UNIVERSIDADE FEDERAL DE SANTA CATARINA

ROBERTH SOUZA BELTER

RSB FIREWALL LANGUAGE:
LINGUAGEM DE PROGRAMAÇÀO PARA A SEGURANÇA DE REDES

Florianópolis

2007

ROBERTH SOUZA BELTER

RSB FIREWALL LANGUAGE:
LINGUAGEM DE PROGRAMAÇÀO PARA A SEGURANÇA DE REDES

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas de Informação da Universidade Federal de Santa Catarina, como requisito parcial à obtenção do título de bacharel em Sistemas de Informação.

Orientador: Prof. Olinto José Varela Furtado

Florianópolis

2007

Índice

1. INTRODUÇÃO

Hoje em dia, a maioria das redes de informações mundiais estão interligadas de alguma forma, devido ao avanço tecnológico iniciado na década de 90. A partir de então, novas tecnologias sobre segurança de redes começaram a surgir, como firewalls especializados, IDS, proxies, etc.

A idéia da RSB Firewall Language surgiu em um momento de dificuldade para se configurar um servidor firewall usando programas de código-livre. Atualmente, existem ferramentas de grande complexidade e capacidade para se configurar um servidor firewall, mas todas de custo elevado e ferramentas "*free*" normalmente são complicadas de se configurar.

Assim, desenvolveu-se uma linguagem de programação capaz de interagir com programas utilizados para se configurar um firewall em diferentes plataformas, visando mais facilidades e funcionalidades ao programador. A idéia principal é a de que o programador desenvolva um programa na linguagem RSBFL e este programa seja facilmente traduzido para linhas de comando (*scripts*) de programas como o Iptables, para Linux, e IPFW, para FreeBSD.

No desenvolvimento da linguagem RSBFL Firewalls foram utilizados conceitos aprendidos nas disciplinas INE5622 – Introdução a Compiladores, INE5630 – Segurança em Computação Distribuída, INE5613 – Banco de Dados I, INE5609 – Estrutura de Dados e INE5619 – Administração e Gerência de Redes de Computadores e INE5608 – Análise e Projeto de Sistemas.

No capítulo 2 mostra-se uma introdução sobre firewalls e filtragem de pacotes, no capítulo 3 mostra-se uma introdução sobre linguagens de programação, no capítulo 4 mostra-se uma solução existente para configurações de firewalls, no capítulo 5 mostra-se a linguagem RSBFL, sua metodologia, componentes e funcionalidades, no capítulo 6 mostra-se um exemplo de programação de firewall para uma pequena empresa e no capítulo 7 mostra-se algumas vantagens e desvantagens da linguagem RSBFL.

2. FIREWALLS

Atualmente, praticamente o mundo intero está interconectado através de alguma rede para compartilhamento de informações, sendo a rede Internet a mais utilizada para tal. A rede Internet é uma rede descentralizada que não foi projetada para garantir a segurança da informação; portanto, as informações que trafegam sobre ela, se não forem tratadas de maneira correta, podem estar acessíveis para qualquer indivíduo, através da acessibilidade direta da informação ou através de ataques. Existem várias formas de se violar uma rede, mas essas violações nada mais são do que ataques a falhas de protocolos ou serviços de uma rede.

Umas das maneiras de se proteger uma rede e impedir essas violações é através do uso de um firewall, que tem por finalidade impedir as conexões indesejadas a uma determinada rede.

2.1 DEFINIÇÃO DE FIREWALL

Um firewall (*"parede de fogo"*, em inglês) pode ser definido como um conjunto de ferramentas (serviços) com objetivo de proteger serviços e dados de uma rede interna (confiável) de ataques vindos de redes externas (não confiáveis, como por exemplo, a internet).

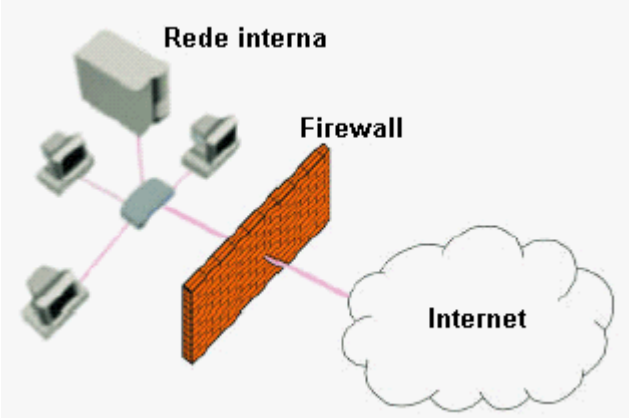


Figura 1 – Mostra um exemplo de firewall entre duas redes.

Segundo a RFC 2979, um firewall deve servir de extremidade para outros servidores (servidores de email, servidores web, etc.) e também realizar uma filtragem de pacotes que trafegam pela rede. Um firewall se baseia basicamente no ip de origem, ip de destino, protocolo e na porta que o pacote irá trafegar, podendo rejeitar ou permitir sua passagem.

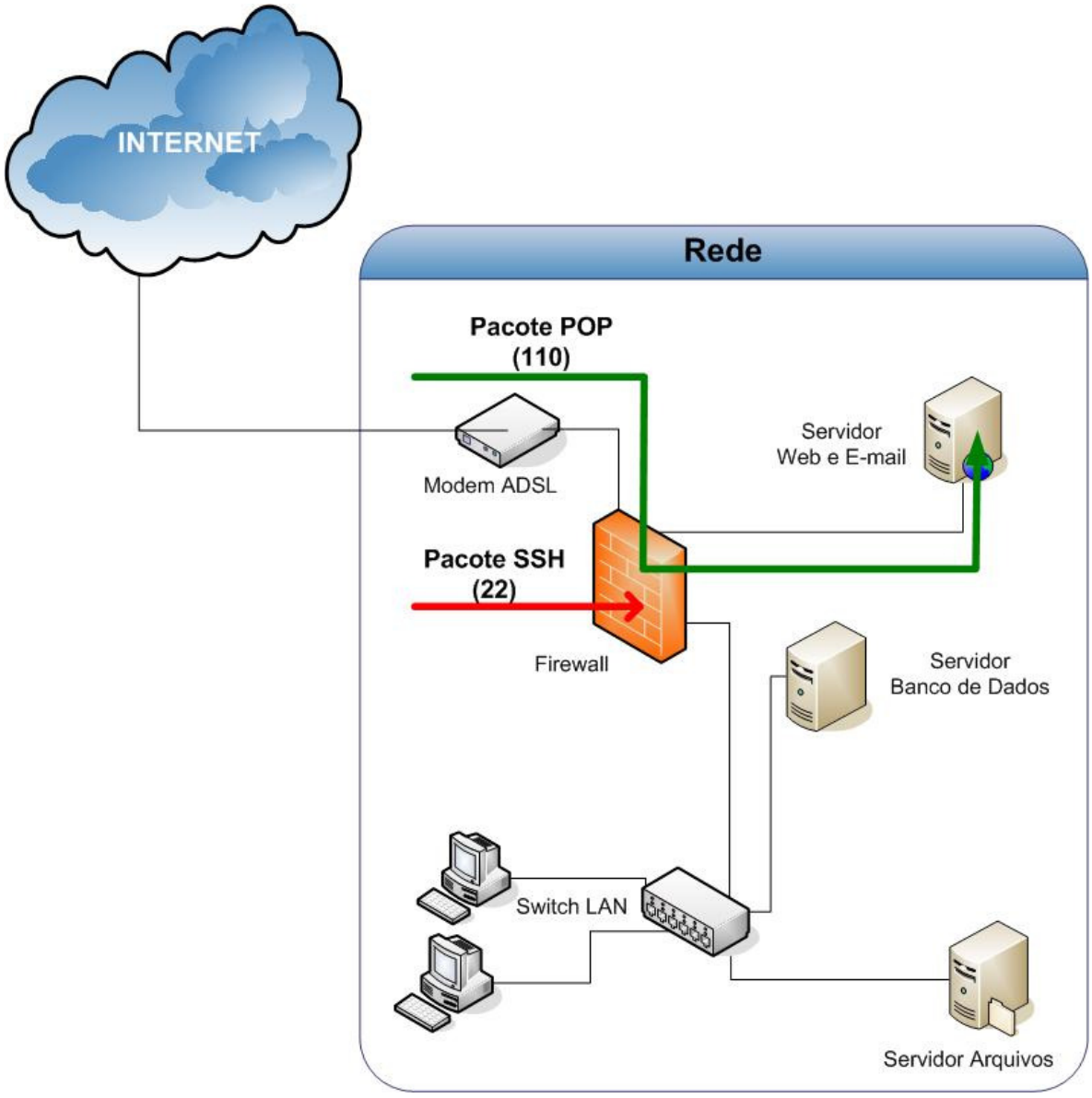


Figura 2 – Mostra a influência de um firewall entre duas redes.

No exemplo da Figura 2, um pacote POP usando a porta 110 está passando pelo firewall e está sendo direcionado para o servidor WEB da rede. Outro pacote SSH usando a porta 22 está sendo bloqueado pelo firewall.

2.2 ELEMENTOS PRINCIPAIS DE UM FIREWALL

Nesta seção serão mostrados os elementos de um firewall envolvidos na filtragem de pacotes: interfaces, pacotes, serviços e portas.

2.2.1 Interfaces

Interfaces de um firewall são os dispositivos físicos por onde trafegam os pacotes. Normalmente um firewall possui duas interfaces, uma para rede interna e uma para a rede externa (comumente a rede Internet).

2.2.2 Pacotes

Todas as transações realizadas em uma rede de computadores envolvem pacotes. Quando uma determinada máquina deseja enviar uma mensagem à outra máquina (email, ou uma requisição a um determinado site, etc.), esta mensagem é dividida em partes com um determinado tamanho de bytes. Estas partes da mensagem são definidas como pacotes.

Para um firewall, um pacote contém os seguintes campos:

- *origem*: define o IP ou rede de origem do pacote.
- *destino*: define o IP ou rede de destino do pacote.
- *porta de origem*: define a porta de conexão usada ao sair da origem.

- *porta de destino*: define a porta de conexão usada ao chegar no destino.
- *protocolo*: define o tipo de protocolo usado na mensagem (TCP, UDP, GRE, etc.).
- *interface de origem*: define a interface do firewall em que o pacote está entrando.
- *interface de destino*: define a interface do firewall em que o pacote está saindo.

2.2.3 Serviços e Portas

Um firewall é constituído por um sistema operacional (normalmente usadas distribuições Linux), e um conjunto de processos (*daemons*) que realizam a filtragem de pacotes ou outros serviços como HTTP Proxy (squid) e DNS (bind/named). Alguns serviços para se comunicar com outras máquinas utilizam umas janelas específicas chamadas portas. A baixo, seguem alguns exemplos de serviços e portas utilizadas:

– 21 – FTP
– 22 – SSH
– 23 – Telnet
– 25 – SMTP
– 80 – HTTP
– 110 – POP3
– 443 – HTTPS

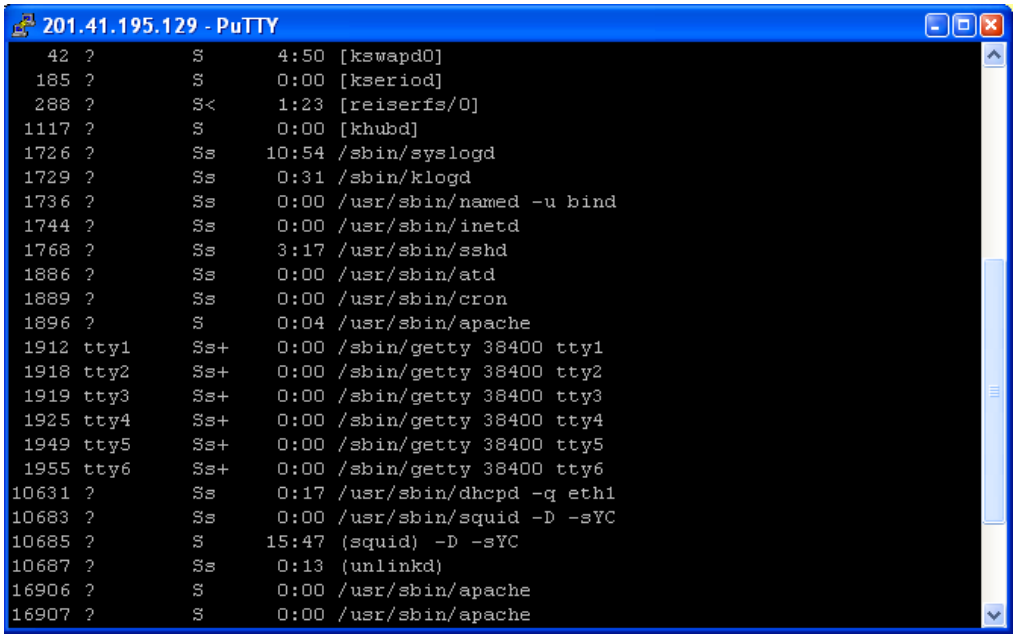


Figura 3 – Mostra os processos em execução em um sistema Linux.

Nas próximas sessões serão mostrados duas ferramentas utilizadas para filtragem de pacotes e conexões em firewalls baseados em software livre, o Iptables para Linux e o IPFW para FreeBSD.

2.2.3.1 Iptables

O Iptables é um programa baseado em linhas de comando, desenvolvido pela netfilter, que armazena regras de filtragem de pacotes em tabelas de roteamento.

As principais tabelas do Iptables são: INPUT, que armazena regras de roteamento para pacotes que estão entrando na máquina; OUTPUT, que armazena regras de roteamento para pacotes que estão saindo da máquina; FORWARD, que armazena regras de roteamento para pacotes que estão "atravessando" a máquina, ou seja, entrando por uma interface com destino a uma outra interface; PREROUNTING NAT, que armazena regras de roteamento para pacotes onde o destino deve ser alterado; e POSTROUTING NAT que armazena regras de roteamento para pacotes onde a origem deve ser alterada.

Vejamos um exemplo de linha de comando Iptables:

iptables -A FORWARD -p TCP -s 192.168.1.0/24 --dport 110 -j ACCEPT

A tag "-A" indica qual tabela a regra será inserida, no caso a tabela FORWARD. A tag "-p" indica o protocolo dos pacotes, no caso TCP. A tag "-s" indica a origem dos pacotes, no caso a rede 192.168.1.0/24. A tag "–dport" indica a porta de destino dos pacotes, no caso a porta 110 (pop). E a tag "-j" indica o que será feito com os pacotes, no caso aceitando a passagem.

De uma forma mais explicita, a linha de comando acima está indicando ao Iptables que será inserido na tabela FORWARD uma regra que aceita os pacotes vindos da rede 192.168.1.0/24 com destino a qualquer rede através da porta 110 (pop).

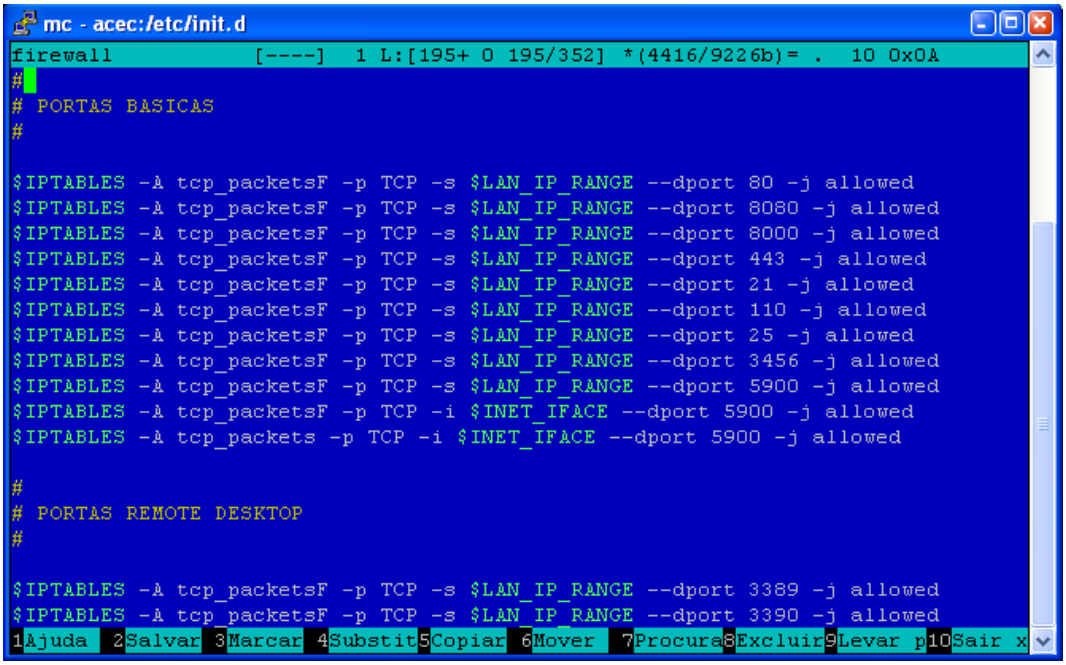


Figura 4 - Mostra um script Iptables sendo editado pelo editor *mcedit* no sistema Linux.

2.2.3.2 IPFW

O IPFW é o filtro de pacotes nativo do sistema operacional FreeBSD. Possui as mesmas funcionalidades do Iptables, que são as de decidir se um determinado pacote será aceito ou descartado ao passar pelo firewall.

Abaixo um exemplo de linha de comando IPFW:

ipfw add allow tcp from 192.168.7.0/24 to any 110 in via dc0

A tag “add” indica que está sendo incluída uma nova regra de roteamento no IPFW. A tag “allow” indica que o pacote será aceito ao fazer a passagem. A tag “tcp” indica o protocolo do pacote. A tag “from 192.168.7.0/24” indica que a origem do pacote é a rede 192.168.7.0/24. A tag “to any” indica que o destino do pacote é para qualquer lugar. A tag “110” indica que a porta do pacote é a porta 110 (POP). E as tags “in via dc0” indica que o origem do pacote é a interface de rede dc0.

De uma forma mais explicita, a linha de comando acima está indicando ao IPFW que será inserida uma regra que aceita os pacotes vindos da rede 192.168.7.0/24 com destino a qualquer rede através da porta 110 (pop).

3. INTRODUÇÃO A LINGUAGENS DE PROGRAMAÇÃO

Neste capítulo, serão apresentados os conceitos e os elementos principais que envolvem uma linguagem de programação.

3.1 TEORIA DA COMPUTAÇÃO

Teoria da computação é o estudo do que pode ou não ser efetivamente computável, de quais formas e com que complexidade. Ela divide estes problemas em três classes:

- Problemas Indecidíveis: impossíveis de serem solucionados de forma computacional.
- Problemas Intratáveis: Possíveis de resolver, porém impossível devido a l imitações de hardware.
- Problemas Tratáveis: Possíveis de ser solucionados.

Para resolver estes problemas, são estudados dois conceitos fundamentais:

- Procedure: Conjunto finito de instruções que podem ser executados mecanicamente em um espaço de tempo.
- Algoritmo: Procedure que tem uma parada garantida, independente de suas entradas.

3.2 LINGUAGENS FORMAIS

Uma linguagem pode ser definida como uma forma de comunicação. Mais precisamente, pode ser definida como um conjunto de elementos (símbolos) e um conjunto de regras para combinar estes elementos, de forma a ser entendida por uma certa comunidade.

Os principais conceitos de linguagens são:

- Alfabeto: conjunto finito e não vazio de símbolos de uma linguagem.
- Sentença: agrupamento finito de símbolos de uma linguagem.

Portanto, entende-se que linguagens formais são linguagens que podem ser representadas de maneira finita e precisa, através de métodos matemáticos.

3.3 GRAMÁTICA

A gramática de uma linguagem pode ser definida como um conjunto de regras que definem as combinações válidas dos elementos de uma linguagem em uma sentença.

Abaixo, um exemplo de gramática aplicada a língua portuguesa:

<sentença> :: = <sujeito> <predicado>

<sujeito> :: = <substantivo>

| <artigo> <substantivo>

| <artigo> <adjetivo> <substantivo>

<predicado> :: = <verbo> <objeto>

<substantivo> :: = João | Maria | cachorro | livro | pão | escola
<artigo> :: = o | a | à | ó
<adjetivo> :: = pequeno | bom | bela
<verbo> :: = morde | le | olha | foi
<objeto> :: = <substantivo>
| <artigo> <substantivo>
| <artigo> <adjetivo> <substantivo>

Notação:

< ..... > : categoria sintática ou gramatical;
::= : definido por
| : ou (alternativa)
α ::= β : regra de sintaxe (ou regra gramatical ou regra de produção)

3.4 ANÁLISE DE UMA LINGUAGEM

Analisar uma linguagem significa verificar se uma sentença sobre ela foi escrita de forma correta. A análise de uma linguagem pode ser feita através de três passos que serão descritos a seguir.

3.4.1 Análise Léxica

A análise léxica de uma linguagem consiste em verificar se os símbolos escritos em uma sentença pertencem ao alfabeto de uma linguagem.

Os símbolos de uma linguagem podem ser definidos através de símbolos fixos como gato, cachorro, casa; ou através de expressões regulares, como por exemplo:

' .* ' – Seqüência de caracteres entre aspas.

{L}* {D}* – Seqüência de letras, seguida por uma seqüência de dígitos.

3.4.2 Análise Sintática

A análise sintática de uma linguagem consiste em verificar se os símbolos escritos em uma sentença foram escritos de acordo com a definição da gramática da linguagem.

Ex.: João foi à escola.

<sentença> :: = <sujeito> <predicado>

<sentença> :: = <substantivo> <verbo> <objeto>

<sentença> :: = <substantivo> <verbo> <artigo> <substantivo>

<sentença> :: = João foi à escola

A frase, portanto, está sintaticamente correta.

3.4.3 Análise Semântica

A análise semântica de uma linguagem consiste em verificar a sentença escrita possui um significado perante a definição de uma linguagem.

Ex.: João foi à escola. A frase João foi à escola possui um significado perante a língua portuguesa, porém se fosse invertido os substantivos da frase, passando a ter escola foi à João, a frase estaria correta sintaticamente, porém não teria um significado lógico.

4. SOLUÇÕES EXISTENTES - FWBUILDER

O FWBuilder é uma ferramenta para gerenciamento de firewalls, com a função de gerar scripts para as seguintes aplicações: Iptables, IPfilter, IPFW, Cisco PIX e OpenBSD PF. Através dele é possível montar as regras através de uma interface gráfica e gerar o script para a aplicação escolhida. Para gerar o script, o IPFW "compila" as regras definidas na interface em um arquivo XML que é traduzido para o script escolhido.

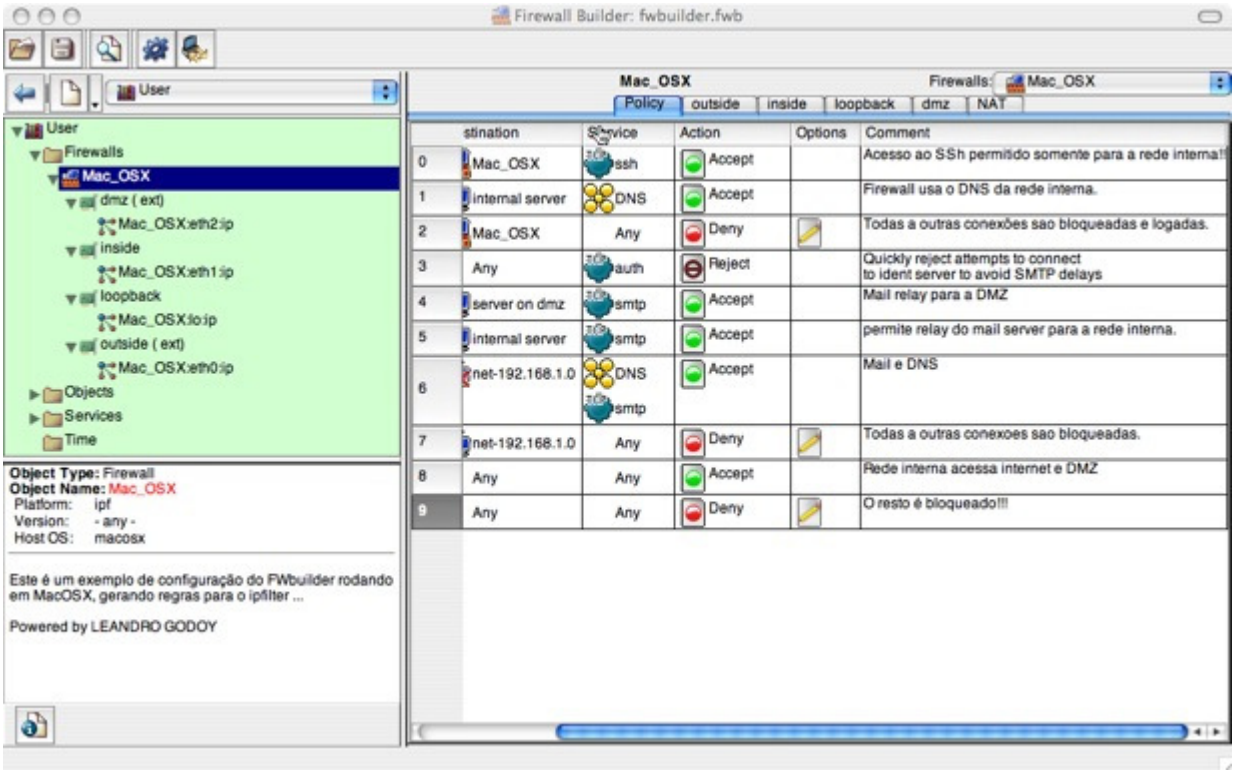


Figura 5 – Mostra a interface de configuração do FWBuilder.

Possui as seguintes funcionalidades:

- Permite comentários nas regras.
- Permite criar grupos de regras.
- Possui templates já prontos para os firewalls mais simples.

- Suporte a módulos.
- Sistema Multiplataforma.

A desvantagem do uso do FWBuilder é que ele é uma ferramenta paga e de código fechado.

5. A LINGUAGEM RSB FIREWALL

O que motivou o desenvolvimento da linguagem RSBFL foi justamente a dificuldade de configuração de firewalls em servidores Linux utilizando ferramentas de código-livre. Foi presenciado em algumas empresas a utilização dos softwares acima e as dificuldades encontradas nas configurações, levando às vezes a irritação dos técnicos e insatisfação dos clientes.

A partir destas dificuldades, pensou-se em desenvolver uma linguagem multiplataforma que trate estes problemas de forma mais dinâmica e organizada, dando uma maior facilidade de leitura ao usuário.

5.1 FERRAMENTAS

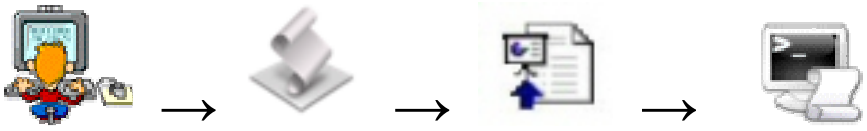
Para desenvolver esta linguagem, foi utilizado como sistema operacional o ambiente Debian GNU/Linux 4.0 etch (http://www.debian.org). Para desenvolver a linguagem em si, foi utilizado o sistema GALS (http://gals.sourceforge.net) que é um gerador de analisadores léxicos e sintáticos em conjunto com Java (http://java.sun.com). E para desenvolver o software que fará a interface da linguagem, foi utilizado PHP5 (http://www.php.net) e MySQL Server 5.0 (http://www.mysql.com).

5.2 METODOLOGIA

O objetivo da linguagem RSBFL é transformar códigos em alto nível definidos pelo usuário em scripts de firewall para as plataformas Linux, através do Iptables; e FreeBSD, através do IFPW. Esta transformação é realizada através de um interpretador, que lê um programa definido pelo usuário e traduz este código para linhas de comando, criando um script Iptables ou

IPFW. Este programa pode ser escrito através de um editor ou definido graficamente através de uma interface de gerenciamento.

Resumidamente, as etapas de transformação ocorrem da seguinte forma:



1 2 3 4

1. O usuário terá acesso a uma interface gráfica onde ele poderá definir sua programação para o firewall através de um editor ou de uma interface de gerenciamento.
2. Com a programação definida, inicia-se o processo de interpretação do programa através das análises léxica, sintática e semântica. Após a análise, é feita a geração de um código intermediário.
3. O interpretador lê o código intermediário e cria um mapeamento para as linhas de comando do script.
4. O script é ativado no firewall, colocando em atividade as regras definidas.

5.3 DEFINIÇÃO DOS OBJETOS DA LINGUAGEM

Os objetos básicos e os tipos de dados da linguagem foram baseados na forma de definição de regras do Iptables e IPFW.

No sistema Iptables, existem 5 tabelas básicas para roteamento de pacotes que foram descritas acima: INPUT, OUTPUT, FORWARD, PREROUTING NAT e POSTROUTING NAT.

Para definição de uma regra de filtragem, o Iptables usa a sintaxe:

iptables -A *tabela* -p *protocolo* -i *interface de origem* -o *interface de destino* -s *rede de origem* -d *rede de destino* --sport *porta de origem* --dport *porta de destino* -j *ação*

E o IFPW usa a sintaxe:

ipfw add *ação protocolo* from *rede de origem* to *rede de destino porta* in via *interface de origem* out via *interface de destino*

Notando a semelhança entre os dois tipos de comando, foi possível identificar objetos e ações semelhantes nos dois softwares, que proporcionou a definição das classes e tipo de dados da linguagem, conforme explicado nos próximos itens.

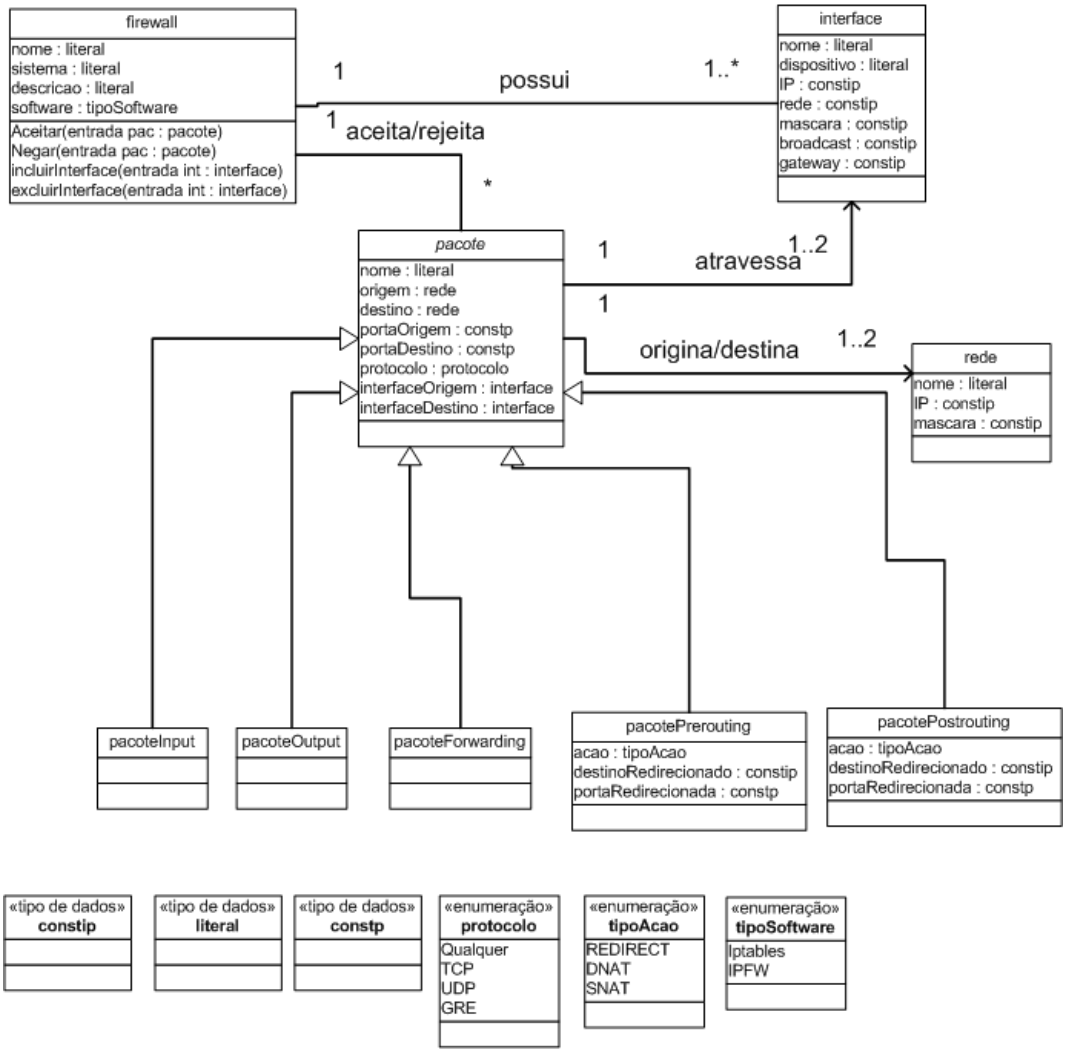


Figura 6 – Mostra o diagrama de classes da linguagem RSBFL.

5.3.1 Tipos de Dados

1. **literal**: definido por ' {L} (_? ({L} | {D}))* ' – Letras de “a” a “z”, dígitos de 1 a 9 ou _, que esteja entre aspa ('), começando obrigatoriamente por uma letra . Ex.: 'Rafael'.
2. **constp**: definido por {D}+ (\-{D}+)? – Quaisquer dígitos entre 0 e 9 (simbolizado por {D}), sendo obrigatório um, que pode ser seguido de um – (traço) juntamente com quaisquer dígitos entre 0 a 9. Identifica uma porta ou um conjunto de portas. Ex.: 22, 0-4458.
3. **constip**: definido por '{D}{D}?{D}?\.{D}{D}?{D}?\.{D}{D}?{D}?\.{D}{D}?{D}?' – Seqüência de quatro casas separadas por um . (ponto) contendo pelo menos um dígito de 0 a 9 (simbolizado por {D}), podendo haver mais dois dígitos de 0 a 9. Identifica o IP de um determinado host. Ex.: 192.168.1.1.
4. **protocolo**: definido por Qualquer, TCP, UDP e GRE. Identifica o tipo de protocolo de um pacote.
5. **tipoAcao**: definido por REDIRECT, DNAT e SNAT. Identifica o tipo de ação a ser tomada pelo firewall em relação ao pacote quando da necessidade de redirecioná-lo para outro destino.
6. **tipoSoftware**: definido por Iptables e IPFW. Identifica o tipo de software que será utilizado no firewall para filtragem de pacotes

5.3.2 Classes

1. **firewall**: A classe firewall é o elemento principal do sistema. Nela é definido o tipo de software utilizado para filtragem de pacotes e se o pacote será aceito ou rejeitado. Possui aos seguintes atributos e métodos:

nome: atributo define o nome do firewall. Ex.: Firewall Matriz

sistema: atributo define o sistema operacional utilizado na máquina. Ex.: Linux.

descricao: atributo define uma descrição qualquer sobre a maquina, como por exemplo a versão do mesmo. Ex.: Debian GN Linux 4.0 Etch.

software: atributo define o tipo de software que fará a filtragem de pacotes, no caso Iptables ou IPFW.

Aceitar(): método que define a aceitação de um pacote durante a filtragem. Ex.: fw.Aceitar(SSH_teste)

Negar(): método que define a rejeição de um pacote durante a filtragem. Ex.: fw.Negar(SSH_teste)

incluirInterface(): método que realiza uma inclusão de interface em um firewall. Ex.: fw.incluirInterface(Rede_Interna)

excluirInterface(): método que realiza uma exclusão de interface em um firewall. Ex.: fw.excluirInterface(Rede_Interna)

2. **interface**: A classe interface define as configurações de uma interface conectada ao firewall. Possui aos seguintes atributos:

   nome: atributo que define o nome da interface. Ex.: Rede_Interna

   dispositivo: atributo que define o nome físico da interface. Ex.: eth0 em sistemas Linux. Ex.: eth0

   IP: atributo que define o endereço IP da interface. Ex.: 192.168.1.1

   rede: atributo que define a rede da interface. Ex.: 192.168.1.0

   mascara: atributo que define a mascara de rede de uma interface Ex.: 255.255.255.0

   broadcast: atributo que define o broadcast da rede. Ex.: 192.168.1.255.

   gateway: atributo que define o gateway (rota padrão) de uma interface: Ex.: 192.168.1.10.

3. **rede**: A classe rede define os hosts e redes que fazem partes das redes vinculadas ao firewall. Nesta classe pode-se definir tanto uma máquina específica quanto uma rede inteira. Possui aos seguintes atributos:

   nome: define o nome do host o rede: Ex.: Servidor_WEB.

   IP: define o ip do host ou rede. Ex.: 192.168.1.8.

mascara: define a mascara de rede do host ou rede. Ex.: 255.255.255.0

4. **pacote**: A classe pacote é uma superclasse que define as classes pacoteInput, pacoteOutput, pacoteForwarding, pacotePrerouting e pacotePostrouting. As três primeiras classes são idênticas a classe pacote, já as classes prerouting e postrouting incluem mais três atributos que serão descritos no item 5. A função da classe pacote é definir os tipos de pacotes que entram e saem do firewall. Possui aos seguintes atributos:

   nome: define o nome do pacote. Ex. SSH_WEB_SERVER

   origem: define a rede de origem do pacote: Ex.: Rede_Interna.

   destino: define a rede de destino do pacote: Ex.: Rede_Externa.

   portaOrigem: define a porta de origem do pacote: Ex.: 555.

   portaDestino: define a porta de destino do pacote: Ex.: 22.

   protocolo: define o protocolo do pacote: Ex.: TCP

   interfaceOrigem: define a interface de entrada do pacote: Ex.: Interface_Interna.

   interfaceDestino: define a interface de destino do pacote: Ex.: Interface_Externa.

5. **pacotePrerouting e pacotePostrouting**: As duas classes tem a função de executar o redirecionamento de pacotes. Além dos atributos herdados da classe pacote, possuem aos seguintes atributos

   ação: ação que deve ser tomada no direcionamento do pacote. Ex.: DNAT.

   destinoRedirecionado: define o endereço para qual o pacote deve ser redirecionado. Ex.: Servidor_WEB.

   portaRedirecionada: define a porta para qual o pacote deve ser redirecionado. Ex.: 80.

5.4 DEFINIÇÕES REGULARES E TOKENS

Com base nos tipos de dados definidos acima, foram definidas as seguintes expressões regulares para a linguagem:

- *L* : [a-zA-Z] – Corresponde as letras do alfabeto.

- *D* : [0-9] – Corresponde aos algoritmos de 0 a 9.
- *tab* : [\s\n\t\r] – Corresponde aos espaços, tabulações e quebras de linha que devem ser ignorados pelo interpretador.
- *comentario* : "/*" [^"*"]* "*"+ ([^ "*" "/"] [^"*"]* "*"+)* "/" – Corresponde aos comentários que também devem ser ignorados pelo interpretador.

E com base nas classes, propriedades e métodos; foram criados os seguintes tokens:

- *constp* : {D}+ (\-{D}+)? – Corresponde a quaisquer dígitos entre 0 e 9, sendo obrigatório um, que pode ser seguido de um – (traço) juntamente com quaisquer dígitos entre 0 a 9.
- *constip* : '{D}{D}?{D}?\.{D}{D}?{D}?\.{D}{D}?{D}?\.{D}{D}?{D}?' – Corresponde a uma seqüência de quatro casas separadas por um . (ponto) contendo pelo menos um dígito de 0 a 9, podendo haver mais dois dígitos de 0 a 9
- literal : ' {L} (_? ({L} | {D}))* ' – Corresponde a uma letra (L) que pode ser seguido por um _ (underline) e uma letra ou dígito (D) entre aspa (').
- *vazio* : '' – Corresponde a um literal vazio.
- *id* : {L} (_? ({L} | {D}))* – Corresponde a uma letra (L) que pode ser seguido por um _ (underline) e uma letra ou dígito (D). Abaixo segue a lista predefinida de ids da linguagem:

  Programa, defina, Objetos, Funcoes, funcao, inicio, fim, declare, firewall, interface, rede, pacoteInput, pacoteOutput, pacoteForwarding, pacotePrerouting , pacotePostrouting, nome, sistema, descricao, software, dispositivo, ip, mascara, broadcast, gateway, origem, destino, portaOrigem, portaDestino, protocolo, interfaceOrigem, interfaceDestino, acao, destinoRedirecionado, portaRedirecionada, DNAT, SNAT, REDIRECT, Iptables, IPFW, Qualquer, TCP, UDP, GRE, incluirInterface, excluirInterface, Aceitar, Negar.

- *símbolos especiais*: '' – Esta categoria de símbolos é composta por operadores, delimitadores e separadores, Listados a seguir:

; , . , = , (,) , { , } , * , / , : , '.

5.5 GRAMÁTICA

A gramática da linguagem RSBFL foi definida da seguinte forma:

<programa> ::= Programa id <bloco> ;

<bloco> ::= <bloco_obj> <bloco_func> inicio <corpo> fim ".";

<bloco_obj> ::= defina Objetos inicio <def_obj> fim ";" ;

<def_obj> ::= declare id ":" <tipo> <def_obj> | î ;

<tipo> ::= firewall "{" <corpo_fw> "}" | interface "{" <corpo_iface> "}" | rede "{" <corpo_rede> "}" | pacoteInput "{" <corpo_pacote> "}" | pacoteOutput "{" <corpo_pacote> "}" | pacoteForwarding "{" <corpo_pacote> "}" | pacotePrerouting "{" <corpo_nat> "}" | pacotePostrouting "{" <corpo_nat> "}" ;

<corpo_fw> ::= "." nome "=" literal ";" "." sistema "=" literal ";" "." descricao "=" literal ";" "." software "=" <soft> ";" ;

<soft> ::= Iptables | IPFW ;

<corpo_iface> ::= "." nome "=" literal ";" "." dispositivo "=" literal ";" "." ip "=" constip ";" "." rede "=" constip ";" "." mascara "=" constip ";" <bc> ;

<bc> ::= "." <bc1> | î ;

<bc1> ::= broadcast <bc2> <bc3> | gateway <bc2> ;

<bc2> ::= "=" <cip> ";" ;

<bc3> ::= "." gateway <bc2> | î ;

<cip> ::= constip | vazio ;

<corpo_rede> ::= "." nome "=" literal ";" "." ip "=" constip ";" "." mascara "=" constip ";" ;

<corpo_pacote> ::= "." nome "=" literal ";" "." origem "=" <chost> ";" "." destino "=" <chost> ";" "." portaOrigem "=" constp ";" "." portaDestino "=" constp ";" "." protocolo "=" <prot> ";" "." interfaceOrigem "=" <chost> ";" "." interfaceDestino "=" <chost> ";" ;

<chost> ::= id | Qualquer ;

<prot> ::= Qualquer | TCP | UDP | GRE ;

<corpo_nat> ::= "." nome "=" literal ";" "." origem "=" <chost> ";" "." destino "=" <chost> ";" "." portaOrigem "=" constp ";" "." portaDestino "=" constp ";" "." protocolo "=" <prot> ";" "." interfaceOrigem "=" <chost> ";" "." interfaceDestino "=" <chost> ";" "." acao "=" <tacao> ";" "." destinoRedirecionado "=" <chost> ";" "." portaRedirecionada "=" constp ";" ;

<tacao> ::= DNAT | SNAT | REDIRECT ;

<bloco_func> ::= defina Funcoes inicio <def_func> fim ";" ;

<def_func> ::= funcao id "(" ")" "{" <corpo> "}" <def_func> | î ;

<corpo> ::= <comando> ";" <corpo> | î ;

<comando> ::= id <comando1> ;

<comando1> ::= "(" ")" | "." <comando2> ;

<comando2> ::= Aceitar "(" id ")" | Negar "(" id ")" | incluirInterface "(" id ")" | excluirInterface "(" id ")" ;

De acordo com a gramática, o programa escrito na linguagem RSBFL teria três blocos diferenciados:

1. O primeiro bloco para definição de objetos, iniciado através da sintaxe *defina Objetos*. Neste bloco serão definidos objetos do tipo firewall, interface, rede e pacotes. Ex.:
   Declaração de um host e pacotes vindos deste host.

```
declare rd0: rede {

    .nome = 'Host_Suporte';
    .ip = '200.143.128.138';
    .mascara = '255.255.255.255';
}

declare pacote_ssh: pacoteInput {

    .nome = ‘Suporte_SSH';
    .origem = Host_Suporte;
    .destino = Firewall;
    .portaOrigem = 0-65535;
```

```
        .portaDestino = 22;
        .protocolo = TCP;
        .interfaceOrigem = Qualquer;
        .interfaceDestino = Rede_Externa;

    }

    declare pacote_rdp: pacoteInput {

        .nome = 'Suporte_RDP';
        .origem = Host_Suporte;
        .destino = Firewall;
        .portaOrigem = 0-65535;
        .portaDestino = 3389;
        .protocolo = TCP;
        .interfaceOrigem = Qualquer;
        .interfaceDestino = Rede_Externa;

    }
```

2. O segundo bloco para definição de funções, iniciado através da sintaxe *defina Funcoes*. Neste bloco serão definidas funções para agrupamento de regras: Ex.: Uma função para ativação de acesso externo do host suporte (declarado no item acima) na porta 22 e 3389.

```
    funcao libera_suporte() {

        fw.Aceitar(pacote_ssh);
        fw.Aceitar(pacote_rdp);

    }
```

3. O terceiro bloco para dar início ao programa e chamar as funções iniciais. Ex.:

inicio

regras_redeInterna();
libera_suporte();

fim.

De forma mais sucinta, um programa escrito na linguagem RSBFL ficaria no seguinte formato:

Programa *id*;

defina Objetos
inicio

declare *id*: *tipo* {

propriedades (ex.: .nome = 'host_suporte')

}

fim;

defina Funcoes
inicio

funcao *id*() {

chamada de funções. (ex.: fw.Aceitar(pcotex))

}

fim;

inicio

chamada de funções.(ex.:libera_suporte())

fim.

5.6 SEMÂNTICA DA LINGUAGEM RSBFL

Com base na gramática da linguagem e nos tipos de dados escolhidos , foram definidos os seguintes aspectos semânticos para a linguagem:

- Os objetos não podem ser declarados mais de uma vez.
- Somente 1 (um) objeto firewall pode ser declarado.
- Os atributos IP, Rede, Broadcast e Gateway não podem ter um valor maior que 254 em cada casa.
- O atributo Máscara somente pode ter os valores: 255.255.255.255,255. 255.255.254, 255.255.255.252, 255.255.255.248, 255.255.255.240, 255.255.255.224, 255.255.255.192, 255.255.255.128, 255.255.255.0, 255.255.254.0, 255.255.252.0, 255.255.248.0, 255.255.240.0, 255.255.224.0, 255.255.192.0, 255.255.128.0, 255.255.0.0, 255.254.0.0,

255.252.0.0, 255.248.0.0, 255.240.0.0, 255.224.0.0, 255.192.0.0, 255.128.0.0, 255.0.0.0, 254.0.0.0, 252.0.0.0, 248.0.0.0, 240.0.0.0, 224.0.0.0, 192.0.0.0 e 128.0.0.0, 0.0.0.0.

- Os atributos Origem, Destino e destinoRedirecionado só podem assumir o valor "Qualquer" ou um objeto da classe Rede.

- Os atributos portaOrigem, portaDestino e portaRedirecionada não podem ter um valor maior que 65535.

- Os atributos interfaceOrigem e interfaceDestino só podem assumir o valor "Qualquer" ou um objeto da classe Interface.

- Ids de chamada de funções só podem ser objetos da classe firewall.

- Parâmetros das funções incluirInterface e excluirInterface só podem ser objetos da classe interface.

- Parâmetros das funções Aceitar e Negar só podem ser objetos das classes pacoteInput, pacoteOutput, pacoteForwarding, pacotePrerouting e pacotePostrouting.

- Chamadas de funções só podem ser executadas para funções previamente declaradas.

5.7 SCRIPTS IPTABLES E IPFW

Seguindo a organização da gramática, os scripts também foram organizados em três blocos: definição de objetos, definição de funções e chamada de inicialização do script.

O script Iptables foi organizado no seguinte formato:

```
#!/bin/sh

#
# Definicao de Objetos
#
```

```
RD0_NOME=”Host_Suporte”
RD0_IP=”200.143.128.138”
RD0_MASCARA=”255.255.255.255”

PACOTE_SSH_NOME=”Suporte_SSH”
PACOTE_SSH_ORIGEM=”200.143.128.138”
PACOTE_SSH_DESTINO=”200.180.9.132”
PACOTE_SSH_PORTA_ORIGEM=”0:65535”
PACOTE_SSH_PORTA_DESTINO=”22”
PACOTE_SSH_PROTOCOLO=”TCP”
PACOTE_SSH_INTERFACE_ORIGEM=”ALL”
PACOTE_SSH_INTERFACE_DESTINO=”eth0”

PACOTE_RDP_NOME=”Suporte_RDP”
PACOTE_RDP_ORIGEM=”200.143.128.138”
PACOTE_ RDP _DESTINO=”200.180.9.132”
PACOTE_ RDP _PORTA_ORIGEM=”0:65535”
PACOTE_ RDP _PORTA_DESTINO=”3389”
PACOTE_ RDP _PROTOCOLO=”TCP”
PACOTE_ RDP _INTERFACE_ORIGEM=”ALL”
PACOTE_ RDP _INTERFACE_DESTINO=”eth0”

#
# Definicao de Funcoes
#

libera_suporte() {

        /sbin/iptables –A INPUT –p $PACOTE_SSH_PROTOCOLO \
        –s $PACOTE_SSH_ORIGEM –d $PACOTE_SSH_DESTINO \
        –i $PACOTE_SSH_INTERFACE_ORIGEM \
```

```
        –o $PACOTE_SSH_INTERFACE_DESTINO \
        --sport $PACOTE_SSH_PORTA_ORIGEM \
        --dport $PACOTE_SSH_PORTA_DESTINO –j ACCEPT

        /sbin/iptables –A INPUT –p $PACOTE_RDP_PROTOCOLO \
        –s $PACOTE_ RDP_ORIGEM –d $PACOTE_ RDP _DESTINO \
        –i $PACOTE_ RDP _INTERFACE_ORIGEM \
        –o $PACOTE_ RDP _INTERFACE_DESTINO \
        --sport $PACOTE_ RDP _PORTA_ORIGEM \
        --dport $PACOTE_ RDP _PORTA_DESTINO –j ACCEPT

}

modulos() {

        /sbin/depmod -a

         /sbin/modprobe ip_conntrack
         /sbin/modprobe ip_tables
         /sbin/modprobe iptable_filter
         /sbin/modprobe iptable_mangle
         /sbin/modprobe iptable_nat
         /sbin/modprobe ipt_LOG
         /sbin/modprobe ipt_MASQUERADE
         /sbin/modprobe ip_nat_ftp
         /sbin/modprobe ip_conntrack_ftp

         echo "1" > /proc/sys/net/ipv4/ip_forward

}
```

```
libera() {

        /sbin/iptables -P INPUT ACCEPT
        /sbin/iptables -P OUTPUT ACCEPT
        /sbin/iptables -P FORWARD ACCEPT
        /sbin/iptables -F
        /sbin/iptables -X
        /sbin/iptables -t nat -F

}

Iniciar() {

        libera_suporte

}

#
# Chamadas de inicialização do script
#

case "$1" in

    start)

        # carregando modulos
        modulos

        # esvaziando as tabelas e definindo regras padroes
        libera
```

```
        # permitindo a passagem de pacotes estabelecidos
        /sbin/iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

        Iniciar

        /sbin/iptables -P INPUT DROP
        /sbin/iptables -P OUTPUT DROP
        /sbin/iptables -P FORWARD DROP

    ;;
    stop)

        libera

    ;;
    restart)

      $0 stop
      sleep 1
      $0 start

    ;;
esac
```

As funções módulos() e libera() são funções necessárias ao script iptables. A função módulos() carrega os módulos principais do iptables e a função libera esvazia as tabelas de roteamento para inclusão de novas regras.

Já o script IPFW foi organizado no seguinte formato:

```
#!/bin/sh
```

```
#
# Definicao de Objetos
#

RD0_NOME=”Host_Suporte”
RD0_IP=”200.143.128.138”
RD0_MASCARA=”255.255.255.255”

PACOTE_SSH_NOME=”Suporte_SSH”
PACOTE_SSH_ORIGEM=”200.143.128.138”
PACOTE_SSH_DESTINO=”200.180.9.132”
PACOTE_SSH_PORTA_ORIGEM=”0:65535”
PACOTE_SSH_PORTA_DESTINO=”22”
PACOTE_SSH_PROTOCOLO=”TCP”
PACOTE_SSH_INTERFACE_ORIGEM=”ALL”
PACOTE_SSH_INTERFACE_DESTINO=”eth0”

PACOTE_RDP_NOME=”Suporte_RDP”
PACOTE_RDP_ORIGEM=”200.143.128.138”
PACOTE_ RDP _DESTINO=”200.180.9.132”
PACOTE_ RDP _PORTA_ORIGEM=”0:65535”
PACOTE_ RDP _PORTA_DESTINO=”3389”
PACOTE_ RDP _PROTOCOLO=”TCP”
PACOTE_ RDP _INTERFACE_ORIGEM=”ALL”
PACOTE_ RDP _INTERFACE_DESTINO=”eth0”

#
# Definicao de Funcoes
#

libera_suporte() {
```

```
        ipfw add allow $PACOTE_SSH_PROTOCOLO from $PACOTE_SSH_ORIGEM to
        $PACOTE_SSH_DESTINO $PACOTE_SSH_PORTA_DESTINO in via
        $PACOTE_SSH_INTERFACE_ORIGEM out via
        $PACOTE_SSH_INTERFACE_DESTINO

        ipfw add allow $PACOTE_RDP_PROTOCOLO from $PACOTE_RDP_ORIGEM to
        $PACOTE_RDP_DESTINO $PACOTE_RDP_PORTA_DESTINO in via
        $PACOTE_RDP_INTERFACE_ORIGEM out via
        $PACOTE_RDP_INTERFACE_DESTINO


}

geral() {

        ipfw -f flush

        ipfw add 1 check-state

        ipfw add deny all from any to any frag

        ipfw add allow tcp from any to any established

}


Iniciar() {

        libera_suporte
```

```
}

#
# Chamadas de inicialização do script
#

case "$1" in

    start)

        # esvaziando as tabelas e definindo regras padroes
        geral


        Iniciar

    ;;
    stop)

        libera

    ;;
    restart)

      $0 stop
      sleep 1
      $0 start

    ;;
esac
```

A função libera() é uma função necessária ao script IPFW que esvazia as tabelas de roteamento para inclusão de novas regras e adiciona as regras de negação padrão.

5.8 INTERFACE

Nesta sessão será mostrada a interface de gerenciamento e o editor de programas da linguagem RSBFL e toda a dinâmica para criação de objetos e regras para o firewall.

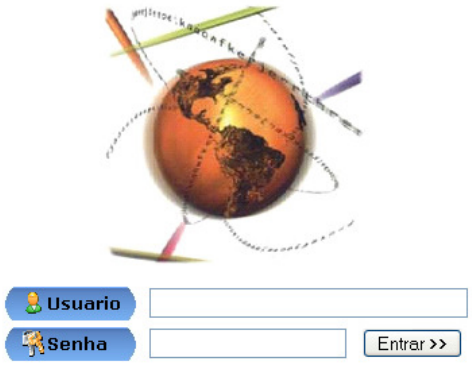


Figura 7 – Mostra a interface para autenticação do usuário no sistema.

Figura 8 – Mostra a tela principal do sistema.

A Figura 8 mostra a tela principal do sistema, a tela inicial da interface de gerenciamento. Nela são apresentadas as opções de navegação do usuário e os créditos do trabalho de conclusão de curso. Nas opções de navegação, temos no menu a esquerda as opções de escolha entre a interface de gerenciamento e o editor do programa.

Figura 9 – Mostra as opções de edição do programa.

5.8.1 A Interface de Gerenciamento

A interface de gerenciamento foi desenvolvida para que se pudesse criar os objetos e funções da linguagem RSBFL de forma dinâmica e posteriormente serem editados no editor de programas. Estas configurações são armazenadas em um banco de dados MySQL e posteriormente lidas por um script php para criação do código do programa.

Figura 10 – Mostra os menus para criação de objetos: Firewall, Interfaces, Redes e Hosts, e Regras.

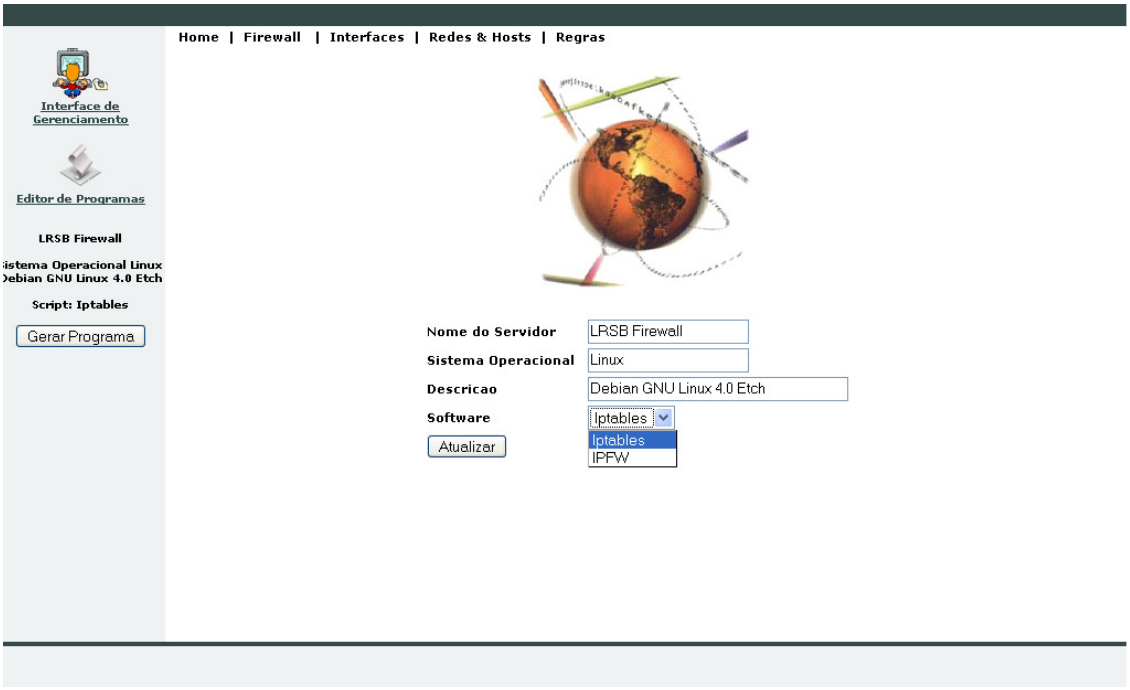


Figura 11 – Mostra a interface de definição do objeto firewall.

A Figura 11 mostra a interface de configuração do firewall. Nela é possível definir o nome do servidor, sistema operacional, descrição do servidor e software que será utilizado no mapeamento do script. Corresponde a um objeto da classe firewall descrita na sessão 4.4.2.

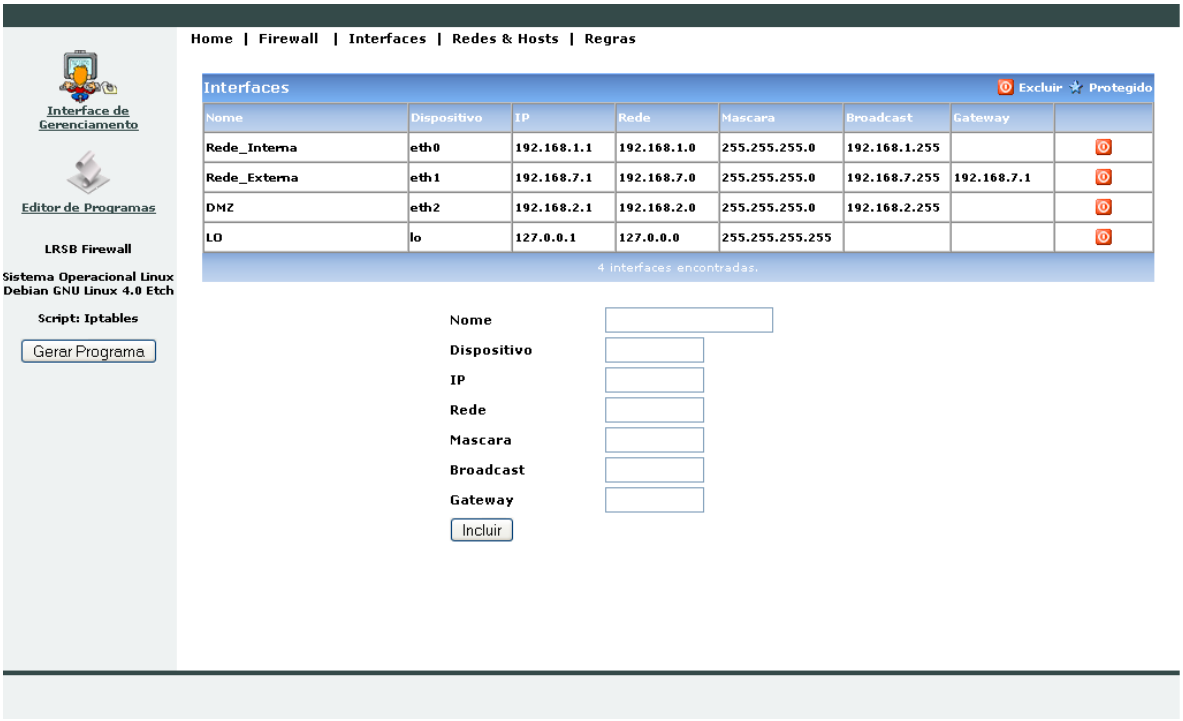


Figura 12 – Mostra a interface de definição dos objetos interfaces.

A Figura 12 mostra a interface de configuração das interfaces do firewall. Na tabela acima são mostrados os itens nome, dispositivo, ip, rede, mascara, broadcast, gateway e uma opção para exclusão da interface. Caso apareça um símbolo de protegido, significa que a interface está em uso por alguma regra de roteamento. Abaixo são mostrados os campos novamente para a inclusão de uma nova interface. Esta interface de configuração corresponde a definição dos objetos classe interface, descrita na sessão 4.4.2.

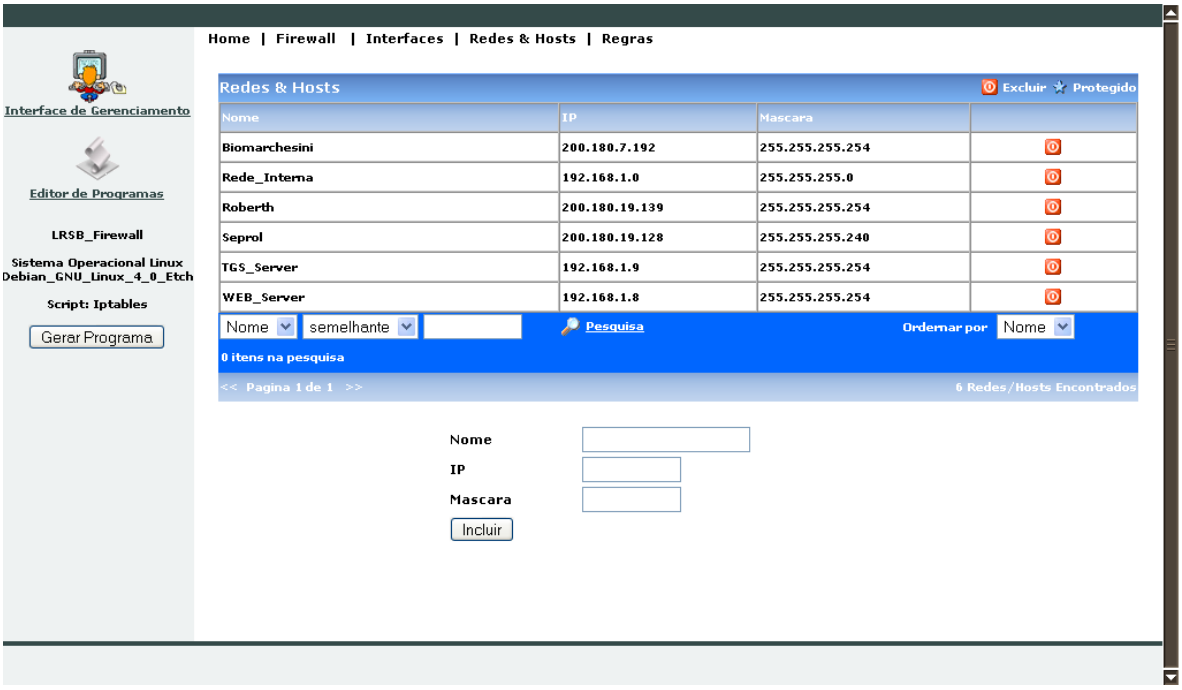


Figura 13 – Mostra a interface de definição dos objetos redes.

A Figura 13 mostra a interface de configuração das redes e hosts do firewall. Na tabela acima são mostrados os itens nome, ip, mascara, e uma opção para exclusão da rede. Caso apareça um símbolo de protegido, significa que a rede está em uso por alguma regra de roteamento. Abaixo é mostrado um formulário de pesquisa para facilitar a localização das redes. Mais abaixo são mostrados os campos novamente para a inclusão de uma nova rede. Esta interface de configuração corresponde a definição dos objetos classe rede, descrita na sessão 4.4.2.

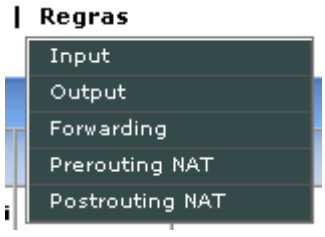


Figura 14 – Mostra o menu para as interfaces de tipos de pacotes.

A Figura 14 mostra os itens que podem ser escolhidos no menu regras, no caso as cinco formas de roteamento de pacotes.

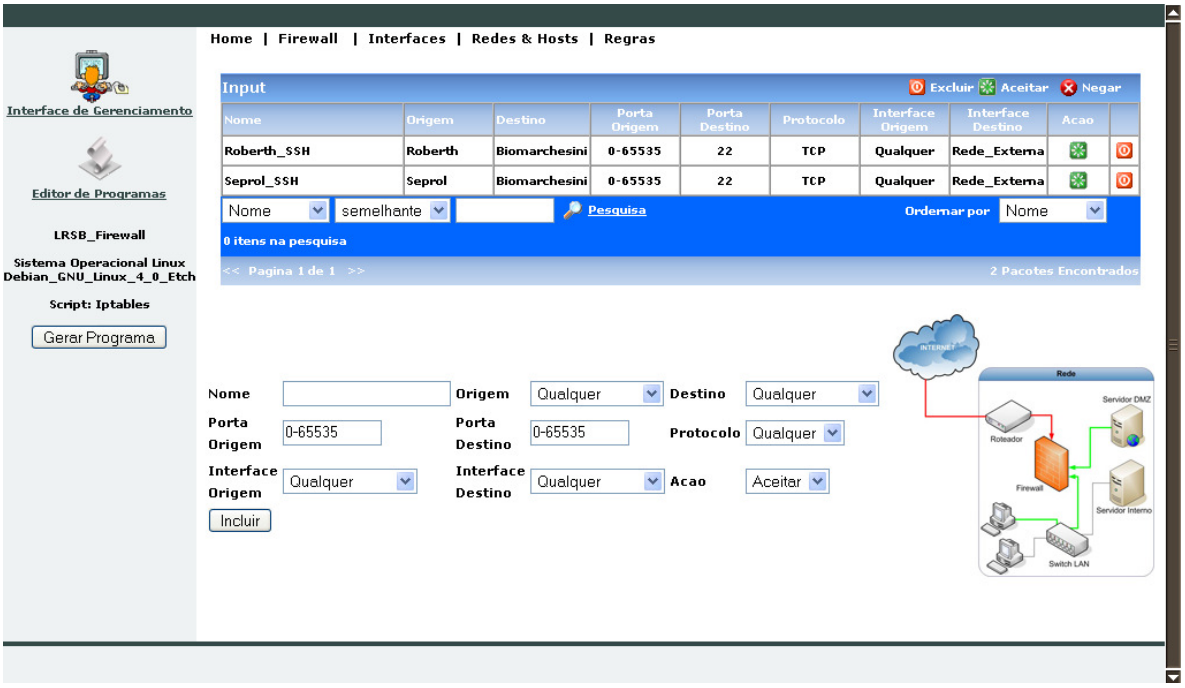


Figura 15 – Mostra a interface de definição dos objetos pacotes input.

A Figura 15 mostra a interface de configuração dos pacotes de entrada no firewall. Na tabela acima são mostrados os itens nome, origem, destino, porta de origem, porta de destino, protocolo, interface de origem, interface de destino, uma opção para aceitar e negar o pacote e uma opção para exclusão do pacote. Abaixo é mostrado um formulário de pesquisa para facilitar a localização dos pacotes. Mais abaixo são mostrados os campos novamente para a inclusão de um novo pacote e um gráfico dando uma explicação do que seria pacotes do tipo Input. Esta interface de configuração corresponde a definição dos objetos classe pacoteInput, descrita na sessão 4.4.2.

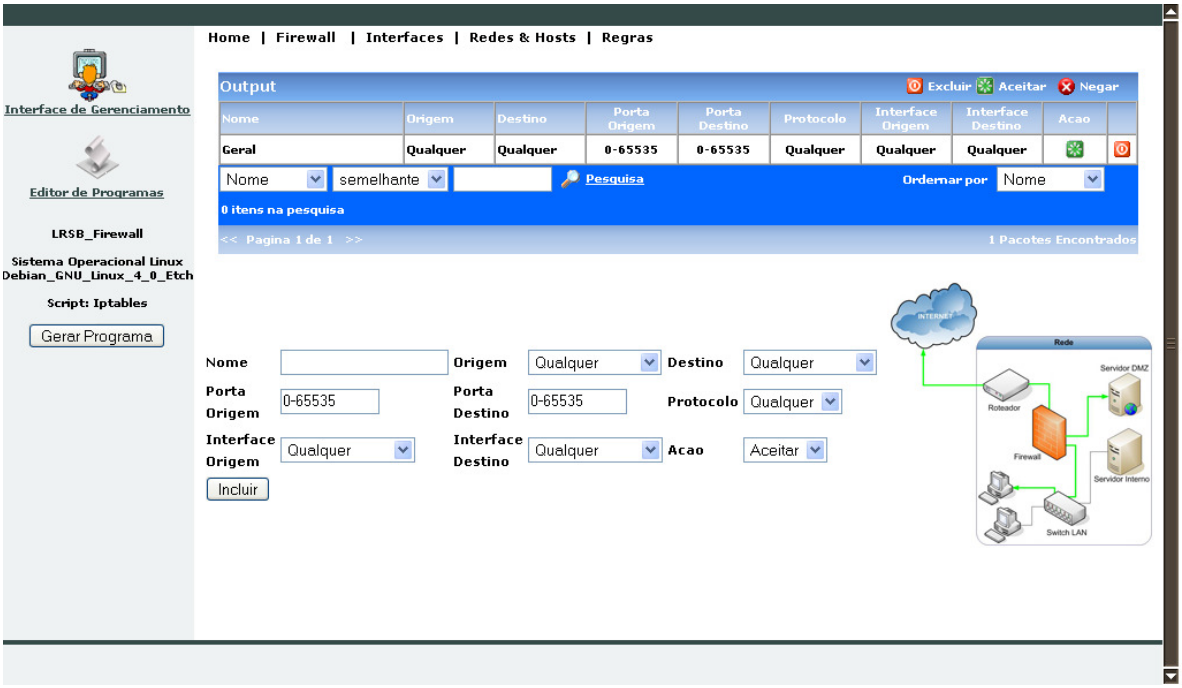


Figura 16 – Mostra a interface de definição dos objetos pacotes output.

A Figura 16 mostra a interface de configuração dos pacotes de saída do firewall. Na tabela acima são mostrados os itens nome, origem, destino, porta de origem, porta de destino, protocolo, interface de origem, interface de destino, uma opção para aceitar e negar o pacote e uma opção para exclusão do pacote. Abaixo é mostrado um formulário de pesquisa para facilitar a localização dos pacotes. Mais abaixo são mostrados os campos novamente para a inclusão de um novo pacote e um gráfico dando uma explicação do que seria pacotes do tipo Output. Esta interface de configuração corresponde a definição dos objetos classe pacoteOutput, descrita na sessão 4.4.2.

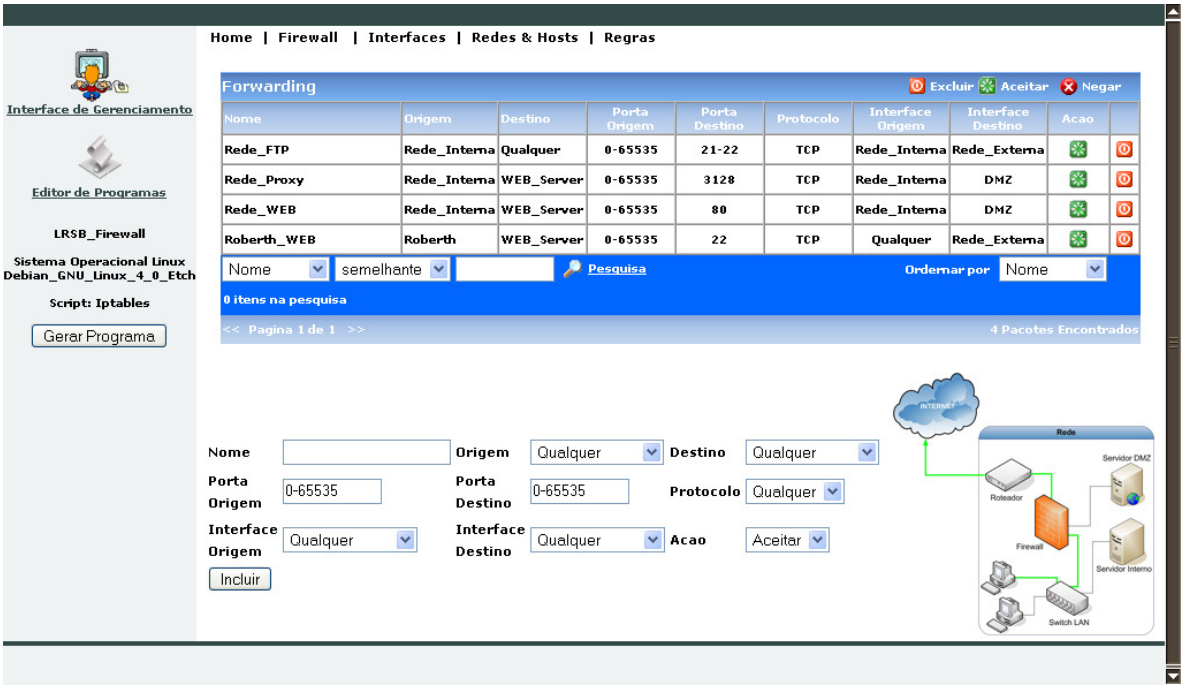


Figura 17 – Mostra a interface de definição dos objetos pacotes forwarding.

A Figura 17 mostra a interface de configuração dos pacotes que atravessam o firewall. Na tabela acima são mostrados os itens nome, origem, destino, porta de origem, porta de destino, protocolo, interface de origem, interface de destino, uma opção para aceitar e negar o pacote e uma opção para exclusão do pacote. Abaixo é mostrado um formulário de pesquisa para facilitar a localização dos pacotes. Mais abaixo são mostrados os campos novamente para a inclusão de um novo pacote e um gráfico dando uma explicação do que seria pacotes do tipo Forwarding. Esta interface de configuração corresponde a definição dos objetos classe pacoteForwarding, descrita na sessão 4.4.2.

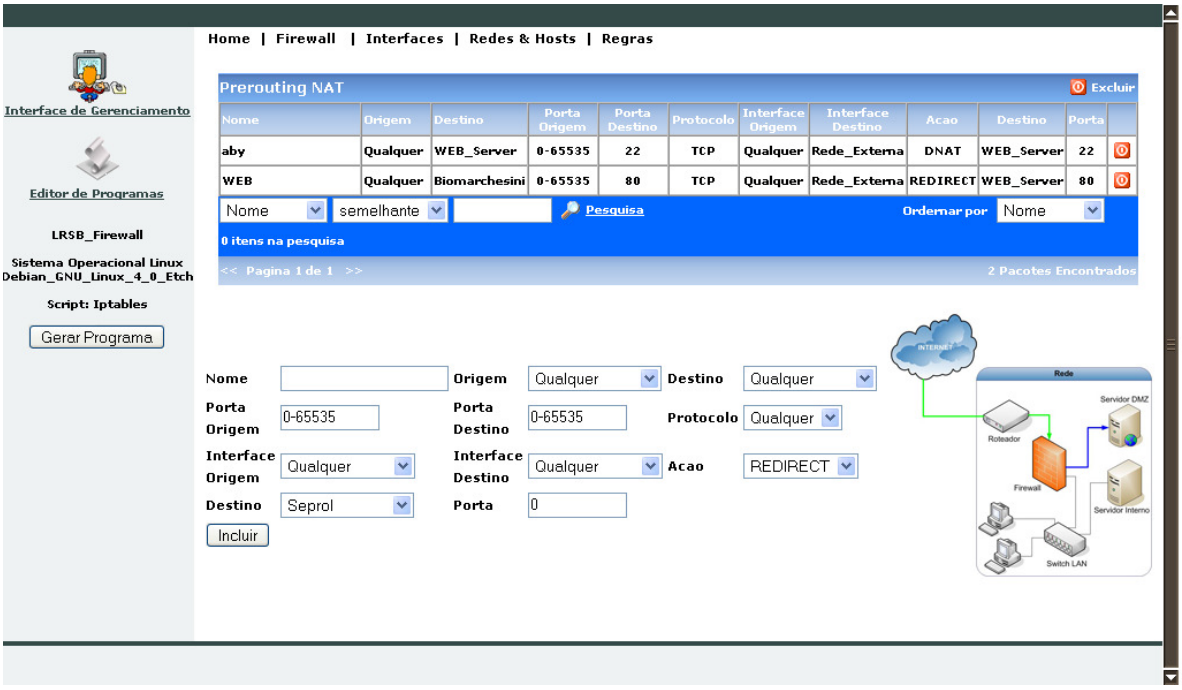


Figura 18 – Mostra a interface de definição dos objetos pacotes prerouting nat.

A Figura 18 mostra a interface de configuração dos pacotes que devem ser redirecionados no firewall, antes do roteamento. Na tabela acima são mostrados os itens nome, origem, destino, porta de origem, porta de destino, protocolo, interface de origem, interface de destino, ação, destino a ser redirecionado, porta a ser redirecionada e uma opção para exclusão do pacote. Abaixo é mostrado um formulário de pesquisa para facilitar a localização dos pacotes. Mais abaixo são mostrados os campos novamente para a inclusão de um novo pacote e um gráfico dando uma explicação do que seria pacotes do tipo Prerouting NAT. Esta interface de configuração corresponde a definição dos objetos classe pacotePrerouting, descrita na sessão 4.4.2.

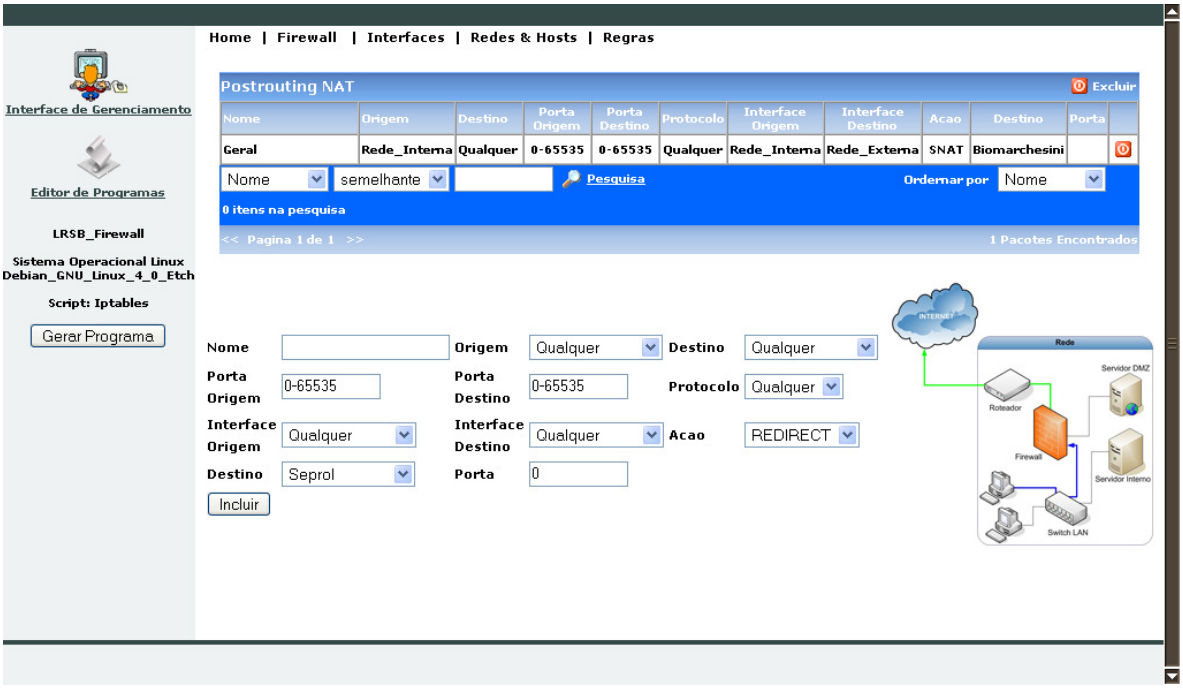


Figura 19 – Mostra a interface de definição dos objetos pacotes postrouting nat.

A Figura 19 mostra a interface de configuração dos pacotes que devem ser redirecionados no firewall, depois do roteamento. Na tabela acima são mostrados os itens nome, origem, destino, porta de origem, porta de destino, protocolo, interface de origem, interface de destino, ação, destino a ser redirecionado, porta a ser redirecionada e uma opção para exclusão do pacote. Abaixo é mostrado um formulário de pesquisa para facilitar a localização dos pacotes. Mais abaixo são mostrados os campos novamente para a inclusão de um novo pacote e um gráfico dando uma explicação do que seria pacotes do tipo Postrouting NAT. Esta interface de configuração corresponde a definição dos objetos classe pacotePostrouting, descrita na sessão 4.4.2.

Figura 20 – Mostra o botão para geração do programa.

A Figura 20 mostra um informativo sobre o nome, descrição e sistema operacional do firewall; e um botão para gerar um programa de acordo com as definições da interface de gerenciamento e transferi-lo para o editor de programas.

5.8.2 O Editor de Programas

Na interface de edição de programas é possível editar um programa definido pela interface de gerenciamento ou editar programas definidos pelo usuário.

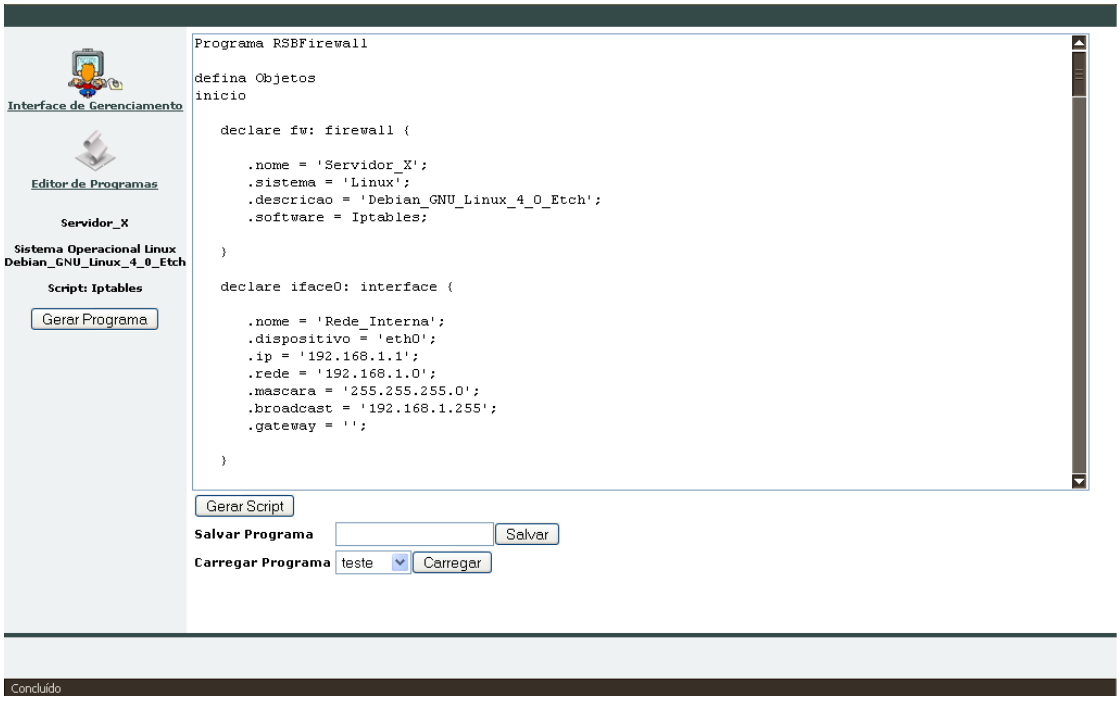


Figura 21 – Mostra a interface para edição de programas.

A Figura 21 mostra a interface do editor de programas. O botão *Gerar Script* inicia a execução da análise léxica, sintática e semântica do código escrito. Após a análise, é feita a geração do script a partir do código escrito.

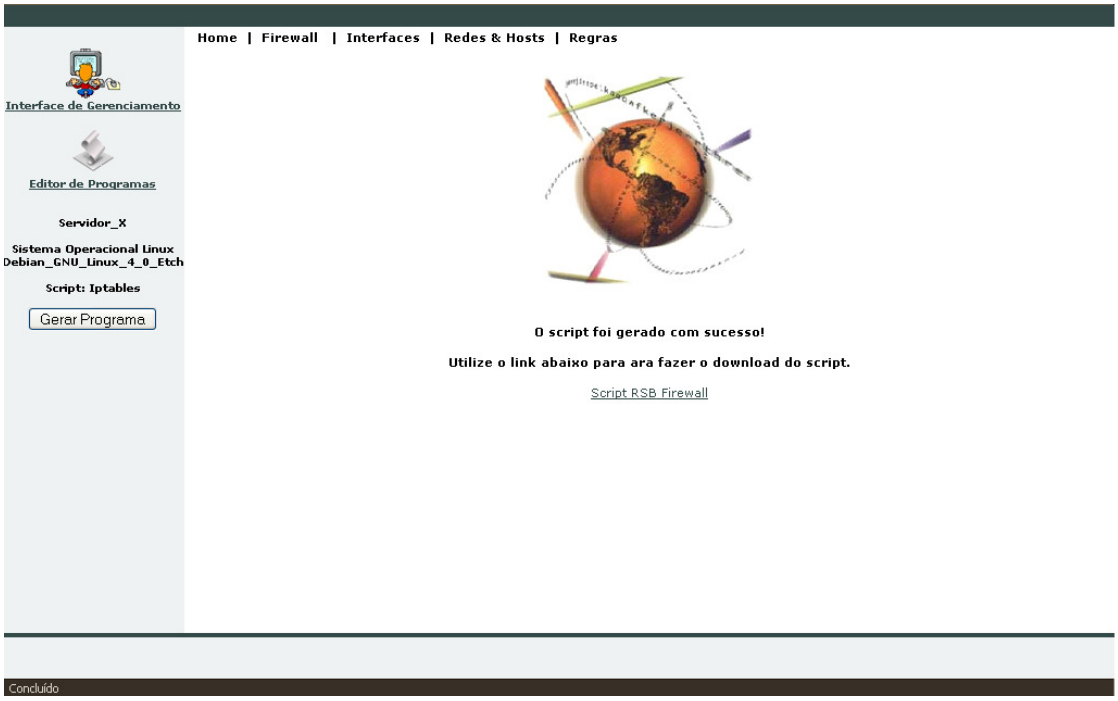


Figura 22 – Mostra a interface de download do script.

A Figura 22 mostra a interface de download do script gerado.

6. EXEMPLO DE CONFIGURAÇÃO DE UMA REDE NA RSBFL

Sendo a rede de uma empresa X:

- 15 estações de trabalho com sistema operacional Windows XP, estando configuradas na faixa de IP 192.168.1.0/24.
- 1 servidor firewall debian com iptables interligando a rede interna (192.168.1.0/24) a um modem ADSL (192.168.10.1), com IP 192.168.1.1 da interface interligada a rede interna e IP 192.168.10.2 da interface que interliga o modem ADSL (Rede Internet).

Deseja-se configurar o firewall com as seguintes restrições:

1. Permitir pacotes de entrada no firewall pela porta 22 (SSH) e 53(DNS) vindos da rede interna e do IP 200.145.768.223 (IP da empresa responsável pelo suporte).
2. Permitir qualquer tipo de saída de pacotes do firewall para ambas as redes.
3. Permitir a passagem de pacotes da rede interna para a rede Internet, pelas portas 80 (HTTP), 443(HTTPS), 21 (FTP) e 3389 (Remote Desktop).

Iniciaremos a configuração pela interface de gerenciamento:

Figura 23 – Mostra a definição do objeto firewall.

A Figura 23 mostra a configuração do sistema operacional do firewall e o software a ser utilizado, no caso o Iptables.

Figura 24 – Mostra a definição dos objetos interfaces.

A Figura 24 mostra a configuração das interfaces definidas na configuração do firewall, no caso a interface Interna e a interface Externa.

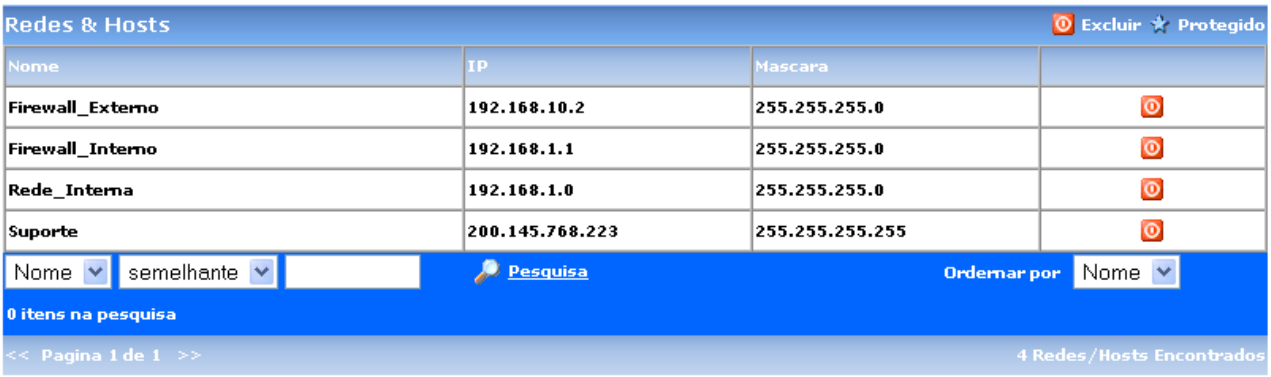


Figura 25 – Mostra a definição dos objetos redes.

A Figura 25 mostra a definição da Rede_Interna, do host Suporte, e dos IPs Internos e Externos do firewall, definidos do requisitos citados a cima. A rede Externa (Internet) não precisa ser definida pois pode ser tratada como qualquer.

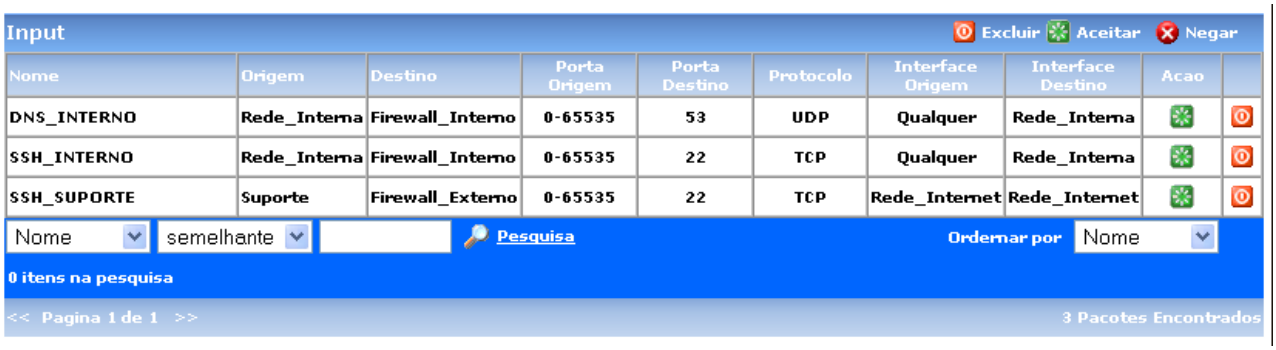


Figura 26 – Mostra a definição dos objetos pacotes input.

A Figura 26 mostra a definição das regras de Input definidas no item 1 dos requisitos.

Figura 27 – Mostra a definição dos objetos pacotes output.

A Figura 27 mostra as regras de Output definidas no item 2 dos requisitos.

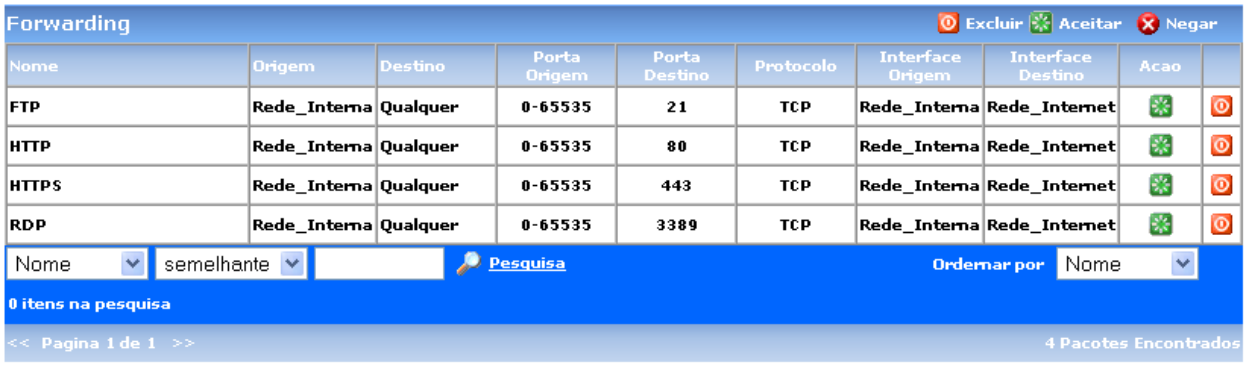


Figura 28 – Mostra a definição dos objetos pacotes forwarding.

A Figura 28 mostra as regras de Forwarding definidas no item 3 dos requisitos.

Com as regras definidas, foi gerado o código do programa através da opção *Gerar Programa* da interface de gerenciamento, obtendo o seguinte programa:

Programa RSBFirewall

defina Objetos
inicio

declare fw: **firewall** {

.nome = 'Servidor_X';

```
   .sistema = 'Linux';
   .descricao = 'Debian_GNU_Linux_4_0_Etch';
   .software = Iptables;

}

declare iface0: interface {

   .nome = 'Rede_Interna';
   .dispositivo = 'eth0';
   .ip = '192.168.1.1';
   .rede = '192.168.1.0';
   .mascara = '255.255.255.0';
   .broadcast = '192.168.1.255';
   .gateway = '';

}

declare iface1: interface {

   .nome = 'Rede_Internet';
   .dispositivo = 'eth1';
   .ip = '192.168.10.2';
   .rede = '192.168.10.0';
   .mascara = '255.255.255.0';
   .broadcast = '192.168.10.255';
   .gateway = '';

}

declare rd0: rede {
```

```
  .nome = 'Firewall_Externo';
  .ip = '192.168.10.2';
  .mascara = '255.255.255.0';

}

declare rd1: rede {

  .nome = 'Rede_Interna';
  .ip = '192.168.1.0';
  .mascara = '255.255.255.0';

}

declare rd2: rede {

  .nome = 'Suporte';
  .ip = '200.145.768.223';
  .mascara = '255.255.255.255';

}

declare rd3: rede {

  .nome = 'Firewall_Interno';
  .ip = '192.168.1.1';
  .mascara = '255.255.255.0';

}
```

```
declare pacotei0: pacoteInput {

  .nome = 'SSH_SUPORTE';
  .origem = Suporte;
  .destino = Firewall_Externo;
  .portaOrigem = 0-65535;
  .portaDestino = 22;
  .protocolo = TCP;
  .interfaceOrigem = Rede_Internet;
  .interfaceDestino = Rede_Internet;

}

declare pacotei1: pacoteInput {

  .nome = 'SSH_INTERNO';
  .origem = Rede_Interna;
  .destino = Firewall_Interno;
  .portaOrigem = 0-65535;
  .portaDestino = 22;
  .protocolo = TCP;
  .interfaceOrigem = Qualquer;
  .interfaceDestino = Rede_Interna;

}

declare pacotei2: pacoteInput {

  .nome = 'DNS_INTERNO';
  .origem = Rede_Interna;
  .destino = Firewall_Interno;
```

```
  .portaOrigem = 0-65535;
  .portaDestino = 53;
  .protocolo = UDP;
  .interfaceOrigem = Qualquer;
  .interfaceDestino = Rede_Interna;

}

declare pacoteo0: pacoteOutput {

  .nome = 'Geral';
  .origem = Qualquer;
  .destino = Qualquer;
  .portaOrigem = 0-65535;
  .portaDestino = 0-65535;
  .protocolo = Qualquer;
  .interfaceOrigem = Qualquer;
  .interfaceDestino = Qualquer;

}

declare pacotef0: pacoteForwarding {

  .nome = 'HTTPS';
  .origem = Rede_Interna;
  .destino = Qualquer;
  .portaOrigem = 0-65535;
  .portaDestino = 443;
  .protocolo = TCP;
  .interfaceOrigem = Rede_Interna;
  .interfaceDestino = Rede_Internet;
```

```
}

declare pacotef1: pacoteForwarding {

  .nome = 'FTP';
  .origem = Rede_Interna;
  .destino = Qualquer;
  .portaOrigem = 0-65535;
  .portaDestino = 21;
  .protocolo = TCP;
  .interfaceOrigem = Rede_Interna;
  .interfaceDestino = Rede_Internet;

}

declare pacotef2: pacoteForwarding {

  .nome = 'RDP';
  .origem = Rede_Interna;
  .destino = Qualquer;
  .portaOrigem = 0-65535;
  .portaDestino = 3389;
  .protocolo = TCP;
  .interfaceOrigem = Rede_Interna;
  .interfaceDestino = Rede_Internet;

}

declare pacotef3: pacoteForwarding {
```

```
    .nome = 'HTTP';
    .origem = Rede_Interna;
    .destino = Qualquer;
    .portaOrigem = 0-65535;
    .portaDestino = 80;
    .protocolo = TCP;
    .interfaceOrigem = Rede_Interna;
    .interfaceDestino = Rede_Internet;

  }

fim;

defina Funcoes
inicio

  funcao Iniciar() {

    fw.incluirInterface(iface0);
    fw.incluirInterface(iface1);

    fw.Aceitar(pacotei0);
    fw.Aceitar(pacotei1);
    fw.Aceitar(pacotei2);

    fw.Aceitar(pacoteo0);

    fw.Aceitar(pacotef0);
    fw.Aceitar(pacotef1);
    fw.Aceitar(pacotef2);
    fw.Aceitar(pacotef3);
```

```
  }
```

fim;

inicio

```
  Iniciar();
```

fim.

Com o programa definido, gerou-se então o script iptables, obtendo-se o seguinte resultado:

```
#!/bin/sh

#
# Definicao de Objetos
#

#
# Firewall
#
# Nome = Servidor_X
# Sistema = Linux
# Descricao = Debian_GNU_Linux_4_0_Etch
# Software = Iptalbes
#
```

```
#
# Interfaces
#

IFACE0_NOME=”Rede_Interna”
IFACE0_DISPOSITIVO=”eth0”
IFACE0_IP=”192.168.1.1”
IFACE0_REDE=”192.168.1.0”
IFACE0_MASCARA=”255.255.255.0”
IFACE0_BROADCAST=”192.168.1.255”

IFACE1_NOME=”Rede_Internet”
IFACE1_DISPOSITIVO=”eth1”
IFACE1_IP=”192.168.10.2”
IFACE1_REDE=”192.168.10.0”
IFACE1_MASCARA=”255.255.255.0”
IFACE1_BROADCAST=”192.168.10.255”
IFACE1_GATEWAY=”192.168.10.1”

#
# Redes
#

RD0_NOME=”Firewall_Externo”
RD0_IP=”192.168.10.2”
RD0_MASCARA=”255.255.255.0”

RD1_NOME=”Rede_Interna”
RD1_IP=”192.168.1.0”
RD1_MASCARA=”255.255.255.0”
```

```
RD2_NOME=”Suporte”
RD2_IP=”200.145.768.223”
RD2_MASCARA=”255.255.255.255”

RD3_NOME=”Firewall_Interno”
RD3_IP=”192.168.1.1”
RD3_MASCARA=”255.255.255.0”

#
# Pacotes Input
#

PACOTEI0_NOME=”SSH_SUPORTE”
PACOTEI0_ORIGEM=”200.145.768.223”
PACOTEI0_DESTINO=”192.168.10.2”
PACOTEI0_PORTA_ORIGEM=”0:65535”
PACOTEI0_PORTA_DESTINO=”22”
PACOTEI0_PROTOCOLO=”TCP”
PACOTEI0_INTERFACE_ORIGEM=”eth1”
PACOTEI0_INTERFACE_DESTINO=”eth1”

PACOTEI1_NOME=”SSH_INTERNO”
PACOTEI1_ORIGEM=”192.168.1.0”
PACOTEI1_DESTINO=”192.168.1.1”
PACOTEI1_PORTA_ORIGEM=”0:65535”
PACOTEI1_PORTA_DESTINO=”22”
PACOTEI1_PROTOCOLO=”TCP”
PACOTEI1_INTERFACE_ORIGEM=”all”
PACOTEI1_INTERFACE_DESTINO=”eth0”

PACOTEI2_NOME=”DNS_INTERNO”
```

```
PACOTEI2_ORIGEM=”192.168.1.0”
PACOTEI2_DESTINO=”192.168.1.1”
PACOTEI2_PORTA_ORIGEM=”0:65535”
PACOTEI2_PORTA_DESTINO=”53”
PACOTEI2_PROTOCOLO=”UDP”
PACOTEI2_INTERFACE_ORIGEM=”ALL”
PACOTEI2_INTERFACE_DESTINO=”eth0”

#
# Pacotes Output
#

PACOTEO0_NOME=”Geral”
PACOTEO0_ORIGEM=”0.0.0.0”
PACOTEO0_DESTINO=”0.0.0.0”
PACOTEO0_PORTA_ORIGEM=”0:65535”
PACOTEO0_PORTA_DESTINO=”0:65535”
PACOTEO0_PROTOCOLO=”ALL”
PACOTEO0_INTERFACE_ORIGEM=”ALL”
PACOTEO0_INTERFACE_DESTINO=”ALL”

#
# Pacotes Forwarding
#

PACOTEF0_NOME=”HTTPS”
PACOTEF0_ORIGEM=”192.168.1.0”
PACOTEF0_DESTINO=”0.0.0.0”
PACOTEF0_PORTA_ORIGEM=”0:65535”
PACOTEF0_PORTA_DESTINO=”443”
PACOTEF0_PROTOCOLO=”TCP”
```

PACOTEF0_INTERFACE_ORIGEM=”eth0”
PACOTEF0_INTERFACE_DESTINO=”eth1”

PACOTEF1_NOME=”FTP”
PACOTEF1_ORIGEM=”192.168.1.0”
PACOTEF1_DESTINO=”0.0.0.0”
PACOTEF1_PORTA_ORIGEM=”0:65535”
PACOTEF1_PORTA_DESTINO=”21”
PACOTEF1_PROTOCOLO=”TCP”
PACOTEF1_INTERFACE_ORIGEM=”eth0”
PACOTEF1_INTERFACE_DESTINO=”eth1”

PACOTEF2_NOME=”RDP”
PACOTEF2_ORIGEM=”192.168.1.0”
PACOTEF2_DESTINO=”0.0.0.0”
PACOTEF2_PORTA_ORIGEM=”0:65535”
PACOTEF2_PORTA_DESTINO=”3389”
PACOTEF2_PROTOCOLO=”TCP”
PACOTEF2_INTERFACE_ORIGEM=”eth0”
PACOTEF2_INTERFACE_DESTINO=”eth1”

PACOTEF3_NOME=”HTTP”
PACOTEF3_ORIGEM=”192.168.1.0”
PACOTEF3_DESTINO=”0.0.0.0”
PACOTEF3_PORTA_ORIGEM=”0:65535”
PACOTEF3_PORTA_DESTINO=”80”
PACOTEF3_PROTOCOLO=”TCP”
PACOTEF3_INTERFACE_ORIGEM=”eth0”
PACOTEF3_INTERFACE_DESTINO=”eth1”

#

```
# Definicao de Funcoes
#

Iniciar () {

        /sbin/iptables –A INPUT –p $PACOTEI0_PROTOCOLO \
        –s $PACOTEI0_ORIGEM –d $ PACOTEI0_DESTINO \
        –i $PACOTEI0_INTERFACE_ORIGEM \
        –o $PACOTEI0_INTERFACE_DESTINO \
        --sport $PACOTEI0_PORTA_ORIGEM \
        --dport $PACOTEI0_PORTA_DESTINO –j ACCEPT

        /sbin/iptables –A INPUT –p $PACOTEI1_PROTOCOLO \
        –s $PACOTEI1_ORIGEM –d $ PACOTEI1_DESTINO \
        –i $PACOTEI1_INTERFACE_ORIGEM \
        –o $PACOTEI1_INTERFACE_DESTINO \
        --sport $PACOTEI1_PORTA_ORIGEM \
        --dport $PACOTEI1_PORTA_DESTINO –j ACCEPT

        /sbin/iptables –A INPUT –p $PACOTEI2_PROTOCOLO \
        –s $PACOTEI2_ORIGEM –d $ PACOTEI2_DESTINO \
        –i $PACOTEI2_INTERFACE_ORIGEM \
        –o $PACOTEI2_INTERFACE_DESTINO \
        --sport $PACOTEI2_PORTA_ORIGEM \
        --dport $PACOTEI2_PORTA_DESTINO –j ACCEPT
        /sbin/iptables –A OUTPUT –p $PACOTEO0_PROTOCOLO \
        –s $PACOTEO0_ORIGEM –d $ PACOTEO0_DESTINO \
        –i $PACOTEO0_INTERFACE_ORIGEM \
        –o $PACOTEO0_INTERFACE_DESTINO \
        --sport $PACOTEO0_PORTA_ORIGEM \
        --dport $PACOTEO0_PORTA_DESTINO –j ACCEPT
```

```
        /sbin/iptables –A FORWARD –p $PACOTEF0_PROTOCOLO \
        –s $PACOTEF0_ORIGEM –d $ PACOTEF0_DESTINO \
        –i $PACOTEF0_INTERFACE_ORIGEM \
        –o $PACOTEF0_INTERFACE_DESTINO \
        --sport $PACOTEF0_PORTA_ORIGEM \
        --dport $PACOTEF0_PORTA_DESTINO –j ACCEPT

        /sbin/iptables –A FORWARD –p $PACOTEF1_PROTOCOLO \
        –s $PACOTEF1_ORIGEM –d $ PACOTEF1_DESTINO \
        –i $PACOTEF1_INTERFACE_ORIGEM \
        –o $PACOTEF1_INTERFACE_DESTINO \
        --sport $PACOTEF1_PORTA_ORIGEM \
        --dport $PACOTEF1_PORTA_DESTINO –j ACCEPT

        /sbin/iptables –A FORWARD –p $PACOTEF2_PROTOCOLO \
        –s $PACOTEF2_ORIGEM –d $ PACOTEF2_DESTINO \
        –i $PACOTEF2_INTERFACE_ORIGEM \
        –o $PACOTEF2_INTERFACE_DESTINO \
        --sport $PACOTEF2_PORTA_ORIGEM \
        --dport $PACOTEF2_PORTA_DESTINO –j ACCEPT

        /sbin/iptables –A FORWARD –p $PACOTEF3_PROTOCOLO \
        –s $PACOTEF3_ORIGEM –d $ PACOTEF3_DESTINO \
        –i $PACOTEF3_INTERFACE_ORIGEM \
        –o $PACOTEF3_INTERFACE_DESTINO \
        --sport $PACOTEF3_PORTA_ORIGEM \
        --dport $PACOTEF3_PORTA_DESTINO –j ACCEPT

}
```

```
modulos() {

        /sbin/depmod -a

         /sbin/modprobe ip_conntrack
         /sbin/modprobe ip_tables
         /sbin/modprobe iptable_filter
         /sbin/modprobe iptable_mangle
         /sbin/modprobe iptable_nat
         /sbin/modprobe ipt_LOG
         /sbin/modprobe ipt_MASQUERADE
         /sbin/modprobe ip_nat_ftp
         /sbin/modprobe ip_conntrack_ftp

         echo "1" > /proc/sys/net/ipv4/ip_forward

}

libera() {

        /sbin/iptables -P INPUT ACCEPT
        /sbin/iptables -P OUTPUT ACCEPT
        /sbin/iptables -P FORWARD ACCEPT
        /sbin/iptables -F
        /sbin/iptables -X
        /sbin/iptables -t nat -F

}

#
# Chamadas de inicialização do script
```

```
#

case "$1" in

    start)

        # carregando modulos
        modulos

        # esvaziando as tabelas e definindo regras padroes
        libera

        /sbin/iptables -P INPUT DROP
        /sbin/iptables -P OUTPUT DROP
        /sbin/iptables -P FORWARD DROP

        Iniciar

    ;;
    stop)

        libera

    ;;
    restart)

      $0 stop
      sleep 1
      $0 start

    ;;
```

```
esac
```

7. VANTAGENS E DESVANTAGENS

Vantagens:

1. Definição prática dos objetos na interface de gerenciamento.
2. Facilidade na organização de objetos e funções.
3. Facilidade na ativação/desativação de regras (funções).
4. Facilidade de migração entre os softwares Iptables e IPFW.

Desvantagens:

1. Nem todas as funcionalidades do Iptables e do IPFW estão sendo mapeadas pela RSBFL.
2. Não é possível a edição e compilação do código pelo console do servidor.

8. CONCLUSÃO

A LRSB Firewalls foi desenvolvida com a intuição de facilitar a criação de regras de filtragem de pacotes e de se poder utilizar um mesmo código em diferentes plataformas. No desenvolvimento do projeto viu-se que foi possível desenvolver a linguagem de maneira que o código ficasse de forma organizada e que esse mesmo código pudesse ser traduzido para outros ambientes, que no caso do projeto foi utilizado Iptables, para Linux, e IPFW, para FreeBSD.

A ferramenta GALS (Gerador de Analisadores Léxicos e Sintáticos) foi de extrema importância para o desenvolvimento do projeto. Através dela, foi possível desenvolver toda a parte de análise léxica e sintática da linguagem, sua gramática e expressões regulares de forma prática e eficiente.

A gramática da linguagem, organizada em blocos (bloco de objetos, funções e chamadas iniciais de funções), tornou a criação e edição de regras muito práticas. Com a possibilidade de definições de funções, foi possível habilitar e desabilitar facilmente um conjunto de regras agrupadas, definidas em função específica, apenas habilitando e desabilitando (através do uso de comentários) a chamada da função no bloco de chamadas de funções.

Na interface de gerenciamento foi possível definir os objetos e ações de maneira dinâmica e prática. Nela foi possível definir os objetos já corrigindo erros léxicos, sintáticos e semânticos, que poderiam ser cometidos através da edição do programa, no editor de programas.

9. TRABALHOS FUTUROS

1. Realizar o mapeamento completo das funções do Iptables e do IPFW.
2. Executar o mapeamento das regras de filtragem de pacotes para outros softwares, além do Iptables e IPFW.
3. Executar o mapeamento de softwares com outras funcionalidades, como por exemplo, o squid, que é um filtro de endereços HTTP.
4. Realizar o mapeamento do programa criado para a interface de gerenciamento. Atualmente é possível apenas realizar a geração do programa através dos objetos definidos na interface de gerenciamento.
5. Desenvolver um editor dirigido pela sintaxe da linguagem.

REFERÊNCIAS

FREEBSD FOUNDATION. **FreeBSD Handbook:** IPFW. Disponível em: <http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/firewalls-ipfw.html>. Acesso em: 12 mai. 2007.

FURTADO, Olinto José Varela. **Introdução a Compiladores**. Disponível em: <http://www.inf.ufsc.br/~olinto/>. Acesso em: 12 abr. 2007.

GESSER, Carlos Eduardo. **GALS:** O Gerador de Analisadores Léxicos e Sintáticos. Disponível em: <http://gals.sourceforge.net>. Acesso em: 02 fev. 2007.

LAGAN, Michael M. **Linux Online.** Disponível em: <http://www.linux.org>. Acesso em: 23 jan. 2007.

LINUX NA REDE. **FWBuilder:** Firewall para Iptables,Ipfw,Ipfilter,PIX …Disponível em: < http://www.linuxnarede.com.br/artigos/fullnews.php?id=87>. Acesso em: 27 abr. 2007.

MYSQLAB. **MySQL:** The world's most popular open source database. Disponível em: <http://www.mysql.com>. Acesso em: 25 mar. 2007.

NETCITADEL, LLC. **FWBuilder:** FWBuilder. Disponível em: <http://www.fwbuilder.org>. Acesso em: 27 abr. 2007.

NET FILTER CORE TEAM. **Iptables:** Firewalling, NAT, and packet mangling for Linux. Disponível em: <http://www.iptables.org>. Acesso em: 27 jan. 2007.

PAULA, Fábio Berbert de. **Viva o Linux:** Firewall e NAT em FreeBSD com controle de banda e redirecionamento de portas e IPs. Disponível em: <http://www.vivaolinux.com.br/artigos/verArtigo.php?codigo=2509&pagina=6>. Acesso em: 12 abr. 2007.

SOBRAL, João Bosco Mangueira. **Segurança em Computação Distribuída**. Disponível em: <http://www.inf.ufsc.br/~bosco/>. Acesso em: 12 abr. 2007.

SOFTWARE IN THE PUBLIC INTEREST. **Debian:** O Sistema Operacional Universal. Disponível em: <http://www.debian.org>. Acesso em: 23 jan. 2007.

SUN MICROSYSTEMS, INC. **Java:** Java Technology: <http://java.sun.com>. Acesso em: 23 mar. 2007.

THE PHP GROUP. **PHP:** Hypertext Precessor: <http://www.php.net>. Acesso em: 20 fev. 2007.

THE RFC ARCHIVE. **RFC 2979:** Behavior of and Requirements for Internet Firewalls: <http://www.rfc-archive.org/getrfc.php?rfc=2979>. Acesso em: 05 mai. 2007.