Representação e Aplicação de Políticas de Segurança em Firewalls de Redes Híbridas

Maurício Fiorenza[1], Diego Kreutz[1], Rodrigo Mansilha[1], Douglas D. J. de Macedo[2], Eduardo Feitosa[3], Roger Immich[4]

[1]Universidade Federal do Pampa (UNIPAMPA)

[2]Universidade Federal de Santa Catarina (UFSC)

[3]Universidade Federal do Amazonas (UFAM)

[4]Universidade Federal do Rio Grande do Norte (UFRN)

Resumo. *O gerenciamento de políticas de segurança em firewalls de redes híbridas é um processo desafiador, principalmente devido a diversidade de soluções e fabricantes (*e.g.*, Cisco NGFW, Check Point, Fortigate, IPTables), cada um com suas linguagens, interfaces e modelos de operação. Neste trabalho é proposta uma linguagem genérica para representação de políticas de segurança utilizadas em firewalls, denominada FWlang. A linguagem foi especificada para representar os seis tipos de políticas de firewalls modernos, incluindo ACL, NAT 1to1, NAT Nto1, traffic shapping, roteamento estático e filtros de URL, e implementada e incorporada à solução de gerenciamento de firewalls FWunify. A avaliação demonstra o potencial de simplificação apresentado pela linguagem, chegando a uma redução de 72% no número de termos necessários para aplicar um determinado grupo de políticas a três firewall diferentes.*

1. Introdução

Uma rede híbrida pode incorporar diversas soluções de firewall, tanto tradicionais (*e.g.*, Cisco NGFW, Palo Alto, Barracuda, Sophos, pfSense, IPTables), como SDN (*i.e.*, baseadas em OpenFlow, P4). Contudo, o uso combinado desses paradigmas acrescenta desafios ao gerenciamento de firewalls. Um administrador, para aplicar uma política de segurança, necessitará especificar e executar os comandos relativos à política, utilizando a sintaxe específica de cada tipo de firewall instalado na rede (exemplos práticos, detalhados, podem ser vistos no Apêndice A da versão estendida [Fiorenza et al., 2021]). Essa prática pode levar a erros de configuração, que comprometam a segurança das informações trafegadas ou causem a indisponibilidade de serviços na rede. Para além disto, há um aumento na complexidade no que diz respeito ao conhecimento sobre as múltiplas tecnologias de firewall, levando a necessidade de equipes mais técnicas e, consequentemente, mais caras. Estudos recentes demonstram que falhas no processo de configuração de equipamentos ou serviços foram responsáveis por 86% das informações vazadas em 2019 [IBM X-Force, 2020] e preveem que, até 2023, 99% das violações em firewalls serão causadas por configurações incorretas e não por falha das soluções [Gartner, 2019].

Com o objetivo de mitigar estes problemas, são necessárias linguagens genéricas e extensíveis para a representação de políticas de segurança. Algumas linguagens são dedicadas a famílias específicas de equipamentos ou versões de software, como é o caso da *RichLanguage* [Firewalld, 2021], criada para facilitar o gerenciamento de firewalls em ambientes Linux, e a LAI [Tian et al., 2019], para o gerenciamento de regras do tipo ACL (*Access Control List*) nos ambientes de WAN da Alibaba. Outras linguagens são aplicáveis em uma gama mais ampla de firewalls [Bodei et al., 2018, Pozo et al., 2009,

Zhang et al., 2007], mas têm seu escopo limitado a representação de regras do tipo ACL ou NAT (*Network Address Translation*). É importante observar que as linguagens podem variar significativamente entre versões de uma mesma linha/fabricante de firewall. Esse é o caso dos *firmwares* 8.2 e 8.4 do firewall Cisco ASA, onde a composição de uma regra de NAT muda totalmente de uma versão para outra.

Neste trabalho propomos a FWlang: uma linguagem genérica, abstrata e extensível para gerenciar políticas de segurança de firewalls de redes híbridas. Uma linguagem genérica reduz o número de termos necessários para gerenciar um conjunto de múltiplos firewalls em redes híbridas, que podem incluir variedades de ambiente de rede, fabricante, linha e versão. Uma linguagem em nível adequado de abstração tende a facilitar aprendizagem e evitar erros de configuração. Uma linguagem extensível permite representar políticas utilizadas em firewalls modernos, como *traffic shaping*, filtros de URL ou regras de roteamento. Acompanhando avanços recentes na área de gerência, a FWlang é baseada no conceito de redes baseadas em intenções ou IBNs (*Intent-Based Networking*) [Clemm et al., 2019, Zeydan and Turk, 2020].

As contribuições deste trabalho são: (a) levantamento de requisitos e especificação de uma linguagem genérica para representar políticas de segurança de firewalls modernos; (b) concretização de uma instância da linguagem FWlang para o contexto de IBNs, utilizando intenções como forma de representação estruturada de políticas de segurança, e integração com recursos da solução FWunify para aplicação automática das políticas nos firewalls da rede; e (c) avaliação empírica da linguagem em um ambiente híbrido composto por firewalls Cisco ASA, IPTables e OpenFlow. Na avaliação, demonstramos o impacto na redução do número de termos necessários para representar políticas de firewalls utilizando *datasets* de ambientes reais. Adicionalmente, demonstramos o funcionamento prático da automação e corretude da aplicação das regras em ambientes reais.

O restante deste trabalho está estruturado da seguinte forma. Na Seção 2 são apresentadas a especificação e a gramática da FWlang. Os resultados das avaliações e os trabalhos relacionados são discutidos nas Seções 3 e 4, respectivamente. Por fim, na Seção 5 são apresentadas as considerações finais.

2. Linguagem para Representação de Políticas de Firewalls

2.1. Especificação

Intent-Based Networking, ou IBN, é um conceito que tem sido explorado recentemente pela indústria [Cisco Systems, 2017, VMware, 2020] e pela academia [Singh et al., 2020, Zeydan and Turk, 2020, Sanvito et al., 2018]. O principal objetivo de uma IBN é agregar novos níveis de automação e inteligência no gerenciamento das redes através de representações de alto nível de abstração denominadas intenções.

Em consonância com os avanços recentes no contexto de gerenciamento de redes baseadas em intenções, propomos a FWlang, uma linguagem estruturada de representação de intenções, voltada a ampliar a manutenibilidade e a confiabilidade no gerenciamento de firewalls em redes híbridas. Através da FWlang, usuários com os mais variados graus de instrução – desde pouco conhecimento técnico específico (*e.g.*, administrador de sistemas que não atua na configuração de firewalls) até experiência prática com administração de firewalls – devem ser capazes de definir e aplicar políticas de segurança, sem precisar acessar e conhecer as sintaxes e especificidades de quaisquer tipos de firewalls.

O processo de construção e formalização da FWlang foi iniciado através da definição de um conjunto de políticas consideradas fundamentais. Para determinar esse conjunto, foi realizado um estudo englobando: (a) políticas implementadas em quatro ambientes reais, (b) documentação de fabricantes de equipamentos de segurança, e (c) literatura científica [Scarfone and Hoffman, 2009, Krit and Haimoud, 2017, Bodei et al., 2018]. Os ambientes reais[1] são compostos por dois firewalls de borda responsáveis pela segurança de redes de várias centenas de máquinas, um firewall interno, que protege uma rede de filiais da instituição, e um firewall de um servidor Web, que é mantido em um centro de dados terceirizado. As documentações oficiais consultadas incluem materiais publicados por algumas das principais fabricantes de firewalls modernos, como Cisco, Palo Alto, Check Point e Fortinet.

A Tabela 1 apresenta um resumo das principais políticas utilizadas pelos firewalls analisados no mapeamento realizado. Podemos observar que as soluções de mercado consolidadas, como Cisco, Palo Alto, Fortinet e Check Point, possuem em comum as políticas dos tipos ACL, filtro de URL, *traffic shaping*, roteamento estático, NAT 1to1 e NAT Nto1. Algumas soluções, como a Palo Alto, possuem também recursos para expressar políticas mais específicas, como *Decryption*[2] e *DoS Protection*[3]. Como essas políticas não são comuns às diferentes soluções de firewall, elas foram excluídas do conjunto inicial.

Tabela 1. Políticas suportadas por firewalls modernos

Solução	ACL	Filtro de URL	Traffic Shaping	Roteamento estático	NAT 1to1	NAT Nto1
Cisco NGFW	✓	✓	✓	✓	✓	✓
Palo Alto NGFW	✓	✓	✓	✓	✓	✓
Fortinet NGFW	✓	✓	✓	✓	✓	✓
Check Point NGFW	✓	✓	✓	✓	✓	✓
PFSense	✓		✓	✓	✓	✓
IPTables, Mignis	✓				✓	✓
FlowTracker, P4GUARD	✓					

Soluções livres, como o pfSense [pfSense, 2020], suportam cinco das seis regras suportadas por firewalls comerciais. A política encontrada com menor frequência em soluções livres é a de filtro de URL, que é tipicamente atribuída a outras soluções livres da camada de aplicação, como o Squid[4]. A maioria das demais soluções, como o IPTables [Netfilter, 2021], Mignis [Adão et al., 2014], SDN/Fireflow [Fiessler et al., 2018], SDN/FlowTracker [Vinh Tran and Ahn, 2016] e SDN/P4GUARD [Datta et al., 2018] limitam-se a políticas dos tipos ACL e NAT.

Como resultado do estudo, selecionamos seis políticas de firewall para especificar a primeira versão da FWlang. Essas políticas são resumidas na Tabela 2. Por exemplo, uma política do tipo ACL tem seus requisitos divididos em dois grupos. O primeiro inclui os termos que são obrigatórios (destacados em itálico na tabela) para descrição da política, como origem e destino dos pacotes, tipo de tráfego a ser tratado, a ação a ser tomada (bloquear/permitir), e a ordem de prioridade de execução da regra no firewall. O segundo grupo inclui campos opcionais oferecidos por algumas das soluções de firewall

[1]Os detalhes são omitidos para preservar as instituições e empresas.
[2]`https://tinyurl.com/palo-alto-decryption`
[3]`https://tinyurl.com/palo-alto-dos`
[4]`http://www.squid-cache.org`

analisadas, como ativação do serviço de registro de eventos (*logs*), definição do intervalo de tempo que a regra ficará ativa e descrição.

Tabela 2. Mapeamento de políticas de firewall consideradas na elaboração da FWlang

Política	ACL	Filtro de URL	Traffic Shaping	Roteamento estático	NAT 1to1	NAT Nto1
Requisitos	*Origem* *Destino* *Tráfego* *Ação* *Prioridade* Serviço de logs Intervalo de tempo Descrição	*Origem* *Destino* *Tráfego* *Ação/Largura de banda* *Prioridade* Serviço de logs Intervalo de tempo Descrição	*Origem* *Destino* *Tráfego* *Largura de banda* *Prioridade* Serviço de logs Intervalo de tempo Descrição	*Origem/Destino* *Gateway* Intervalo de tempo Descrição	*Origem* *Destino* *Tráfego* Serviço de logs Descrição	*Origem* *Destino* Serviço de logs Descrição

Para definir a gramática da FWlang, consideramos usar como ponto de partida as linguagens LAI [Tian et al., 2019] e NILE [Jacobs et al., 2018]. Optamos por seguir a linguagem NILE baseados nos seguintes aspectos. A LAI atende os requisitos de uma solução específica para o gerenciamento de ACLs simples (apenas marcadores como origem, destino e ação) em roteadores de uma WAN privada e incorpora essencialmente termos e sintaxe específica de ACLs de roteadores (*e.g*, *scope*, *isolate*, *fix*). A NILE, por sua vez, foi projetada para atender requisitos comuns de uma rede, como políticas de ACL e *traffic shaping* em dispositivos de rede que implementam e executam as funções virtuais da rede. A linguagem foi concebida para ser uma representação intermediária entre a descrição de uma intenção em linguagem natural e os comandos a serem aplicados na rede para o gerenciamento de NFVs. Em síntese, a NILE utiliza termos que consideramos intuitivos, corriqueiros e simples (*e.g.*, *from*, *to*, *block/allow*, *throughput*) para administradores de sistemas expressarem políticas utilizadas em redes corporativas.

A gramática da FWlang é apresentada na Figura 1 em notação EBNF (*extended Backus–Naur form*). A FWlang adota da NILE operadores básicos como *from* e *to*, para definir origem e destino, ou marcadores para definir se um tráfego é permitido ou bloqueado (*allow/block*). Para representar políticas de segurança de firewalls modernos, a FWlang estende a NILE adicionando ACLs, NATs, regras de roteamento, *traffic shaping*[5] e filtros de URL, que exigem operadores adicionais para expressar requisitos como prioridade, categorias de tráfegos, *gateways* ou controle do serviço de *logs*. É importante observar que a gramática proposta pode ser estendida futuramente caso necessário (*e.g.*, incorporar outros tipos de políticas de firewalls ou de outros *appliances* de segurança).

A Figura 2 apresenta três exemplos de intenções de segurança de firewalls escritas com a FWlang. O primeiro exemplo (Figura 2(a)) descreve uma política do tipo ACL identificada como "deny-netA-h100-http", que bloqueia o tráfego HTTP (*block traffic('http')*) da rede de origem 10.0.0.0/24 para o host 200.19.0.100 (*to endpoint('200.19.0.100')*). É importante observar que a regra será adicionada (*add*) aos firewalls indicados nos marcadores *firewall* antes de todas as demais regras existentes (*order before('all')*).

O segundo exemplo (Figura 2(b)) define uma política do tipo *traffic shaping*, que limita a largura de banda para o tráfego udp/5555 (*for traffic('udp/5555')*) da rede 10.0.0./24, para o host 200.19.0.100, a 100 Mbps (*with throughput('100Mbps')*). Nesse

[5]Estende os recursos de *traffic shaping* da linguagem NILE.

Figura 1. Gramática FWlang no formato EBNF

```
<intent>        ::= 'define intent' intent_name ':' <commands>
<commands>      ::= <command> {'\n' <command> }
<command>       ::= (<name> | <locations> | <rules> | <targets>
                | <qos> | <middleboxes> | <order>) + [ <optional> ]
<name>          ::= 'name' <text>
<locations>     ::= 'from' <object> 'to' <object>
<object>        ::= 'endpoint('value | all')' | 'range('value')'
                | 'category('value | all')'  | 'gateway('value')'
<rules>         ::= (allow | block) <traffic>
<targets>       ::= 'for' <traffic>
<qos>           ::= 'with' <metrics>
<metrics>       ::= 'throughput('value')'
<middleboxes>   ::= (add | del) <middlebox> {(',') <middlebox>}
<middlebox>     ::= 'middlebox('mid_id')' | 'firewall('fw_id')'
                | middleboxes('mid_id' (',') 'mid_id') | firewalls('fw_id' (',') 'fw_id')
<order>         ::= 'order' <position>
<position>      ::= 'before('rule_name' | 'all ')' | 'after('rule_name' | 'all')'
<traffic>       ::= 'traffic('value' | 'all')' | port('[<tuple>]')
<tuple>         ::= 'protocol:' value | 'src_port:' value | 'dst_port:' value
<optional>      ::= <interval> | <log> | <description>
<interval>      ::= 'start' <data_time> '\n' 'end' <data_time>
<data_time>     ::= 'datetime('value')' | 'date('value')  | 'hour('value')'
<log>           ::= logging <options>
<options>       ::= (enable | disable)
<description>   ::= 'description' <text>
<text>          ::= 'text('rule_name' | 'rule_description')'
```

Figura 2. Exemplos de intenções de firewall escritas com a FWlang

```
define intent acl:
name        text('deny-netA-h100-http')
from        range('10.0.0.0/24')
to          endpoint('200.19.0.100')
block       traffic('http')
order       before('all')
add/del     firewall('cisco-1'),firewall('iptables-1'),firewall('openflow-1')
```

(a) *Intenção Access List*

```
define intent traffic_shaping:
name        text('limit-net-h100-100m')
from        range('10.0.0.0/24')
to          endpoint('200.19.0.100')
order       before('all')
for         traffic('udp/5555')
with        throughput('100Mbps')
add/del     firewall('openflow-1')
```

(b) *Intenção Traffic Shaping*

```
define intent url_filter:
name        text('filter-social-media')
from        range('10.0.0.0/24')
to          category('social-media')
block       traffic('all')
order       after('filter-phishing')
logging     enable
add         firewall('checkpoint-1')
```

(c) *Intenção Filtro de URL*

caso, a ordem indica que a nova regra será adicionada antes de todas as outras políticas (*order before('all')*) de *traffic shaping* presentes no firewall "openflow-1".

O terceiro exemplo (Figura 2(c)) é um filtro de URL. Ele estabelece que todo tráfego gerado pela rede 10.0.0.0/24, que passa pelo firewall "checkpoint-1", com destino a categoria de tráfego "social-media", deve ser bloqueado (*block traffic('all')*). Essa política deve ser adicionada após a regra pré-existente "filter-phishing" e ter o serviço de *logs* habilitado (*logging enable*).

2.2. Implementação

A FWlang estende a linguagem NILE com o objetivo de representar políticas utilizadas em firewalls convencionais e de nova geração. Para concretizar uma instância da FWlang, foram adicionados diversos novos marcadores para atender os requisitos mapeados na Tabela 2. Por exemplo, os indicadores de origem e destino do tráfego *from/to* ganharam marcadores auxiliares para representar sub-redes (*range('10.0.0.0/24')*), categorias de tráfego (*category('social-media')*) ou ainda a representação de *gateways* (*gateway('10.0.1.1')*) em políticas de roteamento.

Conciliar o gerenciamento de diversas soluções de firewall em uma linguagem unificada sucinta desafios técnicos, como priorização de intenções. Para adicionar ordem de prioridade às intenções de ACL, *traffic shaping* e filtros de URL, foi utilizado o conceito de antes e depois (*before/after*), presente na inicialização de serviços de sistemas GNU/Linux. Para conseguir estabelecer ordem de prioridade, foram adicionados dois novos marcadores, o nome (*name text('policy name')*) e a ordem (*order before('policy name')*). O nome permite referenciar as intenções para estabelecer uma ordem de prioridade, enquanto que o marcador ordem (antes ou depois) irá determinar a posição relativa ou absoluta das regras geradas para os firewalls. No caso do marcador de ordem, há também o valor padrão *all*, que pode ser utilizado para inserir uma regra antes ou após todas as existentes.

No caso de firewalls, são necessários os marcadores de adicionar (*add*) e remover (*del*) para definir se a regra representada pela intenção deve ser adicionada ou removida. O firewall, ou a lista de firewalls, ao qual a política será aplicada/removida é identificado pelos marcadores *middlebox*, *firewall*, *middleboxes* ou *firewalls* (e.g., *add firewalls('cisco-1','iptables-1')*. Apesar dos termos *firewall* e *firewalls* serem potencialmente mais familiares aos administradores de sistemas, os termos *middlebox* e *middleboxes* foram mantidos na linguagem por serem mais gerais, isto é, representarem quaisquer tipos de *appliances* de redes, como firewalls, produtos específicos para inspeção de tráfego SSL, entre outros. Apesar do foco da FWlang ser firewalls, nada impede que a linguagem seja estendida para outros tipos de *appliances* no futuro.

Um protótipo funcional da FWlang foi desenvolvido e disponibilizado publicamente em conjunto com a FWunify[6]. A arquitetura da FWunify permite o gerenciamento integrado de variadas soluções de firewalls em ambientes de rede híbridos [Fiorenza et al., 2020]. Atualmente, ela incorpora as características necessárias para o gerenciamento de firewalls Cisco ASA 5520, IPTables e baseados em OpenFlow (*e.g.*, Open vSwitch).

[6]`https://github.com/mmfiorenza/fwunify`

Detalhes adicionais das políticas (resumidas na Tabela 2), da gramática em notação EBNF e da implementação da FWunify (*e.g.*, linguagem, módulos e bibliotecas) podem ser encontrados nos Apêndices B, C e D da versão estendida [Fiorenza et al., 2021], respectivamente.

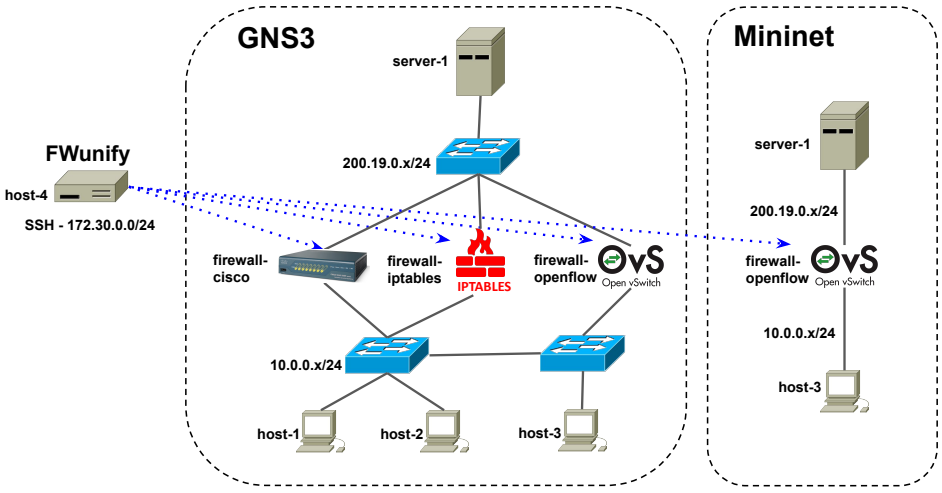
3. Avaliação

Na avaliação da FWlang, foram analisados os seguintes aspectos: (a) redução de complexidade na representação de políticas reais de segurança (Seção 3.2); (b) intuitividade da FWlang (Seção 3.3); (c) corretude do processo de tradução das intenções em regras específicas dos firewalls (Seção 3.4); e (d) eficácia das regras geradas e aplicadas nos firewalls (Seção 3.5). Detalhes complementares das avaliações, como dados do questionário aplicado (Seção 3.3) e relatos técnicos da aplicação das regras nos diferentes tipos de firewalls (Seções 3.4 e 3.5), podem ser encontrados na versão estendida do trabalho [Fiorenza et al., 2021]).

3.1. Ambientes de testes

Os testes da aplicação das intenções representadas na FWlang foram realizados em dois ambientes, conforme ilustrado na Figura 3. O primeiro ambiente, baseado no GNS3[7], é composto por: (a) três firewalls – Cisco ASA-5520 (firewall-cisco), IPTables (firewall-iptables) e Open vSwitch (firewall-openflow) – posicionados entre as redes interna (10.0.0.0/24) e externa (200.19.0.0/24) do cenário; (b) um servidor Ubuntu Server 18.04 (server-1), conectado logicamente as interfaces externas dos firewalls; (c) três hosts executando Lubuntu 16.04 (host-1, host-2 e host-3), que utilizam, respectivamente, os firewalls Cisco, IPTables e OpenFlow como *gateway* principal; e (d) um host Ubuntu 18.04 (host-4) executando o FWunify e conectado aos firewalls por uma interface específica para gerenciamento.

Figura 3. Ambientes de testes



O GNS3 possui uma limitação para testes de desempenho, mais especificamente com relação à largura de banda[8]. As conexões disponibilizadas pelo GNS3 são limitadas a

[7] https://www.gns3.com/
[8] https://tinyurl.com/gns3-limitation

larguras de banda baixas (de 1 a 4 Mbps dependendo do sistema hospedeiro). Além disso, segundo resultados empíricos externos e próprios, a vazão oscila de forma imprevisível, inviabilizando testes de desempenho mais confiáveis. Consequentemente, para realizar uma demonstração de uma intenção de *traffic shaping* na prática, foi criado um segundo ambiente de testes, utilizando o Mininet[9]. O ambiente é composto pelo *host-4, host-3*, firewall-openflow e *server-1*, conforme ilustrado na Figura 3.

3.2. Redução de Complexidade

Considerando o contexto da diversidade de sintaxe, a inerente complexidade e propensão a erros humanos, como apontado por relatórios técnicos e estudos especializados [Gartner, 2019, IBM X-Force, 2020], linguagens genéricas baseadas em intenções de nível mais abstrato (*i.e.*, sem precisar conhecer detalhes técnicos da rede e das sintaxes específicas de diferentes firewalls), como a FWlang, representam uma forma de simplificar a representação de políticas de segurança e automatizar o processo de tradução e aplicação das regras nos firewalls. Para demonstrar o potencial de simplificação e redução da propensão a erros foram analisados conjuntos de regras em ambiente de produção de quatro instituições reais, compostos por: (a) 22 políticas ACL de um firewall interno que controla o tráfego entre filiais (Sophos); (b) 35 políticas ACL de um firewall protegendo um servidor Web em um *data center* compartilhado (IPTables); (c) 38 políticas ACL de um firewall de borda (Cisco); e (d) 47 políticas de um segundo firewall de borda, sendo 8 ACLs e 39 regras de NAT 1to1 (MikroTik).

Para efeitos de comparação, consideramos o número de termos fixos (da sintaxe) necessários para a descrição e aplicação das políticas em três firewalls diferentes (Cisco, IPTables e OpenFlow), utilizando a FWlang e as sintaxes específicas de cada solução. No gráfico da Figura 4 podemos observar uma redução no número de termos utilizados para aplicar as políticas através de intenções da FWlang. A comparação leva em consideração os comandos necessários para aplicação manual das regras nos firewalls tradicionais.

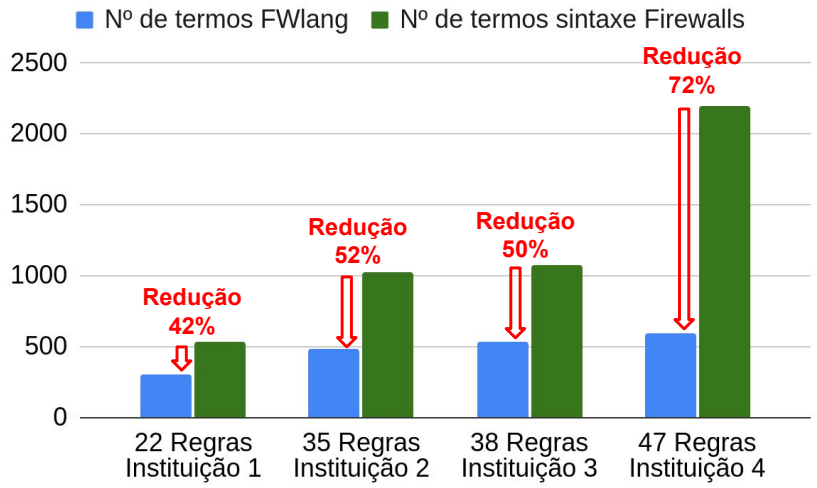


Figura 4. Número de termos em diferentes sintaxes

[9]http://mininet.org/

A redução no número de termos para os quatro conjuntos de regras foi de 42%, 52%, 50% e 72%. A redução mais significativa no quarto conjunto de regras é resultado do número elevado de termos utilizados para descrição das políticas de NAT. É importante observar também que o número de termos cresce exponencialmente à medida que novas soluções de firewall são adicionadas ao ambiente. Os números indicam que a FWlang simplifica a representação de políticas de firewalls. Quanto mais diversificado for o conjunto de soluções de firewall a serem gerenciadas, maior será a simplificação da gestão, uma vez que o administrador dos sistemas precisa compreender e representar as intenções de segurança uma única vez utilizando a sintaxe da FWlang.

3.3. Intuitividade da FWlang

Para avaliar aspectos subjetivos da linguagem FWlang, criamos um questionário online[10] e o enviamos para para profissionais com experiência em administração de sistemas e/ou firewalls. O questionário contém questões que visam avaliar a intuitividade[11] e complexidade[12] da linguagem quando comparada com exemplos utilizando sintaxes específicas de firewalls atuais.

Seguindo o exemplo de pesquisa recente que avalia a utilização de linguagens baseadas em intenções pelos usuários [Scheid et al., 2020], utilizamos três políticas de segurança (Filtro de URL, *Traffic Shaping* e ACL) no questionário online para guiar a avaliação dos usuários. Todas as políticas são apresentadas em FWlang e na sintaxe específica dos firewalls. O questionário também apresenta questões sobre o nível de experiência do respondente.

A pesquisa contou com um total de 22 participantes, dos quais 100% são administradores de sistemas, e 57% possuem experiência prática com administração de firewalls. Mais de 85% dos respondentes consideram a representação das políticas utilizando FWlang significativamente mais intuitiva e menos complexa quando comparada com as sintaxes utilizadas pelos firewalls. Os demais (15% – respondentes que possuem experiência prática com firewalls) consideram a FWlang equivalente a outras sintaxes para alguns casos particulares. Por exemplo, filtros de URLs são regras simples de expressar em soluções tradicionais, o que levou alguns dos participantes a considerar a sintaxe do firewall Palo Alto tão simples quanto a sintaxe da FWlang. Em resumo, os resultados positivos com relação a FWlang podem ser observados nos dois tipos de respondentes, isto é, os administradores de sistemas sem (subgrupo A) e com (subgrupo B) experiência prática de gerenciamento de firewalls. Os resultados sugerem que a FWlang é mais intuitiva e menos complexa para grupos de usuários com diferentes níveis de conhecimento no que diz respeito ao gerenciamento de firewalls.

3.4. Avaliação do processo de tradução

A avaliação da acurácia do processo tradução das intenções pode ser realizada através da especificação, tradução automática e consequente verificação manual das regras (ou

[10] `https://forms.gle/iSQt27j26Xcnp2hT6`

[11] Intuitividade considerando aspectos como facilidade de interpretação de todos os termos, nível de conhecimento prévio exigido para interpretação e facilidade para memorização.

[12] Complexidade considerando aspectos como quantidade de termos utilizados, verbosidade e legibilidade.

instruções de baixo nível) geradas, análogo ao realizado em trabalhos existentes na literatura [Jacobs et al., 2018, Zhang et al., 2007]. Para avaliar a acurácia da tradução utilizando a FWlang e a solução FWunify foram definidas três tipos de políticas de segurança. A primeira, do tipo ACL, realiza o bloqueio do tráfego ICMP entre as redes 10.0.0.0/24 e 200.19.0.0/24. A segunda remove uma regra NAT 1to1, que realiza o redirecionamento do tráfego recebido na porta 80 do IP (externo) 200.19.0.50 para a porta 90 do IP (interno) 10.0.0.50. Por fim, uma política de *traffic shaping* é adicionada aos firewalls para limitar a 30 Mbps o tráfego FTP entre a rede 10.0.0.0/24 e o IP 200.19.0.100.

Os comandos resultantes do processo de tradução via FWlang e FWunify foram comparados com o resultado esperado coletado em documentações oficiais das soluções de firewall, demonstrando que atendem rigorosamente a sintaxe exigida para cada tipo de firewall. Adicionalmente, é importante ressaltar que os comandos gerados foram todos testados e validados manualmente nos respectivos firewalls. Os resultados detalhados das traduções das três intenções em FWlang estão disponíveis na versão estendida [Fiorenza et al., 2021]

3.5. Avaliação da eficácia das políticas aplicadas

A avaliação da eficácia das instruções de baixo nível, geradas a partir de uma descrição de mais alto nível, como as intenções em IBNs, pode ser realizada através da tradução, aplicação automática e avaliação das políticas utilizando cenários e ferramentas externas, similar ao realizado em trabalhos existentes na literatura [Riftadi and Kuipers, 2019, Soule et al., 2018]. Para realizar a avaliação da eficácia da tradução e aplicação das intenções de segurança em FWlang foram especificadas duas políticas, uma do tipo ACL e outra do tipo *traffic shaping*.

A ACL utilizada, que pode ser vista na Figura 2(a), interrompe o tráfego HTTP (porta tcp/80) entre as máquinas *host-1*, *host-2* e *host-3* e o servidor *server-1* do ambiente de testes descrito na Seção 3.1. A intenção foi traduzida e enviada aos três firewalls (Cisco, IPTables e OpenFlow) através da solução FWunify. Após aplicar a ACL, para testar a conectividade na porta tcp/80 do *server-1*, foi executado o um *shell script* Bash (*port-response.sh*[6]) que utiliza o comando *netcat*[13] do Linux. O *script* é programado para realizar uma tentativa de conexão a cada 5 segundos.

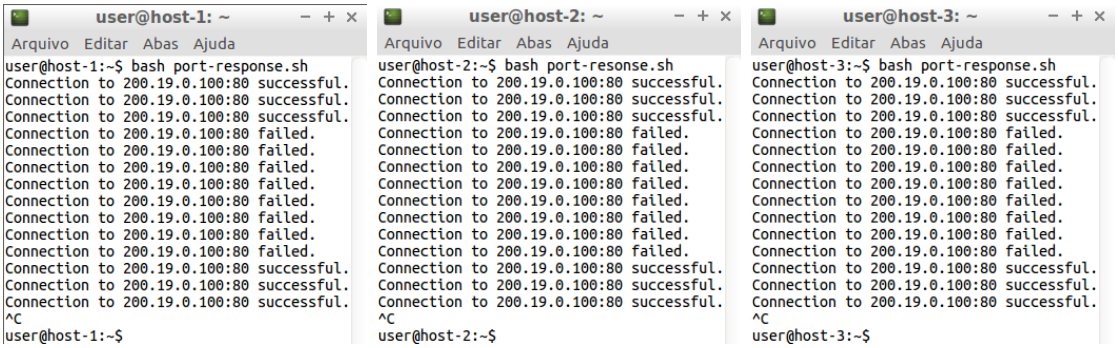
As Figuras 5(a), 5(b) e 5(c) apresentam os resultados da execução do *script* em cada um dos hosts, antes da aplicação da política, durante o período em que a política estava ativa e após a remoção da regra dos respectivos firewalls. Como pode ser verificado, durante o período de vigência da política, nenhum dos hosts conseguiu realizar a conexão com o servidor.

O segundo teste especificou e aplicou a intenção de *traffic shaping* da Figura 2(b). Essa intenção limita o tráfego da porta udp/5555 a 100Mbps (`throughput('100Mbps')`) entre a rede do *host-3* (10.0.0.0/24) e o servidor *server-1*, cujo link é de 1Gbps. A interconexão do host e do servidor é estabelecida através do Open vSwitch (OpenFlow versão 1.3).

A medição da vazão entre o host e o servidor foi realizada através da ferramenta

[13] `https://docs.oracle.com/cd/E86824_01/html/E54763/netcat-1.html`

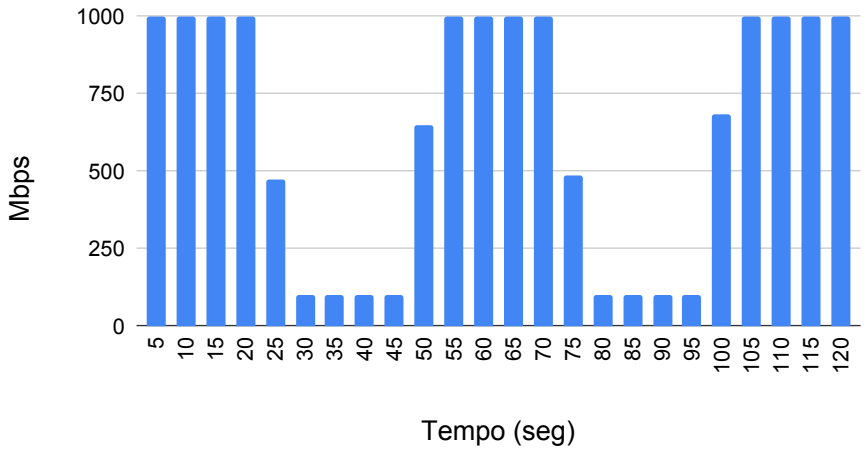
Figura 5. Prova de conceito para aplicação e remoção de ACL



(a) Execução do *script* no host-1 (b) Execução do *script* no host-2 (c) Execução do *script* no host-3

iperf[14]. O gráfico da Figura 6 apresenta o comportamento da vazão da conexão antes, durante e depois de duas aplicações e remoções consecutivas da intenção de *traffic shaping*. Como pode ser observado, a vazão reduz para 100 Mbps quando a intenção é aplicada ao firewall e retorna a 1 Gbps após a remoção da intenção. Os registros (*logs*) e detalhes de execução do teste estão disponíveis no Apêndice H da versão estendida [Fiorenza et al., 2021].

Figura 6. Prova de conceito para aplicação e remoção de *Traffic Shaping*



4. Trabalhos Relacionados

Existem essencialmente dois grupos de linguagens para representar políticas de segurança, como pode ser observado na Tabela 3. O primeiro grupo é composto pelas linguagens específicas de domínio, como RichLanguage [Firewalld, 2021], PDLz [Lobo et al., 2012], LAI [Tian et al., 2019] e NILE [Jacobs et al., 2018]. A RichLanguage e a PDLz são exemplos de linguagens criadas para representar regras de firewalls específicos, com os baseados no IPTables. A LAI é um caso interessante por ser uma linguagem projetada para gerenciar políticas ACLs em WANs (*Wide Area Networks*)

[14] `https://iperf.fr/`

privadas. Mais especificamente, a linguagem permite a adição, remoção e migração de regras de controle de tráfego nos diferentes roteadores que compõem uma WAN.

A linguagem NILE foi projetada para o gerenciar serviços no contexto de NFV (*Network Functions Virtualization*). Além de realizar o encadeamento de NFVs, a linguagem permite também a aplicação de políticas simples de ACL e *traffic shaping* no tráfego entre as funções virtualizadas. Neste primeiro grupo, como pode ser observado na Tabela 3, outras linguagens, como a RichLanguage, suportam também políticas de NAT (*Network Address Translation*) dos tipos 1to1 e Nto1.

Tabela 3. Linguagens para representação de políticas de segurança

	Políticas utilizadas em firewalls modernos							
Linguagem	**ACL**	**NAT 1to1**	**NAT Nto1**	**Traffic Shaping**	**URL Filter**	**Roteamento estático**	**Baseada em Intenções**	**Domínio de Aplicação**
RichLanguage	✓	✓	✓	✗	✗	✗	✗	IPTables
PDLz	✗	✗	✗	✗	✗	✓	✗	IPTables
LAI	✓	✗	✗	✗	✗	✗	✓	WAN
NILE	✓	✗	✗	✓	✗	✗	✓	NFVs
FLIP	✓	✗	✗	✗	✗	✗	✗	Qualquer
FWS	✓	✓	✓	✗	✗	✗	✗	Qualquer
FIRMATO	✓	✗	✗	✗	✗	✗	✗	Qualquer
AFPL2	✓	✓	✓	✗	✗	✗	✗	Qualquer
FWlang	✓	✓	✓	✓	✓	✓	✓	Qualquer

O segundo grupo é formado pelas linguagens que propõem uma abstração mais geral, isto é, que pode ser utilizada para representar políticas de diversos tipos de firewalls. Este é o caso de linguagens como a FLIP [Zhang et al., 2007], FIRMATO [Bartal et al., 2004], FWS [Bodei et al., 2018], AFPL2 [Pozo et al., 2009] e FWlang. Linguagens como FWS e AFPL2 contemplam a representação de políticas do tipo ACL, NAT 1to1 e NAT Nto1. Neste grupo de linguagens, é importante destacar que a única linguagem a suportar os seis tipos (ACL, NAT 1to1, NAT Nto1, *traffic shapping*, filtros de URL e roteamento estático - ver detalhes na Seção 2) de políticas de firewalls modernos é a FWlang.

Outra característica que merece destaque é o uso de intenções para representar políticas de firewalls. O desenvolvimento e a adoção das redes baseadas em intenções (IBNs) vem ganhando força e popularização devido ao apoio de grandes fabricantes de equipamentos de rede e segurança, como a Cisco, Juniper, Huawei e VMware, e o aumento do número de pesquisas sobre o tema [Zeydan and Turk, 2020, Wei et al., 2020]. Para simplificar a representação de políticas de rede ou segurança, linguagens como a LAI e NILE são baseadas na definição de intenções. Similarmente, seguindo essa tendência de mercado e pesquisa, e principalmente a premissa de simplificar a representação das políticas de segurança da rede, a FWlang também utiliza intenções. Resumidamente, com intenções, o operador não precisa mais aprender e conhecer a sintaxe e semântica específica das soluções de firewalls, potencializando a redução de custos de implantação e operação através de uma curva de aprendizagem menor e simplicidade e uniformidade na descrição de políticas de segurança.

5. Considerações Finais

A utilização de uma linguagem de apoio para a configuração de equipamentos de rede é cada vez mais necessária devido a complexidade atual das redes híbridas. Neste trabalho, foi apresentada a especificação, implementação e avaliação da FWlang, uma linguagem baseada em intenções para representar políticas de segurança aplicáveis em diferentes tipos de firewalls. A utilização de intenções permite descrever políticas de segurança de uma forma simples e intuitiva, independente das especificidades ou sintaxe de cada tipo de solução de firewall. A FWlang é a única linguagem que suporta os seis tipos de políticas de segurança (ACL, NAT 1to1, NAT Nto1, *traffic shaping*, filtro de URL e roteamento estático) utilizadas em firewalls de nova geração, que serviram como base para determinar os marcadores suportados pela linguagem.

A FWlang foi implementada e incorporada à solução FWunify, utilizada para automatizar e simplificar a tradução e aplicação de regras em firewalls de redes híbridas. A linguagem foi avaliada quanto a sua capacidade de reduzir a propensão a erros no processo de gerenciamento de firewalls. Os resultados demonstram que a linguagem pode reduzir em até 72% no número de termos utilizados para especificar um conjunto de políticas de segurança para a configuração de diferentes tipos de firewalls. A FWlang foi avaliada também em termos da acurácia do processo de tradução de políticas de segurança, a efetividade das políticas traduzidas.

Como trabalhos futuros, podemos destacar: (a) a extensão da linguagem FWlang para incorporar outros tipos de políticas específicas a linhas de equipamentos, como *DoS Protection* da Palo Alto; (b) realizar uma avaliação da usabilidade da linguagem FWlang, considerando fatores como curva de aprendizagem, retenção ao longo do tempo, taxa de erros por usuário e satisfação subjetiva [Voronkov et al., 2017]; (c) expandir a avaliação do número de termos utilizados na descrição de políticas para incluir outros conjuntos de regras.

Referências

Adão, P., Bozzato, C., Rossi, G. D., Focardi, R., and Luccio, F. L. (2014). Mignis: A semantic based tool for firewall configuration. In *IEEE 27th Computer Security Foundations Symposium*, pages 351–365.

Bartal, Y., Mayer, A., Nissim, K., and Wool, A. (2004). Firmato: A novel firewall management toolkit. *ACM Trans. Comput. Syst.*, 22:381–420.

Bodei, C., Degano, P., Galletta, L., Focardi, R., Tempesta, M., and Veronese, L. (2018). Language-independent synthesis of firewall policies. In *IEEE EuroS&P*, pages 92–106. IEEE.

Cisco Systems (2017). A Rede Baseada em Intenção. `https://www.cisco.com/c/dam/global/pt_br/solutions/pdfs/intent-networking-wp-cte-pt-br.pdf`.

Clemm, A., Ciavaglia, L., Granville, L. Z., and Tantsura, J. (2019). Intent-Based Networking - Concepts and Overview. Internet-Draft draft-clemm-nmrg-dist-intent-03, Internet Engineering Task Force.

Datta, R., Choi, S., Chowdhary, A., and Park, Y. (2018). P4Guard: Designing P4 based firewall. In *IEEE Military Communications Conference (MILCOM)*, pages 1–6. IEEE.

Fiessler, A., Lorenz, C., Hager, S., and Scheuermann, B. (2018). FireFlow-high performance hybrid SDN-firewalls with OpenFlow. In *IEEE 43rd LCN*, pages 267–270. IEEE.

Fiorenza, M., Kreutz, D., Mansilha, R., de Macedo, D. D. J., Feitosa, E., and Immich, R. (2021). Representação e aplicação de políticas de segurança em firewalls de redes híbridas (versão estendida). `https://arxiv.kreutz.xyz/sbrc2021_fwlang_versao_estendida.pdf`.

Fiorenza, M. M., Kreutz, D., and Mansilha, R. (2020). Gerenciamento de firewalls em redes híbridas. In *Anais do XX Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*.

Firewalld (2021). Rich Language Documentation. https://firewalld.org/documentation/man-pages/firewalld.richlanguage.html.

Gartner (2019). Technology Insight for Network Security Policy Management. https://www.gartner.com/en/documents/3902564/technology-insight-for-network-security-policy-managemen.

IBM X-Force (2020). X-Force Threat Intelligence Index. https://www.ibm.com/downloads/cas/DEDOLR3W.

Jacobs, A. S., Pfitscher, R. J., Ferreira, R. A., and Granville, L. Z. (2018). Refining network intents for self-driving networks. In *Proceedings of the Afternoon Workshop on Self-Driving Networks*, SelfDN 2018, page 15–21, New York, NY, USA. Association for Computing Machinery.

Krit, S.-d. and Haimoud, E. (2017). Overview of firewalls: Types and policies: Managing windows embedded firewall programmatically. In *International Conference on Engineering & MIS (ICEMIS)*, pages 1–7. IEEE.

Lobo, J., Marchi, M., and Provetti, A. (2012). Firewall configuration policies for the specification and implementation of private zones. In *IEEE International Symposium on Policies for Distributed Systems and Networks*, pages 78–85. IEEE.

Netfilter (2021). Documentation about the netfilter/iptables project. http://www.netfilter.org/documentation/.

pfSense (2020). The pfSense Documentation. https://docs.netgate.com/manuals/pfsense/en/latest/the-pfsense-documentation.pdf.

Pozo, S., Varela-Vaca, A., and Gasca, R. (2009). AFPL2, an abstract language for firewall ACLs with NAT support. In *Second International Conference on Dependability*, pages 52–59. IEEE.

Riftadi, M. and Kuipers, F. (2019). P4i/o: Intent-based networking with P4. In *IEEE Conference on NetSoft*, pages 438–443. IEEE.

Sanvito, D., Moro, D., Gulli, M., Filippini, I., Capone, A., and Campanella, A. (2018). Onos intent monitor and reroute service: enabling plug&play routing logic. In *4th IEEE NetSoft*, pages 272–276. IEEE.

Scarfone, K. and Hoffman, P. (2009). Guidelines on firewalls and firewall policy. *NIST Special Publication*, 800:41.

Scheid, E. J., Widmer, P., Rodrigues, B. B., Franco, M. F., and Stiller, B. (2020). A controlled natural language to support intent-based blockchain selection. In *IEEE ICBC*, pages 1–9. IEEE.

Singh, A., Aujla, G. S., and Bali, R. S. (2020). Intent-based network for data dissemination in software-defined vehicular edge computing. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–9.

Soule, R., Basu, S., Marandi, P. J., Pedone, F., Kleinberg, R., Sirer, E. G., and Foster, N. (2018). Merlin: A language for managing network resources. *IEEE/ACM Trans. Netw.*, 26(5):2188–2201.

Tian, B., Zhang, X., Zhai, E., Liu, H. H., Ye, Q., Wang, C., Wu, X., Ji, Z., Sang, Y., Zhang, M., Yu, D., Tian, C., Zheng, H., and Zhao, B. Y. (2019). Safely and automatically updating in-network ACL configurations with intent language. In *Proceedings of the ACM SIGCOMM*, page 214–226. ACM.

Vinh Tran, T. and Ahn, H. (2016). Flowtracker: A SDN stateful firewall solution with adaptive connection tracking and minimized controller processing. In *ICSN*, pages 1–5.

VMware (2020). What is intent-based networking (IBN)? https://www.vmware.com/topics/glossary/content/intent-based-networking.

Voronkov, A., Iwaya, L. H., Martucci, L. A., and Lindskog, S. (2017). Systematic literature review on usability of firewall configuration. *ACM Comput. Surv.*, 50(6).

Wei, Y., Peng, M., and Liu, Y. (2020). Intent-based networks for 6g: Insights and challenges. *Digital Communications and Networks*, 6(3):270–280.

Zeydan, E. and Turk, Y. (2020). Recent advances in intent-based networking: A survey. In *IEEE 91st Vehicular Technology Conference (VTC-Spring)*, pages 1–5. IEEE.

Zhang, B., Al-Shaer, E., Jagadeesan, R., Riely, J., and Pitcher, C. (2007). Specifications of a high-level conflict-free firewall policy language for multi-domain networks. In *Proceedings of the 12th ACM SACMAT*, page 185–194.