Helton Molina Sapia

Apoio à tradução da política de segurança para regras de *firewall* utilizando uma linguagem de modelagem visual: SPML2

São José do Rio Preto
2016

Helton Molina Sapia

Apoio à tradução da política de segurança para regras de *firewall* utilizando uma linguagem de modelagem visual: SPML2

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de São José do Rio Preto.

Orientador: Prof. Dr. Rogério Eduardo Garcia

São José do Rio Preto
2016

Sapia, Helton Molina.
Apoio à tradução da política de segurança para regras de *firewall* utilizando uma linguagem de modelagem visual : SPML2 / Helton Molina Sapia. – São José do Rio Preto, 2016
121 p.: il. , tabs.

Orientador: Rogério Eduardo Garcia
Dissertação (Mestrado) – Universidade Estadual Paulista "Júlio de Mesquita Filho", Instituto de Biociências, Letras e Ciências Exatas.

1. Computação – Matemática. 2. Redes de computadores - Medidas de segurança. 3. Comutação de pacotes (Transmissão de dados). 4. *Firewalls* (Medidas de segurança de computadores) - Métodos experimentais. 5. Programação visual (Computação) I. Garcia, Rogério Eduardo. II. Universidade Estadual Paulista "Júlio de Mesquita Filho". Instituto de Biociências, Letras e Ciências Exatas. III. Título.
CDU – 681.3.025

Ficha catalográfica elaborada pela Biblioteca do IBILCE
UNESP - Câmpus de São José do Rio Preto

Helton Molina Sapia

Apoio à tradução da política de segurança para regras de *firewall* utilizando uma linguagem de modelagem visual: SPML2

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de São José do Rio Preto.

Comissão Examinadora

Prof. Dr. Rogério Eduardo Garcia
UNESP - Presidente Prudente
Orientador

Prof. Dr. Paulo Lício de Geus
UNICAMP - Campinas

Prof. Dr. Milton Hirokazu Shimabukuro
UNESP - Presidente Prudente

São José do Rio Preto
17 de Junho de 2016

Dedico este trabalho a minha esposa Lucimari e aos meus filhos: João Paulo, Maria Victoria, João Miguel e Maria Clara. Vocês são a razão para eu me empenhar em ser uma pessoa melhor!

Agradecimentos

Agradeço, em primeiro lugar, a DEUS, por derramar infinitas bençãos em meu caminho. Quando eu acreditei não ter mais forças para seguir adiante, DEUS enviou alguém em meu socorro. OBRIGADO SENHOR!!! Como disse o Apóstolo Paulo "O que nos resta dizer? Se Deus está a nosso favor, quem estará contra nós?" (Romanos 8,31).

Agradeço em especial a toda a minha família, pela paciência e pelas orações.

Agradeço ao meu orientador Rogério, pelos ensinamentos que me permitiram alcançar o título de mestre.

Agradeço aos meus colegas de mestrado, pelo apoio e amizade.

Agradeço aos meus amigos, vocês fizeram a diferença.

“Crux sacra sit mihi lux!
Nunquam draco sit mihi dux!
Vade retro Satana!
Nunquam suade mihi vana!
Sunt mala quae libas.
Ipse venena bibas!”
Crux Sancti Patris Benedicti

Resumo

As telecomunicações e as redes de computadores permitiram a integração mundial, mas junto com os benefícios também surgiram problemas, como as ameaças a dados que trafegam pela rede, o que justifica o controle das comunicações, mais especificamente o controle sobre o tráfego de pacotes. O tráfego de pacotes entre redes deve ser protegido, de modo a evitar acessos não autorizados. A proteção deve ocorrer no perímetro de rede, fronteira entre uma rede interna e a Internet. O perímetro de uma rede é protegido por *firewall* para que não ocorra acesso não autorizado. O *firewall* realiza a filtragem dos pacotes de dados que trafegam entre redes, aplicando regras para selecionar o pacote de dados que pode alcançar uma rede. Escrever as regras de funcionamento de um *firewall* parece ser uma tarefa simples, entretanto, a necessidade de implementar várias regras para uma ou várias redes, torna o conjunto de regras complexo e de difícil entendimento. Nesse contexto, a utilização de uma representação gráfica para a visualização das regras que expressam a política de segurança apresenta-se como recurso para facilitar o entendimento do que deve ser implementado. E, consequentemente, pode minimizar os defeitos nas regras que configuram o *firewall*. Para tal, foi proposto o uso da SPML – *Security Policy Modeling Language*, que visa à criação de uma representação visual da política de segurança de acesso a redes utilizando notação gráfica. Neste trabalho foi definida uma extensão da SPML, a SPML2, além de formalizar a sintaxe e semântica das duas linguagens. Foram, ainda, verificadas a eficácia e a eficiência da utilização da SPML2 como apoio ao administrador de redes na tarefa de traduzir a política de segurança para regras em linguagem nativa de um *firewall* por meio de um experimento controlado.

Palavras-chave: Modelagem da Política de Segurança. Modelagem Visual. Aprendizado de Configuração de *Firewall*. Métodos Formais. Experimento Controlado.

Abstract

Telecommunications and computer networks enabled global integration, but with the benefits also appeared problems such as threats to data traveling over the network, which justifies the control of communications, specifically control over data traffic. The data traffic between networks must be protected to prevent unauthorized access. The protection should occur in the network perimeter, the boundary between the internal network and the Internet. The firewall is responsible for protecting the network perimeter to prevent any unauthorized access. A firewall performs the filtering of data packets that travel between networks by applying rules that selected the ones that can access the network. To write firewall rules seems to be a simple task. However, the need to implement several rules for one or more networks makes complex the set rules and increase the difficult to understand it. Thus, the use of a graphical representation for visualization of rules that express the security policy is presented as a way to make easy to understand what should be implemented. Consequently, this can minimize the defects in the rules that configure the firewall. To this end, it proposed the use of SPML - Security Policy Modeling Language, which aims to create a visual notation that represent the network access security policy. This work defined an extension of SPML, the SPML2, and formalized the syntax and semantics of the two languages. Although, the efficiency and effectiveness for use of SPML2 to support the network administrator in the translation task of the security policy rules, in the firewall native language, had been checked through a controlled experiment.

Keywords: Security Policy Modeling Language. Visual Modeling. Learning Firewall Configuration. Formal Methods. Controlled Experiment.

Lista de Figuras

Lista de Tabelas

Sumário

CAPÍTULO 1

Introdução

1.1 Contextualização e Formulação do Problema

Utilizar a Internet [1] se tornou fundamental para muitas organizações independentemente de tamanho e de pertencer a iniciativa pública ou privada. As redes de computadores das organizações contêm servidores que podem ser acessados por vários equipamentos presentes na Internet(Tanenbaum & Wetherall, 2011).

O acesso à Internet permite aos computadores locais o alcance e a interação com recursos da rede exterior e, também, permite o alcance de computadores externos a recursos locais, o que expõe os serviços das redes locais à possibilidade de ataques, tornando vulneráveis os serviços e, consequentemente, os dados mantidos em computadores das redes internas (Goodrich & Tamasia, 2013). Para oferecer proteção a serviços e dados da rede interna contra ataques oriundos do restante da Internet é necessário utilizar um *firewall* (Forouzan & Mosharraf, 2013).

O *firewall* deve ser configurado para implementar restrições sobre o tráfego de rede determinando as ações que podem ser executadas por determinados sujeitos nos objetos de rede. Tais restrições mapeiam a política de segurança geralmente expressa em um texto, que descreve o que a rede externa pode alcançar na rede interna e vice-versa (Goodrich & Tamasia, 2013). A política de segurança é complexa e está sujeita a defeitos tanto na sua elaboração quanto na sua tradução para a

[1] Rede de computadores que interconecta centenas de milhões de dispositivos computacionais ao redor do mundo (Kurose & Ross, 2013).

linguagem utilizada pelos *firewalls* que a implementam (Wool, 2004, 2010).

É crucial que as configurações de *firewall* expressem exatamente a política de segurança. Quaisquer configurações defeituosas podem resultar em bloqueio indesejado, acesso não autorizado, ou, até mesmo, a possibilidade de uma pessoa não autorizada alterar as configurações de segurança. Facilitar o processo de mapeamento da política de segurança para regras em linguagem nativa de *firewall*, que representa corretamente a política de segurança, pode diminuir a possibilidade de defeitos em configurações.

Para decidir sobre qual ação deve ser tomada para determinado pacote de dados (liberar ou bloquear), o *firewall* compara as informações do cabeçalho do pacote com as regras. As regras escritas em linguagem nativa de *firewall* utilizam informações contidas no cabeçalho do pacote como *protocolo*, *endereço IP* e *número de porta*.

A variedade de serviços que podem ser ofertados em uma rede de equipamentos computacionais e a própria rede estão em constante crescimento, assim, o número de regras em uma configuração típica de *firewall* tende a crescer. Incluir corretamente novas regras é uma tarefa complexa que ocorre com frequência (Bandara et al., 2009).

Deve-se ainda considerar a ordem das regras: duas regras indicando ações diferentes, sendo uma bloqueando e outra liberando, associadas a um mesmo tipo de tráfego de rede, pode gerar um erro de configuração, uma vez que a política de segurança pode ser escrita em qualquer ordem (Ben Souayeh Ben Youssef & Bouhoula, 2010).

O administrador de redes precisa ter o correto entendimento da política de segurança para, a partir da política de segurança, elaborar o conjunto de regras em linguagem nativa de *firewall* que expresse a política. A modificação do conjunto de regras em linguagem nativa de *firewall* provocada por uma alteração da política de segurança não deve conter defeito que possa causar falhas, ou seja, liberar o pacote que deveria ser bloqueado ou bloquear o pacote que deveria ser liberado.

Criar ou alterar regras em linguagem nativa de um *firewall* exige que o administrador de redes tenha o domínio sintático e semântico da linguagem de configuração do produto de *firewall* que está sendo utilizado, o que particularmente pode ser complexo e difícil de dominar.

As regras em linguagem nativa de *firewall* devem implementar o que está expresso na política de segurança, logo, o correto entendimento da política também é fundamental para a corretude das regras. O uso de uma representação gráfica torna

claro para o administrador de redes os controles definidos na política de segurança.

1.2 Motivação e Justificativa

Firewalls, em geral, são relativamente fáceis de configurar. Entretanto, a política de segurança que orienta a operação é complexa e sujeita a defeitos tanto na sua elaboração, quanto na tradução para a linguagem utilizada pelos softwares que a implementam (Wool, 2004, 2010). Erros de configuração de *firewalls* são frequentes, até mesmo em especificações realizadas por administradores experientes (Bandara et al., 2009).

Ao examinar trinta e sete *firewalls* em redes de equipamentos computacionais de organizações de diversos tamanhos e de diversos setores como telecomunicações, financeiras, concessionárias de energia, segmento médico e automotivo, Wool (2004) constatou que todos os *firewalls* foram mal configurados e, consequentemente, todas as redes estavam vulneráveis. O estudo abordou organizações com diferentes níveis de utilização da tecnologia, incluindo desde organizações com departamento próprio para implementar e manter a política de segurança até empresas de pequeno porte que utilizavam soluções gratuitas. No resumo do estudo apresentado na Tabela 1.1 pode ser observado que foram analisados *firewalls* que possuíam desde cinco até duas mil, seiscentas e setenta e uma regras em linguagem nativa, e o número médio de regras considerando os trinta e sete *firewalls* foi de cento e quarenta e quatro. Independentemente da quantidade de regras, todos os *firewalls* apresentavam erros de configuração.

Tabela 1.1: Estatística das características dos *firewalls* analisados por Wool (2004)

Propriedade	**Mínimo**	**Máximo**	**Média**
Número de regras	5	2671	144
Número de objetos	24	5847	968
Número de interfaces	2	13	4,1

Em novo estudo também elaborado por Wool (2010), considerando os conjuntos de regras de oitenta e quatro *firewalls* de diferentes versões e de diferentes fabricantes, foi constatado que todos os *firewalls* apresentam problemas de configuração e, consequentemente, continuam vulneráveis. Nesse novo estudo por questões de confidencialidade não foram divulgadas as informações detalhadas sobre os conjuntos de regras. Com as informações apresentadas, foi possível apresentar a distribuição da quantidade de conjuntos de regras por fornecedor e versão do produto de *firewall*, cuja distribuição pode ser observada na Tabela 1.2. Como demonstrado na

distribuição, o produto do fornecedor Check Point Firewall-1 versão 4.1 foi o que apresentou a maior quantidade de conjuntos de regras para análise.

Tabela 1.2: Distribuição dos conjuntos de regras por fornecedor e versão do *firewal* adaptado de Wool (2010)

Fornecedor	**Versão**	**Quantidade de conjuntos de regras**	**%**
Check Point Firewall-1	4.0	4	4,8
Check Point Firewall-1	4.1	30	35,7
Check Point Firewall-1	NG/NG-FP3	17	20,2
Check Point Firewall-1	NG R55	3	3,6
Cisco PIX	4.4	3	3,6
Cisco PIX	5.0-5.2	7	8,3
Cisco PIX	6.0-6.2	11	13,1
Cisco PIX	6.3-7.0	9	10,7
Total		84	100

Nesse segundo estudo (Wool, 2010) foram analisados *firewalls* que possuíam desde duas regras até três mil duzentos e cinquenta e nove regras em linguagem nativa. Novamente, independentemente da quantidade de regras, todos os *firewalls* apresentavam erros de configuração.

A política de segurança é expressa em regras e realizar a manutenção dessas regras sem perturbar o que está expresso na política de segurança é uma tarefa complexa que ocorre com bastante frequência. Deve-se considerar, ainda, que existe uma ampla variedade de protocolos que precisam ser suportados e que as redes de equipamentos computacionais estão em constante crescimento, assim o número de regras em uma configuração típica de *firewall* tende a crescer (Bandara et al., 2009).

A segurança operacional é um dos pilares que sustentam a segurança entre redes de equipamentos computacionais. Conforme Bandara et al. (2009), a definição e a manutenção da política de segurança expressa em regras de um *firewall* tornam-se tarefas complexas e sujeitas a muitos erros com o aumento do tamanho das redes.

A motivação deste trabalho é minimizar os defeitos nas regras em linguagem nativa de *firewall* e a justificativa é que fornecer suporte para que o administrador de redes tenha o correto entendimento da política de segurança colabora com a produção do conjunto de regras em linguagem nativa de *firewall* que expresse essa política.

A maioria dos trabalhos encontrados na literatura apresenta técnicas de apoio à validação da política de segurança, tendo como ponto de partida as regras em linguagem nativa de *firewall*, e as confronta com a política de segurança (Yuan

et al., 2006, El-Atawy et al., 2007, Al-Shaer et al., 2009, Gawanmeh, 2014, Moussa et al., 2014).

Jin-hua et al. (2008) apresentam o framework PBNM – *Policy-based network management* – no qual a política de segurança deve ser validada com as respectivas regras de linguagem nativa de *firewall* antes de ser implementada no *firewall*. Para realizar essa validação é proposta uma especificação formal utilizando a teoria dos conjuntos. Essa especificação formal não oferece uma representação de fácil compreensão para o administrador de redes.

Os estudos de Wool (2004, 2010) mostram que é comum existirem *firewalls* com um conjunto de regras que não expressam a política de segurança definida e, portanto, tornam as redes vulneráveis. Para colaborar com o procedimento de mapeamento da política de segurança em regras nativas de *firewall*, Trevisani & Garcia (2008) sugerem a utilização de uma notação gráfica que modele a política de segurança, pois destacam que a utilização da representação visual da política de segurança em uma notação gráfica facilita o seu entendimento e, consequentemente, a sua manutenção.

1.3 Objetivo e Metodologia

O objetivo geral é contribuir para o processo de mapeamento da política de segurança para regras em linguagem nativa de *firewall* utilizando uma abordagem visual, garantindo que essas regras efetivamente implementem o que determina a política de segurança, diminuindo a possibilidade do administrador de redes cometer erros em configurações.

Objetiva-se, ainda, assegurar que: (i) a abordagem visual escolhida, a SPML (*Security Policy Modeling Language*), proposta por (Trevisani & Garcia, 2008), tenha uma representação gráfica que permita expressar as funcionalidades previstas em um *firewall* e (ii) que o processo de elaboração dessa representação gráfica possua as validações sintática e semântica, (iii) a fim de garantir que o modelo criado seja traduzido para regras corretas em linguagem nativa de *firewall* a partir de uma metalinguagem definida.

Para atender esses objetivos foi definida uma extensão da proposta inicial da SPML (a SPML2) para expressar as funcionalidades previstas em um *firewall*. Em seguida, foram definidos os formalismos sintático e semântico para a elaboração do modelo visual e, obedecendo a esses formalismos, foi desenvolvida uma ferramenta que implemente a abordagem alternativa para representação visual na modelagem da política de segurança, a SPML2. Na sequência, foi realizado um experimento

para avaliar o uso da SPML2.

Os objetivos específicos são:

- Desenvolver e validar os formalismos sintático e semântico da SPML;
- A partir do formalismo da SPML, desenvolver e validar os formalismos sintático e semântico da SPML2;
- Desenvolver a ferramenta para a modelagem definida na SPML2;
- Verificar se a utilização da SPML2 contribui para a eficiência e para a eficácia do processo de mapeamento da política de segurança para regras em linguagem nativa de *firewall*.

Tendo em vista os objetivos estipulados, e seguindo um cronograma previamente definido, as atividades que compõem a metodologia do trabalho são:

1. Busca por abordagens alternativas que apoiam o administrador de redes na tarefa de produzir regras de *firewall* em conformidade com a políticas de segurança.
2. Estudo aprofundado da versão inicial da SPML (abordagem escolhida), realizando uma análise crítica da sua simbologia e suas limitações.
3. Definição dos formalismos sintático e semântico para a SPML. Com essa atividade busca-se identificar pontos a serem melhorados na metalinguagem definida. Essa metalinguagem é utilizada para tradução para a linguagem nativa do *firewall*.
4. Definição da SPML2, evolução da SPML, incluindo seus formalismos sintático e semântico e resolvendo os problemas encontrados na versão inicial.
5. Desenvolvimento de uma ferramenta (SP2Model) capaz de implementar os modelos SPML2 obedecendo aos formalismos sintático e semântico.
6. Condução de um experimento controlado para avaliar o uso da SPML2.

1.4 Organização

O restante do trabalho está organizado em seis Capítulos. São eles:

- No Capítulo 2 é apresentado o conceito de Segurança da Informação em Redes Computacionais, bem como o processo e as técnicas utilizados para implementá-la, dando especial atenção ao *firewall*;
- No Capítulo 3 é discorrido sobre as abordagens alternativas ao processo tradicional para a configuração da política de segurança em um produto de *firewall*;
- No Capítulo 4 é apresentada a representação formal sintática e semântica para a SPML2 utilizando da gramática EBNF, de árvores de análise sintática e da Notação Z;
- No Capítulo 5 é apresentada a SPML2 juntamente à ferramenta web que a implementa, a SP2Model.
- No Capítulo 6 é apresentado o experimento, além de sua motivação, objetivos e justificativa. Também são apresentados a metodologia e o cronograma de execução;
- Por fim, no Capítulo 7 são apresentadas as conclusões e trabalhos futuros.

CAPÍTULO 2

Segurança Entre Redes

2.1 Considerações Iniciais

Neste capítulo são discutidos os elementos para prover segurança na comunicação entre redes. Na Seção 2.2 é apresentado o conceito de *firewall*, destacando as categorias em que os *firewalls* podem ser classificados e as técnicas para controlar o acesso e impor a política de segurança em um *firewall*. Em seguida, na Seção 2.3 são apresentados o filtro de pacotes, as camadas da pilha de protocolos da Internet e um cenário para facilitar o entendimento do processo de mapeamento de uma política de segurança para regras em linguagem nativa de *firewall*. Por fim, na Seção 2.4, são expostas as considerações finais.

Para que ocorra uma comunicação segura entre redes é desejável que essa comunicação possua algumas propriedades específicas como: **confidencialidade**, em que somente o remetente e o destinatário pretendido devem poder entender o conteúdo da mensagem transmitida; **autenticação do ponto final**, que possibilita ao remetente e ao destinatário confirmar a identidade da outra parte envolvida na comunicação; **integridade** da mensagem, que assegura que o conteúdo da comunicação não seja alterado, por acidente ou por má intenção, durante a transmissão e **segurança operacional**, que garante que as redes da organização não sejam comprometidas por atacantes conectados à Internet pública (Kurose & Ross, 2010).

Mecanismos de criptografia podem garantir confidencialidade, autenticação de ponto final e integridade da mensagem (Kurose & Ross, 2010), porém convém res-

saltar que os mecanismos de criptografia não são foco deste trabalho. Para garantir a segurança operacional, a utilização de *firewall* constitui a principal ferramenta para implementar a proteção da fronteira entre redes de computadores (Forouzan & Mosharraf, 2013). Além de *firewalls*, também são utilizados os sistemas de detecção de invasão (IDS, do inglês *intrusion detecting system*).

Enquanto o *firewall* tem como objetivo implementar o controle de tráfego em conformidade com a política de segurança da organização, o IDS trabalha realizando inspeção em todo o tráfego de pacotes detectando o pacote ou o conjunto de pacotes, suspeitos de serem potencialmente maliciosos (Kurose & Ross, 2010). Como neste trabalho o foco está na implementação do controle de tráfegos proposto na política de segurança, IDS não faz parte desse estudo.

2.2 *Firewalls*

Um *firewall* é uma combinação de hardware e software que examina o tráfego de pacotes de dados entre redes de equipamentos computacionais, permitindo que alguns pacotes passem e outros sejam bloqueados, de acordo com a política de segurança (Stallings, 2008).

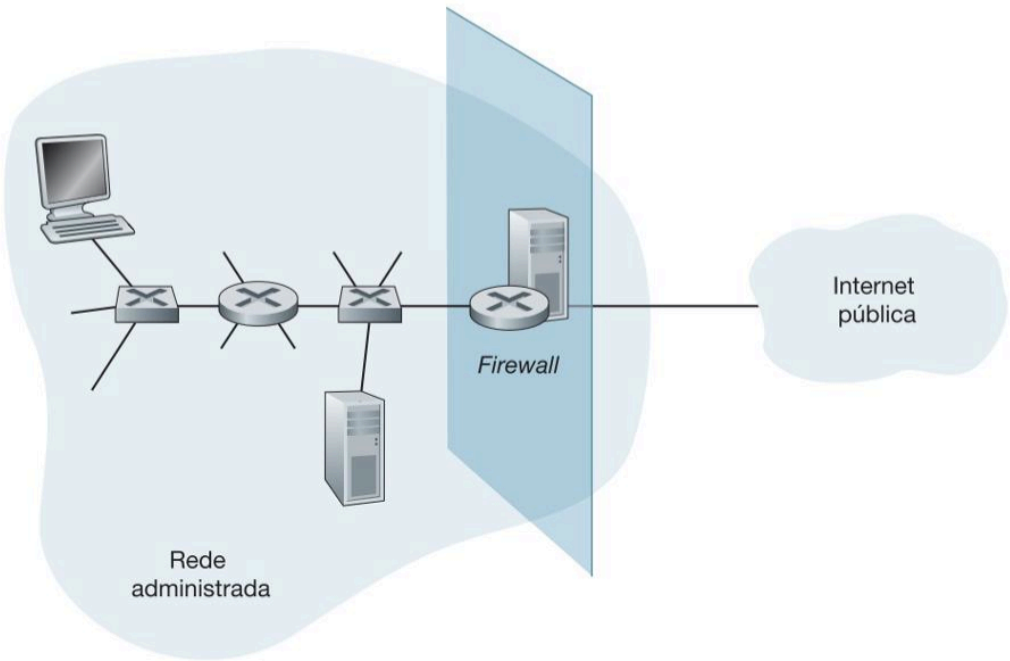


Figura 2.1: Posição do *Firewall*: entre a rede administrada e a Internet pública (Kurose & Ross, 2010, p. 539)

Como pode ser observado na Figura 2.1, o *firewall* está localizado na fronteira

entre a rede administrada e a Internet pública. Sua posição permite que ele inspecione a comunicação entre o mundo externo e os recursos da rede interna realizando a filtragem do tráfego de pacotes originados e destinados entre essas redes, estabelecendo um perímetro de segurança.

Para garantir a segurança operacional, Stallings (2008) afirma que um *firewall* necessita cumprir objetivos, como:

- Todo o tráfego externo endereçado à rede protegida e todo o tráfego originado na rede protegida com destino para redes externas devem ser entregues para o *firewall*.
- Considerando todo o tráfego entregue ao *firewall*, somente o tráfego autorizado, como definido pela política de segurança local, pode ser entregue ao seu destino.
- O próprio *firewall* deve ser imune a acesso não autorizado.

Kurose & Ross (2013) explicam que os *firewalls* podem ser classificados em três categorias:

1. Filtros de pacotes tradicionais, que examinam o cabeçalho de cada pacote isoladamente.
2. Filtros de pacotes com controle de estado, que examinam a tabela de conexões, rastreando o conjunto de pacotes que pertençam a uma determinada conexão.
3. *Gateways* de aplicação, que funcionam como um servidor específico para cada aplicação disponibilizada.

Stallings (2008) cita quatro técnicas, utilizadas pelos *firewalls*, para controlar o acesso e impor a política de segurança:

1. Controle de serviço: determina os tipos de serviços de Internet que podem trafegar entre as redes.
2. Controle de direção: define a direção em que as solicitações de inicialização de determinados serviços podem ocorrer.
3. Controle de usuário: controla o acesso a um determinado serviço baseado na identificação do usuário que está solicitando acessá-lo.

4. Controle de comportamento: controla como determinados serviços são utilizados tendo como base nas informações que estão sendo transportadas.

As técnicas para o controle de usuário e para o controle de comportamento são implementadas por *gateways* de aplicação. Para implementar as técnicas de controle de serviço e de controle de direção o *firewall* utiliza um filtro de pacotes (tradicional ou com controle de estado).

Nesta pesquisa é abordada a implementação do controle de serviço e do controle de direção, previstos na política de segurança, em um produto de *firewall*. Assim, o presente estudo concentra-se nos filtros de pacotes, portanto *gateways* de aplicação estão fora do escopo deste trabalho e não são abordados neste texto.

2.3 Protocolos e Filtro de Pacotes

O filtro de pacotes tradicional, ou o filtro de pacotes com controle de estado, aplica um conjunto de regras a cada pacote IP (Figura 2.6) que entra ou sai do *firewall* decidindo se encaminha ou descarta aquele pacote que pode ter vindo da rede interna para a Internet ou o contrário.

A rede de equipamentos computacionais e, consequentemente, a Internet são organizados em camadas, como pode ser observado na Figura 2.2. O conjunto dos protocolos das várias camadas é chamado de pilha de protocolos (Socolofsky & Kale, 1991).

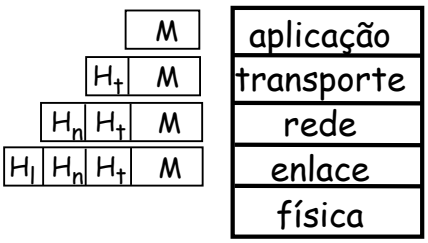


Figura 2.2: Pilha de protocolos da Internet de cinco camadas sendo: M a mensagem produzida na camada de aplicação; H_t o cabeçalho da camada de transporte; H_n o cabeçalho da camada de rede; H_l o cabeçalho da camada de enlace (Kurose & Ross, 2013).

A camada física é responsável pela transmissão dos bits por um canal de comunicação, assim, sua função é garantir que quando um lado transmitir um bit 1, o outro lado o receba como bit 1. A camada de enlace provê a entrega entre hospedeiros no mesmo enlace e, se necessário, resolve questões de controle de acesso a um

meio de comunicação compartilhado. A camada de rede é responsável pela movimentação do pacote do hospedeiro origem até o hospedeiro destino, determinando as rotas que guiam os pacotes. A camada de transporte é responsável por manter uma conversação entre a aplicação do hospedeiro origem com a aplicação do hospedeiro destino. Finalmente, a camada de aplicação é responsável por executar os serviços de rede, os quais são distribuídos por diversos sistemas finais. A aplicação de um sistema final utiliza os protocolos dessa camada para trocar pacotes de informação com a aplicação em outro sistema final (Kurose & Ross, 2010) (Tanenbaum & Wetherall, 2011).

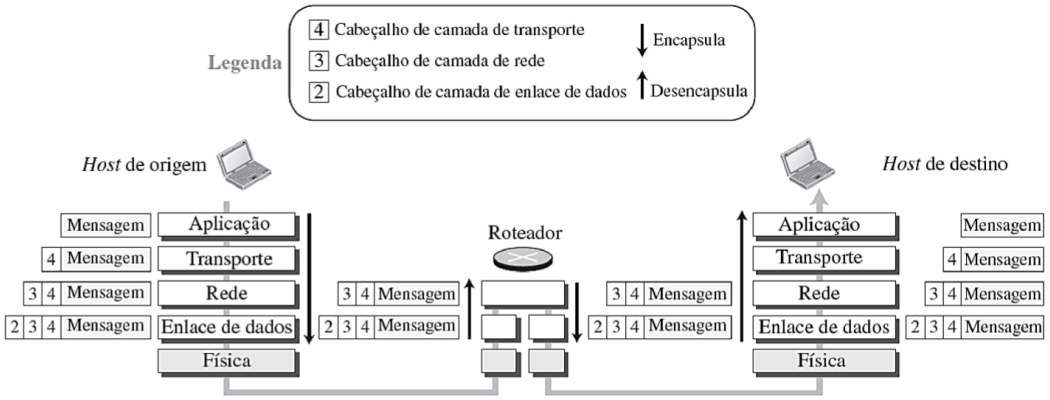


Figura 2.3: Comunicação na Internet (Forouzan & Mosharraf, 2013, p. 13).

Na Figura 2.3 pode ser observada a comunicação entre um *host* de origem e um *host* de destino. Quando um *host* de origem envia uma $Mensagem$ a um *host* de destino, essa mensagem é produzida na camada de aplicação e é entregue à camada de transporte, que adiciona um cabeçalho 4 à mensagem. A junção da $Mensagem$ da camada de aplicação com o cabeçalho 4 da camada de transporte formam o segmento da camada de transporte. O segmento da camada de transporte é entregue à camada de rede, que adiciona um cabeçalho 3. O segmento da camada de transporte e o cabeçalho 3 da camada de rede juntos criam o datagrama da camada de rede. O datagrama da camada de rede é entregue, então, à camada de enlace, que adiciona ao datagrama suas informações de cabeçalho 2, criando um quadro da camada de enlace. Em cada camada, a junção do cabeçalho com o campo de carga útil (pacote da camada superior) é denominado PDU – *Protocol Data Unit*. Quando o PDU é entregue a uma camada inferior ele é encapsulado e quando o PDU é entregue a uma camada superior ele é desencapsulado.

A comunicação na camada de aplicação ocorre entre dois serviços de rede em

execução, chamados processos. Em cada sistema final, um *socket* é associado ao processo, de forma que a comunicação é realizada com o processo associado. Como a comunicação ocorre entre dois *sockets*, deve ser definido um endereço para o *socket* local e outro endereço para o *socket* remoto.

O endereço de um *socket* identifica o processo e o equipamento onde o serviço de rede está em execução. Para endereçar um equipamento é utilizado um número IP e para identificar um processo é usado um número de porta, assim o endereço de um *socket* é a combinação de um endereço IP e um número de porta.

Existem duas versões de protocolo IP em uso na Internet, o IPv4 e o IPv6. Um endereço IPv4 pode ser público ou privado (Rekhter et al., 1996). Os endereços públicos são roteáveis na Internet, enquanto os endereços privados não podem ser roteados para fora das redes internas de uma organização.

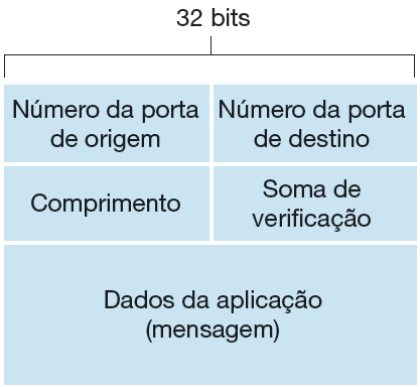


Figura 2.4: Estrutura do segmento UDP (Kurose & Ross, 2013, p. 148)

Filtros de pacotes inspecionam informações dos cabeçalhos dos pacotes da camada de rede e da camada de transporte. O cabeçalho da camada de transporte contém várias informações, sendo que interessa a este trabalho as informações “Porta Origem” e “Porta Destino”. “Porta Origem” contém o número da porta que identifica o *socket* ligado ao processo do lado transmissor e “Porta Destino” contém o número da porta que identifica o *socket* ligado ao processo do lado do receptor.

Os protocolos mais comuns para Internet utilizados na camada de transporte são o *UDP* e o *TCP*. A estrutura do segmento *UDP* pode ser observada na Figura 2.4 e a estrutura do segmento *TCP* pode ser observada na Figura 2.5. Tanto no cabeçalho do segmento *UDP* quanto *TCP*, contêm as seguintes informações: “Porta Origem” e “Porta Destino”.

Dentre as várias informações contidas no cabeçalho da camada de rede, para este trabalho interessam “IP Origem” (transmissor), “IP Destino” (receptor) e “Pro-

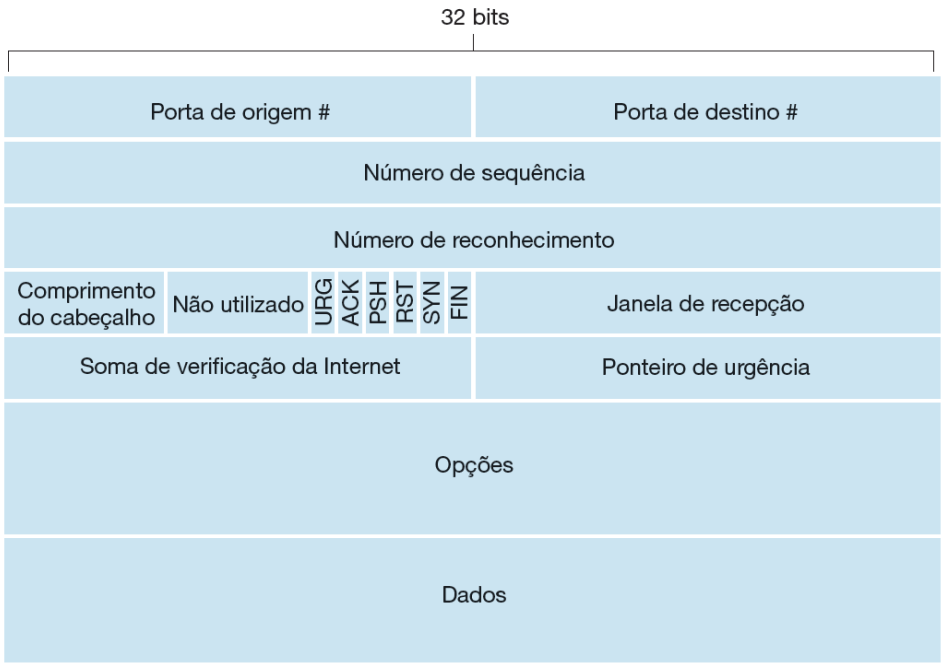


Figura 2.5: Estrutura do segmento TCP (Kurose & Ross, 2013, p. 172)

tocolo" (camada de transporte).

O principal protocolo da camada de rede é o protocolo IP e a versão mais comum é o *IPv4*, cujo formato do datagrama pode ser observado na Figura 2.6. Problemas como a escassez de endereços *IPv4* levaram à criação de uma nova versão do protocolo IP, o *IPv6*, que é a nova geração do IP (*IPng*). O protocolo *IPv6* expande a capacidade de endereçamento do *IPv4* em 2^{96} vezes, além de resolver outros problemas. Neste trabalho são utilizadas informações contidas no cabeçalho do datagrama *IPv4*.

A ação de encaminhar ou descartar um pacote é definida por regras que utilizam as informações contidas no cabeçalho do pacote da camada de rede (campo de protocolo, endereço IP de origem e endereço IP de destino) e as informações contidas no cabeçalho do pacote da camada de transporte (endereço de origem e destino que informam respectivamente o número da porta do *socket* associado ao processo de origem e o número da porta do *socket* associado ao processo de destino). Cada regra é associada a uma interface de rede. Um *firewall* posicionado entre duas redes possui duas interfaces de rede, sendo que cada interface deve conectar-se a uma rede distinta.

Se o pacote não corresponder a nenhuma regra, então, uma ação padrão é tomada. Existem duas políticas possíveis para a ação padrão:

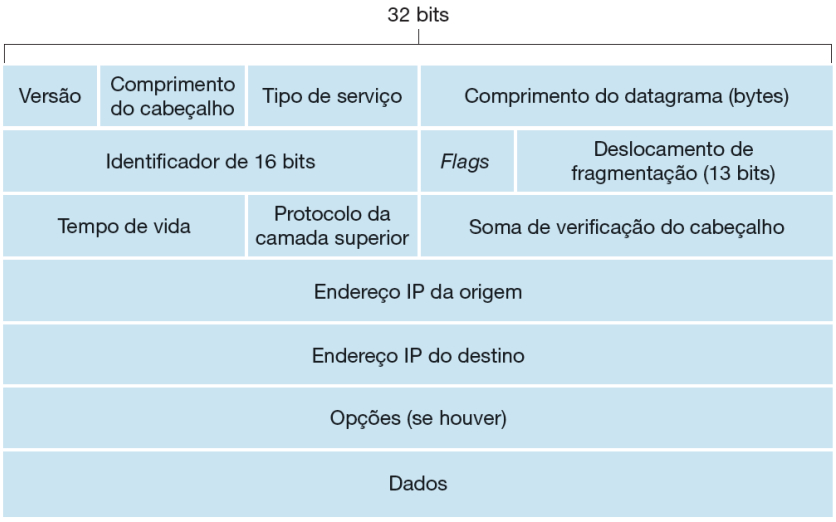


Figura 2.6: Formato do datagrama IPv4 (Kurose & Ross, 2013, p. 246)

- Descartar: o pacote que não for expressamente permitido é descartado, e;
- Encaminhar: o pacote que não for expressamente proibido é encaminhado.

2.3.1 Cenário exemplo

Para ilustrar a política de segurança é tomada como base uma instituição fictícia cujo cenário é ilustrado na Figura 2.7. Nesse cenário, computadores de diferentes departamentos de uma filial acessam serviços disponíveis em servidores na matriz. Na política de segurança da instituição está definido que todos os computadores da filial podem acessar o servidor Web e o servidor DNS, com exceção dos computadores do departamento financeiro, que não podem acessar o servidor Web. O servidor de SSH pode ser acessado apenas pelo computador *Adm1*.

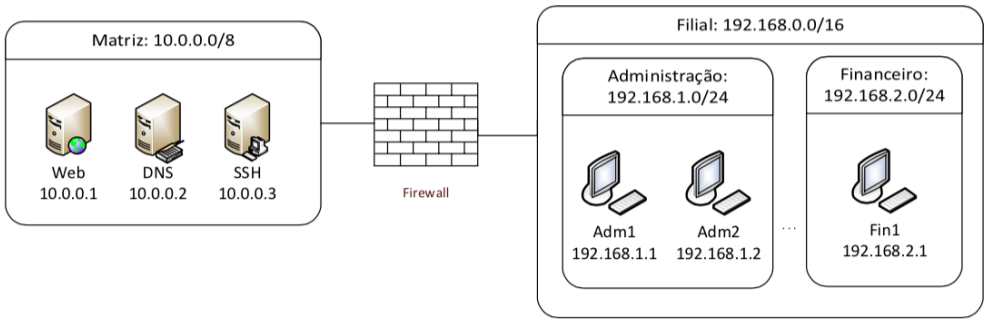


Figura 2.7: Organização da rede de uma instituição fictícia.

O texto da política de segurança deve ser compreendido pelo administrador de redes que, conhecendo os endereços IP dos objetos de rede e as portas associadas aos serviços de rede, interpreta a política de segurança e a traduz em regras considerando as informações utilizadas pelo filtro de pacotes. Na Tabela 2.1 pode ser observado o conjunto de regras correspondente à política de segurança da instituição descrita no cenário. As regras *R1* e *R2* permitem que todos os computadores da filial acessem o servidor DNS e o servidor Web, a regra *R3* bloqueia o acesso dos computadores do departamento financeiro ao servidor Web, a regra *R4* bloqueia o acesso ao servidor SSH a partir de qualquer computador da filial e, por último, a regra *R5* permite que apenas *Adm1* realize o acesso ao servidor SSH.

Tabela 2.1: Exemplos de regras associadas a informações dos cabeçalhos dos pacotes.

Regra	Ação	Protocolo	IP Origem	Porta Origem	IP Destino	Porta Destino
R0	Bloquear	*	*	*	*	*
R1	Permitir	UDP	192.168.0.0/16	*	10.0.0.2	53
R2	Permitir	*	192.168.0.0/16	*	10.0.0.1	80
R3	Bloquear	*	192.168.2.0/24	*	10.0.0.1	80
R4	Bloquear	TCP	192.168.0.0/16	*	10.0.0.3	22
R5	Permitir	TCP	192.168.1.1	*	10.0.0.3	22

Para que um filtro de pacotes possa implementar as regras de filtragem descritas na Tabela 2.1, essas regras precisam ser reescritas como regras em linguagem nativa de *firewall*. Existem vários programas que realizam a filtragem do tráfego de pacotes de rede e cada programa possui sua própria linguagem para escrita das regras. Por exemplo, no *Packet Filter*, um programa de filtro de pacotes desenvolvido por Daniel Hartmeier e atualmente mantido pela equipe de desenvolvimento do OpenBSD (OPENBSD, 2016), o conjunto de regras em linguagem nativa de *firewall* seria escrito em instruções como na Definição 2.1.

Para garantir que o *firewall* atribui segurança na comunicação entre redes, conforme prevê a política de segurança, é preciso validar se as regras escritas em linguagem nativa de *firewall* implementam o que determina a política de segurança.

Assim, empiricamente, para implementar o controle das comunicações entre redes de computadores, o administrador de redes precisa entender a política de acesso à rede para, depois, traduzir essa política em regras na linguagem de *firewall*. O administrador de redes deve certificar-se de que não haja erros. Para tal deve-se validar as regras em língua nativa de *firewall*. Como observado na Figura 2.8, o procedimento tem três fases principais: compreender, traduzir e validar.

Definição 2.1: Exemplo de regras de filtragem – sintaxe do *Packet Filter*

```
pass proto udp from 192.168.0.0/16 to 10.0.0.2 port 53
pass from 192.168.0.0/16 to 10.0.0.1 port 80
block from 192.168.2.0/24 to 10.0.0.1 port 80
block proto tcp from 192.168.0.0/16 to 10.0.0.3 port 22
pass proto tcp from 192.168.0.0/16 to 10.0.0.3 port 22
```

1. Compreender: fase na qual o administrador de redes identifica na política de segurança quais ações (autorizar o tráfego ou não autorizar o tráfego) estão associadas aos objetos de rede e aos serviços de rede.

2. Traduzir: fase na qual o administrador de redes, tendo a compreensão das ações previstas na política de segurança e conhecendo os endereços IP dos objetos de rede e as portas dos serviços, traduz essas ações para regras em linguagem nativa de *firewall*.

3. Validar: fase na qual o administrador de redes verifica se o conjunto de regras em linguagem nativa de *firewall* implementam o que determina a política de segurança.

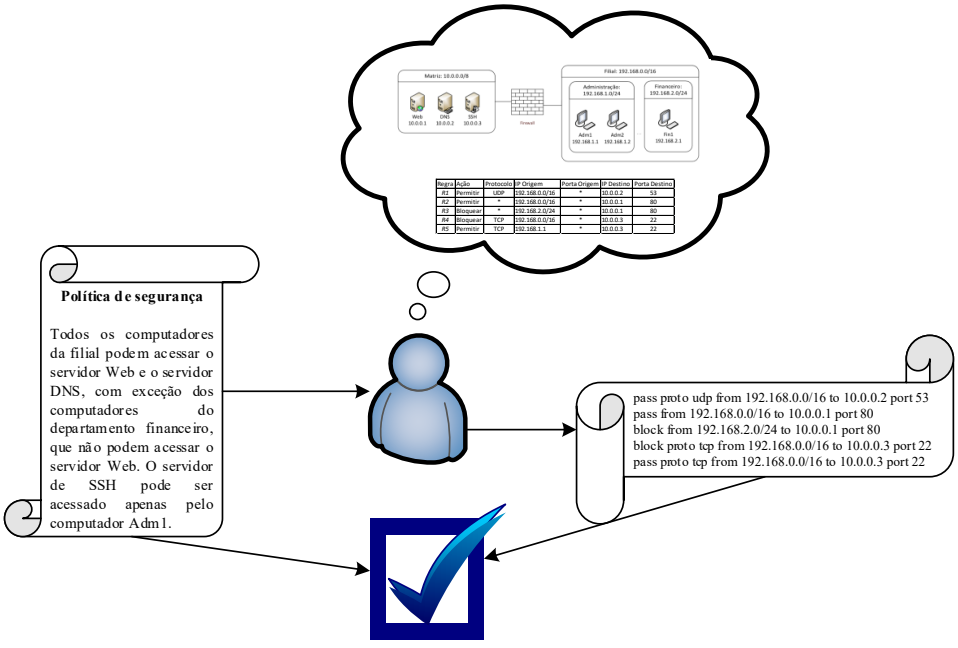


Figura 2.8: Processo de mapeamento da política de segurança em regras de *firewall*.

No filtro de pacotes as regras são analisadas em ordem sequencial. Yuan et al. (2006) destacam a existência de duas abordagens, relacionadas a seguir:

- o pacote é comparado a todas as regras e a última que coincidir com os dados do cabeçalho do pacote é executada;
- o pacote é comparado com cada uma das regras e a primeira que coincidir com os dados do cabeçalho é executada.

O filtro de pacotes *Packet Filter* utiliza a primeira abordagem, de modo que a política padrão (encaminhar ou bloquear) deve ser definida no início das regras de filtragem.

2.4 Considerações Finais

Neste capítulo são apresentadas as propriedades de uma comunicação segura entre redes: confidencialidade, autenticação do ponto final, integridade da mensagem e segurança operacional. A segurança operacional é implementada principalmente por *firewalls*, que é um mecanismo disponível para aplicar segurança na interligação entre redes, pela utilização do filtro de pacotes, implementando o que determina a política de segurança.

A configuração do *firewall* tem como documento base a política de segurança da instituição. Conforme exposto na Subseção 2.3.1, da política de segurança, o administrador de redes deve extrair as informações necessárias para criar as regras que irão expressar a política de segurança em instruções de linguagem nativa de *firewall*. As regras em linguagem nativa de *firewall* são elaboradas considerando as técnicas de controle de acesso (serviço e direção), a política de ação padrão (descartar ou encaminhar) e a abordagem para sequência de análise das regras.

Assim, para obter segurança na comunicação entre redes é preciso colaborar com o administrador de redes no procedimento de mapeamento da política de segurança para regras em linguagem nativa de *firewall*.

No Capítulo 3 são apresentadas técnicas que colaboram com o administrador de redes no procedimento de mapeamento da política de segurança em regras nativas de *firewall* em cada uma de suas fases: compreender, traduzir e validar.

CAPÍTULO 3

Modelagem da Política de Segurança

3.1 Considerações Iniciais

Como apresentado no Capítulo 2, um *firewall* só atribui segurança na comunicação entre redes se as suas regras implementam o que determina a política de segurança. O administrador de redes precisa **compreender** o que determina a política de segurança para, então, **traduzir** a política de segurança para regras em linguagem nativa de *firewall* e, para certificar-se de que não existam erros, é preciso **validar** se as regras em linguagem nativa de *firewall* implementam o que determina a política de segurança. Conforme destacado, o procedimento possui três fases importantes: compreender, traduzir e validar.

Neste capítulo são discutidas as técnicas que colaboram com o administrador de redes no mapeamento da política de segurança em regras nativas de *firewall*. Na próxima seção são exibidas as técnicas para validação de políticas de segurança. A Seção 3.2 cita as técnicas que discutem sobre a compreensão e a tradução da política de segurança em regras nativas de *firewall*. Na Seção 3.3 é apresentada a SPML. Na Seção 3.4 é discutido sobre o protótipo que implementa a SPML. Por fim, na Seção 3.5 são expostas as considerações finais, incluindo as considerações sobre a SPML que servirão de base para estender a proposta original da SPML.

3.2 Política de Segurança: Compreensão, Tradução e Validação

A maioria das técnicas localizadas na literatura apoia a fase de validação da política de segurança. El-Atawy et al. (2007) e Al-Shaer et al. (2009) apresentam técnicas para validação de políticas de segurança baseadas em testes de tráfegos gerados, utilizando perfis personalizados de utilizadores mais comuns. Al-Shaer et al. (2009) afirmam que a geração de tráfego aleatório para testar a aplicação da configuração de *firewall* não é correta nem viável, uma vez que requer um número elevado de casos de teste para uma cobertura razoável. Ambos os trabalhos sugerem que em uma sessão de teste, a partir de um conjunto de diferentes políticas de segurança, são geradas as listas de controle de acesso autorizados de acordo com os perfis personalizados de utilizadores. Por meio da automatização desse processo foram gerados vários casos de testes, os quais possibilitam a coleta e análise dos dados. O resultado dessa análise mostra que o teste de segurança automatizado além de viável oferece maior grau de confiança do que os testes aleatórios ou manuais.

Gawanmeh (2014) e Moussa et al. (2014), também desenvolveram trabalhos de pesquisa com o objetivo de criar soluções automatizadas para detectar inconsistências de configuração de *firewalls* por meio de verificação formal. A novidade nesses trabalhos é que são considerados *firewalls* distribuídos.

Os trabalhos de Jin-hua et al. (2008) e Ben Souayeh Ben Youssef & Bouhoula (2010) apresentam formas alternativas para expressar a política de segurança com o objetivo de facilitar a sua compreensão, posteriormente a política de segurança é gerada em linguagem nativa de *firewall*.

Jin-hua et al. (2008) sugerem que há necessidade de verificar as configurações de *firewall* antes e depois de serem implantadas e apresentam o *framework Policy-based network management* (PBNM), que pode ser observado na Figura 3.1. O PBNM, utilizando especificação formal, realiza a análise das relações entre as regras produzidas pelo administrador de redes e a política de segurança (*Enforcement Validation Engine*). Após realizar a validação, o mecanismo efetua a entrega das regras validadas para o administrador. O *Enforcement Validation Engine* em conjunto ao *Policy Enforcement Point* realizam o monitoramento e o pós-teste para a ferramenta de análise.

Ben Souayeh Ben Youssef & Bouhoula (2010) utilizam mecanismos baseados em sistemas de inferência e especificação lógica para validar se o conjunto de regras em linguagem nativa de *firewall* está em conformidade com os perfis de política

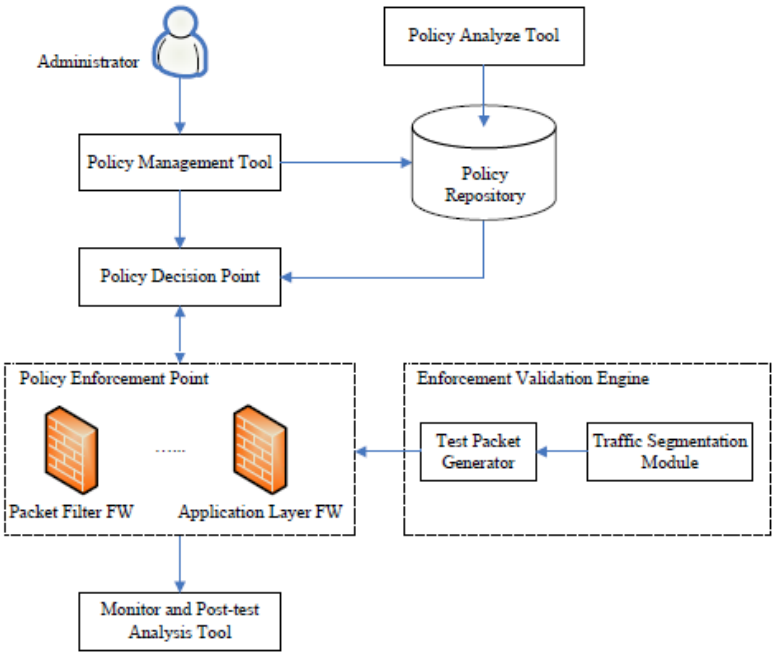


Figura 3.1: *Framework* gerenciamento de redes baseado em políticas (Jin-hua et al., 2008).

de segurança previamente configurados. Posteriormente, as regras validadas são entregues escritas em linguagem nativa de *firewall*.

O trabalho de Trevisani & Garcia (2008) apresenta a SPML – *Security Policy Modeling Language*, uma abordagem que utiliza a representação gráfica para modelar a política de segurança, e posteriormente, traduzir essa política de segurança em regras nativas de *firewall*. A SPML foi utilizada como base para conceber e evoluir a presente pesquisa. Foram estudados os trabalhos produzidos referente a SPML (monografias e artigo), além de reuniões com os autores do trabalho.

3.3 Abordagem Visual para Modelagem da Política de Segurança

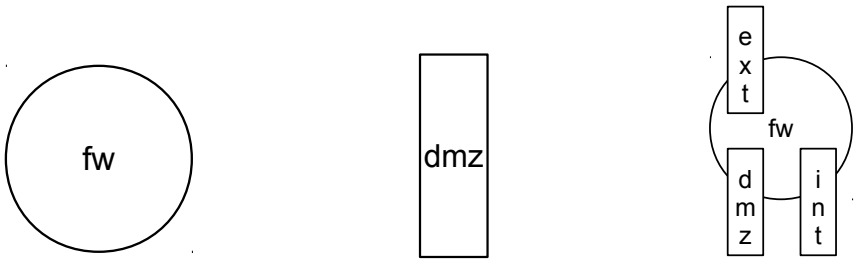
Para utilizar uma representação gráfica, Trevisani & Garcia (2008) propuseram uma linguagem para modelagem da política de segurança, a SPML – *Security Policy Modeling Language*. A notação da SPML propõe dois diagramas para representar graficamente um *firewall*: um diagrama para as regras de tradução e outro para as regras de filtragem. Os diagramas apresentam componentes interconectados de modo a representar graficamente as funcionalidades de um *firewall*.

Cada componente representa uma funcionalidade do *firewall* e, de acordo com a funcionalidade, são definidos os atributos do componente. Nem todos os atributos são visíveis na notação gráfica, mas são necessários para possibilitar a tradução da notação gráfica para a linguagem de configuração nativa de um *firewall*. A SPML classifica os componentes como: componentes do *firewall*, componentes de filtragem, componentes de tradução e componentes externos. Eles são apresentados a seguir.

3.3.1 *Firewall* e seus componentes

Para definir as características e a configuração de um *firewall* a SPML possui basicamente dois componentes: *firewall* e interface. Para o componente *firewall*, um círculo é utilizado para representar o par hardware+software, que executa as funções do *firewall*, apresentado na Figura 3.2(a). O componente *firewall* possui os seguintes atributos: nome do *firewall*, domínio, *gateway*, DNS e um *flag*, que indica se os pacotes recebidos fragmentados devem ser remontados antes de serem entregues ao destino.

Um *firewall* pode possuir uma ou mais interfaces de rede, que permitem a conexão dos componentes externos ao *firewall*. Não é permitido que um componente externo conecte-se diretamente ao *firewall*, sendo necessário o uso do componente interface. A interface é representada graficamente por um retângulo (Figura 3.2(b)) e possui os seguintes atributos: nome, dispositivo, endereço IP, máscara de sub-rede e antispoof. Um *firewall* com três interfaces de rede é apresentado na Figura 3.2(c).



(a) *Firewall*

(b) Interface

(c) *Firewall* com 3 interfaces

Figura 3.2: Os Componentes *Firewall* e Interface, adaptado de Trevisani & Garcia (2008)

3.3.2 Componentes de filtragem

Os componentes de filtragem na SPML representam como o *firewall* realiza a filtragem de pacotes. Eles representam os pacotes por um segmento de reta com uma seta, como apresentado na Figura 3.3(a).

Esse componente é utilizado no diagrama de filtragem. Quando a seta aponta para o *firewall* indica que o fluxo é de entrada. E se apontar para o componente externo indica que o fluxo é de saída. Por padrão, todo fluxo é com controle de estado e para indicar que o *firewall* não precisa manter o estado da conexão, o componente deve ser representado por uma linha com duas setas como representado na Figura 3.3(b).

(a) Fluxo Web com controle de estado

(b) Fluxo sem controle de estado

Figura 3.3: Os Componentes de filtragem, adaptado de Trevisani & Garcia (2008)

Os atributos de um fluxo são: nome, versão do endereço IP, protocolo, porta de origem, porta de destino, controle de estado e identificador. O identificador deve ser único para os fluxos de entrada e é utilizado para relacionar um fluxo de entrada a um ou vários fluxos de saída.

3.3.3 Componentes de tradução

Os componentes de tradução são divididos em componente para tradução de endereço de rede, *NAT*, e componente para redirecionamento de serviços, *RDR* (Srisuresh & Holdrege, 1999). Para a versão do endereço IPv6 não há necessidade de tradução, pois todos os *hosts* de uma rede tem endereços públicos (Brito, 2013). Esses componentes são utilizados no diagrama de tradução e a conexão entre eles é representada por um segmento de reta tracejado com uma seta na extremidade, como apresentado na Figura 3.4(a).

O componente de tradução de endereço de rede (Figura 3.4(b)) é representado por um círculo contendo um número, ele expressa o *SrcNat*, que deve estar conectado a uma ou mais entidades externas, e o *DestNat*, que deve estar conectado a uma interface do *firewall*. O componente indica a tradução 1:N (um para muitos) ou 1:1 (um para um). O componente *SrcNat* possui os atributos: versão do endereço IP, protocolo, porta de origem, porta de destino e NID (acrônimo de *NAT ID*, um número inteiro usado para identificar o componente ao seu *DestNat*).

Um *DestNat* possui somente o atributo *NAT ID*, pois os demais atributos necessários são definidos no seu correspondente *SrcNat*.

Para representar o redirecionamento de serviço de rede é utilizado um retângulo (Figura 3.4(c)) com os cantos arredondados. O componente de redirecionamento, do mesmo modo que o *NAT*, contém um número mais a direita do componente, que indica o *SrcRdr* e o *DestRdr*. Os atributos de um *SrcRdr* são: versão do endereço IP, protocolo, porta de origem, porta de destino e RID (acrônimo de *RDR ID*, um número inteiro usado para identificar o componente ao seu *DstRdr*).

O *DestRdr* deve ser conectado a uma entidade externa que contém a porta na qual devem ser entregues os pacotes redirecionados. Seus atributos são porta de destino e RID.

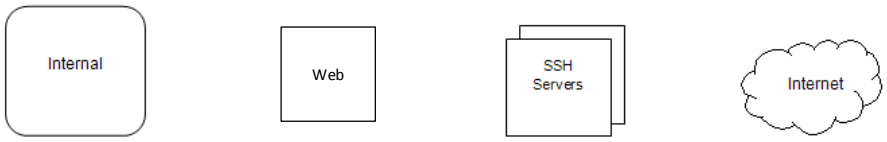
(a) Conexão de componentes (b) *NAT* (c) *RDR*

Figura 3.4: Os Componentes de tradução, adaptado de Trevisani & Garcia (2008)

3.3.4 Entidades Externas

Os componentes externos ao *firewall* – as entidades externas – podem ser a origem ou o destino de um fluxo, de uma tradução ou de um redirecionamento. As entidades externas não podem conectar-se entre si. Elas obrigatoriamente precisam conectar-se ao *firewall*.

O componente *rede* é uma rede TCP/IP, representado graficamente por um quadrado com os cantos arredondados (Figura 3.5(a)). Os atributos de uma rede são nome, prefixo de rede e máscara de sub-rede. *Host* é o componente utilizado para qualquer dispositivo capaz de receber um endereço IP, representado por um quadrado (Figura 3.5(b)), e seus atributos são nome: domínio, endereço IP e máscara da sub-rede. *Hostlist* é o componente que define um grupo de *hosts* que pertencem a uma mesma política de segurança, representado graficamente por dois quadrados sobrepostos (Figura 3.5(c)), e seus atributos são nome: lista de *hosts* previamente definidos no componente *host*. Finalmente, o componente *inet* – representado graficamente por uma nuvem (Figura 3.5(d)) – é usado para definir a Internet ou uma rede qualquer que não tenha o seu prefixo IP conhecido, sendo seu único atributo o nome.



(a) Rede de equipamentos (b) Host (c) Lista de Hosts (d) Rede de prefixo desconhecido

Figura 3.5: As Entidades Externas, adaptado de Trevisani & Garcia (2008)

3.3.5 Diagrama de Filtragem

Para apresentar o diagrama de filtragem, considera-se a política de segurança de uma universidade. A rede acadêmica deve ser isolada da rede administrativa e todos os equipamentos têm acesso à Internet com controle de *URL*[1] solicitada, exceto os *hosts* dos professores que podem acessar a Internet diretamente. Um servidor *proxy squid* presente na rede desmilitarizada (*DMZ*) controla o que pode ser acessado. O diagrama de filtragem que modela a política de segurança descrita é mostrado na Figura 3.6. Nela podem ser observados os identificadores de 1 a 4, sendo que o identificador 1 representa o tráfego originado na rede acadêmica e destinado ao servidor *proxy squid*, o identificador 2 indica o tráfego originado no servidor *proxy squid* e destinado à Internet, o identificador 3 indica o tráfego originado nos *hosts* dos professores destinado à Internet e, finalmente, o identificador 4 indica o tráfego originado na rede administrativa e destinado ao servidor *proxy squid*.

3.3.6 Diagrama de Tradução de Endereços

Para que os pacotes de redes privadas possam trafegar na Internet, é necessário que ocorra a tradução de endereço de rede para um endereço público. A representação de um diagrama de tradução que realiza a tradução conforme definido pela política de segurança da universidade pode ser observada na Figura 3.7.

Supondo que exista um *host* na rede interna, que possua endereço IPv4 privado, e que esse *host* necessite ser acessado a partir da Internet, é preciso especificar mais um diagrama de tradução. Esse segundo diagrama de tradução expressa o redirecionamento dos acessos originados na Internet para o *hots* localizado na rede interna. O diagrama de tradução que implementa o redirecionamento da porta 80 para um *host* na rede interna pode ser visualizado na Figura 3.8.

[1]Uniform Resource Locator

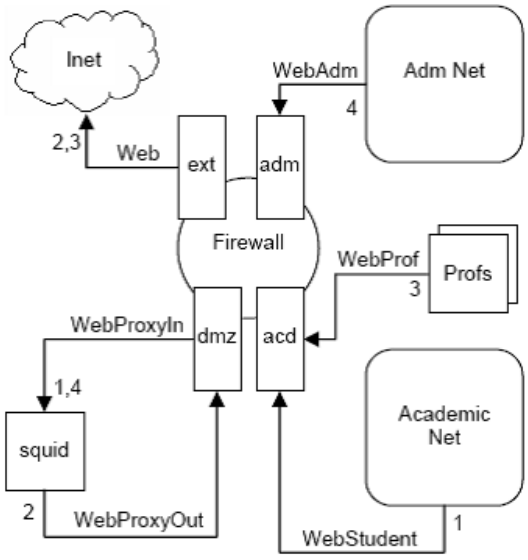


Figura 3.6: Diagrama de filtragem, adaptado de Trevisani & Garcia (2008)

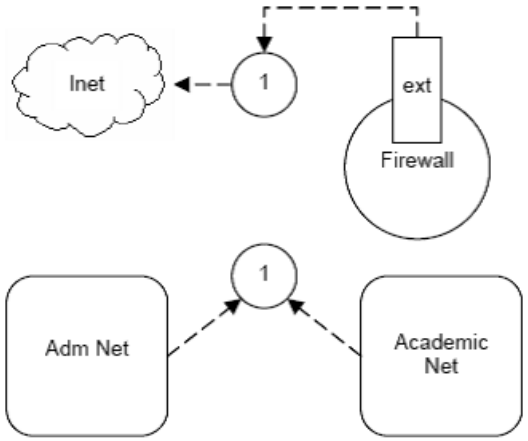


Figura 3.7: Diagrama de tradução, adaptado de Trevisani & Garcia (2008)

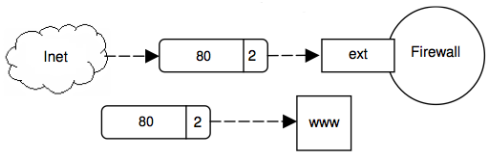


Figura 3.8: Diagrama de tradução (redirecionamento), adaptado de Trevisani & Garcia (2008)

3.3.7 Análise Sequencial das Regras

O *firewall* analisa as regras em sequência, conforme descrito no Capítulo 2, podendo utilizar duas abordagens para decidir quais regras processar. Para permitir

que a SPML defina a abordagem de ordem em que deve processar as regras, como também apresentar uma visualização adequada da sequência de processamento dos fluxos, é proposto por Trevisani & Garcia (2008) uma visualização gráfica transversal, a *SideView*. Uma modelagem SPML que determina a ordem das regras e a abordagem do *firewall* é mostrada na Figura 3.9.

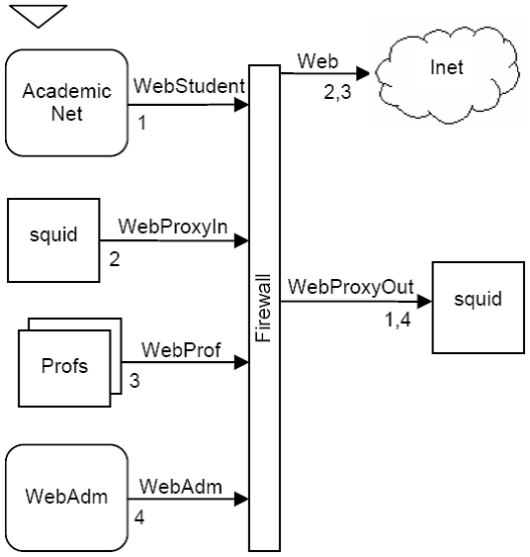


Figura 3.9: Diagrama de filtragem – sequência das regras, adaptado de Trevisani & Garcia (2008)

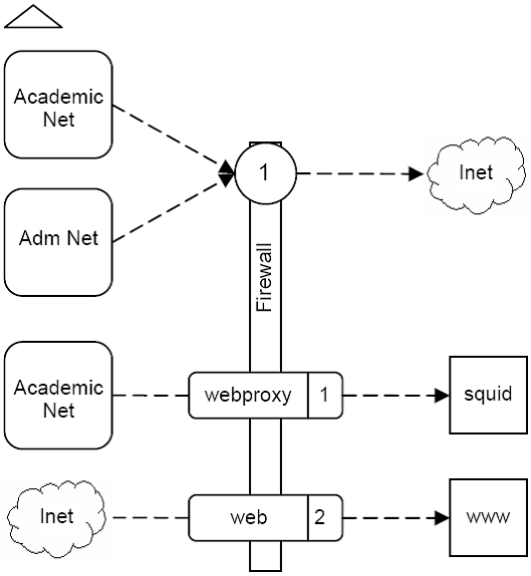


Figura 3.10: Diagrama de tradução – sequência das regras, adaptado de Trevisani & Garcia (2008)

Na *SideView*, para selecionar a abordagem considerada na análise das regras, é utilizado um triângulo, localizado no topo à esquerda. Quando o triângulo aponta para baixo, o *firewall* analisa todas as regras antes de tomar qualquer ação sobre o pacote, como pode ser observado na Figura 3.9.

Analogamente ao diagrama de filtragem, o diagrama de tradução também requer a definição da ordem das regras e a abordagem a ser utilizada, Uma modelagem contemplando essas características é apresentada na Figura 3.10. Nesse caso, o triângulo apontando para cima indica que a primeira regra que coincidir com os parâmetros determinados é executada.

3.4 O Protótipo *Security On Hands*

A proposta da SPML é que, uma vez que esteja elaborada a modelagem gráfica da política de segurança, o administrador de redes possa gerar automaticamente as regras em linguagem nativa de *firewall* a partir da modelagem. Para realizar a conversão de modelos em um conjunto de regras em linguagem de *firewall* foi desenvolvido um protótipo chamado SoH (*Security on Hands*), escrito em Java. Tal protótipo é organizado em módulos que realizam a edição dos diagramas e a tradução para configuração de *firewall*. O detalhamento da SoH e sua arquitetura foge ao escope deste trabalho.

3.5 Considerações Finais

Neste capítulo são apresentadas as fases do procedimento de conversão da política de segurança em regras nativas de *firewall*: compreender, traduzir e validar. Também são listados trabalhos que apresentam técnicas para apoiar o administrador de redes em alguma dessas fases do procedimento.

Foram localizados vários trabalhos que apoiam a fase de validação (El-Atawy et al., 2007, Al-Shaer et al., 2009, Gawanmeh, 2014, Moussa et al., 2014). Jin-hua et al. (2008) e Ben Souayeh Ben Youssef & Bouhoula (2010). Além de validar, traduzem as regras resultantes da validação para regras em linguagem nativa de *firewall*. Em seu trabalho, Jin-hua et al. (2008) sugerem que além da validação das regras implementadas no *firewall* também existe a necessidade de validar as regras antes de serem implantadas no *firewall*. E apresentam uma técnica de validação baseada na especificação formal e análise das relações entre as regras e a política de segurança. Dando apoio à fase de compreensão e tradução, foi localizado apenas o trabalho de Trevisani & Garcia (2008).

A SPML inicialmente apresentada por Trevisani & Garcia (2008) não possui

formalismos sintático e semântico necessários a uma linguagem. Os formalismos sintático e semântico são necessários para a correta tradução da representação visual produzida em SPML para regras em linguagem nativa de *firewall*. O Capítulo 4 apresenta o formalismo sintático e o formalismo semântico para a SPML desenvolvido neste trabalho de pesquisa.

O protótipo SoH foi implementado considerando a especificação inicial da SPML. Embora as restrições sintáticas da SPML tenham sido consideradas como parte do protótipo, no desenvolvimento não foi realizada a validação utilizando definições sintática e semântica especificadas formalmente.

Alguns problemas foram observados, por exemplo:

1. Observa-se que a SPML utiliza um mesmo elemento gráfico (o círculo) para definir dois elementos do modelo – *firewall* e componentes de tradução – o que pode ser considerado ambiguidade;

2. Há um símbolo para tráfego com controle de estado e outro símbolo para tráfego sem controle de estado, mas um parâmetro é usado para determinar se o controle de estado é ou não utilizado;

3. O uso de múltiplos diagramas (tradução e filtragem, por exemplo) para expressar uma política de segurança dificulta seu entendimento imediato, pois é necessário interpretar ambos para tomar conhecimento do que, de fato, está modelado;

4. Não existe uma representação gráfica que defina o bloqueio de um pacote, sendo impossível representar o que o *firewall* deve bloquear.

5. As restrições sintáticas da SPML não foram especificadas formalmente, embora tenham sido implementadas como parte do protótipo SoH. O mesmo vale para a definição semântica. As definições sintática e semântica possibilitam validar se a política de segurança modelada está expressa em conformidade com as regras da SPML.

CAPÍTULO 4

Formalismo para SPML

4.1 Considerações Iniciais

Uma política de segurança expressa em regras modeladas em SPML deve ser traduzida em uma especificação equivalente para o *firewall*. A notação gráfica apresentada no Capítulo 3 (proposta original da SPML) não possui uma definição formal, do ponto de vista de uma linguagem. Como uma linguagem, faz-se necessário prover recursos para verificação de regras sintáticas e semânticas. E, para isso, é necessária uma especificação formal da sintaxe e da semântica da SPML.

Convém destacar que este trabalho além de definir as especificações formais, sintática e semântica, da SPML também apresenta a sua evolução, a SPML2. No Capítulo 5 são apresentadas as alterações nos formalismos sintático e semântico para acolher tais alterações.

Tal especificação formal é utilizada não só para validar as representações gráficas que expressam uma política de segurança, mas também é empregada na tradução dessa política de segurança em um conjunto de regras que define o comportamento de um *firewall*. Assim, neste capítulo são apresentadas as especificações de uma metalinguagem capaz de acolher todos os elementos da política de segurança definidos na SPML. Uma metalinguagem é uma linguagem usada para descrever outra, afirma Sebesta (2011). E, neste caso, a metalinguagem ora proposta (e apresentada em forma textual) deve ser capaz de expressar a política definida nas representações gráficas do diagrama em outra linguagem (específica para o *firewall*).

É importante garantir que uma abordagem alternativa para representação visual possua especificação formal, pois métodos formais têm mostrado grande sucesso nas áreas de sistemas críticos. O formalismo provê modelos mais completos, mais consistentes e confiáveis para serem usados em softwares (Hussain et al., 2011).

4.2 Definição léxica da SPML

Inicialmente, é preciso definir quais são os elementos básicos da metalinguagem, ou seja, quais as palavras consideradas chave para a metalinguagem. Considerando que a metalinguagem deve mapear os elementos usados para a configuração da política de segurança em um *firewall* e presentes na SPML, os lexemas que compõem a linguagem devem ser identificados por *tokens*. Os *tokens* identificados para a metalinguagem proposta são:

- *program*: conjunto de linhas que definirão o *firewall*;
- *line*: definição de algum componente do *firewall*;
- *fw*: definição de um firewall;
- *if*: definição de uma interface;
- *extent*: definição de uma entidade externa;
- *ht*: definição de um host;
- *htl*: definição de uma lista de hosts;
- *net*: definição de uma rede de equipamentos;
- *un*: definição de uma rede cujo endereço é desconhecido;
- *flw*: definição de regras de filtragem de fluxos;
- *ifl*: definição de um fluxo de entrada;
- *af*: definição da versão do endereço ip;
- *state*: definição do tipo da conexão;
- *ofl*: definição de fluxo de saída;
- *napt*: definição de regras de tradução e redirecionamento;

- *srcnat*: definição de fluxo de entrada para tradução;
- *dstnat*: definição de fluxo de saída já traduzido;
- *srcrdr*: definição de fluxo de entrada a ser redirecionado;
- *dstrdr*: definição de fluxo de saída já redirecionado;
- *napt*: definição de regras de tradução e redirecionamento;
- *device*: definição do dispositivo da interface;
- *netmask*: definição da máscara de subrede;
- *antispoof*: definição da utilização do recurso antispoof;
- *domain*: definição do domínio;
- *gateway*: definição do endereço ip do gateway;
- *dns*: definição do endereço IP do servidor de DNS;
- *scrub*: definição da utilização do recurso de remontagem de pacotes;
- *proto*: definição do protocolo da camada de transporte;
- *srcport*: definição da porta de origem;
- *dstport*: definição da porta de destino;
- *fwname*: definição do nome do *firewall*;
- *ifname*: definição do nome da interface;
- *htname*: definição do nome do host;
- *htlname*: definição do nome da lista de hosts;
- *netname*: definição do nome da rede;
- *uname*: definição do nome da rede sem endereço ip;
- *iflname*: definição do nome de um fluxo de entrada de dados;
- *ofname*: definição do nome de um fluxo de saída de dados.

4.3 Regras de Sintaxe da SPML

Regras sintáticas devem ser definidas para que uma máquina seja capaz de analisar sentenças expressas em uma metalinguagem. Desse modo, é preciso definir um conjunto de regras que, no caso de linguagens de programação, é usualmente definido em BNF (*Backus-Naur Form*) (Sebesta, 2011, Chomsky, 1956, Backus & Naur, 1959). Entretanto, a versão estendida **EBNF** faz-se necessário, uma vez que a BNF restringe o uso de expressões que facilitam a definição da metalinguagem para a SPML (Ramos et al., 2009). Na Definição 4.1 é apresentada a gramática utilizando a representação **EBNF** para a SPML.

Os *tokens* <fwname>, <ifname>, <htname>, <htlname>, <netname>, <uname>, <iflname>, <oflname>, <ip>, <netmask>, <domain>, <gateway> e <dns> não foram definidos neste documento, pois utilizam a definição contida no apêndice A da RFC-3986 (Berners-Lee et al., 2005).

O mapeamento na metalinguagem – considerando os *tokens* (Seção 4.2) e a EBNF (Definição 4.1) apresentada – do modelo em SPML mostrado na Figura 3.6 é descrito na Definição 4.2. Nessa definição observa-se que há uma sequência específica – por exemplo, deve-se definir primeiramente o *firewall* para, depois, definir as interfaces a ele associadas.

4.4 Árvore de Análise Sintática

Para verificar a estrutura da linguagem, é utilizada uma árvore de análise sintática. A árvore de análise sintática é uma estrutura hierárquica em que cada nó é rotulado com um símbolo não terminal, e cada nó folha é rotulado com um símbolo terminal. Uma árvore de análise sintática (árvore de derivação) representa as sequências de derivação em uma gramática livre de contexto (Sebesta, 2011). Deve existir uma árvore de derivação para toda e qualquer cadeia derivável partindo da raiz da gramática (Aho et al., 2008) – uma gramática livre de contexto é **não ambígua** se, para toda e qualquer cadeia pertencente à linguagem, existir uma única sequência de derivações mais à esquerda e uma única sequência de derivações mais à direita (Ramos et al., 2009).

Assim, cada subárvore descreve uma instância de uma abstração de uma sentença expressa segundo a gramática que define a linguagem (Sebesta, 2011). E a derivação proporciona um método para a construção de uma cadeia específica de terminais partindo de um não-terminal inicial (Louden, 2004).

Para ilustrar as derivações a partir da EBNF são apresentada algumas árvores de

Definição 4.1: Gramática da SPML expressa em EBNF

```
<program> :: = <line>{<line>}EOF
<line> ::= <fw>
         | <if>
         | <flw>
         | <extent>
<fw> ::= fw(<fwname>,<domain>,<gateway>,<dns>,<scrub>)
<if> ::= if(<ifname>,<device>,<ip>,<netmask>,<antispoof>,<fwname>)
<antispoof> ::= yes | no
<scrub> ::= yes | no
<ht> ::= ht(<htname>,<domain>,<ip>,<netmask>)
<htl> ::= htl(<htlname>,<htname>{,<htname>})
<net> ::= net(<netname>,<ip>,<netmask>)
<un> ::= un(<uname>)
<flw> ::= <ifl> | <ofl> | <napt>
<ifl> ::= ifl(<iflname>,
              <af>,
              <proto>,
              <srcport>,
              <dstport>,
              <state>,
              <extent>,
              <fid>)
<af> ::= IPv4 | IPv6
<proto> ::= tcp
          | udp
          | icmp
<srcport> ::= 0 .. 65535
<sdstport> ::= 0 .. 65535
<state> ::= statefull | stateless
<extent> ::= <htname>
           | <htlname>
           | <netname>
           | <uname>
<ofl> ::= ofl(<oflname>,<extent>,<fid>{,<fid>})
<napt> ::= <srcnat>
         | <dstnat>
         | <srcrdr>
         | <dstrdr>
<srcnat> ::= srcnat(<af>,
                    <proto>,
                    <srcport>,
                    <dstport>,
                    <extent>,
                    <nid>)
<dstnat> ::= dstnat(<extent>,<nid>)
<srcrdr> ::= srcrdr(<af>,<proto>,<srcport>,<dstport>,<extent>,<rid>)
<dstrdr> ::= dstrdr(<extent>,<dstport>,<rid>)
```

Definição 4.2: Tradução do modelo apresentado em SPML para a metalinguagem

```
1  fw(fwUniv, unesp.br, 1.2.3.3, 8.8.8.8, yes)
2  if(ext, eth0, 1.2.3.2, 255.255.255.252, yes, fwUniv)
3  if(adm, eth1, 192.168.0.1, 255.255.255.0, yes, fwUniv)
4  if(acd, eth2, 10.0.0.1, 255.0.0.0, yes, fwUniv)
5  if(dmz, eth3, 172.16.0.1, 255.255.255.224, yes, fwUniv)
6  ht(squid, unesp.br, 172.16.0.10, 255.255.255.224)
7  ht(www, unesp.br, 172.16.0.11, 255.255.255.224)
8  ht(prof1, unesp.br, 10.1.0.1, 255.0.0.0)
9  ht(prof2, unesp.br, 10.1.0.2, 255.0.0.0)
10 ht(prof3, unesp.br, 10.1.0.3, 255.0.0.0)
11 ht(prof4, unesp.br, 10.1.0.4, 255.0.0.0)
12 ht(prof5, unesp.br, 10.1.0.5, 255.0.0.0)
13 htl(profs, prof1, prof2, prof3, prof4, prof5)
14 net(admnet, 192.168.0.0, 255.255.255.0)
15 net(acdnet, 10.0.0.0, 255.0.0.0)
16 un(inet)
17 ifl(webstudent, IPv4, tcp, 0, 80, statefull, acdnet, 1)
18 ifl(webproxyout, IPv4, tcp, 0, 80, statefull, squid, 2)
19 ifl(webprof, IPv4, tcp, 0, 80, statefull, profs, 3)
20 ifl(webadm, IPv4, tcp, 0, 80, statefull, admnet, 4)
21 ofl(webproxyin, squid, 1, 4)
22 ofl(web, inet, 2, 3)
23 dstnat(inet, 1)
24 srcnat(IPv4, tcp, 0, 80, admnet, 1)
25 srcnat(IPv4, tcp, 0, 80, acdnet, 1)
26 srcrdr(IPv4, tcp, 0, 80, acdnet, 1)
27 dstrdr(squid, 3128, 1)
28 srcrdr(IPv4, tcp, 0, 80, inet, 2)
29 dstrdr(www, 80, 2)
```

derivações [1]. A árvore de análise sintática da instrução da linha 1 da Definição 4.2 é mostrada na Figura 4.1, na qual observa-se a derivação à direita para o *token* **fw**.

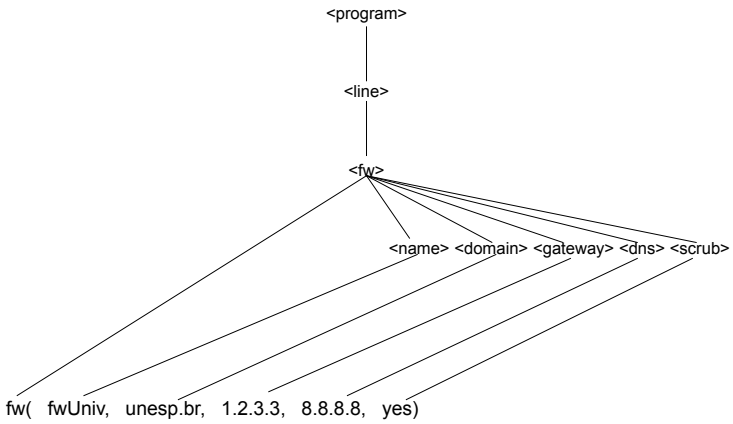


Figura 4.1: Árvore de derivação da instrução **fw**

A árvore de análise sintática da instrução da linha 2 da Definição 4.2, que define uma interface, é apresentada na Figura 4.2. Nota-se que na definição da interface já é informado o seu respectivo *firewall*.

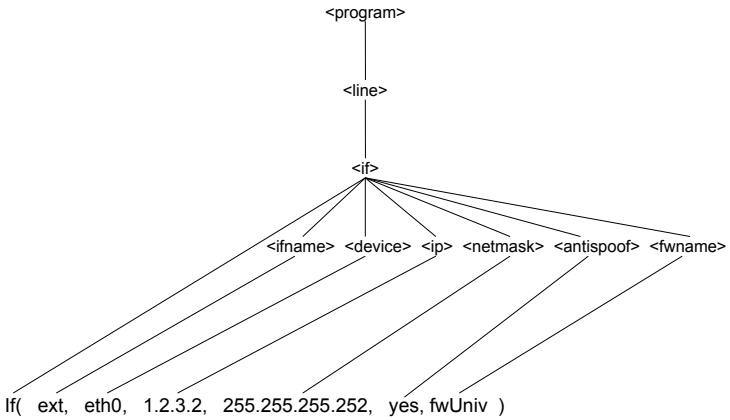


Figura 4.2: Árvore de derivação da instrução **if**

A definição de um *host*, detalhada pela árvore de análise sintática apresentada na Figura 4.3, é a derivação da instrução da linha 6 da Definição 4.2.

A árvore de análise sintática da instrução da linha 13 da Definição 4.2, que define uma lista de *hosts* é apresentada na Figura 4.4. Os *hosts* utilizados nessa instrução devem ser previamente criados por instruções que derivam do *token* **ht**.

[1] Uma sentença de cada instrução definida na EBNF é apresentada nesta seção.

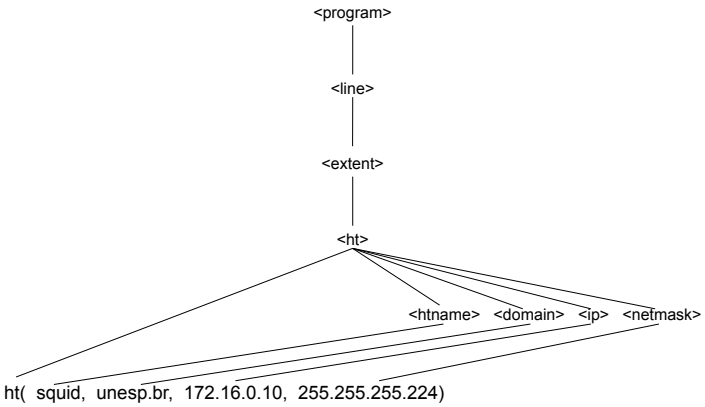


Figura 4.3: Árvore de derivação instrução **ht**

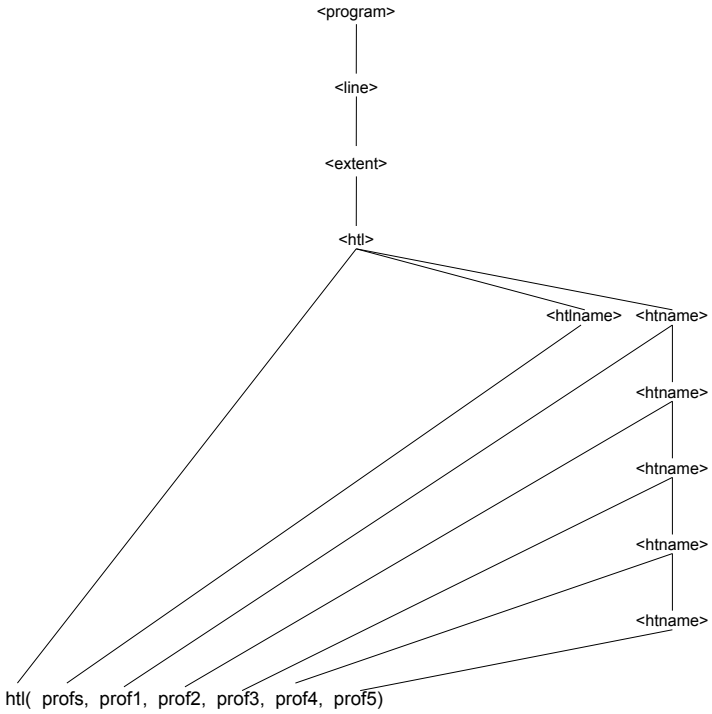


Figura 4.4: Árvore de derivação da instrução **htl**

A Definição 4.2 possui várias redes e para exemplificar a árvore de análise sintática da instrução que define uma rede de equipamentos, utilizou-se a instrução **net** da linha 14 conforme apresentado na Figura 4.5.

A árvore de análise sintática da instrução da linha 16 da Definição 4.2, que define uma rede sem endereço IP atribuído, é apresentada na Figura 4.6. A Internet é definida utilizando esta instrução.

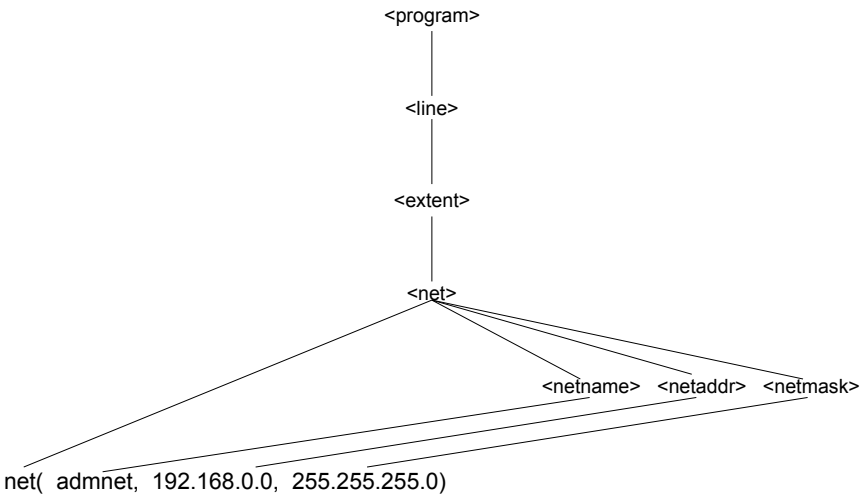


Figura 4.5: Árvore de derivação da instrução **net**

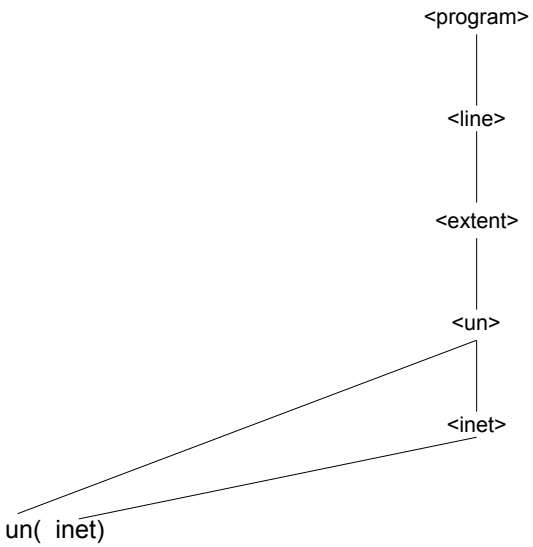


Figura 4.6: Árvore de derivação da instrução **un**

Os fluxos de entrada são aqueles endereçados ao *firewall*. A árvore de análise sintática da instrução da linha 17 da Definição 4.2 define um fluxo endereçado ao *firewall*, conforme apresentado na Figura 4.7.

O fluxo de saída utiliza um identificador previamente definido em um fluxo de entrada. A árvore de análise sintática da instrução da linha 21 da Definição 4.2, que define um fluxo que parte do *firewall*, é apresentada na Figura 4.8.

No fluxo a ser traduzido o endereço IP, *token* **srcnat**, seu identificador é pre-

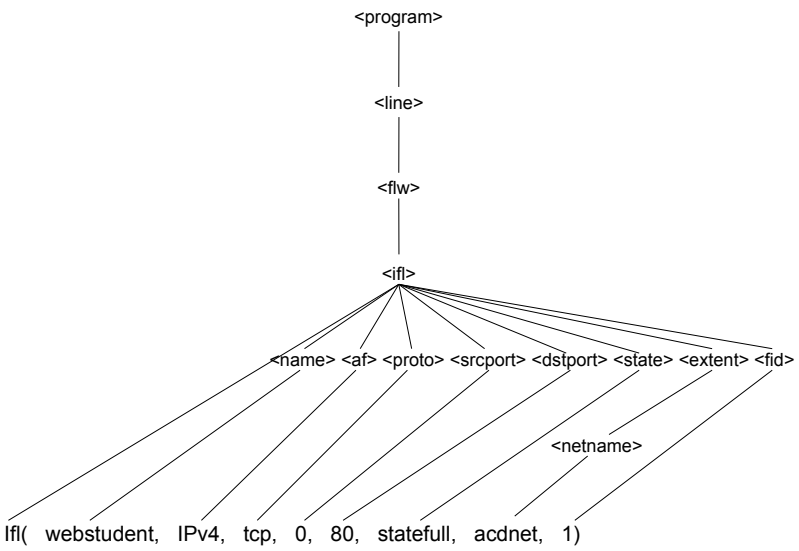


Figura 4.7: Árvore de derivação da instrução **ifl**

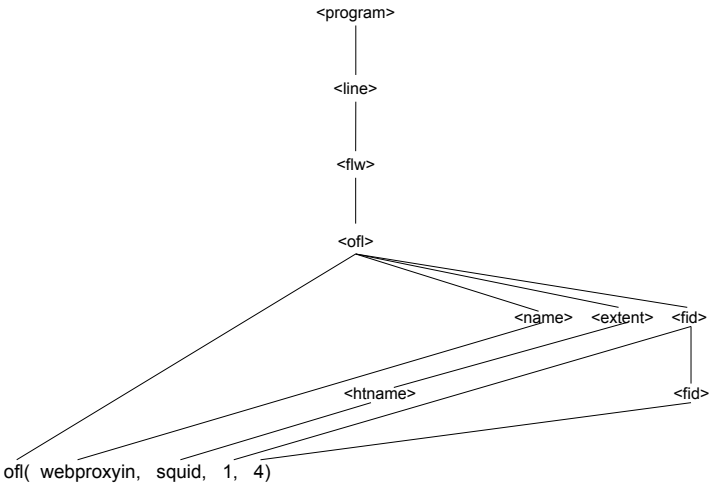


Figura 4.8: Árvore de derivação da instrução **ofl**

viamente definido por um *token* **dstnat**. A árvore de análise sintática da instrução da linha 24 da Definição 4.2 define um fluxo a ser traduzido o endereço IP, como pode ser visto na Figura 4.9. A árvore de análise sintática da instrução da linha 23 da Definição 4.2, que define um fluxo em que já foi traduzido o endereço IP, é apresentado na Figura 4.10.

A árvore de análise sintática da instrução da linha 26 da Definição 4.2, que define um fluxo a ser redirecionado, é apresentada na Figura 4.11.

O *token* **dstrdr** requer a porta de destino da entidade externa para onde o pacote

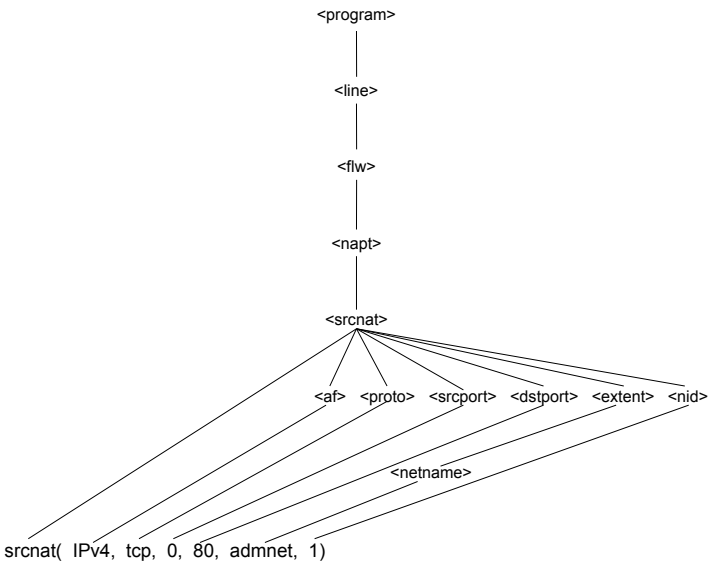


Figura 4.9: Árvore de derivação da instrução **srcnat**

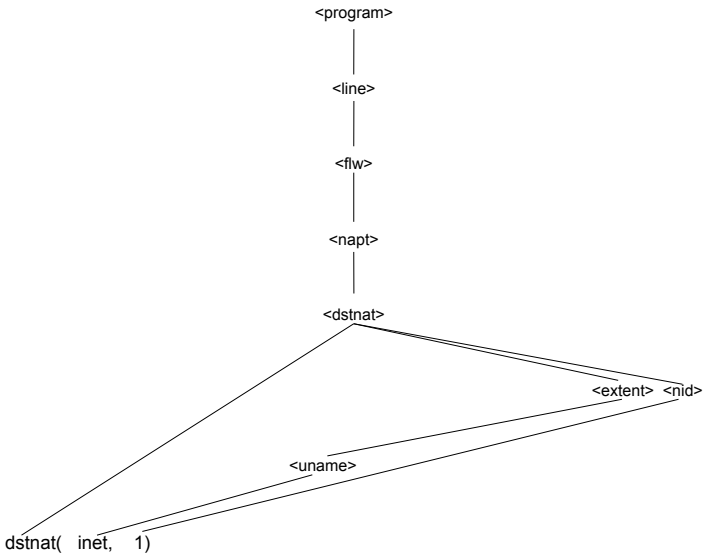


Figura 4.10: Árvore de derivação da instrução **dstnat**

é redirecionado. A árvore de análise sintática da instrução da linha 27 da Definição 4.2, que define um fluxo já redirecionado, é apresentada na Figura 4.12.

4.5 Semântica SPML

Similarmente a uma linguagem de programação, as sentenças da metalinguagem definida para a SPML requerem que suas restrições e condições sejam avaliadas e

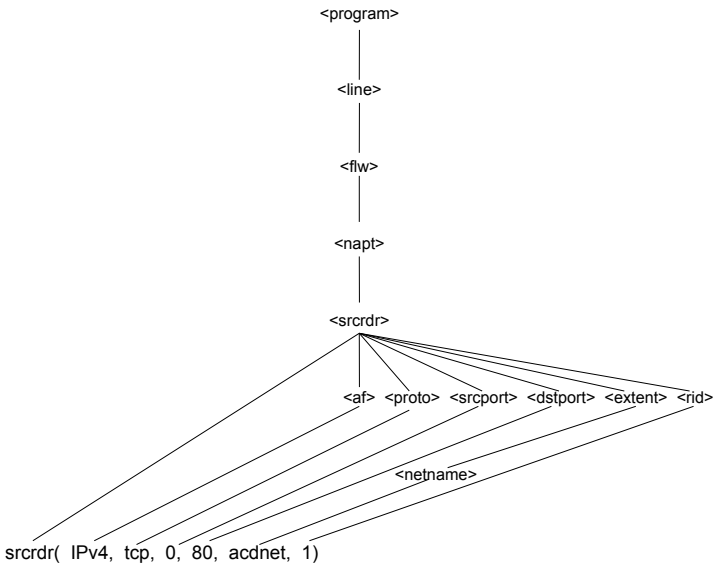


Figura 4.11: Árvore de derivação da instrução **srcrdr**

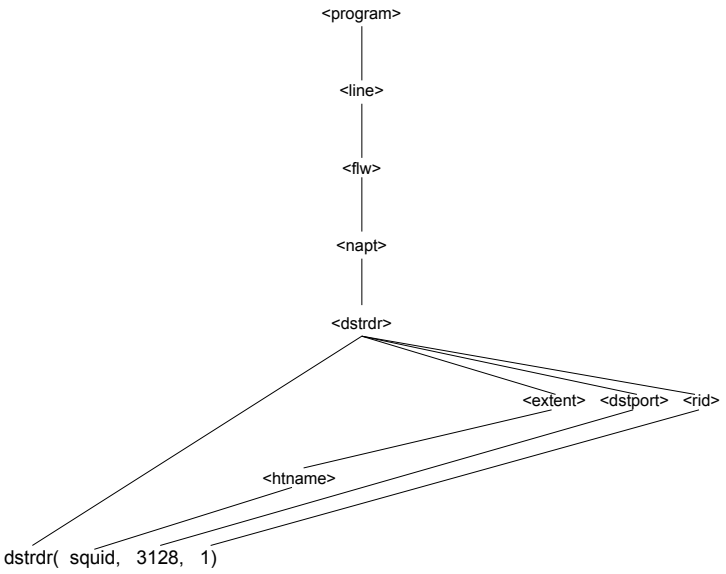


Figura 4.12: Árvore de derivação da instrução **dstrdr**

consideradas na tradução para um *firewall*.

Para expressar o formalismo semântico necessário à SPML é utilizada a notação **Z**, na qual a especificação de sistemas baseia-se em teoria dos conjuntos e cálculo de predicados. Inicialmente, para especificar um sistema usando uma notação **Z**, é necessário definir os tipos base para, depois, definir as operações sobre os tipos base, juntamente às respectivas restrições.

Considerando a especificação EBNF apresentada para análise sintática, uma definição formal em Z é proposta para a metalinguagem da SPML. Os tipos base[2] da SPML são apresentados na Notação 1.

$$[Firewall, Interface, ExtEntity]$$

Notação 1: Tipos Base em SPML **Z**.

Definidos os tipos base, é necessário definir os conjuntos, as relações e os predicados dos conjunto. Essas definições para a EBNF da SPML são apresentadas na Notação 2. Dentre os vários predicados, como exemplo, destaca-se $dom \quad ifs \subseteq if$, que indica que o domínio da relação ifs é subconjunto do conjunto das interfaces if. As demais não são comentadas por se tratar de interpretação da notação matemática.

[2] O tipo base $\mathbb{N}$ é nativo em linguagem **Z**

$$
\begin{array}{|l}
\hline
SPML \\
\hline
fw : \mathbb{F}\ Firewall \\
if : \mathbb{F}\ Interface \\
ifs : Interface \pfun Firewall \\
ifl : EntEntity \pfun Interface \\
ofl : INterface \pfun ExtEntity \\
dstnat : EntEntity \pfun Interface \\
srcnat : Interface \pfun ExtEntity \\
srcrdr : ExtEntity \pfun Interface \\
dstrdr : Interface \pfun ExtEntity \\
ht, htl, net, un : \mathbb{F}\ ExtEntity \\
htls : ht \pfun htl \\
fid : \mathbb{N} \\
nid : \mathbb{N} \\
rid : \mathbb{N} \\
\hline
\mathrm{dom}\ ifs \subseteq if \\
\mathrm{ran}\ ifs \subseteq fw \\
\mathrm{dom}\ ifl \subseteq ht \cup htl \cup net \cup un \\
\mathrm{ran}\ ifl \subseteq if \\
\mathrm{dom}\ ofl \subseteq if \\
\mathrm{ran}\ ofl \subseteq ht \cup htl \cup net \cup un \\
\mathrm{dom}\ dstnat \subseteq ht \cup htl \cup net \cup un \\
\mathrm{ran}\ dstnat \subseteq if \\
\mathrm{dom}\ srcnat \subseteq if \\
\mathrm{ran}\ srcnat \subseteq ht \cup htl \cup net \cup un \\
\mathrm{dom}\ srcrdr \subseteq ht \cup htl \cup net \cup un \\
\mathrm{ran}\ srcrdr \subseteq if \\
\mathrm{dom}\ dstrdr \subseteq if \\
\mathrm{ran}\ dstrdr \subseteq ht \cup htl \cup net \cup un \\
ht \cap htl \cap net \cap un = \emptyset \\
\mathrm{dom}\ htls \subseteq ht \\
\mathrm{ran}\ htls \subseteq htl \\
\hline
\end{array}
$$

Notação 2: Conjuntos, relações e predicados iniciais SPML em **Z**.

É preciso definir um valor inicial para todos os conjuntos e relações descritos na notação **Z**. A definição do estado inicial dos conjuntos e relações é apresentada na Notação 3.

Uma política de segurança expressa em SPML pode ter somente um *firewall*. Essa pré-condição é validada pelo predicado $fw = \emptyset$. A operação que adiciona um

Startup

$\Delta SPML$

$fw' = \emptyset$
$if' = \emptyset$
$ifs' = \emptyset$
$ifl' = \emptyset$
$ofl' = \emptyset$
$dstnat' = \emptyset$
$srcnat' = \emptyset$
$srcrdr' = \emptyset$
$dstrdr' = \emptyset$
$ht' = \emptyset$
$htl' = \emptyset$
$net' = \emptyset$
$un' = \emptyset$
$htls' = \emptyset$
$fid' = 0$
$nid' = 0$
$rid' = 0$

Notação 3: Inicialização da SPML.

firewall é apresentada na Notação 4.

$$
\begin{array}{|l}
\hline
\textit{AddFirewall} \\
\Delta SPML \\
\textit{firewall}? : \textit{Firewall} \\
\hline
\textit{fw} = \emptyset \\
\textit{firewall}? \notin \textit{fw} \\
\textit{fw}' = \textit{fw} \cup \{\textit{firewall}?\} \\
\hline
\end{array}
$$

Notação 4: Adiciona um *firewall*.

Uma interface não pode ser inserida mais de uma vez, essa restrição é garantida pelo predicado $interface? \notin if$. E uma interface deve ser adicionada a um *firewall* previamente criado, pré-condição assegurada pelo predicado $firewall? \in fw$. A operação que adiciona uma interface é apresentada na Notação 5.

$$
\begin{array}{|l}
\hline
\textit{AddInterface} \\
\Delta SPML \\
\textit{interface}? : \textit{Interface} \\
\textit{firewall}? : \textit{Firewall} \\
\hline
\textit{interface}? \notin \textit{if} \\
\textit{firewall}? \in \textit{fw} \\
\textit{if}' = \textit{if} \cup \{\textit{interface}?\} \\
\textit{ifs}' = \textit{ifs} \cup \{(\textit{interface}? \mapsto \textit{firewall}?)\} \\
\hline
\end{array}
$$

Notação 5: Adiciona uma interface.

A operação que adiciona um *host* é apresentada na Notação 6. Sua única pré condição é que o *host* não tenha sido criado anteriormente. Esse requisito é garantido pelo predicado $host? \notin ht$.

Para criar um grupo de equipamentos vinculados a uma mesma regra da política de segurança é necessário criar uma lista de *hosts* e, em seguida, adicionar *hosts* a essa lista. A operação que adiciona uma lista de hosts é apresentada na Notação 7.

O *host* pode ser adicionado uma única vez em cada lista, pré condição assegurada pelo predicado $host? \notin dom\ htls$. A operação que adiciona um *host* a uma lista de *hosts* é apresentada na Notação 8.

$$
\begin{array}{|l}
\hline
AddHost \\
\Delta SPML \\
host? : ExtEntity \\
\hline
host? \notin ht \\
ht' = ht \cup \{host?\} \\
\hline
\end{array}
$$

Notação 6: Adiciona um *host*.

$$
\begin{array}{|l}
\hline
AddHostList \\
\Delta SPML \\
hostlist? : ExtEntity \\
\hline
hostlist? \notin htl \\
htl' = htl \cup \{hostlist?\} \\
\hline
\end{array}
$$

Notação 7: Adiciona uma lista de *hosts*.

$$
\begin{array}{|l}
\hline
AddHostToHostList \\
\Delta SPML \\
host? : ExtEntity \\
hostlist? : ExtEntity \\
\hline
host? \in ht \\
host? \notin \mathrm{dom}\, htls \\
hostlist? \in htl \\
htls' = htls \cup \{(host? \mapsto hostlist?)\} \\
\hline
\end{array}
$$

Notação 8: Adiciona um *host* a uma lista de *hosts*.

Note que as entidades externas *host*, *hostlist*, *rede* e *rede com endereço IP desconhecido* devem ser únicas. Apesar dessa restrição não constar nas operações *AddHost*, *AddHostList*, *AddNetwork* e *AddUnkNet*, ela é assegurada pelo predicado $ht \cap htl \cap net \cap un = \emptyset$, presente na definição dos conjuntos, relações e predicados (Figura 2). A operação que adiciona uma rede é apresentada na Notação 9. E a operação que adiciona uma rede com endereço IP desconhecido é apresentada na Notação 10.

$$
\begin{array}{|l}
\hline
AddNetwork \\
\Delta SPML \\
network? : ExtEntity \\
\hline
network? \notin net \\
net' = net \cup \{network?\} \\
\hline
\end{array}
$$

Notação 9: Adiciona uma rede.

$$
\begin{array}{|l}
\hline
AddUnkNet \\
\Delta SPML \\
unknet? : ExtEntity \\
\hline
unknet? \notin un \\
un' = un \cup \{unknet?\} \\
\hline
\end{array}
$$

Notação 10: Adiciona uma rede com endereço desconhecido.

Um fluxo de entrada somente pode ser criado se a entidade externa que origina os pacotes existir. Essa pré condição é garantida pelo predicado $externalentity? \in ht \cup htl \cup net \cup un$. A operação que adiciona um fluxo de entrada é apresentada na Notação 11.

Um fluxo de saída somente pode ser criado depois do seu respectivo fluxo de entrada, conforme a restrição definida pelo predicado $flowid? > 0 \land flowid? \leq fid$. A operação que adiciona um fluxo de saída é apresentada na Notação 12.

A política de segurança pode realizar mais que uma tradução de endereço IP. A cada tradução criada, o identificador da tradução é incrementado. A operação que adiciona o destino de uma tradução é apresentada na Notação 13.

AddIncomingFlow

$\Delta SPML$
$externalentity? : ExtEntity$
$interface? : Interface$

$externalentity? \in ht \cup htl \cup net \cup un$
$interface? \in if$
$ifl' = ifl \cup \{(externalentity? \mapsto interface?)\}$
$fid' = fid + 1$

Notação 11: Adiciona um fluxo de entrada.

AddOutgoingFlow

$\Delta SPML$
$externalentity? : ExtEntity$
$interface? : Interface$
$flowid? : \mathbb{N}$

$externalentity? \in ht \cup htl \cup net \cup un$
$interface? \in if$
$flowid? > 0 \land flowid? \leq fid$
$ofl' = ofl \cup \{(interface? \mapsto externalentity?)\}$

Notação 12: Adiciona um fluxo de saída.

AddDestinationNAT

$\Delta SPML$
$externalentity? : ExtEntity$
$interface? : Interface$

$externalentity? \in ht \cup htl \cup net \cup un$
$interface? \in if$
$dstnat' = dstnat \cup \{(externalentity? \mapsto interface?)\}$
$nid' = nid + 1$

Notação 13: Adiciona o destino de uma tradução.

Cada origem de tradução criada deve ser adicionada ao conjunto $srcnat$. Assim a operação $srcnat' = srcnat \cup \{(externalentity? \mapsto interface?)\}$ garante a obediência da condição necessária. A operação que adiciona a origem de uma tradução é apresentada na Notação 14.

$$
\begin{array}{|l}
\hline
AddSourceNAT \\
\Delta SPML \\
externalentity? : ExtEntity \\
interface? : Interface \\
translationid? : \mathbb{N} \\
\hline
externalentity? \in ht \cup htl \cup net \cup un \\
interface? \in if \\
translationid? > 0 \land translationid? \leq nid \\
srcnat' = srcnat \cup \{(interface? \mapsto externalentity?)\} \\
\hline
\end{array}
$$

Notação 14: Adiciona a origem de uma tradução.

Segundo a linguagem **Z**, sempre que se utiliza a variante Δ para adicionar um esquema, todas as variáveis do esquema adicionado devem ser definidas (Notação 2). Entretanto, apenas as variáveis modificadas são apresentadas (mostrando a modificação sofrida) para não tornar a definição **Z** da SPML extensa. Como pode ser observado na Notação 15, $srcrdr$ e rid são redefinidas, enquanto outras variáveis do mesmo esquema não aparecem, mesmo fazendo parte da definição inserida, por não sofreram alteração. Similarmente, na Notação 16 é apresentada a operação que adiciona o destino de um redirecionamento, indicando apenas a alteração de $dstrdr$.

4.6 Considerações Finais

Neste capítulo são apresentados o formalismos sintático e semântico necessários a SPML para que ela seja considerada uma linguagem. Inicialmente são apresentados os lexemas para todos os *tokens* da SPML e, na sequência o formalismo sintático utilizando a EBNF. Finalmente é apresentado o formalismo semântico expresso em notação **Z**.

Durante a elaboração do formalismo sintático observou-se que a SPML define atributos que não são utilizados no mapeamento da política de segurança em regras de *firewall*, tais como: *scrub*, *domain* e *antispoof*. Considerando que o objetivo da

$$
\begin{array}{|l}
\hline
\textit{AddSourceRDR} \\
\Delta SPML \\
externalentity? : ExtEntity \\
interface? : Interface \\
\hline
externalentity? \in ht \cup htl \cup net \cup un \\
interface? \in if \\
srcrdr' = srcrdr \cup \{(externalentity? \mapsto interface?)\} \\
rid' = rid + 1 \\
\hline
\end{array}
$$

Notação 15: Adiciona a origem de um redirecionamento.

$$
\begin{array}{|l}
\hline
\textit{AddDestinationRDR} \\
\Delta SPML \\
externalentity? : ExtEntity \\
interface? : Interface \\
redirectionid? : \mathbb{N} \\
\hline
externalentity? \in ht \cup htl \cup net \cup un \\
interface? \in if \\
redirectionid? > 0 \land redirectionid? \leq rid \\
dstrdr' = dstrdr \cup \{(interface? \mapsto externalentity?)\} \\
\hline
\end{array}
$$

Notação 16: Adiciona o destino de um redirecionamento.

SPML é mapear a política de segurança, atributos não relacionados a este contexto não devem estar incorporados ao formalismo.

Os formalismos sintático e semântico apresentados definem as regras de escrita de modelos na linguagem SPML, bem como o comportamento que deve ser seguido para que os modelos estejam corretos.

A maior parte do formalismo aqui apresentado é utilizada para garantir as validações sintática e semântica da extensão da SPML (a SPML2) apresentada no Capítulo 5.

CAPÍTULO 5

Security Policy Modeling Language 2 - SPML2

Neste capítulo é apresentada a *Security Policy Modeling Language 2* (SPML2), uma evolução da SPML que elimina a ambiguidade de símbolos, possibilita uma representação mais clara da política de segurança utilizando apenas um diagrama, e permite representar o bloqueio de tráfego de pacotes, além de outras melhorias. Ao longo do texto, juntamente com a modelagem SPML2 também são apresentados recortes de instantâneos da interface da ferramenta desenvolvida (SP2Model). Desse modo paralelamente aos modelos SPML2 são apresentados como a ferramenta espera que o usuário especifique elementos da linguagem visual. Adicionalmente é apresentada uma breve descrição da ferramente e sua interface, apenas para mostrar uma visão geral da mesma sem tentar explanar todas as suas funcionalidades – isso é parte de um guia de utilização da ferramenta.

5.1 Cenário exemplo

Para ilustrar como a política de segurança é mapeada usando a SPML2, é utilizado um cenário baseado na instituição fictícia, como ilustrado na Figura 5.1. Nesse cenário, os computadores de diferentes departamentos podem acessar os serviços em servidores hospedados na rede DMZ na sede da empresa. Os computadores da sede, localizados na rede local, exigem tradução de endereços para acessar a Internet. A política de segurança da instituição está definida para permitir que todos

os computadores da filial acessem o servidor *Web* e o servidor *DNS*, com exceção dos computadores do departamento financeiro, que não devem acessar o servidor *Web*. O servidor *SSH* pode ser acessado apenas pelo computador *Adm1*. Os modelos SPML2 ao longo deste capítulo utilizam este cenário como estudo de caso para exemplificar o uso da SPML2.

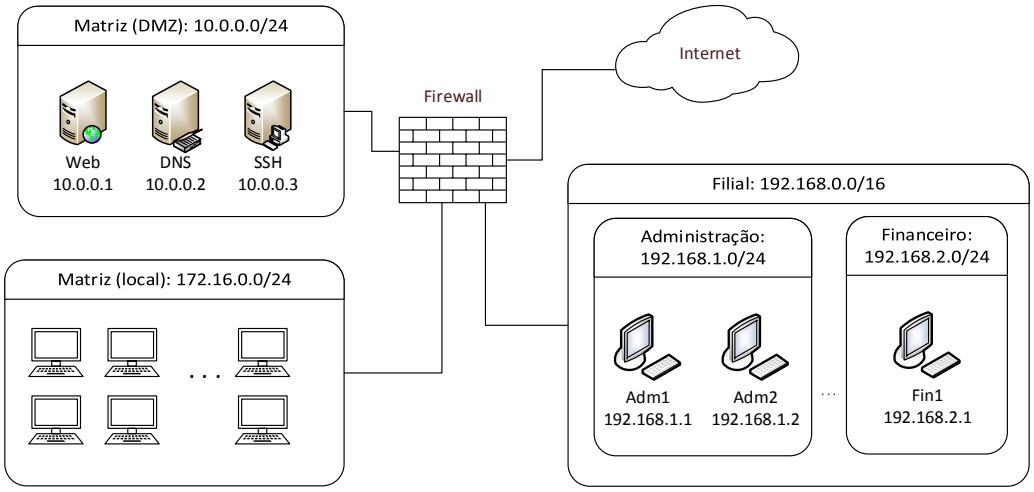


Figura 5.1: Organização de rede da instituição fictícia.

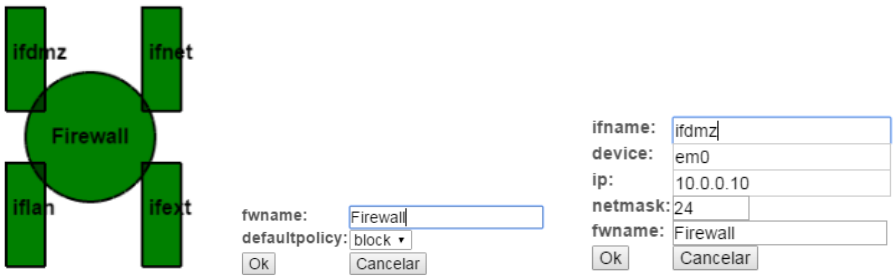
5.2 Notação

A SPML2 é definida por componentes capazes de representar a política de segurança de uma rede. Esses componentes são utilizados para representar o *firewall*, as interfaces e as entidades externas que geram/recebem tráfego de pacotes autorizados pela política de segurança. Caso o modelo tenha mais de um *firewall*, a interconexão deve ocorrer por meio de uma entidade externa que seja comum a ambos.

Cada componente é um recurso de *firewall* e possui uma representação gráfica, juntamente a um conjunto específico de atributos. Alguns desses atributos não são visíveis na notação gráfica. Os componentes SPML2 são organizados em quatro grupos, são eles: (i) componentes de *firewall*, (ii) componentes de tradução, (iii) componentes de filtragem, e (iv) componentes externos. Os componentes de tradução e filtragem determinam a ação que deve ser aplicada ao tráfego de pacotes. Os componentes externos ao *firewall* são entidades externas que denotam a origem ou o destino de um tráfego de pacotes. Entidades externas não podem se interconectar, desse modo, elas precisam, necessariamente, se conectar ao *firewall*. A representação gráfica dos componentes da SPML2 é similar aos componentes da SPML,

mas os atributos de vários componentes da SPML foram alterados na SPML2. Para apresentar os componentes da SPML2 e os seus atributos foi utilizada a ferramenta desenvolvida, a SP2Model[1].

1. O círculo é utilizado para representar *hardwares* e/ou *softwares* que executam funções de *firewall*, conforme Figura 5.2(a). O componente de *firewall* possui dois atributos: o *nome* e *política padrão*, conforme Figura 5.2(b). A política padrão indica as ações (*passar* ou *bloquear*[2]) usadas para os pacotes que não foram explicitamente definidos na política de segurança.



(a) (b) (c)

Figura 5.2: Componentes do *firewall* para a SPML2, (a) *firewall* com quatro interfaces, (b) atributos do *firewall* e (c) atributos da interface.

2. Um *firewall* pode ter uma ou mais interfaces de rede, permitindo assim ligações com componentes externos. Um componente externo não pode se conectar diretamente ao *firewall*, é exigido o uso de um componente de interface. A *interface* é representada graficamente por um retângulo (Figura 5.2(a)), e tem os seguintes atributos: *nome da interface*, *nome do dispositivo*, *endereço IP*, *máscara* e *nome do firewall*, como mostrado na Figura 5.2.(c).

3. Os componentes de filtragem de tráfego são representados por um segmento de reta com uma seta (Figuras 5.3(a), 5.3(b) e 5.3(c)). A cor da linha indica se o pacote deve passar ou ser bloqueado na interface, sendo que preto indica *tráfego de entrada*, azul indica *tráfego de saída* e vermelho indica *bloqueio do tráfego*. A seta ligada à interface indica um *tráfego de entrada* e quando ligada à entidade externa indica um *tráfego de saída*. Os atributos do *tráfego de entrada* são: *ID do tráfego*, *nome tráfego*, *interface associada*, *entidade*

[1] A ferramenta SP2Model é descrita na Seção 5.4

[2] Os componentes de filtragem de tráfego implementam as ações bloquear ou passar, e seguem explicados no item 3.

externa associada, *porta de origem*, *redirecionamento de porta* [3], *protocolo* e *versão do endereço IP*, como mostrado na Figura 5.3(d). O *ID do tráfego* deve ser único para cada tráfego de entrada. Ele é usado para conectar um *tráfego de entrada* em um ou mais *tráfegos de saída*. O *tráfego de saída* possui os seguinte atributos: *nome do tráfego*, *interface associada*, *entidade externa associada*, *porta destino*, *tradução de endereços de rede* [4] e um ou mais *ID de tráfego* (Figura 5.3(f)). O *ID do tráfego* deve ser definido por meio de um *tráfego de entrada*. O *bloqueio de pacotes* possui nove atributos, são eles: *ID do tráfego*, *nome do bloqueio*, *versão do endereço IP*, *interface associada*, *origem da entidade externa associada*, *porta de origem*, *destino externo da entidade associada*, *porta destino* e *protocolo*, como mostrado na Figura 5.3(e).

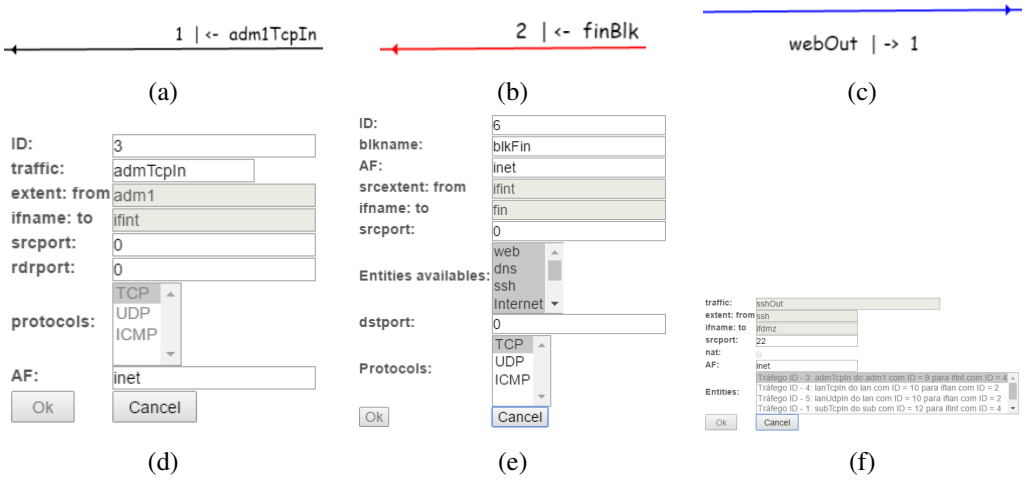


Figura 5.3: Componentes de filtragem de pacotes da SPML2, (a) liberar tráfego de entrada, (b) bloquear tráfego de entrada, (c) liberar saída de um tráfego de entrada, (d) atributos de um tráfego de entrada, (e) atributos de um bloqueio de tráfego e (f) atributos de um tráfego de saída.

4. O componente *tradução* é usado para representar o tráfego de pacotes que passam pela interface e exigem uma tradução (porta e/ou endereço). O componente *tradução* é denotado por um segmento de reta tracejado com uma seta (Figuras 5.4(a) e 5.4(b)). A cor preta indica um *redirecionamento de porta* e a cor azul indica uma *tradução de endereços de rede*. Um pacote de *tráfego de saída* será traduzido quando o atributo *tradução do endereço de rede* estiver marcado, conforme pode ser visualizado na Figura 5.4(c). Um

[3] Aplicável apenas em caso de redirecionamento de tráfego
[4] Aplicável apenas em caso de tradução tráfego de rede.

pacote de *tráfego de entrada* será redirecionado se o atributo *porta de redirecionamento* (*rdrport*) estiver presente (diferente de 0), como apresentado na Figura 5.4(d).

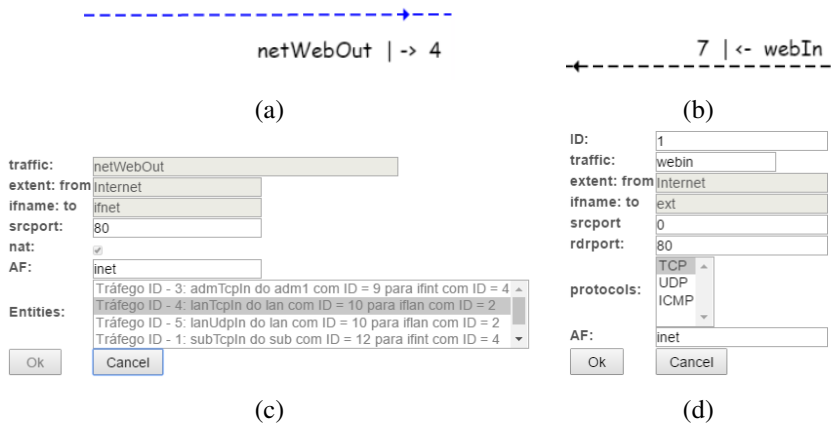


Figura 5.4: Componentes de tradução de pacotes da SPML2, (a) Tradução de tráfego, (b) Redirecionamento de tráfego, (c) Atributos de uma tradução de tráfego, (d) Atributos de um redirecionamento de tráfego.

5. O componente *rede desconhecida* (Figura 5.5(a)) é usado para definir a Internet ou qualquer rede que não tenha prefixo IP definido, seu atributo único é *nome* (Figura 5.5(c)).

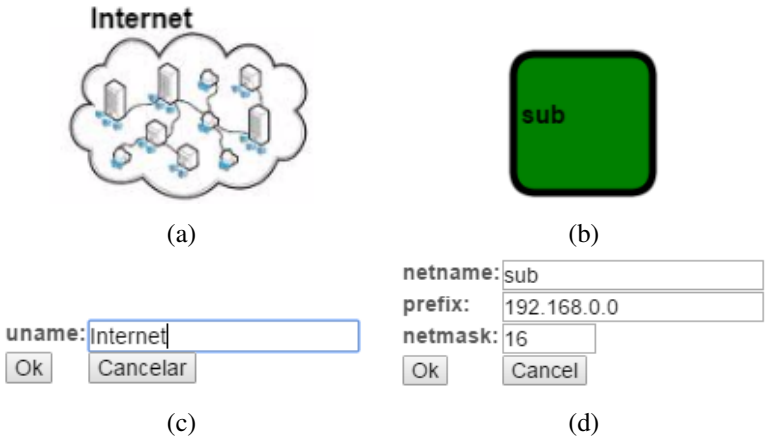


Figura 5.5: Entidades externas da SPML2, (a) rede com prefixo desconhecido, (b) entidade rede, (c) atributo de uma rede desconhecida e (b) atributos da entidade rede.

6. O componente *rede* representa uma rede IP e é denotado graficamente por um quadrado com cantos arredondados (Figura 5.5(b)). Os atributos da entidade *rede* são: *nome*, *prefixo de rede* e *máscara de sub-rede* (Figura 5.5(d)).

7. O componente *host* é utilizado para qualquer dispositivo capaz de receber um endereço IP. A notação gráfica do *host* é um quadrado (Figura 5.6(a)), e seus atributos são: *nome*, *endereço IP* e *máscara de sub-rede* (Figura 5.6(c)).

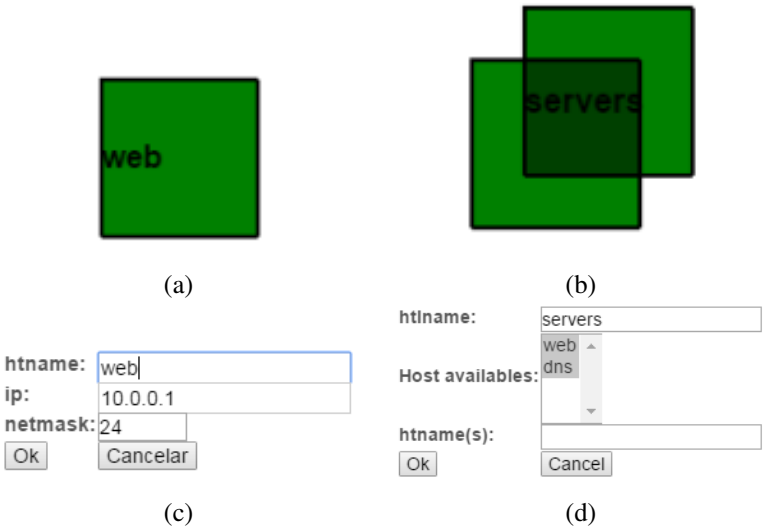


Figura 5.6: Entidades externas da SPML2, (a) entidade *host*, (b) entidade lista de *hosts*, (c) atributos da entidade *host* e (b) atributos da entidade lista de *hosts*.

8. O componente *lista de hosts* é um conjunto de *hosts* que pertence ao mesmo tráfego de política de segurança. A *lista de hosts* é representada graficamente por dois quadrados sobrepostos (Figura 5.6(b)) e seus atributos são: *nome* e dois ou mais *hosts* previamente definidos no componente *host* (Figura 5.6(d)).

5.3 Formalismo SPML2

Para definir o formalismo sintático e semântico da SPML2 é tomado como base formalismo da SPML definido no Capítulo 4. Foram alteradas as definições sintáticas e foram eliminadas restrições semânticas.

5.3.1 Formalismo Sintático

As mudanças mais significativas no formalismo sintático da SPML2 em relação ao formalismo sintático da SPML estão relacionadas com:

- A eliminação de atributos que não estão relacionados diretamente com a política de segurança, tais como domínio, *gateway* e DNS, pode ser observado na Figura 5.7, sendo que na Figura 5.7(a) é apresentada a árvore de derivação

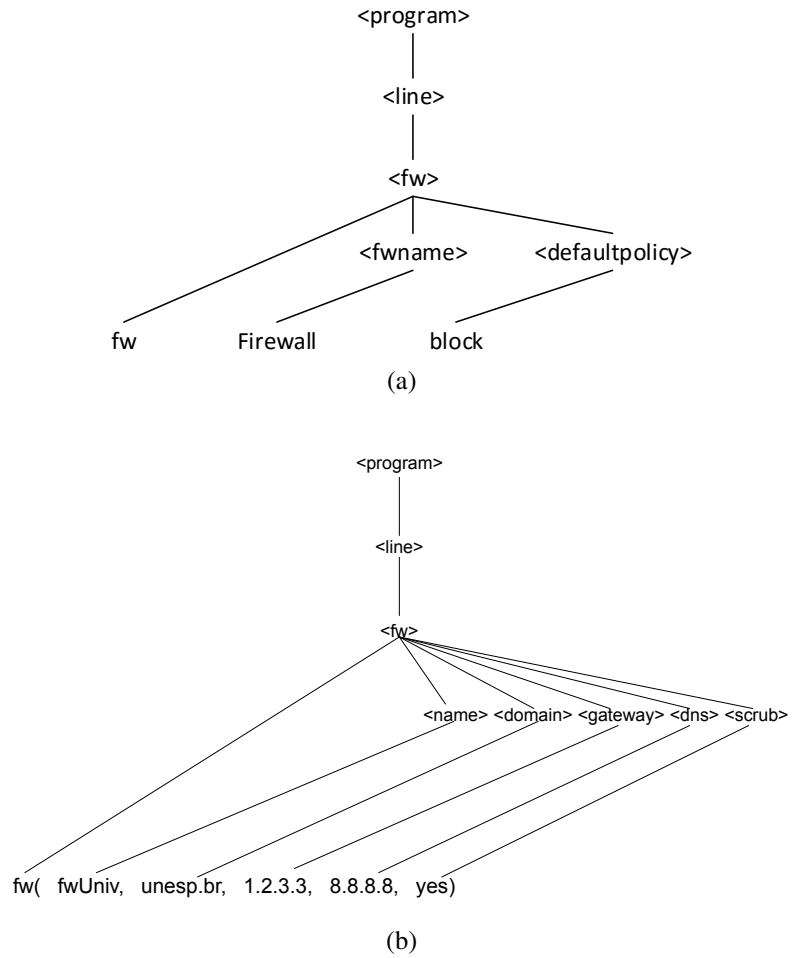


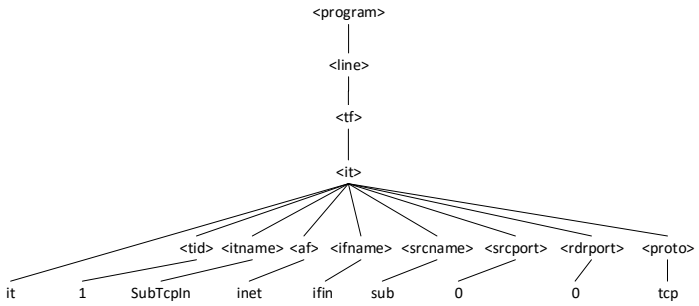
Figura 5.7: Árvore de derivação da instrução fw: (a) na SPML2, (b) na SPML.

da instrução *fw* para a SPML2 e na Figura 5.7(b) é apresentada a árvore de derivação para a mesma instrução (*fw*) para a SPML.

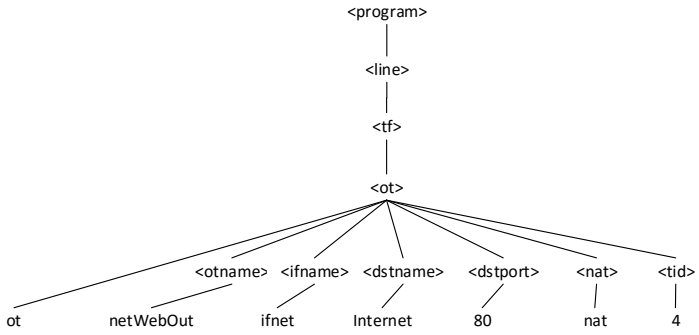
- A eliminação das instruções para tradução e redirecionamento de tráfegos, que agora são atributos da liberação de tráfego (entrada/saída) como é apresentado na Figura 5.8. Na Figura 5.8(a) pode ser observada a árvore de derivação da instrução *it*, sendo que o atributo responsável por indicar o redirecionamento de tráfego é o atributo $< rdrport >$. Na Figura 5.8(b) pode ser observada a árvore de derivação da instrução *ot*, sendo que o atributo responsável por indicar tradução do endereço IP é o atributo $< nat >$.

- Criação de uma instrução para bloqueio de tráfego. A árvore de derivação da nova instrução blk pode ser observada na Figura 5.9.

A gramática da SPML2 expressa em EBNF é apresentada na Definição 5.1.



(a)



(b)

Figura 5.8: Árvore de derivação das instruções: (a) it liberação de tráfego de entrada (atributo $rdrport$) e (b) ot liberação de tráfego de saída (atributo nat).

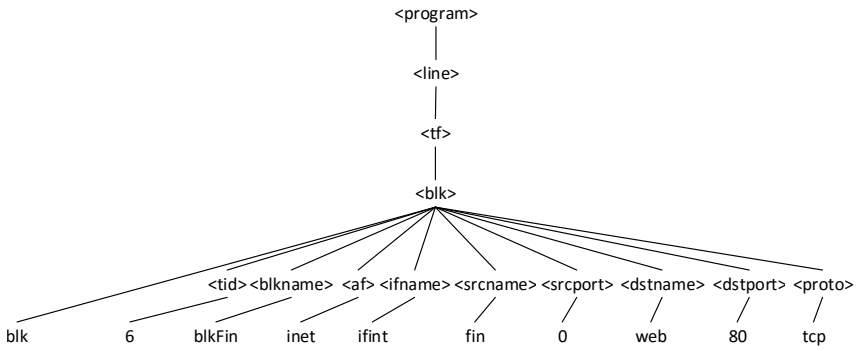


Figura 5.9: Árvore de derivação para a instrução blk da SPML2.

A política de segurança descrita na Seção 5.1 e apresentada utilizando modelo SPML2 na Figura 5.10, foi escrita em metalinguagem de acordo com a gramática EBNF apresentada neste trabalho (veja a Definição 5.1. Pode-se observar que há uma sequência específica no mapeamento da metalinguagem SPML2, por exemplo, deve-se definir o *firewall* inicialmente para, então, definir as interfaces associadas a ele. A metalinguagem do modelo SPML2 está listada na Definição 5.2.

Definição 5.1: Gramática da SPML2 expressa em EBNF.

```
<program> :: = <line>{<line>}EOF
<line> ::= <fw> | <if> | <extent> | <tf>
<fw> ::= fw(<fwname>,<defaultpolicy>)
<defaultpolicy> ::= block | pass
<if> ::= if(<ifname>,<device>,<ip>,<netmask>,<fwname>)
<extent> ::= <ht> | <htl> | <net> | <un>
<ht> ::= ht(<htname>,<ip>,<netmask>)
<htl> ::= htl(<htlname>,<htname>{,<htname>})
<nat> ::= nat |
<net> ::= net(<netname>,<prefix>,<netmask>)
<un> ::= un(<uname>)
<tf> ::= <it> | <ot> | <blk>
<it> ::= it(<tid>,<itname>,<af>,<ifname>,<srcname>,
     <srcport>,<rdrport>,<proto>{,<proto>})
<ot> ::= ot(<otname>,<ifname>,<dstname>,<dstport>,
     <nat>,<tid>{,<tid>})
<blk> ::= blk(<tid>,<blkname>,<af>,<ifname>,<srcextent>,
      <srcport>,<dstextent>,<dstport>,<proto>{,<proto>})
<af> ::= inet | inet6
<proto> ::= tcp | udp | icmp
<srcport> ::= 0 .. 65535
<dstport> ::= 0 .. 65535
<rdrport> ::= 0 .. 65535
```

Definição 5.2: Metalinguagem do modelo SPML2 apresentado na Figura 5.10.

```
fw(Firewall,block)
if(ifdmz,em0,10.0.0.10,24,Firewall)
if(iflan,em2,172.16.0.10,24,Firewall)
if(ifnet,em3,10.10.10.10,30,Firewall)
if(ifint,em1,192.168.0.10,24,Firewall)
ht(web,10.0.0.1,24)
ht(dns,10.0.0.2,24)
ht(ssh,10.0.0.3,24)
un(Internet)
ht(adm1,192.168.1.1,24)
net(lan,172.16.0.0,24)
net(fin,192.168.2.0,24)
net(sub,192.168.0.0,16)
it(1,subTcpIn,inet,ifint,sub,0,0,tcp)
it(2,subUdpIn,inet,ifint,sub,0,0,udp)
it(3,admTcpIn,inet,ifint,adm1,0,0,tcp)
it(4,lanTcpIn,inet,iflan,lan,0,0,tcp)
it(5,lanUdpIn,inet,iflan,lan,0,0,udp)
ot(webOut,ifdmz,web,80,,1)
ot(dnsOut,ifdmz,dns,53,,2)
ot(sshOut,ifdmz,ssh,22,,3)
ot(netWebOut,ifnet,Internet,80,nat,4)
ot(netDnsOut,ifnet,Internet,53,nat,5)
blk(6,blkFin,inet,ifint,fin,0,web,80,tcp)
```

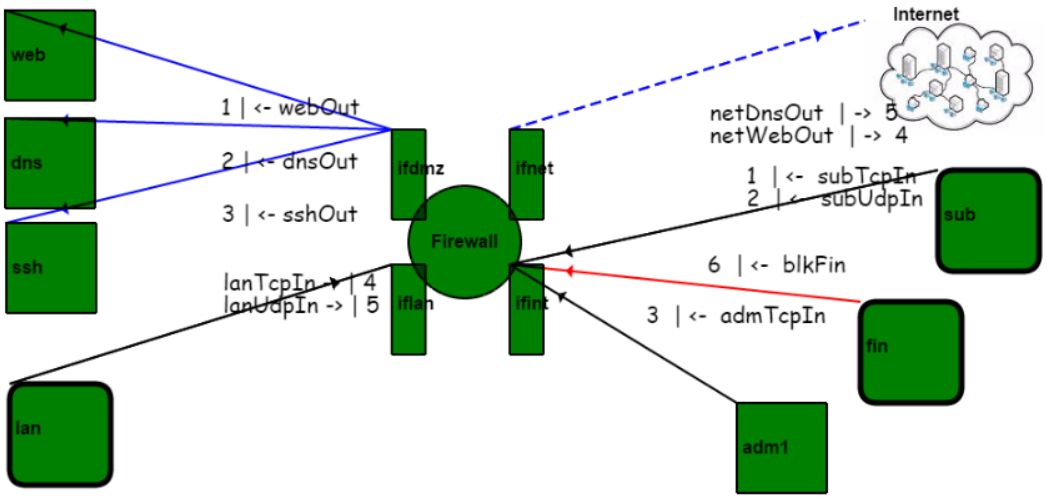


Figura 5.10: Política de segurança modelada em SPML2.

5.3.2 Formalismo Semântico

O formalismo semântico da SPML2, em relação ao formalismo semântico da SPML, foi alterado retirando alguns conjuntos e relações que não são mais utilizados. Convém destacar que os tipos base continuam os mesmos. Os conjuntos, as relações e os predicados dos conjuntos da SPML2 são apresentados na Notação 17.

A definição de um valor inicial para todos os conjuntos e relações descritos na notação **Z**, necessária ao formalismo semântico da SPML2, é apresentada na Notação 18.

5.4 The SP2Model – uma ferramenta *web*

O principal propósito da SPML2 é permitir a representação de uma política de segurança por meio de um modelo gráfico e gerar automaticamente o conjunto de regras em linguagem nativa de *firewall*. A SP2Model é uma ferramenta *web* que permite criar modelos visuais de política de segurança de acordo com SPML2, incluindo todos os componentes descritos na SPML2. A SP2Model foi desenvolvida em HTML5 e JavaScript.

A ferramenta SP2Model foi elaborada obedecendo os formalismos sintático e semântico da SPML2. Na Figura 5.11 pode ser observada a tela principal da ferramenta, na parte superior é possível escolher os comandos, tais como, inserir *firewall* ou entidades externas, inserir tráfegos, entre outros. Na barra lateral (a esquerda) é possível selecionar o tipo de entidade ou de tráfego que deseja inserir.

Na Figura 5.12 é apresentado um instantâneo da ferramenta SP2Model com o

SPML2

$$
\begin{array}{l}
fw : \mathbb{F}\ Firewall \\
if : \mathbb{F}\ Interface \\
ifs : Interface \pfun Firewall \\
blk : ExtEntity \pfun Interface \\
it : ExtEntity \pfun Interface \\
ot : Interface \pfun ExtEntity \\
ht, htl, net, un : \mathbb{F}\ ExtEntity \\
htls : ht \pfun htl \\
tid : \mathbb{N} \\
\hline
\mathrm{dom}\ ifs \subseteq if \\
\mathrm{ran}\ ifs \subseteq fw \\
\mathrm{dom}\ blk \subseteq ht \cup htl \cup net \cup un \\
\mathrm{ran}\ blk \subseteq if \\
\mathrm{dom}\ it \subseteq ht \cup htl \cup net \cup un \\
\mathrm{ran}\ it \subseteq if \\
\mathrm{dom}\ ot \subseteq if \\
\mathrm{ran}\ ot \subseteq ht \cup htl \cup net \cup un \\
ht \cap htl \cap net \cap un = \emptyset \\
\mathrm{dom}\ htls \subseteq ht \\
\mathrm{ran}\ htls \subseteq htl
\end{array}
$$

Notação 17: Notação **Z** para a SPML2: conjuntos, relações e predicados.

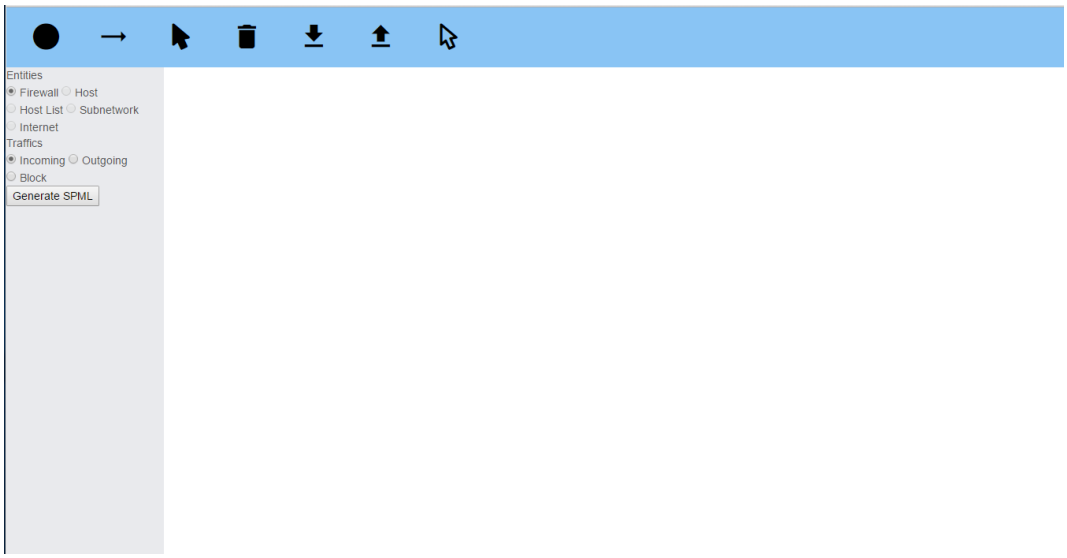


Figura 5.11: Interface da ferramenta Web SP2Model.

$$
\begin{array}{|l}
\hline
Startup \\
\Delta SPML2 \\
\hline
fw' = \emptyset \\
if' = \emptyset \\
blk' = \emptyset \\
it' = \emptyset \\
ot' = \emptyset \\
ht' = \emptyset \\
htl' = \emptyset \\
net' = \emptyset \\
un' = \emptyset \\
htls' = \emptyset \\
tid' = 0 \\
\hline
\end{array}
$$

Notação 18: Notação **Z** para inicialização da SPML2.

firewall e algumas entidades externas configuradas.

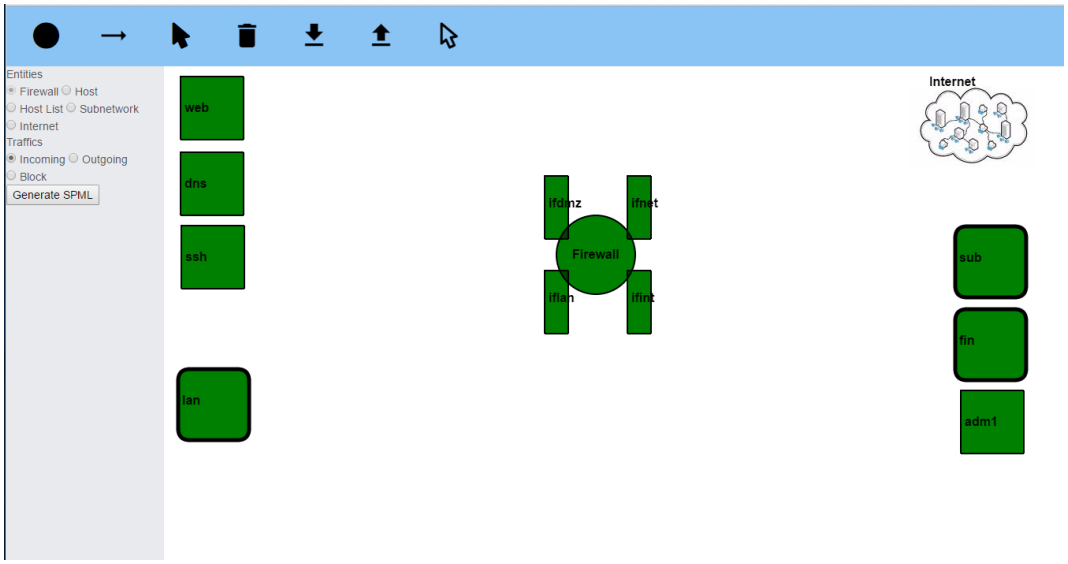


Figura 5.12: Ferramenta Web SP2Model com *firewall*, interfaces e algumas entidades externas modeladas.

Na Figura 5.13 é apresentado um instantâneo da ferramenta SP2Model com uma política de segurança descrita na Seção 5.1 modelada.

A partir da representação gráfica apresentada na Figura 5.10, foi gerado automaticamente o conjunto de regras em linguagem nativa de *firewall*, conforme mostrado

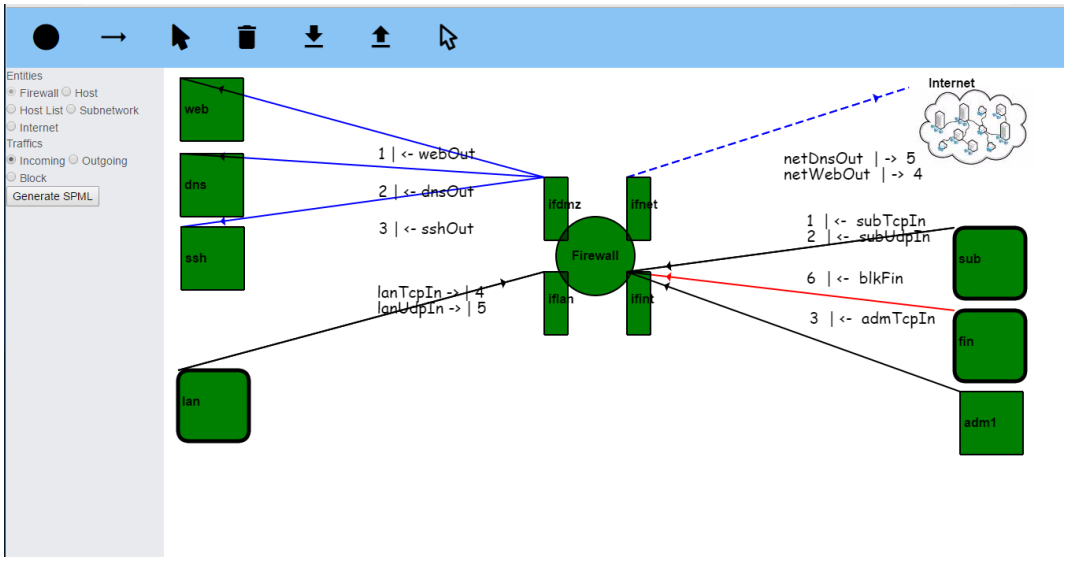


Figura 5.13: Política de segurança sendo modelada na ferramenta Web SP2Model.

na Definição 5.3.

5.5 Considerações Finais

Tendo como base os conceitos de funcionamento de *firewall*, a SPML e o formalismo da SPML desenvolvido neste trabalho, foi desenvolvida a SPML2. A SPML2 resolve problemas da SPML, descritos na Seção 3.5, atributos desnecessários foram eliminados, o bloqueio de tráfego foi criado, o redirecionamento e a tradução de tráfego foram incorporados à liberação de tráfego de entrada e saída, respectivamente. A sintaxe e a semântica da SPML2 foram formalizadas a partir daquela desenvolvida para a SPML.

A SPML2 é avaliada e os resultados e a discussão sobre essa avaliação estão documentados no Capítulo 6.

Definição 5.3: Regras de *firewall* em linguagem nativa – sintaxe do Packet Filter

```
1  #Defaultpolicy
2  block in all
3  #Traffic ID 0
4  pass in on em1 inet proto tcp \
5       from 192.168.0.0/16 to 10.0.0.1 port 80
6  pass out on em0 inet proto tcp \
7       from 192.168.0.0/16 to 10.0.0.1 port 80
8  #Traffic ID 1
9  pass in on em1 inet proto udp \
10      from 192.168.0.0/16 to  10.0.0.2 port 53
```

```
11 pass out on em0 inet proto udp \
12      from 192.168.0.0/16 to 10.0.0.2 port 53
13 #Traffic ID 2
14 pass in on em1 inet proto tcp \
15      from 192.168.1.1 to 10.0.0.3 port 22
16 pass out on em0 inet proto tcp \
17      from 192.168.1.1 to 10.0.0.3 port 22
18 #Traffic ID 3
19 pass in on em2 inet proto tcp \
20      from 172.16.0.0/24 to any port 80
21 pass out on em3 inet proto tcp \
22      from 172.16.0.0/24 to any port 80 nat-to (em3)
23 #Traffic ID 4
24 pass in on em2 inet proto udp \
25      from 172.16.0.0/24 to any port 53
26 pass out on em3 inet proto udp \
27      from 172.16.0.0/24 to any port 53 nat-to (em3)
28 #Traffic ID 5
29 block in on em1 proto tcp \
30      from 192.168.2.0/24 to 10.0.0.1 port 80
```

CAPÍTULO 6

Avaliação da SPML2: o Experimento Controlado

6.1 Considerações Iniciais

Para verificar se a utilização da SPML2 – *Security Policy Modeling Language* – provê ao administrador de redes o correto entendimento da política de segurança e, posteriormente, se a representação gráfica gera as regras em linguagem nativa de *firewall* necessárias, foi realizado um experimento controlado.

O experimento visa analisar o procedimento de mapeamento da política de segurança (textual) para regras em linguagem nativa de *firewall*, utilizando o diagrama SPML2 e a abordagem tradicional, com o propósito de avaliação no que diz respeito à eficiência e à eficácia do ponto de vista do pesquisador no contexto de alunos da graduação avaliando a corretude das regras nativas de *firewall*.

Com o intuito de avaliar o desempenho da SPML2, o experimento foi realizado por um grupo de 15 estudantes de Ciência da Computação. O experimento teve como objetivo principal comparar o mapeamento de políticas de segurança (textual) em língua nativa de *firewall* usando a abordagem visual (SPML2) e a tradicional (Ad Hoc). Como métrica de desempenho utilizou-se a eficácia (acurácia) e a eficiência (tempo despendido).

O experimento foi conduzido utilizando o paradigma *Goal/Question/Metric* (GQM) (Basili et al., 1994) e o modelo proposto por Wohlin et al. (2012).

Para avaliação e comparação da eficiência e eficácia, do processo de elaboração do conjunto de tráfego previsto pela política de segurança, em relação ao conjunto de tráfego controlado pelo *firewall*, foram desenvolvidas as seguintes atividades:

- Configuração tradicional (Ad Hoc): a partir da política de acesso à rede (documento textual), os participantes escrevem o conjunto de regras de *firewall* diretamente no idioma nativo.
- Configuração visual (SPML2): a partir da política de acesso à rede e com o apoio da ferramenta SP2Model, os participantes prepararam o diagrama SPML2 correspondente e, posteriormente, utilizar o SP2Model para exportar as regras na língua nativa de *firewall*.

Os experimentos foram realizados no laboratório de rede da Universidade Estadual Paulista (UNESP), Campus de Presidente Prudente.

6.2 Objetivos e Métricas do Experimento

O principal objetivo do experimento é comparar a abordagem visual (SPML2) à abordagem tradicional (Ad Hoc) durante o processo de criação da configuração e de manutenção de configuração de *firewalls* com foco na eficácia e eficiência.

Assim, a hipótese nula foi formulada para avaliar a significância da abordagem adotada (Ad Hoc ou SPML2) quanto à eficácia (Efe: a tradução da política de acesso à rede para um conjunto de regras em linguagem nativa de *firewall*) e eficiência (Efi: tempo para a conversão da política de acesso à rede para um conjunto de regras em linguagem nativa de *firewall*). Na Tabela 6.1 é apresentada a formalização das hipóteses feitas.

Tabela 6.1: Hipóteses testadas nos experimentos.

Efeito	**Hipóteses**	**Hipóteses formulada**
Eficácia (Efe)	Hipótese nula	$H_0^1 : Efe_{SPML2} \leq Efe_{AdHoc}$
	Hipótese alternativa	$H_a^1 : Efe_{SPML2} > Efe_{AdHoc}$
Eficiência (Efi)	Hipótese nula	$H_0^2 : Efi_{SPML2} \leq Efi_{AdHoc}$
	Hipótese alternativa	$H_a^2 : Efi_{SPML2} > Efi_{AdHoc}$

A variável independente é o tráfego definido pela política de acesso à rede. A política de acesso à rede define o conjunto de tráfego autorizado ($t_A P$) e o conjunto de

tráfego não autorizado ($t_{NA}P$). Um tráfego autorizado a passar pelo *firewall* (t_AP) requer duas ações, entrar por uma interface do *firewall* e sair por outra interface do *firewall*. Um tráfego não autorizado a passar pelo *firewall* ($t_{NA}P$) requer uma ação, ser bloqueado na interface associada. A união do conjunto de tráfego autorizado e do conjunto de tráfego não autorizado definidos pela política de segurança fornece o tráfego da política de acesso à rede, $tP = t_AP \cup t_{NA}P$.

As variáveis dependentes são as regras corretas que controlam o tráfego do *firewall* (Efe) e o tempo (Efi) para mapeá-las. As regras de configuração definem o conjunto de tráfego que deve ser liberado (t_LF) e o conjunto de tráfego que deve ser bloqueado (t_BF). Um tráfego liberado pelo *firewall* (t_LF) deve atravessá-lo, assim, são necessárias duas regras, a primeira permitindo a entrada do pacote por uma interface e a segunda liberando a saída do pacote por outra interface. Um tráfego bloqueado pelo *firewall* (t_BF) é expresso por uma regra de bloqueio. A união de ambos os conjuntos é denotada por $tf = t_LF \cup t_BF$.

O tempo gasto, medido em minutos, representa o tempo necessário para mapear as regras de política de acesso à rede em linguagem nativa de *firewall* usando as abordagens tradicional (Ad Hoc) e visual (SPML2).

A pontuação da eficácia é medida pelo conjunto de tráfego fornecido pela política de acesso à rede, que também está mapeado no conjunto de tráfego controlado pelo *firewall*, $Efe = tP \cap tF$, e a eficiência é medida pelo tempo gasto em minutos, TS, para a preparação de regras de *firewall*, dividido pelo número de regras corretas produzidas, $Efi = TS/Efe$ (tempo médio por regra correta produzida). Cada participante deve produzir o conjunto de regras em linguagem nativa de *firewall* que implementa o que determina a política de segurança em um menor tempo para a avaliação de ambas as abordagens utilizadas.

6.3 Artefatos

Os artefatos utilizados na condução do experimento foram definidos de acordo com os requisitos necessários para a execução das sessões do experimento e com os requisitos de conhecimento prévio para a sua realização. Os artefatos utilizados foram os seguintes: (i) Políticas de Segurança (descritas na Subseção 6.3.1), (ii) Softwares (relacionados na Tabela 6.2), (iii) Formulários (mencionados na Subseção 6.3.2), (iv) Materiais de treinamento (listados na Tabela 6.5) e Planilhas de registro e controle.

Com a realização do experimento, regras em linguagem nativa de *firewall* (entregues em formato de arquivo texto) e modelos visuais da política de segurança

Tabela 6.2: Softwares utilizados para a realização do experimento.

Software	**Utilidade**
Editor de texto	Escrever as regras em linguagem nativa de *firewall*
Google Chrome	Executar a ferramenta SP2Model
SP2Model	Modelar a política de segurança em conformidade com a SPML2

(entregues em formato de arquivo capaz de ser gerado/carregado pela ferramenta SP2Model) são produzidas.

Os artefatos foram planejados e preparados considerando que a coleta de dados deve ser validada para que os resultados do experimento não sejam afetados pelos artefatos, ou seja, os resultados devem ser os mesmos, independentemente de como os artefatos foram utilizados no experimento. Caso contrário, os resultados seriam inválidos (Wohlin et al., 2012).

6.3.1 Políticas de Segurança

No experimento foram utilizadas duas políticas de segurança com diferentes regras de acesso a rede (denominadas Política A e Política B). As políticas foram elaboradas considerando as inconsistências mais comuns entre o tráfego autorizado e o tráfego não autorizado, de acordo com a política de acesso à rede e o tráfego de pacotes liberados/bloqueados implementados no *firewall* (Yuan et al., 2006). O número de regras na linguagem nativa de *firewall* é apresentado na Tabela 6.3.

Tabela 6.3: Número de regras na linguagem nativa de *firewall*.

Política	**Número de regras**
A	31
B	47

6.3.2 Formulários

Foram desenvolvidos formulários para colaborar com a documentação do experimento, permitindo coletar informações acerca dos participantes, bem como manter registro do consentimento necessário. A relação dos formulários desenvolvidos é apresentada na Tabela 6.4.

6.3.3 Treinamento

Com o objetivo de garantir que todos os participantes possuam o conhecimento básico necessário para a execução das atividades propostas, foram preparados qua-

Tabela 6.4: Formulários utilizados para a documentação do experimento.

Documento	Objetivo
1	Termo de consentimento livre e esclarecido
2	Planejamento do experimento (para acompanhamento do condutor)
3	Perfil dos participantes
4	Ficha de acompanhamento

tro treinamentos conforme apresentados na Tabela 6.5:

Tabela 6.5: Treinamentos ministrados para a realização do experimento.

Treinamento	Objetivo
1	Noções de segurança em redes, incluindo o entendimento de uma política de segurança e sua relação com os equipamentos de rede
2	Sintaxe da linguagem de configuração de *firewall – Packet Filter*
3	Modelagem da política de segurança em SPML2 usando a SP2Model
4	Apresentação do experimento

6.4 Experimento Piloto

Para validar o experimento é necessário avaliar os artefatos e o tratamento dado aos artefatos e, para isso, foi realizado e analisado um experimento piloto. Convém salientar que esse experimento piloto não teve o objetivo de obter resultados estatísticos, mas melhorar o projeto do experimento e os artefatos utilizados para o experimento final.

O experimento piloto foi conduzido conforme o planejamento do experimento (Formulário 2 da Tabela 6.4) e foram utilizados todos os demais artefatos mencionados na Seção 6.3.

Desse modo, foram selecionados participantes, que receberam todos os treinamento no primeiro dia e no segundo dia realizaram o experimento. A sequência das atividades e o tempo inicialmente programado para cada atividade pode ser observado na Tabela 6.6.

Durante a realização do experimento notou-se que várias atividades consumiram mais tempo do que o inicialmente planejado, além de que, durante o experimento, ainda restavam muitas dúvidas acerca da sintaxe das regras de *firewall* em linguagem nativa (*Packet Filter*) e do uso da ferramenta SP2Model (que implementa a SPML2).

A experiência obtida na execução do experimento piloto permitiu adequações

Tabela 6.6: Planejamento do experimento piloto

Dia	Tempo	Tarefa		
1	5 min	Apresentação do experimento		
	5 min	Assinatura do termo de consentimento livre e esclarecido		
	10 min	Preenchimento do questionário		
	30 min	Treinamento: Configuração do *firewall*, *Packet Filter* e SPML2		
2	10 min	Organização de participantes		
			Ad Hoc	SPML2
	30 min	Política A (configuração)	Grupo 1	Grupo 2
	10 min	Política A (manutenção)	Grupo 1	Grupo 2
	30 min	Política B (configuração)	Grupo 2	Grupo 1
	10 min	Política B (manutenção)	Grupo 2	Grupo 1

no projeto do experimento final, alterando o planejamento (duração das atividades), bem como os artefatos (melhorando seu conteúdo), tornando claros os pontos que geraram dúvidas ou que a informação era insuficiente para o cumprimento do objetivo de cada atividade.

Vale destacar que os participantes selecionados para a realização do experimento piloto não participaram do experimento final.

6.5 O Experimento Final

Considerando a experiência adquirida na execução do experimento piloto, o cronograma foi ajustado, conforme descrito na Tabela 6.10, bem como os materiais de treinamento também foram ajustados. Considerando as mudanças o experimento final foi conduzido.

Na medida em que os participantes chegaram, foram atribuídos grupos a cada um deles (1 ou 2), seguindo a ordem: o primeiro participante foi alocado ao grupo 1, o segundo participante ao grupo 2, o terceiro participante ao grupo 1 e assim por diante conforme descrito na Subseção 6.5.1. Desse modo a alocação aleatória se deu pela ordem de chegada dos participantes.

6.5.1 Organização do Experimento

O experimento foi conduzido com a participação de 15 alunos de graduação em Ciência da Computação do nono semestre. Todos os alunos concluíram as disciplinas de Redes de Computadores I e Redes de Computadores II. Vale ressaltar que a participação foi voluntária e o perfil que caracteriza a maioria dos participantes é:

- Nunca terem configurado um *firewall* (11 participantes).

- Nunca terem realizado o mapeamento de regras de *firewall* a partir de uma política de acesso à rede (13 participantes).
- Nunca terem estudado técnicas de modelagem de mapeamento de política de acesso à rede (12 participantes).

Considerando a ordem de chegada, os participantes foram divididos em dois grupos (Grupo 1 e Grupo 2), como mostrado na Tabela 6.7.

Tabela 6.7: Número de participantes por grupos.

Grupo	Número de participantes
1	8
2	7

6.5.2 Cronograma de Atividades do Experimento

Para avaliar as tarefas de configuração e manutenção foram realizadas quatro sessões de experimentos. Durante a primeira rodada, o Grupo 1 mapeou a Política A usando a abordagem tradicional (Ad Hoc) e o Grupo 2 mapeou a Política A usando SPML2. Na segunda rodada, o Grupo 2 mapeou a política B utilizando a abordagem tradicional (Ad Hoc), enquanto o Grupo 1 mapeou a política B usando SPML2. A terceira e quarta sessões seguiram a mesma organização, no entanto, foi realizada a tarefa de manutenção.

Na Tabela 6.8 é mostrada a organização das sessões do experimento para a tarefa de configuração de *firewall* e na Tabela 6.9, a organização das sessões do experimento para a tarefa de manutenção de *firewall*.

Tabela 6.8: Organização das sessões do experimento - Configuração

Sessão	Política	Ad Hoc	SPML2
1	A	Grupo 1	Grupo 2
2	B	Grupo 2	Grupo 1

Tabela 6.9: Organização das sessões do experimento - Manutenção

Sessão	Política	Ad Hoc	SPML2
3	A	Grupo 1	Grupo 2
4	B	Grupo 2	Grupo 1

As atividades realizadas seguem apresentadas na Tabela 6.10, na qual é indicado o tempo gasto para cada tarefa. A duração total do experimento foi de 5 dias.

Tabela 6.10: Atividades dos experimentos

Dia	**Tempo**	**Tarefa**		
1	15 min	Apresentação do experimento		
	10 min	Assinatura do termo de consentimento livre e esclarecido		
	15 min	Preenchimento do questionário		
	90 min	Treinamento: Configuração do *firewall*, *Packet Filter* e SPML2		
2	10 min	Organização de participantes		
			Ad Hoc	SPML2
	120 min	Política A (configuração)	Grupo 1	Grupo 2
3	120 min	Política B (configuração)	Grupo 2	Grupo 1
4	100 min	Política A (manutenção)	Grupo 1	Grupo 2
5	100 min	Política B (manutenção)	Grupo 2	Grupo 1

No primeiro dia ocorreu a preparação e treinamento. Além disso, ocorreu uma explicação sobre as tarefas a serem executadas, a assinatura do termo de consentimento livre e esclarecido (TCLE), o preenchimento do questionário pelos voluntários para caracterização do perfil do aluno. Finalmente, as sessões de treinamento foram realizadas para a configuração de *firewalls* usando as regras de abordagem tradicionais e SPML2 (por meio da ferramenta SP2Model). A apresentação do treinamento foi projetada para familiarizar os participantes com as tarefas que deveriam executar.

6.6 Análise e Interpretação dos Resultados

Nesta seção são apresentados os resultados obtidos com o arranjo experimental descrito na Seção 6.5. A fim de esclarecer, esta seção é dividida em duas subseções que apresentam os resultados experimentais obtidos para as tarefas de configuração e manutenção.

6.6.1 Tarefa de Configuração

Nesta subseção são discutidos os resultados obtidos utilizando a configuração experimental relativa à tarefa de configuração descrita na Seção 6.5. A fim de permitir uma avaliação estatística robusta, foram realizados dois testes estatísticos[1] sobre a precisão média e o tempo, são eles: teste de Wilcoxon (Wilcoxon, 1945), e teste t (Devore, 2006). Em ambos os testes o valor retornado indica se existe uma diferença significativa entre os conjuntos de valores, para p-value igual a 0,05.

[1]Para calcular foi utilizada a biblioteca *Scientific Computing Tools for Python – SciPy* (SCIPY, 2016).

Os dados obtidos no experimento, para a tarefa de configuração de *firewall*, utilizados na elaboração deste trabalho de pesquisa podem ser observados na Tabela 6.11.

Tabela 6.11: Dados obtidos em conformidade com as métricas do experimento para a tarefa de configuração de *firewall*.

	Tarefa de Configuração			
	Eficácia [%]		**Eficiência [minutos]**	
Participante	**Ad Hoc**	**SPML2**	**Ad Hoc**	**SPML2**
1	25,81	87,23	10,13	1,88
2	3,23	51,06	53,00	2,17
3	0,00	87,23	120,00	2,29
4	9,68	68,09	21,33	4,50
5	0,00	4,26	120,00	39,00
6	64,52	48,94	3,00	3,09
7	3,23	53,19	70,00	3,40
8	32,26	2,13	5,30	54,00
9	6,38	45,16	38,00	9,00
10	0,00	6,45	120,00	60,50
11	2,13	3,23	45,00	115,00
12	17,02	3,23	11,13	74,00
13	0,00	22,58	120,00	23,14
14	17,02	54,84	9,25	5,82
15	10,64	87,10	12,40	2,22

As médias da eficácia obtida pelas abordagens SPML2 e Ad Hoc são apresentadas na Tabela 6.12. A dispersão e a variação da eficácia entre os grupos são apresentadas respectivamente nas Figuras 6.1 e 6.2, expressas em um diagrama de dispersão e um diagrama de caixa. A abordagem SPML2 obteve os melhores valores nos dois grupos (1 e 2) para a tarefa de configuração de *firewall* considerando o teste de Wilcoxon e o teste t. Para ambos os grupos, a precisão média do SPML2 foi, pelo menos, duas vezes maior do que a abordagem tradicional Ad Hoc. Nas Figuras 6.3 e 6.4, pode ser observado que, em geral, a eficiência e, consequentemente, o tempo gasto na execução da tarefa de configuração de *firewalls*, utilizando a abordagem SPML2, é significativamente melhor do que a abordagem Ad Hoc.

Os tempos médios gastos para a tarefa de configuração, considerando as abordagens Ad Hoc e SPML2, são apresentados na Tabela 6.13. O diagrama de dispersão e o diagrama de caixa referentes ao tempo gasto na tarefa de configuração são apresentados nas Figuras 6.3 e 6.4, respectivamente. A abordagem SPML2 obteve o melhor valor nos dois grupos (1 e 2) para a tarefa de configuração de *firewall*. Con-

Tabela 6.12: Médias da eficácia das abordagens Ad Hoc e SPML2 e os valores dos testes estatísticos considerando o experimento final. Os valores que apresentam os melhores resultados estão em negrito.

	Eficácia [%]		**P-value**	
	Ad Hoc	**SPML2**	**Teste t**	**Wilcoxon**
Grupo 1	17,34	**50,27**	0,00647	0,01348
Grupo 2	7,60	**31,80**		

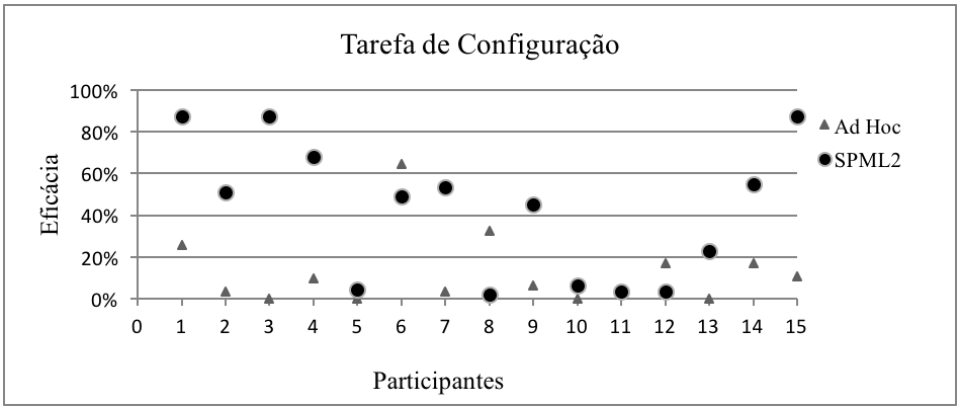


Figura 6.1: Diagrama de dispersão da eficácia para a tarefa de configuração.

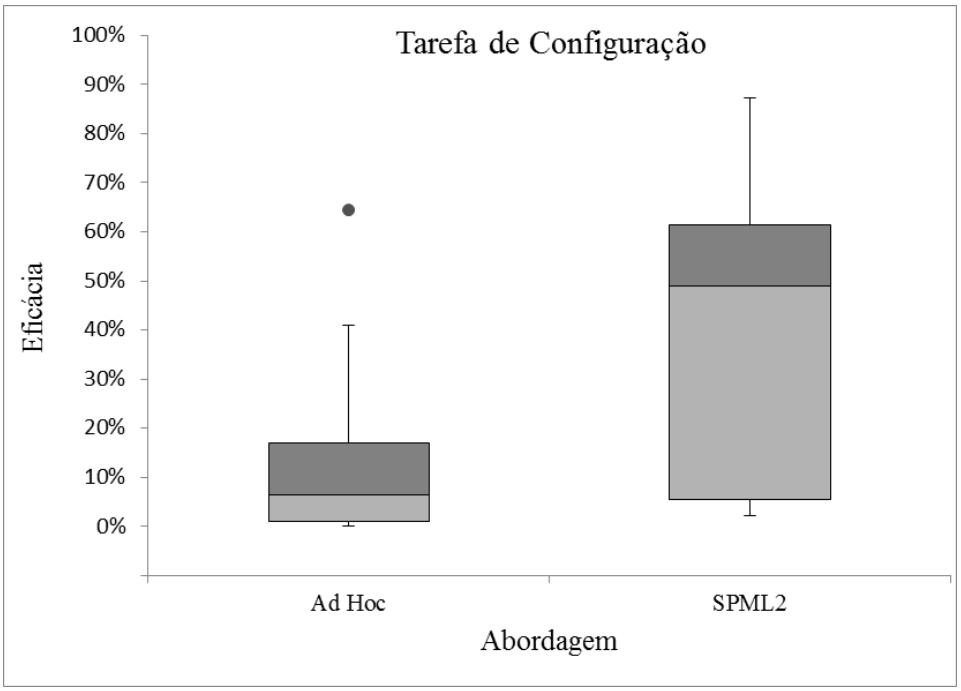


Figura 6.2: *Box plot* da eficácia para a tarefa de configuração.

tudo analisando os valores obtidos pelos testes estatísticos, apesar da SPML2 obter os melhores valores a diferença não foi significativa, deste modo ambas as meto-

dologias são equivalentes quanto a eficiência. Analisando a Figura 6.3 é possível observar que apenas dois alunos passaram mais tempo na SPML2 do que na abordagem Ad Hoc; no entanto, a diferença de tempo é muito maior e produz influência significativa na média.

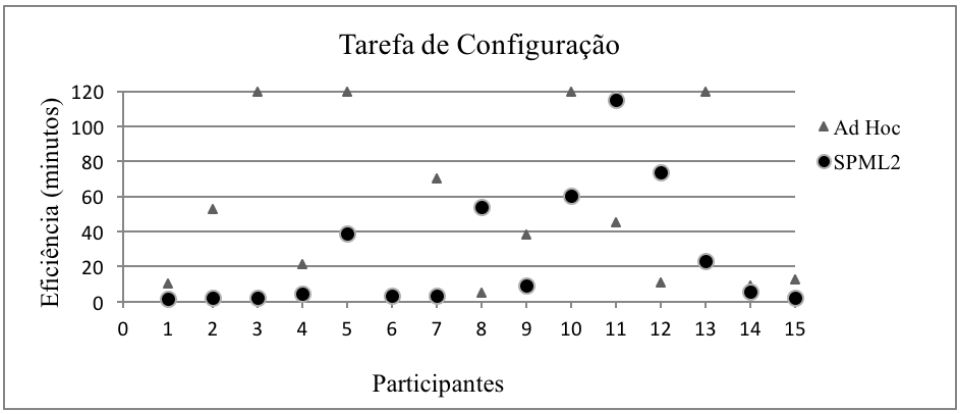


Figura 6.3: Diagrama de dispersão da eficiência para a tarefa de configuração.

Quatro participantes não produziram regras corretas utilizando a abordagem Ad Hoc para a realização da tarefa de configuração de *firewall*, por isso foi atribuído a eles o tempo máximo gasto na sessão.

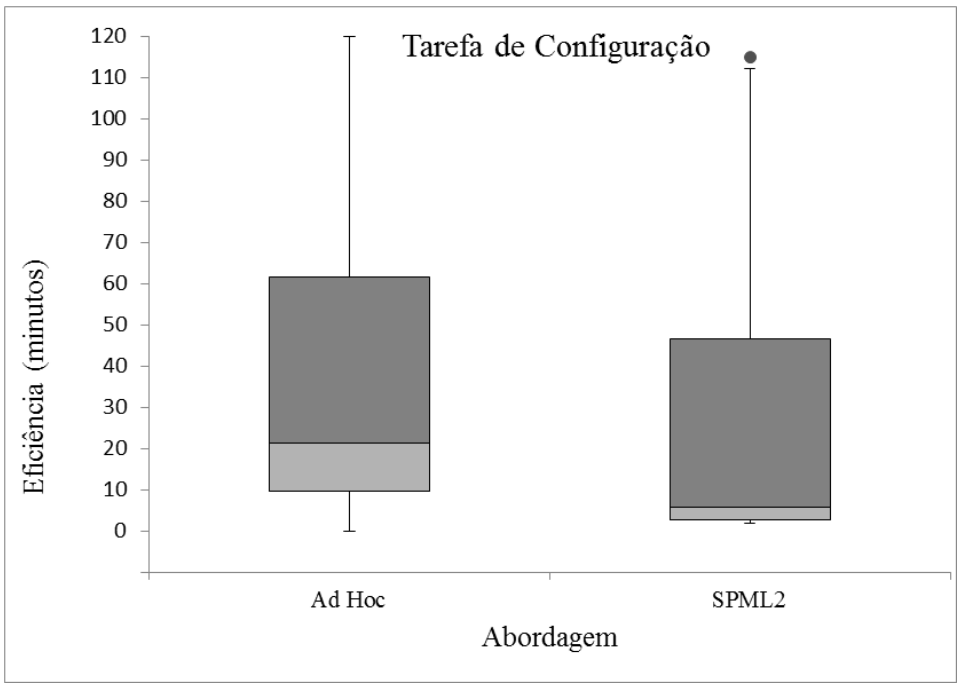


Figura 6.4: *Box plot* da eficiência para a tarefa de configuração.

Convém destacar que a configuração dos tráfegos de entrada e saída na abor-

dagem visual (SPML2) requer que o *firewall*, as interfaces e todas as entidades externas estejam com seus atributos configurados, essa característica pode ter contribuído para a pouca significância da diferença apresentada nos testes estatísticos.

Tabela 6.13: Eficiência Média (tempo gasto) na tarefa de configuração de *firewalls* utilizando as abordagens Ad Hoc e SPML2 e os valores dos testes estatísticos considerando o experimento final. Os valores de tempo mais baixos estão em negrito.

	Eficiência [minutos]		**P-value**	
	Ad Hoc	**SPML2**	**Teste t**	**Wilcoxon**
Grupo 1	50,34	**13,79**	0,12216	0,09383
Grupo 2	50,83	**41,38**		

6.6.2 Tarefa de Manutenção

Esta subseção apresenta os resultados obtidos para a tarefa de manutenção de *firewalls* utilizando as abordagens Ad Hoc e SPML2. Os dados obtidos no experimento, para a tarefa de manutenção de *firewall*, utilizados na elaboração deste trabalho de pesquisa podem ser observados na Tabela 6.11.

Tabela 6.14: Dados obtidos em conformidade com as métricas do experimento para a tarefa de manutenção de *firewall*.

	Tarefa de Manutenção			
	Eficácia [%]		**Eficiência [minutos]**	
Participante	**Ad Hoc**	**SPML2**	**Ad Hoc**	**SPML2**
1	92,86	100,00	0,38	0,33
2	97,62	100,00	0,24	0,27
3	2,38	100,00	35,00	0,55
4	69,05	59,18	0,83	0,62
5	69,05	87,76	0,97	0,23
6	59,52	100,00	0,32	0,18
7	2,38	87,76	23,00	0,42
8	64,29	87,76	0,48	0,49
9	46,94	80,95	1,30	0,68
10	75,51	80,95	1,51	0,82
11	75,51	73,81	1,51	0,81
12	97,96	80,95	1,46	0,41
13	75,51	100,00	0,57	0,79
14	73,47	97,62	0,67	0,56
15	73,47	100,00	0,39	0,50

As eficácias médias obtidas por ambos os métodos são apresentadas na Tabela 6.15. O diagrama de dispersão e o diagrama de caixa para a eficácia podem ser

observados nas Figuras 6.5 e 6.6, respectivamente. A abordagem SPML2 obteve os melhores valores nos dois grupos (1 e 2) para a tarefa de manutenção de *firewall* considerando o teste de Wilcoxon e o teste t. Para ambos os grupos, a eficácia média da SPML2 foi significativamente melhor do que a obtida pela abordagem tradicional Ad Hoc. As médias da SPML2 foram 33% e 13% melhores do que a abordagem Ad Hoc para os grupos 1 e 2 (diferença absoluta), respectivamente. Nas Figuras 6.9 e 6.10 pode ser observado que a eficiência da abordagem SPML2 é significativamente melhor do que a da abordagem Ad Hoc para a tarefa de manutenção.

Tabela 6.15: Eficácias médias obtidas pelas abordagens Ad Hoc e SPML2 e os valores dos testes estatísticos considerando a tarefa de manutenção para o experimento final. Os melhores valores estão em negrito.

	Eficácia [%]		**P-value**	
	Ad Hoc	**SPML2**	**Teste t**	**Wilcoxon**
Grupo 1	57,14	**90,31**	0,00447	0,00286
Grupo 2	74,05	**87,76**		

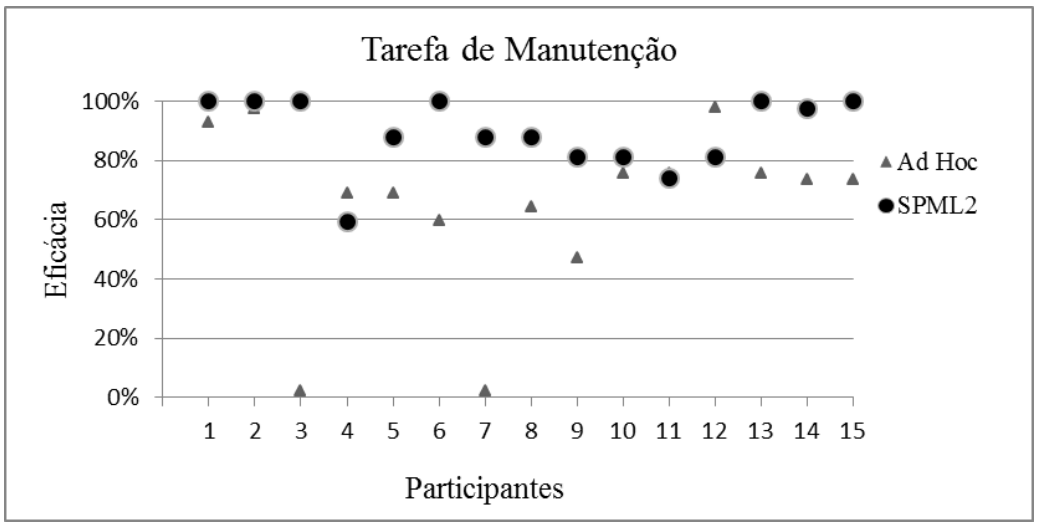


Figura 6.5: Diagrama de dispersão da eficácia para a tarefa de manutenção.

O tempo médio gasto para a tarefa de manutenção, considerando as abordagens Ad Hoc e SPML2, é apresentado na Tabela 6.16. Nas Figuras 6.9 e 6.10 são apresentados os diagramas de dispersão e o diagrama de caixa para a eficiência na tarefa de manutenção de *firewall*, respectivamente. A abordagem SPML2 obteve o melhor valor para ambos os grupos (1 e 2) na tarefa de manutenção de *firewall*. Contudo analisando os valores obtidos pelos testes estatísticos, apesar da SPML2 obter os melhores valores, a diferença não foi significativa para o teste t, porém foi

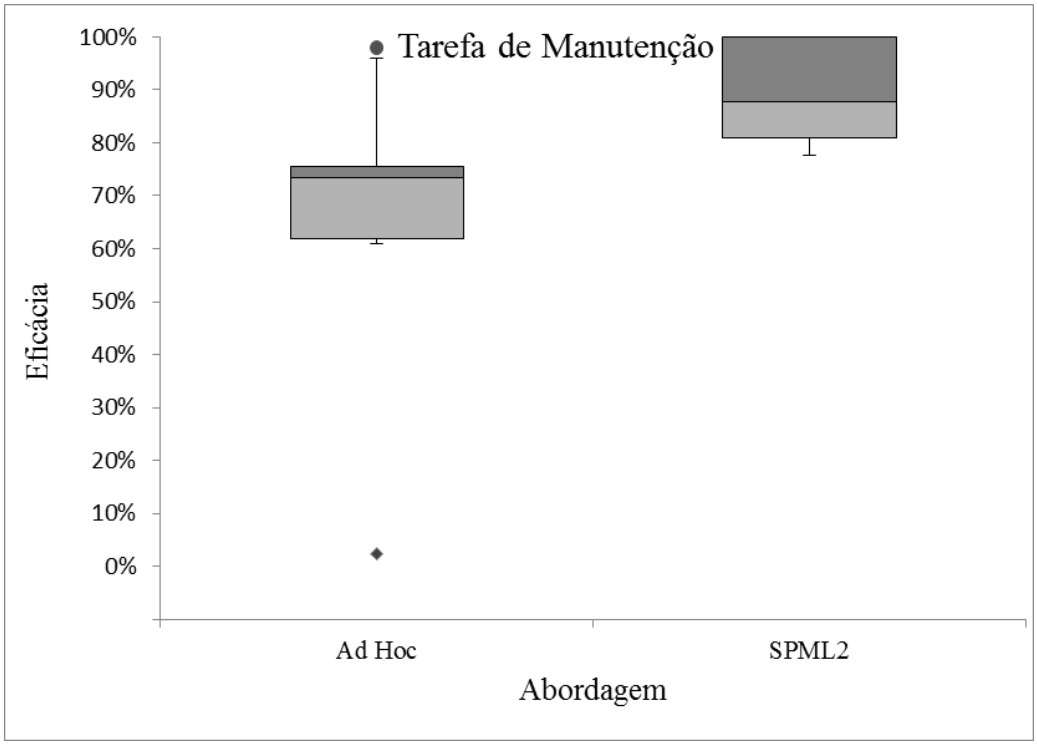


Figura 6.6: Diagrama de caixa da eficácia para a tarefa de manutenção.

Tabela 6.16: Eficiência Média (tempo gasto) utilizando as abordagens Ad Hoc e SPML2 e os valores dos testes estatísticos para a tarefa de manutenção de *firewall* considerando o experimento final. Os valores de tempo mais baixos estão em negrito.

	Eficiência [minutos]		**P-value**	
	Ad Hoc	**SPML2**	**Teste t**	**Wilcoxon**
Grupo 1	7,65	**0,39**	0,14445	0,01243
Grupo 2	1,06	**0,65**		

significativa para o teste de Wilcoxon. Considerando que, de modo geral, o teste de Wilcoxon apresenta precisão maior que teste t para conjuntos pequenos de amostras (menores que 20), é razoável considerar o resultado obtido pelo teste de Wilcoxon. Analisando a Figura 6.9 pode ser observado que apenas três alunos passaram mais tempo na abordagem SPML2 do que na abordagem Ad Hoc; e diferentemente da tarefa de configuração, nenhum participante gastou significativamente mais tempo na abordagem SPML2.

É possível ainda observar que a eficiência e a eficácia obtidas na tarefa de manutenção utilizando a abordagem SPML2 são relativamente superiores do que as obtidas na tarefa de configuração. Essa diferença ocorre porque a tarefa de confi-

guração requer uma compreensão completa das regras de *firewall*. Além disso, os resultados do experimento para a tarefa de manutenção sugerem que o entendimento da modelagem visual é mais claro do que a descrição textual.

O diagrama de dispersão e o diagrama de caixa da eficiência para a tarefa de manutenção de *firewall* elaborados com todos os dados de amostragem podem ser observados nas Figuras 6.7 e 6.8, respectivamente.

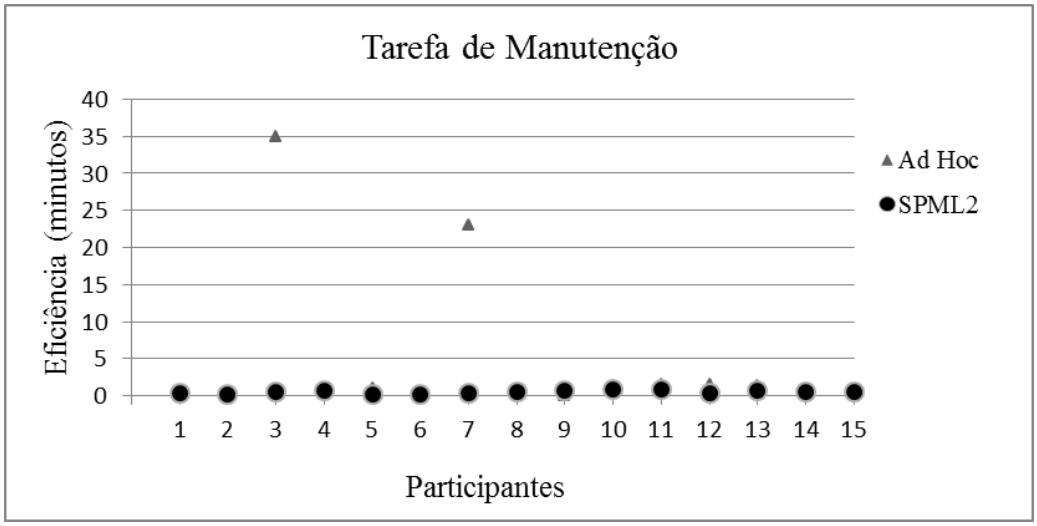


Figura 6.7: Diagrama de dispersão da eficiência para a tarefa de manutenção de *firewall* (com os *outliers*).

Os diagramas apresentados nas Figuras 6.7 e 6.8 foram elaborados com todos os dados de amostragem, os *outliers* provocaram a concentração dos demais dados no diagrama o que atrapalha sua interpretação. Para resolver esse problema foram elaborados outros diagramas (Figuras 6.9 e 6.10).

Para as Figuras 6.9 e 6.10 os valores ausentes referem-se aos participantes *outliers* da amostragem, que redigiram apenas uma regra de *firewall* corretamente na tarefa de manutenção utilizando a abordagem Ad Hoc.

6.7 Ameaças à Validade

As métricas apresentadas na Seção 6.2 foram formuladas considerando minimizar as ameaças à validade de conclusão, necessária para certificar que os dados gerados pelas variáveis independentes, por meio do tratamento, são relevantes para a análise estatística do experimento. As regras produzidas por cada participante foram comparadas a um conjunto de regras previamente testado (oráculo). Assim, a simples comparação é suficiente para determinar quais são as regras corretamente

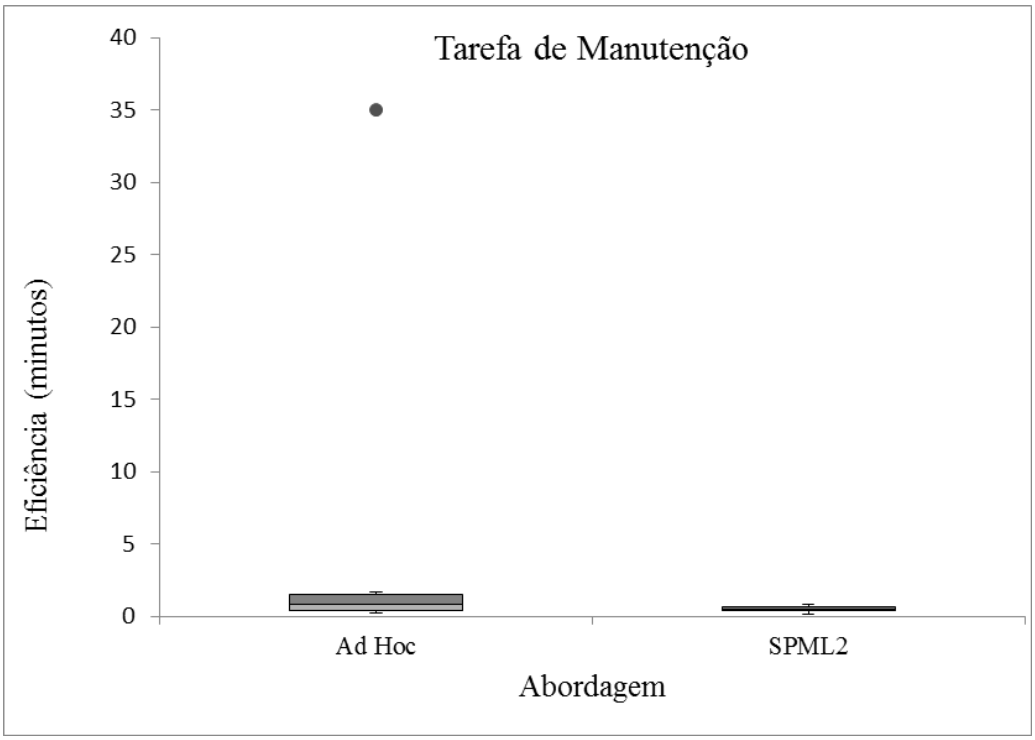


Figura 6.8: Diagrama de caixa da eficiência para a tarefa de manutenção de *firewall* (com os *outliers*).

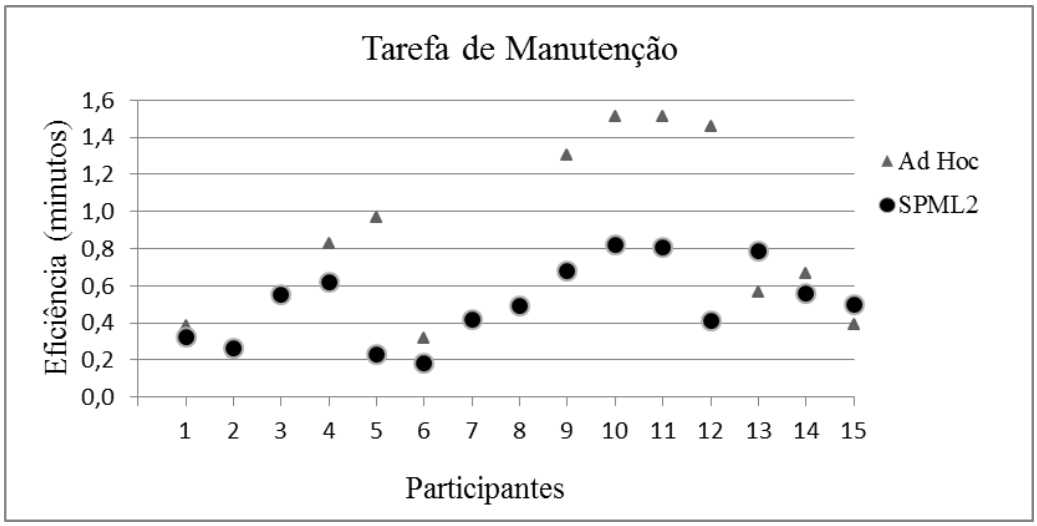


Figura 6.9: Diagrama de dispersão da eficiência para a tarefa de manutenção de *firewall* (sem os *outliers*).

produzidas.

Utilizar duas políticas de segurança diferentes (apresentadas na Subseção 6.3.1) contribui para minimizar as ameaças à validade de construção, pois a configuração

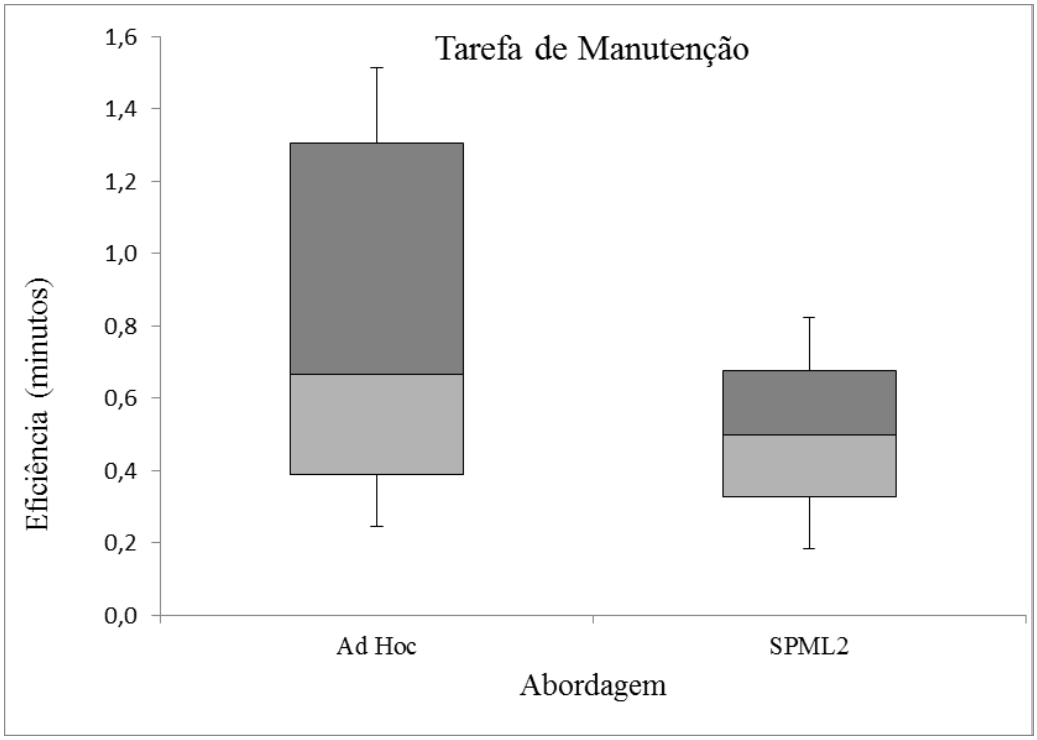


Figura 6.10: Diagrama de caixa da eficiência para a tarefa de manutenção de *firewall* (sem os *outliers*).

de uma política de segurança não deve influenciar na construção da outra. Ou seja, o conhecimento prévio de uma política de segurança (primeira tarefa) não pode influenciar na realização da segunda tarefa. Por isso é importante utilizar duas políticas diferentes, pois o tempo tende a ser minimizado quando se tem o conhecimento prévio.

O treinamento, detalhado na Subseção 6.3.3, foi realizado considerando minimizar as ameaças à validade interna. Assim, todos os indivíduos foram submetidos aos mesmos treinamentos e realizaram as mesmas atividades. Adicionalmente, a aleatoriedade na formação dos grupos, descrita na Seção 6.5, minimiza as ameaças à validade interna do experimento, pois evita o agrupamento tendencioso de indivíduos – um grupo com participantes com maior possibilidade de alcançar eficiência/eficácia em determinada abordagem.

6.8 Considerações Finais

Neste capítulo são apresentados a definição, o planejamento e o cronograma do experimento. A meta do experimento foi analisar o procedimento de mapeamento da política de segurança (textual) para regras em linguagem nativa de *firewall* utili-

zando os diagramas SPML2 e a abordagem tradicional, com o propósito de avaliação, no que diz respeito à eficiência e à eficácia do ponto de vista do pesquisador, no contexto de alunos da graduação, avaliando a corretude das regras produzidas.

CAPÍTULO 7

Conclusão

A segurança operacional é um dos pilares que sustentam a segurança em redes de equipamentos computacionais. Conforme evidenciado por Bandara et al. (2009), a definição e a manutenção da política de segurança expressa em regras de um *firewall* tornam-se tarefas complexas e sujeitas a muitos erros com o aumento do tamanho das redes e da variedade de protocolos. Os estudos de Wool (2004, 2010) mostram que muitos são os *firewalls* vulneráveis, pois suas regras não expressam a política de segurança definida. E segundo Trevisani & Garcia (2008), a utilização de uma notação gráfica que modele a política de segurança em uma notação gráfica facilita o seu entendimento e, consequentemente, a sua manutenção. Além disso, ter uma ferramenta que traduza o modelo da política de segurança em regras de um *firewall* evita que erros de mapeamento tornem o perímetro da rede desprotegido.

Neste trabalho é recapitulada a SPML (Trevisani & Garcia, 2008) – *Security Policy Modeling Language* –, uma linguagem visual para criação de modelo da política de segurança em uma notação gráfica. Os elementos da SPML são apresentados, e um exemplo de política de segurança é modelado em SPML. Após o estudo da SPML notou-se que restrições sintáticas não tinham sido especificadas formalmente, embora tenham sido implementadas como parte da ferramenta SoH. O mesmo vale para a definição semântica.

As definições sintática e semântica possibilitam validar se a política de segurança modelada está expressa em conformidade com as regras da SPML. Assim, neste trabalho, foram definidas uma gramática livre de contexto (EBNF) e uma es-

pecificação em notação **Z** com o objetivo de dar suporte à validação sintática, à validação semântica e à correta tradução do modelo de uma política de segurança gerado em SPML para regras em linguagem nativa de um *firewall*.

É apresentada ainda a SPML2, uma extensão da SPML, como uma ferramenta de apoio ao processo de configuração de *firewall*. A SPML2 foi avaliada por meio de um experimento controlado. A definição da SPML2 e os resultados obtidos com a linguagem visual foram apresentados neste trabalho.

Destacam-se duas principais vantagens da SPML2. Em primeiro lugar, propôs uma nova abordagem para o processo de configuração de *firewall* usando uma representação visual. Em geral, os trabalhos são relacionados com foco na validação das regras de *firewall*, enquanto este trabalho apresenta uma visão holística da configuração de *firewall*, aumentando a visibilidade de todo o processo para o administrador de redes. Em segundo lugar, os resultados demonstram que SPML2 facilita o processo de compreensão e tradução de regras de *firewall*. A eficácia obtida pela abordagem SPML2 foi significativamente melhor do que a abordagem Ad Hoc, conforme apresentado na Seção 6.6, apesar de o tempo médio da SPML2 ser maior do que Ad Hoc para o Grupo 2; apenas dois em quinze estudantes passaram mais tempo usando SPML2.

7.1 Contribuições

A realização deste trabalho permitiu evoluir a SPML pela elaboração de uma extensão que expressasse as funcionalidades previstas em um *firewall*, essa evolução deu origem a SPML2. Foram criados os formalismos sintático e semântico da SPML que serviram de base para os formalismos sintático e semântico da SPML2. Também foi verificado que a SPML2, apoiada pela metalinguagem criada neste trabalho, realiza a correta tradução da política de segurança modelada (representação visual) para regras em linguagem nativa de *firewall*, validadas pelos formalismos sintático e semântico. Esses formalismos permitem que a política de segurança expressa em SPML2 possa ser traduzida para regras em linguagem nativa de qualquer produto de *firewall*.

Ao realizar o experimento obteve-se evidências de que a utilização da abordagem visual SPML2, como passo intermediário no procedimento de modelar a política de segurança em regras nativas de *firewall*, proporcionou maiores eficiência e eficácia do que o procedimento tradicional (Ad Hoc). Para a tarefa de manutenção, a SPML2 foi 33% melhor que a abordagem Ad Hoc para o grupo 1 e 13% melhor do que a abordagem Ad Hoc para o grupo 2 (diferença absoluta de percentuais).

Já para a tarefa de configuração, a SPML2 foi 32% melhor que a abordagem Ad Hoc para o grupo 1 e 24% melhor do que a abordagem Ad Hoc para o grupo 2. Isso corrobora o fato de que a abordagem SPML2 é mais eficaz e mais eficiente que a abordagem Ad Hoc, independentemente da política de segurança e do grupo de indivíduos – as hipóteses nulas foram rejeitadas. Além disso, foi possível observar que a SPML2 facilitou a compreensão dos alunos de Ciência da Computação participantes do experimento quanto ao funcionamento de um *firewall*, embora este fato tenha sido mera observação durante a execução do experimento, sem comprovação estatística.

Espera-se que este trabalho possa colaborar com o administrador de redes na tarefa de compreender corretamente a política de segurança e mapeá-la em regras nativas de um *firewall*, contribuindo para que alterações nessa política não provoquem defeitos na configuração do *firewall* e, dessa forma, possam causar falhas na segurança da comunicação entre redes.

7.2 Trabalhos Futuros

Considerando a melhoria obtida pela modelagem visual SPML2, para os trabalhos futuros pretende-se implementar um novo módulo da SP2Model capaz de traduzir automaticamente as regras em língua nativa de *firewall* para a notação SPML2 (engenharia reversa). Esse aprimoramento é ferramenta útil para a tarefa de manutenção, principalmente por não requerer o mapeamento para modelo SPML2. Pretende-se, ainda, adicionar à SP2Model a possibilidade de traduzir o modelo SPML2 para regras em língua nativa de diferentes produtos de *firewall* (IPTABLES, por exemplo).

Referências

Aho, A. V., Lam, M. S., Sethi, R., e Ullman, J. D. (2008). *Compiladores: princípios, técnicas e ferramentas*. Pearson Addison–Wesley, São Paulo, 2. ed.

Al-Shaer, E., El-Atawy, A., e Samak, T. (2009). Automated pseudo-live testing of firewall configuration enforcement. *Selected Areas in Communications, IEEE Journal on*, 27(3):302–314.

Backus, J. e Naur, P. (1959). The syntax and semantics of the proposed international algebric language of the zurich acm-gamm conference. *Proceedings International Conference on Information Processing*, pp. 125–132.

Bandara, A., Kakas, A., Lupu, E., e Russo, A. (2009). Using argumentation logic for firewall configuration management. In *Integrated Network Management, 2009. IM '09. IFIP/IEEE International Symposium on*, pp. 180–187.

Basili, V. R., Caldiera, G., e Rombach, H. D. (1994). Goal question metric paradigm. Technical report, University of Maryland, College Park, MD, USA.

Ben Souayeh Ben Youssef, N. e Bouhoula, A. (2010). Automatic conformance verification of distributed firewalls to security requirements. In *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, pp. 834–841.

Berners-Lee, T., W3C/MIT, Fielding, R., Software, D., Masinter, L., e Systems, A. (2005). Uniform resource identifier (uri): Generic syntax.

Brito, S. H. B. (2013). *IPv6 O novo protocolo da Internet*. Novatec Editora Ltda, São Paulo, SP.

Chomsky, N. (1956). Three models for the description of language. *Information Theory, IRE Transactions on*, 2(3):113–124.

Devore, J. L. (2006). *Probabilidade e estatística: para engenharia e ciências*. Pioneira Thomson Learning.

El-Atawy, A., Samak, T., Wali, Z., e Al-Shaer, E. (2007). An automated framework for validating firewall policy enforcement. In *Policies for Distributed Systems and Networks, 2007. POLICY '07. Eighth IEEE International Workshop on*, pp. 151–160.

Forouzan, B. A. e Mosharraf, F. (2013). *Redes e Computadores: uma abordagem top-down*. AMGH, Porto Alegre.

Gawanmeh, A. (2014). Automatic verification of security policies in firewalls with dynamic rule sequence. In *Information Technology: New Generations (ITNG), 2014 11th International Conference on*, pp. 279–284.

Goodrich, M. T. e Tamasia, R. (2013). *Segurança de Computadores*. Bookman, Porto Alegre.

Hussain, S., Erwin, H., e Dunne, P. (2011). Threat modeling using formal methods: A new approach to develop secure web applications. In *Emerging Technologies (ICET), 2011 7th International Conference on*, pp. 1–5.

Jin-hua, W., Xiao-su, C., Yi-Zhu, Z., e Jun, N. (2008). A flexible policy-based firewall management framework. In *Cyberworlds, 2008 International Conference on*, pp. 192–194.

Kurose, J. F. e Ross, K. W. (2010). *Redes de computadores e a Internet: uma abordagem top-down*. Addison Wesley, São Paulo, 5. ed.

Kurose, J. F. e Ross, K. W. (2013). *Redes de computadores e a Internet: uma abordagem top-down*. Pearson Education do Brasil, São Paulo, 6. ed.

Louden, K. C. (2004). *Compiladores: princípios e práticas*. Pioneira Thomson Learning, São Paulo, 9. ed.

Moussa, M., Ould-Slimane, H., Boucheneb, H., e Chamberland, S. (2014). A formal framework for verifying inter-firewalls consistency. In *Computers and Communication (ISCC), 2014 IEEE Symposium on*, pp. 1–7.

OPENBSD (2016). Pf: The openbsd packet filter. Disponível em: http://www.openbsd.org/faq/pf/. Acessado em: 24 jan. 2016.

Ramos, M. V. M., José Neto, J., e Vega, I. S. (2009). *Linguagens formais: teoria, modelagem e implementação*. Bookman, Porto Alegre, 1. ed.

Rekhter, Y., Moskowitz, R. G., Karrenberg, D., de Groot, G. J., e Lear, E. (1996). Address allocation for private internets.

SCIPY (2016). Scientific computing tools for python. Disponível em: https://www.scipy.org/index.html. Acessado em: 15 mai. 2016.

Sebesta, R. W. (2011). *Conceitos de linguagens de programação*. Bookman, Porto Alegre, 9. ed.

Socolofsky, T. e Kale, C. (1991). A tcp/ip tutorial. RFC 1180.

Srisuresh, P. e Holdrege, M. (1999). Ip network address translator (nat) terminology and considerations.

Stallings, W. (2008). *Criptografia e segurança de redes*. Pearson Prentice Hall, São Paulo, 4 ed.

Tanenbaum, A. S. e Wetherall, D. (2011). *Redes de computadores*. Pearson Prentice Hall, São Paulo, 5. ed.

Trevisani, K. M. e Garcia, R. E. (2008). Spml: A visual approach for modeling firewall configurations. *MODSEC08: International Conference on Model Driven Engineering Languages and Systems*.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., e Wesslén, A. (2012). *Experimentation in Software Engineering*. Computer Science. Springer.

Wool, A. (2004). A quantitative study of firewall configuration errors. *Computer*, 37(6):62–67.

Wool, A. (2010). Trends in firewall configuration errors: Measuring the holes in swiss cheese. *Internet Computing, IEEE*, 14(4):58–65.

Yuan, L., Chen, H., Mai, J., Chuah, C.-N., Su, Z., e Mohapatra, P. (2006). Fireman: a toolkit for firewall modeling and analysis. In *Security and Privacy, 2006 IEEE Symposium on*, pp. 15 pp.–213.

APÊNDICE
A

Apêndice A

A.1 Políticas de segurança utilizadas no experimento

A.1.1 Política de acesso a redes da instituição A (Configuração)

Esse documento visa definir as regras de acesso entre as redes da instituição e redes externas. As regras de acesso a redes serão reavaliadas periodicamente pelo comitê de segurança da informação e alterações solicitadas deverão ser autorizadas pelo conselho diretivo. O departamento de Tecnologia da Informação poderá, a qualquer tempo, bloquear acessos que possam comprometer a segurança operacional da instituição. A política definida por padrão é bloquear, sendo assim, todo acesso a redes que não for explicitamente definido como permitido por essa política deverá ser bloqueado. A rede acadêmica (ACDNet) deve ser isolada da rede administrativa (ADMNet) para prevenir que estudantes tenham acesso a informações ou serviços de rede não autorizados. Os servidores da instituição serão instalados na rede desmilitarizada (DMZNet).

Informações para configuração do *firewall*

O *firewall* possui quatro interfaces de rede, uma para cada sub-rede cujo o acesso necessita ser controlado. Os atributos necessários para configuração das interfaces

podem ser observados na Tabela A.1.

Tabela A.1: Atributos das interfaces de rede do *firewall*.

Conecta-se a sub-rede	**Interface**	**IP/Máscara**
administrativa	em2	192.168.0.1/24
acadêmica	em1	10.0.0.1/16
desmilitarizada	em3	172.16.0.1/24
Internet	em0	10.10.10.10/30

A seguir são detalhados os acessos/bloqueios organizados pelos recursos que podem ser acessados entre redes. Os prefixos e IP's das sub-redes e *hosts* mencionados nas regras subsequentes estão listados respectivamente na Tabela A.2 e na Tabela A.3. Caso exista alguma exceção a uma determinada regra, essa exceção será listada como um sub tópico da regra.

Tabela A.2: Sub-redes acessadas pela instituição.

Sub-rede	**Prefixo/Máscara**
administrativa	192.168.0.0/24
acadêmica	10.0.0.0/16
desmilitarizada	172.16.0.0/24
Internet	*

- Serviço de DNS (porta 53, protocolo *UDP*) do servidor dns-externo localizado na Internet pode ser acessado por:
  - Todos os *hosts* da sub-rede acadêmica (requer *nat*)
  - Todos os *hosts* da sub-rede administrativa (requer *nat*)
  - Todos os *hosts* da sub-rede desmilitarizada (requer *nat*)
- Serviço *proxy-squid* que opera no servidor *squid* (porta 3128, protocolo *TCP*) pode ser acessado por:
  - Todos os *hosts* da sub-rede acadêmica
  - Todos os *hosts* da sub-rede administrativa
- Serviços de navegação *web* (porta 80 e 443, protocolo *TCP*) localizados na Internet podem ser acessado por:
  - O *host squid* (servidor *proxy-squid*) (requer *nat*)

- Serviço de controle de versões no *host repository* que utiliza o protocolo Git (porta 9418, protocolo *TCP*) e o protocolo *SSH* (porta 22, protocolo *TCP*) pode ser acessado por:
  - Todos os hosts da sub-rede acadêmica
- Serviço de *email* (*smtp* porta 25 e pop3 porta 110, todos protocolo *TCP*) do Google Apps (servidores 173.194.205.108 e 173.194.205.109) pode ser acessado por:
  - Todos os *hosts* da sub-rede administrativa (requer *nat*)
- Serviço *HTTP* no *host* site (porta 8080, protocolo *TCP* a partir de redirecionamento da porta 80) pode ser acessado por:
  - Todos os hosts da Internet (requer *rdr*)
- Serviço de mensagens de controle de rede (protocolo *ICMP*) na Internet pode ser acessado por:
  - Todos os hosts da sub-rede desmilitarizada (requer *nat*)

Tabela A.3: *Hosts* com regras de acesso específicas.

Host	**IP/Máscara**
squid	172.16.0.10/24
site	172.16.0.80/24
repository	172.16.0.20/24
prof1	10.0.1.1/16
prof2	10.0.1.2/16
prof3	10.0.1.3/16
manut	192.168.0.32/24
dns-externo	8.8.8.8/32
gmail1	173.194.205.108/32
gmail2	173.194.205.109/32

A.1.2 Política de acesso a redes da instituição A (Manutenção)

A política de segurança foi modificada pela diretoria, alguns recursos que sofreram alteração com relação a permissão de acesso ou bloqueio. Analise os tópicos listados abaixo e implemente a alteração no arquivo fornecido.

- Serviço *proxy-squid* que opera no servidor *squid* (porta 3128, protocolo *TCP*) pode ser acessado por:
  - Todos os *hosts* da sub-rede acadêmica
  - Todos os *hosts* da sub-rede administrativa
    - O *host manut* não pode acessar o *proxy-squid*.
- Serviços de navegação *web* (porta 80 e 443, protocolo *TCP*) localizados na Internet podem ser acessado por:
  - Os *hosts* dos professores (*prof1*, *prof2* e *prof3*) (requer *nat*)
  - O *host squid* (servidor *proxy-squid*) (requer *nat*)
- Serviço de email (smtp porta 25, pop3 porta 110 e imap porta 143, todos protocolo TCP) do Google Apps (servidores 173.194.205.108 e 173.194.205.109) pode ser acessado por:
  - Todos os *hosts* da sub-rede administrativa (requer *nat*)

A.1.3 Política de acesso a redes da instituição B (Configuração)

Esse documento visa definir as regras de acesso entre as redes da instituição e redes externas. As regras de acesso a redes serão reavaliadas periodicamente pelo comitê de segurança da informação e alterações solicitadas deverão ser autorizadas pelo conselho diretivo. O departamento de Tecnologia da Informação poderá a qualquer tempo bloquear acessos que possam comprometer a segurança operacional da instituição. A política definida por padrão é bloquear, sendo assim todo acesso a redes que não for explicitamente definido como permitido por essa política deverá ser bloqueado. A rede interna (INTNet) deve ser isolada da rede dos servidores que possuem acesso restrito (NESNet) para prevenir que usuários tenham acesso a informações ou serviços de rede não autorizados. Os servidores da instituição que possuem acesso a/da Internet serão instalados na rede desmilitarizada (DMZNet).

Informações para configuração do *firewall*

O *firewall* possui quatro interfaces de rede, uma para cada sub-rede cujo o acesso necessita ser controlado. Os atributos necessários a configuração das interfaces podem ser observados na Tabela A.4.

Tabela A.4: Atributos das interfaces de rede do *firewall*.

Conecta-se a sub-rede	**Interface**	**IP/Máscara**
ninho	em2	192.168.0.1/24
interna	em1	10.0.0.1/16
desmilitarizada	em3	172.16.0.1/24
Internet	em0	10.10.10.10/30

A seguir são detalhados os acessos/bloqueios organizados pelos recursos que podem ser acessados entre redes. Os prefixos e IP's das sub-redes e *hosts* mencionados nas regras subsequentes estão listados respectivamente na Tabela A.5 e na Tabela A.6. Caso exista alguma exceção a uma determinada regra, essa exceção será listada como um sub tópico da regra.

Tabela A.5: Sub-redes acessadas pela instituição.

Sub-rede	**Prefixo/Máscara**
ninho	192.168.0.0/24
interna	10.0.0.0/16
desmilitarizada	172.16.0.0/24
Internet	*

- Serviço de DNS (porta 53, protocolo *UDP*) disponível no *host dns1* e no *host dns2*, localizados na Internet, pode ser acessado por:
  - Todos os *hosts* da sub-rede interna (requer *nat*)
  - Todos os *hosts* da sub-rede desmilitarizada (requer *nat*)
- Serviço *proxy-squid* que opera no servidor *squid* (porta 3128, protocolo *TCP*) pode ser acessado por:
  - Todos os *hosts* da sub-rede interna
- Serviços de navegação *web* (porta 80 e 443, protocolo *TCP*) na Internet podem ser acessado por:
  - Os *hosts fin1* e *fin2* do departamento financeiro da sub-rede interna (requer *nat*)
  - O *host squid* (servidor *proxy-squid*) da sub-rede desmilitarizada (requer *nat*)

- Serviço de *email* (SMTP porta 465, POP3 porta 995 e IMAP porta 993, todos protocolo *TCP*) da LocaWeb (servidor 186.202.140.222) pode ser acessado por:
  - Todos os *hosts* da sub-rede interna (requer *nat*)
- Servidor de aplicação *oas* (portas 7777 e 7877, protocolo *TCP*) hospedado no ninho de servidores pode ser acessado por:
  - Todos os *hosts* da sub-rede interna
- Serviço de Banco de Dados (porta 1521, protocolo *TCP*) hospedado no servidor *odb* pode ser acessado por:
  - Os *hosts* do departamento de Tecnologia da Informação da sub-rede interna
- Serviço do *Oracle Enterprise Manager* (porta 5500, porta *TCP*) hospedado no servidor *odb*, pode ser acessado por:
  - Os *hosts* do departamento de Tecnologia da Informação da sub-rede interna
- Serviço de crédito consignado do banco do brasil (porta 7081, protocolo *TCP*) hospedado no servidor *ccbb*, pode ser acessado por:
  - Os *hosts* do departamento financeiro (*fin3* e *fin4*) da sub-rede interna (requer *nat*)
- Serviço de mensagens de controle de rede (protocolo *ICMP*) na Internet pode ser acessado por:
  - Todos os *hosts* da sub-rede desmilitarizada (requer *nat*)
- Serviço *SSH* (porta 2002, protocolo *TCP*) no servidor *ssh* localizado na rede desmilitarizada pode ser acessado por:
  - o host suporte localizado na Internet (requer *rdr*)
- O serviço *SSH* (porta 22, protocolo *TCP*) de todos os servidores da sub-rede ninho pode ser acessado por:
  - o *host ssh* (servidor) localizado na sub-rede desmilitarizada

Tabela A.6: *Hosts* com regras de acesso específicas.

Host	**IP/Máscara**
squid	172.16.0.10/24
dns1	208.67.222.222/32
dns2	208.67.220.220/32
locaweb	186.202.140.222/32
fin1	10.0.1.1/16
fin2	10.0.1.2/16
fin3	10.0.1.3/16
fin4	10.0.1.4/16
cred1	10.0.2.1/16
cred2	10.0.2.2/16
tes1	10.0.3.1/16
tes2	10.0.3.2/16
tes3	10.0.3.3/16
tes4	10.0.3.4/16
acserver	200.211.20.5/32
energisa	191.242.201.6/32
amtu	191.6.194.125/32
ccbb	170.66.8.142/32
ti	10.0.5.0/24
oas	192.168.0.158/24
odb	192.168.0.111/24
suporte	131.100.210.78/32
ssh	172.16.0.22/24

A.1.4 Política de acesso a redes da instituição B (Manutenção)

A política de segurança foi modificada pela diretoria, alguns recursos que sofreram alteração com relação a permissão de acesso ou bloqueio. Analise os tópicos listados abaixo e implemente a alteração no arquivo fornecido.

- Serviço de consulta de situação cadastral (porta 3006 e 3007, protocolo *TCP*) hospedado no servidor da associação comercial *acserver*, pode ser acessado por:
  - Os *hosts* do departamento de crédito (*cred1* e *cred2*) da sub-rede interna (requer *nat*)
- O serviço de emissão de fatura via Internet (portas 8081,8083,8088, todos protocolo *TCP*) hospedado no servidor da *energisa*, pode ser acessado por:

  - Os *hosts* do departamento de tesouraria (*tes1* e *tes2*) da sub-rede interna (requer *nat*)

- O serviço de controle de vale transporte (porta 8090, protocolo *TCP*) hospedado no servidor da associação municipal de transportes urbanos (*amtu*), pode ser acessado por:

  - Os *hosts* do departamento de tesouraria (*tes3* e *tes4*) da sub-rede interna (requer *nat*)

A.2 Oráculos utilizados no experimento

A.2.1 Oráculo da política A (Configuração)

O oráculo das regras em linguagem nativa de *firewall* da política de segurança da instituição A na sintaxe do *packet filter* para a tarefa de configuração é apresentado na Definição A.1.

A.2.2 Oráculo da política A (Manutenção)

O oráculo das regras em linguagem nativa de *firewall* da política de segurança da instituição A na sintaxe do *packet filter* para a tarefa de manutenção é apresentado na Definição A.2.

A.2.3 Oráculo da política B (Configuração)

O oráculo das regras em linguagem nativa de *firewall* da política de segurança da instituição B na sintaxe do *packet filter* para a tarefa de configuração é apresentado na Definição A.1.

A.2.4 Oráculo da política B (Manutenção)

O oráculo das regras em linguagem nativa de *firewall* da política de segurança da instituição B na sintaxe do *packet filter* para a tarefa de manutenção é apresentado na Definição A.4.

Definição A.1: Oráculo das regras de *firewall* da política de segurança da instituição A – sintaxe do *packet filter* – tarefa de configuração.

```
block in all
pass in on em0 inet proto tcp from any to 10.10.10.10 port 80 \
     rdr-to 172.16.0.80 port 8080
pass in on em1 inet proto tcp from 10.0.0.0/16 to 172.16.0.10 port 3128
pass in on em1 inet proto tcp from 10.0.0.0/16 to 172.16.0.20 port 22
pass in on em1 inet proto tcp from 10.0.0.0/16 to 172.16.0.20 port 9418
pass in on em1 inet proto udp from 10.0.0.0/16 to 8.8.8.8 port 53
pass in on em2 inet proto tcp from 192.168.0.0/24 to 172.16.0.10 port 3128
pass in on em2 inet proto tcp from 192.168.0.0/24 to 173.194.205.108 port 110
pass in on em2 inet proto tcp from 192.168.0.0/24 to 173.194.205.108 port 25
pass in on em2 inet proto tcp from 192.168.0.0/24 to 173.194.205.109 port 110
pass in on em2 inet proto tcp from 192.168.0.0/24 to 173.194.205.109 port 25
pass in on em2 inet proto udp from 192.168.0.0/24 to 8.8.8.8 port 53
pass in on em3 inet proto icmp from 172.16.0.0/24 to any
pass in on em3 inet proto tcp from 172.16.0.10 to any port 443
pass in on em3 inet proto tcp from 172.16.0.10 to any port 80
pass in on em3 inet proto udp from 172.16.0.0/24 to 8.8.8.8 port 53
pass out on em0 inet proto icmp from 172.16.0.0/24 to any nat-to 10.10.10.10
pass out on em0 inet proto tcp from 172.16.0.10 to any port 443 \
     nat-to 10.10.10.10
pass out on em0 inet proto tcp from 172.16.0.10 to any port 80 \
     nat-to 10.10.10.10
pass out on em0 inet proto tcp from 192.168.0.0/24 \
     to 173.194.205.108 port 110 nat-to 10.10.10.10
pass out on em0 inet proto tcp from 192.168.0.0/24 \
     to 173.194.205.108 port 25 nat-to 10.10.10.10
pass out on em0 inet proto tcp from 192.168.0.0/24 \
     to 173.194.205.109 port 110 nat-to 10.10.10.10
pass out on em0 inet proto tcp from 192.168.0.0/24 \
     to 173.194.205.109 port 25 nat-to 10.10.10.10
pass out on em0 inet proto udp from 10.0.0.0/16 to 8.8.8.8 \
     port 53 nat-to 10.10.10.10
pass out on em0 inet proto udp from 172.16.0.0/24 to 8.8.8.8 \
     port 53 nat-to 10.10.10.10
pass out on em0 inet proto udp from 192.168.0.0/24 to 8.8.8.8 \
     port 53 nat-to 10.10.10.10
pass out on em3 inet proto tcp from 10.0.0.0/16 to 172.16.0.10 port 3128
pass out on em3 inet proto tcp from 10.0.0.0/16 to 172.16.0.20 port 22
pass out on em3 inet proto tcp from 10.0.0.0/16 to 172.16.0.20 port 9418
pass out on em3 inet proto tcp from 192.168.0.0/24 to 172.16.0.10 port 3128
pass out on em3 inet proto tcp from any to 172.16.0.80 port 8080
```

Definição A.2: Oráculo das regras de *firewall* da política de segurança da instituição A – sintaxe do *packet filter* – para a tarefa de manutenção.

```
1  block in all
2  pass in on em0 inet proto tcp from any to 10.10.10.10 port 80 \
3       rdr-to 172.16.0.80 port 8080
4  pass in on em1 inet proto tcp from 10.0.0.0/16 to 172.16.0.10 port 3128
5  pass in on em1 inet proto tcp from 10.0.0.0/16 to 172.16.0.20 port 22
6  pass in on em1 inet proto tcp from 10.0.0.0/16 to 172.16.0.20 port 9418
7  pass in on em1 inet proto udp from 10.0.0.0/16 to 8.8.8.8 port 53
8  pass in on em2 inet proto tcp from 192.168.0.0/24 to 172.16.0.10 port 3128
9  pass in on em2 inet proto tcp from 192.168.0.0/24 to 173.194.205.108 port 110
10 pass in on em2 inet proto tcp from 192.168.0.0/24 to 173.194.205.108 port 25
11 pass in on em2 inet proto tcp from 192.168.0.0/24 to 173.194.205.109 port 110
12 pass in on em2 inet proto tcp from 192.168.0.0/24 to 173.194.205.109 port 25
13 pass in on em2 inet proto udp from 192.168.0.0/24 to 8.8.8.8 port 53
14 pass in on em3 inet proto icmp from 172.16.0.0/24 to any
15 pass in on em3 inet proto tcp from 172.16.0.10 to any port 443
16 pass in on em3 inet proto tcp from 172.16.0.10 to any port 80
17 pass in on em3 inet proto udp from 172.16.0.0/24 to 8.8.8.8 port 53
18 pass out on em0 inet proto icmp from 172.16.0.0/24 to any nat-to 10.10.10.10
19 pass out on em0 inet proto tcp from 172.16.0.10 to any port 443 \
20      nat-to 10.10.10.10
21 pass out on em0 inet proto tcp from 172.16.0.10 to any port 80 \
22      nat-to 10.10.10.10
23 pass out on em0 inet proto tcp from 192.168.0.0/24 \
24      to 173.194.205.108 port 110 nat-to 10.10.10.10
25 pass out on em0 inet proto tcp from 192.168.0.0/24 \
26      to 173.194.205.108 port 25 nat-to 10.10.10.10
27 pass out on em0 inet proto tcp from 192.168.0.0/24 \
28      to 173.194.205.109 port 110 nat-to 10.10.10.10
29 pass out on em0 inet proto tcp from 192.168.0.0/24 \
30      to 173.194.205.109 port 25 nat-to 10.10.10.10
31 pass out on em0 inet proto udp from 10.0.0.0/16 to 8.8.8.8 port 53 \
32      nat-to 10.10.10.10
33 pass out on em0 inet proto udp from 172.16.0.0/24 to 8.8.8.8 port 53 \
34      nat-to 10.10.10.10
35 pass out on em0 inet proto udp from 192.168.0.0/24 to 8.8.8.8 port \
36      53 nat-to 10.10.10.10
37 pass out on em3 inet proto tcp from 10.0.0.0/16 to 172.16.0.10 port 3128
38 pass out on em3 inet proto tcp from 10.0.0.0/16 to 172.16.0.20 port 22
39 pass out on em3 inet proto tcp from 10.0.0.0/16 to 172.16.0.20 port 9418
40 pass out on em3 inet proto tcp from 192.168.0.0/24 to 172.16.0.10 port 3128
41 pass out on em3 inet proto tcp from any to 172.16.0.80 port 8080
```

Definição A.3: Oráculo das regras de *firewall* da política de segurança da instituição B – sintaxe do *packet filter* – tarefa de configuração.

```
1  block in all
2  pass in on em0 inet proto tcp from 131.100.210.78 \
3       to (em0) port 2002 rdr-to 172.16.0.22 port 22
4  pass in on em1 inet proto tcp from 10.0.0.0/16 to 172.16.0.10 port 3128
5  pass in on em1 inet proto tcp from 10.0.0.0/16 to 186.202.140.222 port 465
6  pass in on em1 inet proto tcp from 10.0.0.0/16 to 186.202.140.222 port 993
7  pass in on em1 inet proto tcp from 10.0.0.0/16 to 186.202.140.222 port 995
8  pass in on em1 inet proto tcp from 10.0.0.0/16 to 192.168.0.158 port 7777
9  pass in on em1 inet proto tcp from 10.0.0.0/16 to 192.168.0.158 port 7778
10 pass in on em1 inet proto tcp from 10.0.1.1 to any port 443
11 pass in on em1 inet proto tcp from 10.0.1.1 to any port 80
12 pass in on em1 inet proto tcp from 10.0.1.2 to any port 443
13 pass in on em1 inet proto tcp from 10.0.1.2 to any port 80
14 pass in on em1 inet proto tcp from 10.0.1.3 to 170.66.8.142 port 7081
15 pass in on em1 inet proto tcp from 10.0.1.4 to 170.66.8.142 port 7081
16 pass in on em1 inet proto tcp from 10.0.5.0/24 to 192.168.0.111 port 1521
17 pass in on em1 inet proto tcp from 10.0.5.0/24 to 192.168.0.111 port 5500
18 pass in on em1 inet proto udp from 10.0.0.0/16 to 208.67.220.220 port 53
19 pass in on em1 inet proto udp from 10.0.0.0/16 to 208.67.222.222 port 53
20 pass in on em3 inet proto icmp from 172.16.0.0/24 to any
21 pass in on em3 inet proto tcp from 172.16.0.10 to any port 443
22 pass in on em3 inet proto tcp from 172.16.0.10 to any port 80
23 pass in on em3 inet proto tcp from 172.16.0.22 to 192.168.0.0/24 port 22
24 pass in on em3 inet proto udp from 172.16.0.0/24 to 208.67.220.220 port 53
25 pass in on em3 inet proto udp from 172.16.0.0/24 to 208.67.222.222 port 53
26 pass out on em0 inet proto icmp from 172.16.0.0/24 to any \
27      nat-to (em0)
28 pass out on em0 inet proto tcp from 10.0.0.0/16 to 186.202.140.222 port 465 \
29      nat-to (em0)
30 pass out on em0 inet proto tcp from 10.0.0.0/16 to 186.202.140.222 port 993 \
31      nat-to (em0)
32 pass out on em0 inet proto tcp from 10.0.0.0/16 to 186.202.140.222 port 995 \
33      nat-to (em0)
34 pass out on em0 inet proto tcp from 10.0.1.1 to any port 443 \
35      nat-to (em0)
36 pass out on em0 inet proto tcp from 10.0.1.1 to any port 80 \
37      nat-to (em0)
38 pass out on em0 inet proto tcp from 10.0.1.2 to any port 443 \
39      nat-to (em0)
40 pass out on em0 inet proto tcp from 10.0.1.2 to any port 80 \
41      nat-to (em0)
42 pass out on em0 inet proto tcp from 10.0.1.3 to 170.66.8.142 port 7081 \
43      nat-to (em0)
44 pass out on em0 inet proto tcp from 10.0.1.4 to 170.66.8.142 port 7081 \
45      nat-to (em0)
46 pass out on em0 inet proto tcp from 172.16.0.10 to any port 443 \
47      nat-to (em0)
48 pass out on em0 inet proto tcp from 172.16.0.10 to any port 80 \
49      nat-to (em0)
50 pass out on em0 inet proto udp from 10.0.0.0/16 to 208.67.220.220 port 53 \
51      nat-to (em0)
52 pass out on em0 inet proto udp from 10.0.0.0/16 to 208.67.222.222 port 53 \
53      nat-to (em0)
54 pass out on em0 inet proto udp from 172.16.0.0/24 to 208.67.220.220 port 53 \
55      nat-to (em0)
56 pass out on em0 inet proto udp from 172.16.0.0/24 to 208.67.222.222 port 53 \
57      nat-to (em0)
58 pass out on em2 inet proto tcp from 10.0.0.0/16 to 192.168.0.158 port 7777
59 pass out on em2 inet proto tcp from 10.0.0.0/16 to 192.168.0.158 port 7778
60 pass out on em2 inet proto tcp from 10.0.5.0/24 to 192.168.0.111 port 1521
61 pass out on em2 inet proto tcp from 10.0.5.0/24 to 192.168.0.111 port 5500
62 pass out on em2 inet proto tcp from 172.16.0.22 to 192.168.0.0/24 port 22
63 pass out on em3 inet proto tcp from 10.0.0.0/16 to 172.16.0.10 port 3128
64 pass out on em3 inet proto tcp from 131.100.210.78 to 172.16.0.22 port 22
```

Definição A.4: Oráculo das regras de *firewall* da política de segurança da instituição B – sintaxe do *packet filter* – para a tarefa de manutenção.

```
1  block in all
2  pass in on em0 inet proto {tcp} from {131.100.210.78} \
3       to (em0) port 2002 rdr-to 172.16.0.22 port 22
4  pass in on em1 inet proto {tcp} from {10.0.0.0/16} to 172.16.0.10 port 3128
5  pass in on em1 inet proto {tcp} from {10.0.0.0/16} to 186.202.140.222 port 465
6  pass in on em1 inet proto {tcp} from {10.0.0.0/16} to 186.202.140.222 port 993
7  pass in on em1 inet proto {tcp} from {10.0.0.0/16} to 186.202.140.222 port 995
8  pass in on em1 inet proto {tcp} from {10.0.0.0/16} to 192.168.0.158 port 7777
9  pass in on em1 inet proto {tcp} from {10.0.0.0/16} to 192.168.0.158 port 7778
10 pass in on em1 inet proto {tcp} from {10.0.1.1,10.0.1.2} to any port 443
11 pass in on em1 inet proto {tcp} from {10.0.1.1,10.0.1.2} to any port 80
12 pass in on em1 inet proto {tcp} from {10.0.1.3,10.0.1.4} to 170.66.8.142 port 7081
13 pass in on em1 inet proto {tcp} from {10.0.2.1,10.0.2.2} to 200.211.20.5 port 3006
14 pass in on em1 inet proto {tcp} from {10.0.2.1,10.0.2.2} to 200.211.20.5 port 3007
15 pass in on em1 inet proto {tcp} from {10.0.3.1,10.0.3.2} to 191.242.201.6 port 8081
16 pass in on em1 inet proto {tcp} from {10.0.3.1,10.0.3.2} to 191.242.201.6 port 8083
17 pass in on em1 inet proto {tcp} from {10.0.3.1,10.0.3.2} to 191.242.201.6 port 8088
18 pass in on em1 inet proto {tcp} from {10.0.3.3,10.0.3.4} to 191.6.194.125 port 8090
19 pass in on em1 inet proto {tcp} from {10.0.5.0/24} to 192.168.0.111 port 1521
20 pass in on em1 inet proto {tcp} from {10.0.5.0/24} to 192.168.0.111 port 5500
21 pass in on em1 inet proto {udp} from {10.0.0.0/16} \
22      to {208.67.222.222,208.67.220.220} port 53
23 pass in on em3 inet proto {icmp} from {172.16.0.0/24} to any
24 pass in on em3 inet proto {tcp} from {172.16.0.22} to 192.168.0.0/24 port 22
25 pass in on em3 inet proto {tcp} from {172.16.0.10} to any port 443
26 pass in on em3 inet proto {tcp} from {172.16.0.10} to any port 80
27 pass in on em3 inet proto {udp} from {172.16.0.0/24} \
28      to {208.67.222.222,208.67.220.220} port 53
29 pass out on em0 inet proto tcp from {10.0.2.1,10.0.2.2} to 200.211.20.5 port 3006 \
30      nat-to (em0)
31 pass out on em0 inet proto tcp from {10.0.2.1,10.0.2.2} to 200.211.20.5 port 3007 \
32      nat-to (em0)
33 pass out on em0 inet proto tcp from {10.0.3.1,10.0.3.2} \
34      to 191.242.201.6 port 8081 nat-to (em0)
35 pass out on em0 inet proto tcp from {10.0.3.1,10.0.3.2} \
36      to 191.242.201.6 port 8083 nat-to (em0)
37 pass out on em0 inet proto tcp from {10.0.3.1,10.0.3.2} \
38      to 191.242.201.6 port 8088 nat-to (em0)
39 pass out on em0 inet proto tcp from {10.0.3.3,10.0.3.4} \
40      to 191.6.194.125 port 8090 nat-to (em0)
41 pass out on em0 inet proto {icmp} from {172.16.0.0/24} to any nat-to (em0)
42 pass out on em0 inet proto {tcp} from {10.0.0.0/16} to 186.202.140.222 port 465 \
43      nat-to (em0)
44 pass out on em0 inet proto {tcp} from {10.0.0.0/16} to 186.202.140.222 port 993 \
45      nat-to (em0)
46 pass out on em0 inet proto {tcp} from {10.0.0.0/16} to 186.202.140.222 port 995 \
47      nat-to (em0)
48 pass out on em0 inet proto {tcp} from {10.0.1.1,10.0.1.2} to any port 443 \
49      nat-to (em0)
50 pass out on em0 inet proto {tcp} from {10.0.1.1,10.0.1.2} to any port 80 nat-to (em0)
51 pass out on em0 inet proto {tcp} from {10.0.1.3,10.0.1.4} to 170.66.8.142 port 7081
52 pass out on em0 inet proto {tcp} from {172.16.0.10} to any port 443 nat-to (em0)
53 pass out on em0 inet proto {tcp} from {172.16.0.10} to any port 80 nat-to (em0)
54 pass out on em0 inet proto {udp} from {10.0.0.0/16} \
55      to {208.67.222.222,208.67.220.220} port 53 nat-to (em0)
56 pass out on em0 inet proto {udp} from {172.16.0.0/24} \
57      to {208.67.222.222,208.67.220.220} port 53 nat-to (em0)
58 pass out on em2 inet proto {tcp} from {172.16.0.22} to 192.168.0.0/24 port 22
59 pass out on em2 inet proto {tcp} from {10.0.0.0/16} to 192.168.0.158 port 7777
60 pass out on em2 inet proto {tcp} from {10.0.0.0/16} to 192.168.0.158 port 7778
61 pass out on em2 inet proto {tcp} from {10.0.5.0/24} to 192.168.0.111 port 1521
62 pass out on em2 inet proto {tcp} from {10.0.5.0/24} to 192.168.0.111 port 5500
63 pass out on em3 inet proto {tcp} from {10.0.0.0/16} to 172.16.0.10 port 3128
64 pass out on em3 inet proto {tcp} from {131.100.210.78} to 172.16.0.22 port 22
```