é responsável por aprender o que é "bom" ou "ruim" para a rede. Por isto, no Of-IDPS a memória longa é dividida em duas: (1) **longa ruim**: tem

por objetivo identificar as principais ameaças que afligem a rede. Para isto, o histórico de alertas é submetido ao Algoritmo 1. As regras de segurança geradas são gravadas na base de dados Regras Memória Longa Ruim; (2) **longa boa**: ajuda a identificar quais são os principais serviços e *hosts* da rede. Para tanto, o módulo Monitor distingue os fluxos de rede OpenFlow que estão relacionados com alertas de segurança dos que não estão, isto é representado respectivamente pelas bases Fluxos Anormais e Fluxos Normais, na Figura 1. Os campos extraídos da base Fluxos Normais seguem o mesmo padrão dos alertas de segurança, com exceção dos campos identificação e prioridade que sinalizam que estes são fluxos legítimos, e não ameaças. Assim, o processamento da memória longa boa consiste em analisar a base Fluxos Normais, utilizando o Algoritmo 1 e gravar as regras de segurança geradas na base Regras Memória Longa Boa. Outra diferença é que os pacotes que combinarem com estas regras são instalados na rede sem nenhuma restrição.

As regras de segurança geradas pelas memórias são armazenadas em suas respectivas bases de dados e aplicadas na rede pelo módulo Execução (passos 5.1 e 5.2 da Figura 1). Durante a comutação, no controlador OpenFlow, cada pacote é comparado com as regras de segurança das memórias do Of-IDPS. Neste processo, a primeira memória que tiver uma regra que combine com o pacote será aplicada e a respectiva contramedida é adotada. Dependendo da contramedida [Santos et al. 2014], há três possibilidades de encaminhamento: (1) sem restrições (regras da memória longa boa); (2) com restrições de taxa de transferência de dados (para regras com prioridade de segurança baixa e média); (3) ou bloquear completamente (regras com prioridade alta). Caso o pacote não combine com as regras de nenhuma memória, este é encaminhado normalmente.

Os pacotes comutados obrigatoriamente são submetidos à seguinte ordem de análise entre as memórias: sensorial, curta e longa ruim. Contudo, a ordem de processamento da memória longa boa em relação às outras memórias é configurável. Isto foi feito pensando-se em deixar o Of-IDPS flexível para adequar-se a diferentes tipos/necessidades de redes e administradores. Assim, são possíveis três configurações quanto à ordem da memória boa e cada uma tem prós e contras: (1) com a memória longa boa sendo processada antes da sensorial, os fluxos bons/comuns não são afetados pelas contramedidas das outras memórias, evitando problemas de falsos positivos. Todavia, se ocorrerem ataques nos fluxos "bons", estes não serão imediatamente mitigados, pois estarão amparados pela memória longa boa; (2) caso a memória longa boa seja posta antes da memória curta, os ataques muito recentes serão combatidos pela memória sensorial, mesmo que ocorram em fluxos considerados "bons", mas só de forma pontual, pois não serão aplicadas as regras da memória curta. (3) usando a memória longa boa antes da longa ruim, o Of-IDPS inicialmente terá conhecimento prévio do que é bom ou ruim (memórias longa boa e ruim). Porém, o Of-IDPS não hesitará em reagir (memórias sensorial e curta) contra fluxos ordinários (memória longa boa), caso estes representem riscos a rede. Se o problema persistir durante muito tempo, o Of-IDPS aprenderá e mudará sua conclusão a respeito do que é considerado bom ou ruim, tal como na memória humana. É importante ressaltar que a troca da ordem de análise da memória longa boa, só afeta a forma de reação frente aos fluxos considerados ordinários, não afetando a capacidade do Of-IDPS em combater fluxos de redes considerados incomuns.

4. Experimentos e Resultados

Os experimentos apresentados nesta seção visam avaliar a reação autonômica do Of-IDPS, utilizando os métodos apresentados na Seção 3, frente a incidentes de segurança. A avaliação da reação autonômica proposta foi realizada em dois ambientes: (1) LAN simulada; (2) LAN real. A escolha de dois ambientes é devido ao fato que o ambiente virtual proporciona experimentos mais precisos e sem interferências externas, já o ambiente real permite analisar o comportamento do Of-IDPS em uma rede em produção.

O Of-IDPS foi desenvolvido utilizando o arcabouço de controlador OpenFlow Beacon, escolhido por apresentar melhor desempenho se comparado com outros controladores OpenFlow conhecidos [Erickson 2013]. Diferentemente de trabalhos similares, o código-fonte do Of-IDPS está disponível publicamente em `https://github.com/luizsantos/Of-IDPS` sob licença GNU GPLv3[1]. Para estes experimentos, o Of-IDPS foi configurado para utilizar a seguinte ordem de análise de regras entre as memórias: 1ª – sensorial, 2ª – curta, 3ª – longa boa e 4ª – longa ruim. O tempo de retenção dos últimos alertas identificados para as memórias sensorial e curta, são respectivamente 3 e 30 segundos[2].

Na LAN simulada foram realizados experimentos com ataques DDoS TCP-SYN. A simulação é feita utilizando Mininet [Lantz et al. 2010, Antonenko and Smelyanskiy 2013]. Durante o ataque, a vítima executa um servidor HTTP Apache[3] e o ataque DDoS com origem desconhecida (*spoof*) é feito com a ferramenta Hyenae[4]. Foram realizados 3 experimentos variando a quantidade de pacotes maliciosos enviados para a vítima, sendo estas as quantidades: 5.000, 10.000 e 20.000. O objetivo destes experimentos é observar se o Of-IDPS apresenta um comportamento uniforme, na mitigação de ataques, independente do número de pacotes maliciosos tratados. Cada experimento é composto por 15 repetições e foi executado em cenários de rede com e sem o Of-IDPS.

A Tabela 3 apresenta os dados gerados por cada um dos experimentos com DDoS. Analisando principalmente a última coluna da tabela, que apresenta a relação entre o número de pacotes tratados pelas vítimas com e sem o Of-IDPS, é possível constatar que a abordagem proposta foi capaz de mitigar os ataques em todos os experimentos. Por exemplo, no experimento com 10.000 pacotes na rede sem o Of-IDPS (experimento de controle) o atacante enviou 10.019 pacotes, os quais geraram 19.908 pacotes de resposta da vítima. A vítima gerou quase o dobro de pacotes em relação ao atacante, pois o atacante utilizou endereços IPs falsos e aleatórios (*spoof*) para efetuar o ataque DDoS, ou seja, os pacotes de resposta não retornam para o atacante. No experimento com 10.000 pacotes, na rede com Of-IDPS, o atacante enviou 10.200 pacotes e a vítima apenas 1.008 pacotes. Portanto, nos experimentos com 10.000 pacotes, a vítima da rede com Of-IDPS teve uma diminuição de 94,9% dos pacotes maliciosos tratados, se comparada à vítima da rede sem Of-IDPS.

Constatou-se também que a variação do número de pacotes maliciosos enviados durante os ataques DDoS não influenciam drasticamente o comportamento da arquitetura.

[1] `http://www.gnu.org/licenses/` - acessado em 03 de dezembro de 2015.

[2] O uso destes tempos é fundamentado em extensivos testes, que infelizmente não cabem neste artigo.

[3] `http://www.apache.org/` - acessado em 03 de dezembro de 2015.

[4] `http://sourceforge.net/projects/hyenae/` - acessado em 03 de dezembro de 2015.

Por exemplo, os gráficos da Figura 2 apresentam o fluxo médio (linha) e o desvio padrão (sombra) dos pacotes enviados/recebidos pelo atacante e vítima. Analisando os dados do atacante (linha laranja com quadrado) e vítima (linha vermelha com círculo) da rede sem Of-IDPS, verifica-se a contundência dos ataques DDoS, pois a rede é inundada pelos pacotes maliciosos e, por consequência, os recursos da rede e da vítima são comprometidos. Já examinando os dados do acatante (linha azul com asterisco) e vítima (linha verde com triângulo) que utilizam o Of-IDPS, é possível comprovar que a criação de regras autonômicas proposta consegue mitigar os ataques, pois com cerca de 5 segundos de seu início as medidas reativas começam a surtir efeito, restringindo a taxa de transferência de dados dos pacotes maliciosos, o que notoriamente ajuda a conservar recursos da rede e proteger a vítima. Então, conclui-se que o número de pacotes maliciosos enviados durante os ataques não geram distúrbios que invalidam o comportamento da arquitetura, nem a mitigação do problema.

Tabela 3. Dados dos experimentos com ataques DDoS.

Número de pacotes maliciosos enviados	Número pacotes LAN sem Of-IDPS		Número pacotes LAN com Of-IDPS		Comparação entre vítimas (%)*
	Atacante	Vítima	Atacante	Vítima	
5.000	5.020	9.965	5.017	957	90,4
10.000	10.019	19.908	10.200	1.008	94,9
20.000	20.069	39.886	20.323	1.080	97,3

* - Porcentagem de pacotes que a vítima com Of-IDPS deixou de tratar em relação à vítima da rede sem Of-IDPS.
Observação - Durante os experimentos são utilizados pacotes ICMP *echo request/reply* para identificar o inicio e o final de cada teste e também são produzidos naturalmente pacotes ARP durante a comunicação na LAN. Devido a isto há um pequeno acréscimo na quantidade de pacotes em cada experimento.

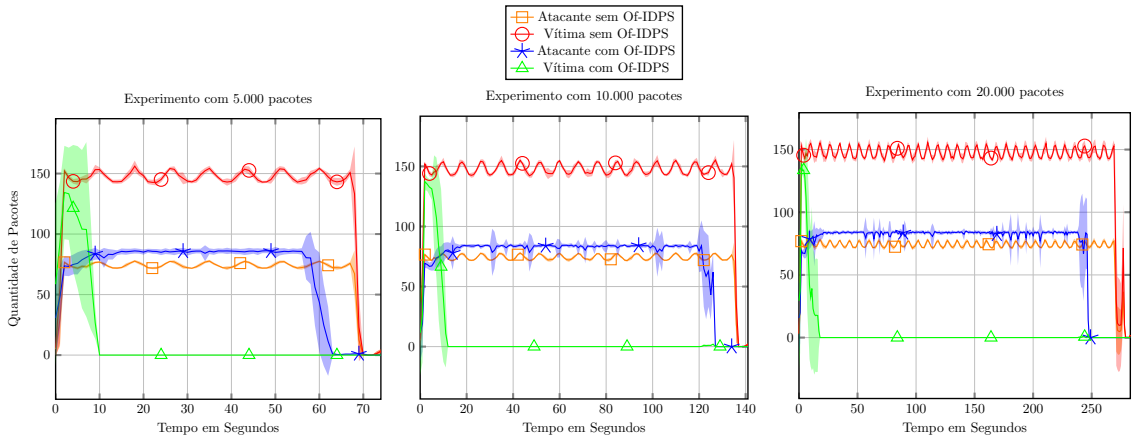

Figura 2. Experimentos variando a quantidade de pacotes no ataque DDoS.

Complementarmente, foram efetuados experimentos com o Of-IDPS em uma LAN real, para verificar a reação autonômica proposta frente às adversidades de um ambiente em produção. Mais especificamente os objetivos foram: (1) verificar a reação do Of-IDPS frente a outros problemas de segurança, além dos já avaliados em [Santos et al. 2014] e do ataque DDoS apresentado anteriormente; (2) analisar se as respostas autonômicas contra problemas de segurança não afetam negativamente *hosts*/serviços de redes legítimos; (3) avaliar a eficácia das memórias longa boa e ruim em utilizar o histórico da rede para identificar os principais *hosts*/serviços e ataques da LAN.

Desta forma, o Of-IDPS foi instalado na LAN do Grupo de Trabalho Early Warning System (GT-EWS) da Rede Nacional de Ensino e Pesquisa (RNP). Tal LAN fornece infraestrutura para o funcionamento de um sistema de alertas antecipados fundamentados principalmente em mensagens de segurança postadas em redes sociais [Santos et al. 2013], e que pode ser acessado via Internet em `http://gtews.cm`.

utfpr.edu.br/ews/. Esta LAN é basicamente composta por: (a) um Controlador OpenFlow no qual está instalado o Of-IDPS; (b) um *host* responsável por coletar postagens no Twitter e Facebook; (c) um servidor Xen[5], com máquinas virtuais que dão suporte aos serviços do projeto, tal como: banco de dados, *Web service*, servidor HTTP, IDS, núcleo de processamento do sistema de alertas antecipados, etc.

A LAN do GT-EWS está conectada na LAN da Universidade Tecnológica Federal do Paraná (UTFPR), que por sua vez está ligada à Internet. Então, é função do Of-IDPS proteger a LAN contra ciberameaças provindas da própria LAN do GT-EWS, da LAN da UTFPR e da Internet.

Para a visualização do comportamento do Of-IDPS na rede, a equipe do GT-EWS desenvolveu uma interface Web que permite ver em tempo de execução as regras de segurança presentes nas memórias do Of-IDPS. Dois fragmentos desta interface são apresentados na Figura 3. Basicamente a interface Web possui um menu lateral esquerdo que dá acesso a cada uma das memórias (ver Figura 3) e uma tabela para visualização dos alertas. Cada linha da tabela é um alerta e apresenta: Prioridade de segurança (*Priority*), identificação do problema que originou a regra (*Descrip.*), IP de origem (*Source*), IP de destino (*Dest.*), protocolo (*Proto.*), porta de origem (*T.Source*), porta de destino (*T.Dest*), suporte (*Support*) e contador de vida (*Life* – só utilizado pela memória curta).

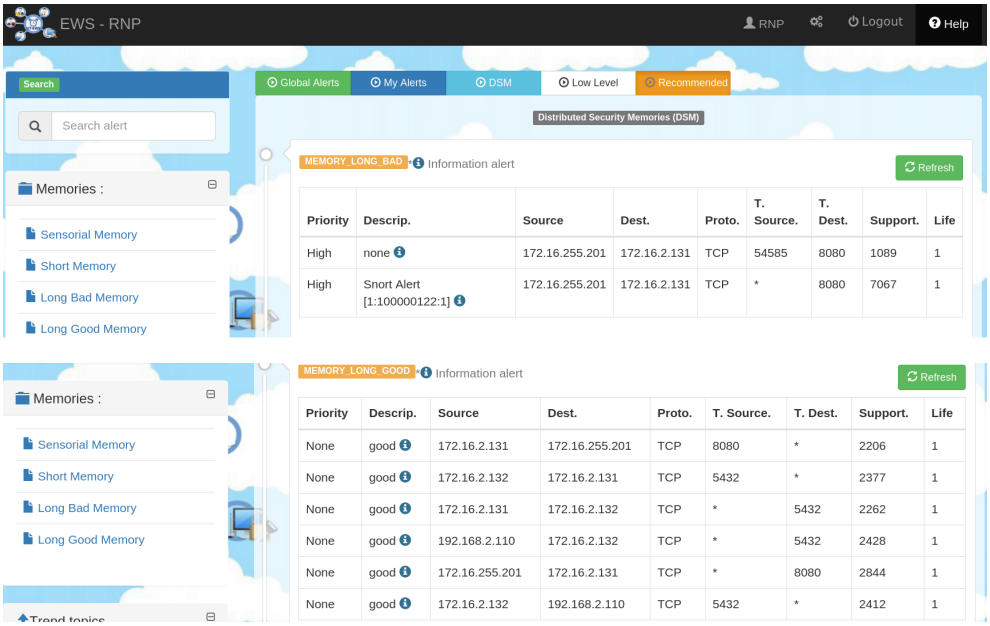

Figura 3. Memórias longa (a) ruim e (b) boa na interface Web do GT-EWS.

O Of-IDPS está em execução na LAN do GT-EWS há 6 meses. Neste período o módulo Monitor (Figura 1) identificou 44 tipos de ciberameaças, que deram origem a 15.098 alertas de segurança. Todo este histórico de alertas processado pela metodologia da memória longa ruim resulta nas duas regras autonômicas presentes na tabela da Figura 3-a. A primeira regra bloqueia pacotes originados de 172.16.255.201, destinados a 172.16.2.131, usando o protocolo TCP, originados da porta cliente 54585 e destinados a porta 8080. Esta regra foi formada por alertas que representam mais de um

[5]http://xenserver.org/ - acessado em 03 de dezembro de 2015.

tipo de problema de segurança, por isto a generalização (*none*) do campo identificação (*Descrip.*). No Of-IDPS pacotes com alta (*high*) prioridade de segurança são bloqueados [Santos et al. 2014]. A segunda regra faz basicamente o mesmo que a primeira, mas generalizando a porta de origem (*) e especificando o tipo de ataque (*Snort Alert [1:100000122:1]*). Ambas regras condizem com o cenário de rede analisado, pois devido à estrutura da LAN da UTFPR, todo acesso provindo da Internet chega na LAN do GT-EWS por intermédio do *host* 172.16.255.201. Também, a maior parte dos acessos da Internet são destinados ao servidor HTTP (*host* 172.16.2.131). Então, na LAN do GT-EWS, a memória longa ruim aprendeu que muitos ataques ocorrem partindo do *host* 172.16.255.201, para 172.16.2.131 e estas ciberameaças atentam principalmente contra vulnerabilidades do servidor HTTP Apache (*Snort Alert [1:100000122:1]*), na porta TCP/8080 e por isso devem ser bloqueadas.

Neste mesmo período de 6 meses, foram também instalados nos *switches* OpenFlow, 10.205.790 fluxos de rede considerados idôneos (base Fluxos Normais da Figura 1). A análise destes fluxos pela memória longa boa pode ser vista na Figura 3-b. A primeira e a penúltima regra indicam que é comum a criação de fluxos de rede entre os *hosts* 172.16.2.131 e 172.16.255.201, usando a porta TCP/8080. Na interface, o campo *Priority* na memória boa é marcado como *None*, simbolizando que não há prioridade/risco de segurança para os pacotes que combinam com estas regras. Já o campo identificação (*Descrip.*) é preenchido com *good*, representando que a regra foi gerada pela memória longa boa. As demais regras demonstram que é ordinário o acesso dos *hosts* 172.16.2.131 (Web *Service* e servidor HTTP) e 192.168.2.110 (Controlador OpenFlow) ao *host* 172.16.2.132 na porta TCP/5432 (banco de dados). As regras criadas pela memória longa boa, também são condizentes com o uso da LAN do GT-EWS, pois nesta rede é muito comum que o *host* 172.16.255.201 acesse o servidor HTTP, que por sua vez, acessa o *host* do banco de dados. Já o controlador OpenFlow, que representa o núcleo do Of-IDPS, armazena as regras de segurança geradas por este, no *host* do banco de dados, para que sejam visualizadas posteriormente no servidor HTTP.

Para possibilitar uma investigação mais minuciosa a respeito da segurança da rede do GT-EWS durante os testes com o Of-IDPS, todos os *hosts* foram monitorados adicionalmente pelas ferramentas OSSEC-HIDS[6] e CACTI[7]. Todos os dados obtidos a respeito da segurança da LAN passaram ainda pela investigação constante dos autores deste trabalho. Todavia, em nenhum momento foi constatado o comprometimento da segurança da LAN do GT-EWS, destacando que a abordagem proposta consegue efetivamente mitigar ciberameaças sem afetar negativamente os serviços legítimos da LAN.

Além dos experimentos com DDoS apresentados neste trabalho, também foram executados testes em menor escala, que empregaram 50.000 e 100.000 pacotes maliciosos. Tais testes tiveram resultados bem similares aos apresentados no artigo. Inclusive o Of-IDPS tende a apresentar resultados ligeiramente melhores ao se aumentar o número de pacotes maliciosos no ataque DDoS, pois os fluxos do ataque ficam com sua taxa de transferência de dados reduzida por mais tempo. Isto pode ser observado na Tabela 3 e Figura 2. Em [Santos et al. 2014], a arquitetura mitigou 89,05% dos pacotes DoS com origem conhecida (IP do atacante). Já no presente trabalho, conseguiu-se mitigar em média

[6] `http://ossec.github.io/` - acessado em 03 de dezembro de 2015.

[7] `http://www.cacti.net/` - acessado em 03 de dezembro de 2015.

94,2% dos pacotes maliciosos em ataques DDoS com origem desconhecida. Ou seja, a nova abordagem conseguiu melhorar 5,15 pontos percentuais em um cenário de ataque mais complexo. Ainda nos experimentos com DDoS, observa-se que o início da reação autonômica contra o ataque foi de 5 segundos e a mitigação do problema se dá com 10 segundos. Tal resultado é considerado aceitável, pois em [Chen et al. 2014] a mitigação do mesmo problema se dá com 40 segundos. Todavia, ambos resultados são satisfatórios se comparados ao tempo que um administrador humano levaria para identificar e mitigar o problema. Ataques que ocorram em menos de 5 segundos podem burlar as memórias sensorial e curta, contudo serão mitigados pela memória longa ruim, caso persistam.

5. Conclusões e Trabalhos Futuros

Os experimentos nas LANs simuladas e real demonstram a efetividade da abordagem proposta em mitigar problemas de segurança. Por exemplo, o Of-IDPS manteve a integridade da LAN do GT-EWS, mesmo depois da investida de 44 tipos de ciberameaças. Nos experimentos com DDoS, a reação autonômica proposta conseguiu mitigar até 97,3% dos pacotes maliciosos, na vítima. Também, constatou-se que a reação autonômica proposta não influencia negativamente os serviços legítimos da rede, principalmente devido ao uso da memória longa boa. Isto pode ser observado na Figura 3-a, que apresenta as regras autonomicamente criadas para bloquear fluxos destinados ao servidor HTTP, que notoriamente traz riscos à rede. Contudo, dificilmente um administrador de rede bloquearia um serviço que é vital para o projeto do GT-EWS, como é o caso. Então, similarmente ao administrador humano, a memória longa boa identifica que o servidor HTTP é um serviço ordinário, e permite seu uso na rede (regras 1 e 5 da Figura 3-b), se contrapondo à memória longa ruim, o que evidencia que as memórias longa boa e ruim conseguem avaliar o histórico da rede e identificar os principais *hosts*/serviços e ataques na LAN.

Assim, por meio da abordagem proposta e dos testes realizados demonstrou-se que é possível criar métodos para auto-proteger LANs de ciberataques, com o mínimo possível de intervenção humana. Como trabalhos futuros pretende-se correlacionar o histórico da LAN com informações externas a respeito de ciberameaças.

Referências

Antonenko, V. and Smelyanskiy, R. (2013). Global network modelling based on mininet approach. In *Proceedings of the 2nd ACM SIGCOMM*, HotSDN '13, pages 145–146, NY, USA. ACM.

Baddeley, A. (1997). *Human Memory: Theory and Practice*. Psychology Press.

Barrère, M., Badonnel, R., and Festor, O. (2011). Supporting vulnerability awareness in autonomic networks and systems with oval. In *Proceedings of the 7th International CNSM'11*, pages 357–363, Laxenburg, Austria. International Federation for Information Processing.

Barrère, M., Badonnel, R., and Festor, O. (2012). Collaborative remediation of configuration vulnerabilities in autonomic networks and systems. In *CNSM'12, 8th international conference and 2012 workshop on SVM*.

Chen, Q., Abdelwahed, S., and Erradi, A. (2014). A model-based validated autonomic approach to self-protect computing systems. *IEEE Internet of Things Journal*.

Chu, Y., Tseng, M., Chen, Y., Chou, Y., and Chen, Y. (2010). A novel design for future on-demand service and security. In *12th ICCT IEEE*, pages 385–388.

Chung, C.-J., Khatkar, P., Xing, T., Lee, J., and Huang, D. (2013). Nice: Network intrusion detection and countermeasure selection in virtual network systems. *Dependable and Secure Computing, IEEE Transactions on*, 10(4):198–211.

Erickson, D. (2013). The beacon openflow controller. In *Proceedings of 2th ACM SIGCOMM*, HotSDN '13, pages 13–18, NY, USA. ACM.

Han, J., Pei, J., Yin, Y., and Mao, R. (2004). Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Min. Knowl. Discov.*, 8(1):53–87.

Hariri, S., Khargharia, B., Chen, H., Yang, J., Zhang, Y., Parashar, M., and Liu, H. (2006). The autonomic computing paradigm. *Cluster Computing*, 9(1):5–17.

Harrington, P. (2012). *Machine Learning in Action*. Manning Publications Co.

Lantz, B., Heller, B., and McKeown, N. (2010). A network in a laptop: Rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM*, Hotnets-IX, pages 1–6, NY, USA. ACM.

Lara, A., Kolasani, A., and Ramamurthy, B. (2013). Network innovation using openflow: A survey. *Communications Surveys Tutorials, IEEE*, PP(99):1–20.

Lara, A. and Ramamurthy, B. (2014). Opensec: A framework for implementing security policies using openflow. In *Proceddings of Globecom 2014 - Communication and Information System Security Symposium*, pages 781–786.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74.

Santos, L. A. F., Campiolo, R., and Batista, D. M. (2014). Uma arquitetura autonômica para detecção e reação a ameaças de segurança em redes de computadores. In *Anais do 4o Workshop em Sistemas Distribuídos Autonômicos, SBRC 2014*, pages 45–48.

Santos, L. A. F., Campiolo, R., Gerosa, M. A., and Batista, D. M. (2013). Deteção de alertas de segurança em redes de computadores usando redes sociais. In *Anais do XXXI SBRC*, pages 791–804.

Sibai, F. M. and Menasce, D. (2011). Defeating the insider threat via autonomic network capabilities. In *Communication Systems and Networks (COMSNETS), 2011 Third International Conference on*, pages 1–10.

Sibai, F. M. and Menasce, D. (2012). Countering network-centric insider threats through self-protective autonomic rule generation. In *Software Security and Reliability (SERE), 2012 IEEE Sixth International Conference on*, pages 273–282.

Symantec (2015). Internet security threat report 2015. Technical Report 20, Symantec.

Tanenbaum, A. S. and Wetherall, D. J. (2013). *Computer Networks*. Pearson Education.

Wang, Y., Zhang, Y., Singh, V., Lumezanu, C., and Jiang, G. (2013). Netfuse: Short-circuiting traffic surges in the cloud. In *2013 IEEE ICC*, pages 3514–3518.

Yuan, E., Esfahani, N., and Malek, S. (2014). A systematic survey of self-protecting software systems. *ACM Trans. Auton. Adapt. Syst.*, 8(4):17:1–17:41.