UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO DE APLICAÇÃO WEB PARA GERENCIAMENTO DE FIREWALL LINUX

RÉGIS MACIEL BORSCHEID

BLUMENAU
2005

2005/2-01

RÉGIS MACIEL BORSCHEID

PROTÓTIPO DE APLICAÇÃO WEB PARA GERENCIAMENTO DE FIREWALL LINUX

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Prof. Francisco Adell Péricas – Orientador

BLUMENAU
2005

2005/2-01

PROTÓTIPO DE APLICAÇÃO WEB PARA GERENCIAMENTO DE FIREWALL LINUX

Por

RÉGIS MACIEL BORSCHEID

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

	_____
Presidente:	Prof. Francisco Adell Péricas, Ms – Orientador, FURB
	_____
Membro:	Prof. Paulo Fernando Silva, Ms – FURB
	_____
Membro:	Prof. Sérgio Stringari, Ms – FURB

Blumenau, 16 de agosto de 2005

Dedico este trabalho a todas as pessoas que me ajudaram diretamente na realização deste e em especial à minha família pelo incentivo para alcançar mais este objetivo.

AGRADECIMENTOS

À Deus, pelo seu imenso amor e graça.

À minha família, que mesmo longe, sempre esteve presente.

Aos meus amigos de faculdade, ao meu amigo Charles Burkhart por suas contribuições com sua experiência na área de redes e *Firewall*.

Ao meu orientador, Francisco Adell Péricas, por ter acreditado na conclusão deste trabalho.

Para findar, agradeço a todos os professores e demais pessoas que me ajudar a conquistar mais esta vitória.

Os bons livros fazem “sacar” para fora o que a pessoa tem de melhor dentro dela.

Lina Sotis Francesco Moratti

RESUMO

Este trabalho apresenta o desenvolvimento de um protótipo de software para gerenciamento de *Firewall* linux via *web*. Apresenta também formas de implementação de *Firewalls* e assuntos relacionados a segurança. Utilizou –se o *Firewall Iptables* como sendo o *Firewall* do projeto, utilizando o algoritmo MD5 (*Message Digest* 5) para fazer a autenticação no servidor da aplicação.

Palavras-chave: Internet. Segurança. Gerência de *firewall*.

ABSTRACT

This work presents the development of a software prototype for managing a linux Firewall base don web. It also presents implementation methods for Firewalls and subjects related to security. The Firewall used in the project was the Iptables, with the MD5 algorithm (Messege Digest 5) to autenticate the application in the server.

Key-words: Internet. Security. Firewall management.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

DMZ – *DeMilitarized Zone*

DNAT – *Destination Network Address Translation*

DNS – *Domain Name System*

FTP – *File Transfer Protocol*

HTTP – *Hipertex Trasfer Protocol*

HTTPS – *HiperText Transfer Protocol over Secure Socker Layer*

IDENT – *Identification Protocol*

LAN – *Local Area Network*

MD5 – *Message Digest* 5

NAT – *Network Address Translation*

SMTP – *Simple Mail Transfer Protocol*

SNAT – *Source Network Address Translation*

SSH – *Secure Shell*

TCP – *Transmission Control Protocol*

UDP – *User Datagram Protocol*

WWW – *World Wide Web*

SUMÁRIO

1 INTRODUÇÃO

Com o crescimento das redes de computadores, torna-se cada vez mais necessária a interligação das redes privadas com as redes públicas. Conseqüentemente, os recursos de hardware e software das empresas ficam expostos a ameaças externas ou mesmo internas, onde falhas de segurança podem causar impactos dos mais diferentes níveis, que podem ir desde constrangimentos até perda de mercado. Em função da necessidade de proteger os dados, os recursos e os próprios computadores, surgiram ferramentas de bloqueio de acessos indesejados, denominadas *Firewall* (Segurança, 2000).

O software de *Firewall* é um dispositivo que fica instalado em um *host* ou servidor que interconecta a rede interna e a rede externa e tem a finalidade de proteger a rede interna filtrando e analisando os pacotes que transitam por através dele. Baseados nessas análises, pode avaliar se os pacotes podem transitar pela rede ou devem ser descartados. Além disso, pode ser instalado dentro de uma rede corporativa com o objetivo de diferenciar e delimitar setores dentro de uma organização (STANGER; LANE, 2002, p.426).

Dentre as funções básicas de um *Firewall* pode-se citar:

a) bloquear dados de entrada que possam conter um ataque de hacker;

b) ocultar informações sobre a rede interna, fazendo com que pareça que todo o trafego de saída seja proveniente do *Firewall* e não da rede, denominado *Network Address Translation* (NAT);

c) filtrar o tráfego de saída.

No sistema operacional Linux essas ferramentas vem sendo aperfeiçoadas constantemente, oferecendo cada vez mais recursos, conseqüentemente aumentado a segurança quando efetuada uma boa implementação.

Atualmente o pacote *Iptables* é o mais utilizado para *Firewall* Linux. Os precursores

desse pacote de *Firewall* são: *Ipfwadm* e o *Ipchains* respectivamente (STANGER; LANE, 2002, p. 424). A configuração de qualquer um desses pacotes de *Firewall* consiste basicamente em desenvolver regras, onde primeiramente todo o tráfego entre duas ou mais redes é bloqueado e em um segundo momento é liberado conforme a necessidade do ambiente onde essa ferramenta está sendo implementada. Mesmo no *Iptables*, uma das ferramentas mais utilizadas, a criação dessas regras não é simples. As mesmas são implementadas em sua maior parte sem o auxílio de ferramentas gráficas, dificultando a criação de regras por pessoas pouco experientes na área.

A solução que será apresentada nesse trabalho é o desenvolvimento de um protótipo aplicação *web* para realizar a configuração e manutenção de um sistema de *Firewall*.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é criar um protótipo de aplicação web para gerenciamento de *Firewall* Linux baseado em *Iptables*.

Os objetivos específicos do trabalho são:

a) implementar algoritmos de criação de regras de *Firewall* a partir de parâmetros passados pelos usuários;

b) utilizar servidor *web* Apache para armazenamento da aplicação e autenticação dos usuários;

c) criar regras para demonstração da aplicação.

1.2 ESTRUTURA DO TRABALHO

A estrutura deste trabalho está dividido em capítulos que serão explicados a seguir.

O primeiro capítulo apresenta a contextualização, problemas de pesquisa, objetivo e justificativa para o desenvolvimento do trabalho.

O segundo capítulo aborda segurança de redes e conceitos como: objetivos de segurança da informação e políticas de segurança.

O terceiro capítulo aborda o tema *Firewall* e está dividido em tipos de *Firewall*, escolha de um *Firewall*, localização de um *Firewall*, filtragem de pacotes e arquitetura de um *Firewall*.

O quarto capítulo aborda o tema *Iptables* e está dividido em: tabela *Filter*, tabela *Nat* e tabela *Mangle*.

O quinto capítulo abrange o desenvolvimento do trabalho, contextualizando a criação de regras de *Firewall*, mostrando a especificação com diagramas de casos de uso, diagrama de atividades, diagrama de classes e diagrama de seqüência. Abrange ainda a implementação e resultados obtidos com o desenvolvimento do protótipo.

O sexto capítulo descreve as conclusões obtidas no desenvolvimento da monografia.

2 SEGURANÇA EM REDES

Segundo Dias (2000), na sociedade da informação, ao mesmo tempo que as informações são consideradas o principal patrimônio de uma organização, elas estão também sob constante risco como nunca estiveram antes. Com isso, a segurança de informações tornou-se um ponto crucial para a sobrevivência das instituições.

A grande demanda por redes de computadores interligadas através da Internet, traz consigo responsabilidades com a segurança dos dados trafegados e armazenados. Existe uma grande preocupação com o funcionamento correto e confiável destas redes, haja visto que a dependência de atividades essenciais, de todos os tipos, crescem a cada dia. Além disso, a Internet vem tendo um crescimento muito significativo de atividades comerciais sendo desenvolvidas por seu intermédio. Por outro lado, os ataques intrusivos a redes de computadores têm crescido tanto em número quanto em quantidade de máquinas envolvidas. Isso faz com que técnicas de segurança se tornem indispensáveis nos sistemas computacionais modernos (NBSO, 2005).

2.1 OBJETIVOS DA SEGURANÇA

Segundo Dias (2000), como existem várias formas de implementação de segurança, os objetivos de segurança variam com o tipo de ambiente computacional e a natureza do sistema. Para identificar os objetivos prioritários para uma determinada organização é essencial fazer uma análise da natureza das aplicações, dos riscos e impactos prováveis em caso de falha de segurança. Para traçar esses objetivos os seguintes quisitos devem ser observados:

a) confidencialidade ou privacidade – proteger as informações contra o acesso de qualquer pessoa não explicitamente autorizada pelo dono da informação, isto é, as

informações e processos são liberados apenas a pessoas autorizadas;

b) integridade de dados – evitar que dados sejam apagados ou de alguma forma alterados, sem a permissão do proprietário da informação. O conceito de dados nesse objetivo é mais amplo, englobando dados, programas, documentação, registros, fitas ou unidades de *backup*. O conceito de integridade está relacionado com o fato de assegurar que dados não foram modificados por pessoas não autorizadas. Em termos de comunicação de dados, a integridade restringe-se à detecção de alterações deliberadas ou acidentais nos dados transmitidos;

c) disponibilidade – proteger os serviços de informática de tal forma que não sejam degradados ou tornados indisponíveis sem a devida autorização. Para um usuário autorizado, um sistema não disponível quando se necessita dele, pode ser tão ruim quanto um sistema inexistente ou destruído. As medidas relacionadas a esse objetivo, podem ser a duplicação de equipamentos ou *backup*. Disponibilidade pode ser definida como a garantia de que os serviços prestados por um sistema são acessíveis, sob demanda, aos usuários autorizados;

d) consistência – certificar-se de que o sistema atua de acordo com as expectativas dos usuários autorizados;

e) isolamento – regular o acesso ao sistema;

f) confiabilidade – garantir que, mesmo em condições adversas, o sistema atuará conforme o esperado.

Apesar de todos os objetivos citados serem importantes, dependendo do tipo de organização, alguns são mais ou menos relevantes: sistemas com necessidades de segurança diferentes devem ser tratados e protegidos também de formas diferentes.

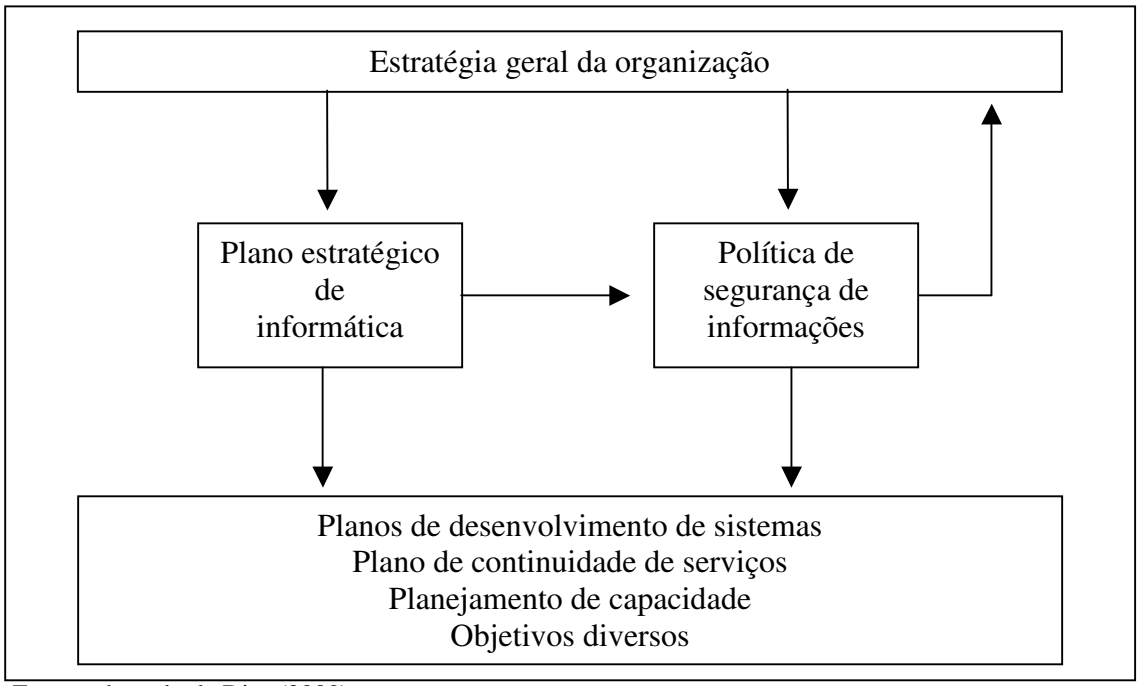
2.2 POLÍTICA DE SEGURANÇA

Uma política de segurança consiste num conjunto formal de regras que devem ser seguidas pelos usuários dos recursos de uma organização. As políticas de segurança devem ter uma implementação realista e definir claramente as áreas de responsabilidade dos usuários, do pessoal de gestão de sistemas e redes e da direção. Devem também adaptar-se a alterações na organização. As políticas de segurança fornecem um enquadramento para as implementações de mecanismos de segurança, definem procedimentos de segurança adequados, processos de auditoria à segurança e estabelecem uma base para procedimentos legais na sequência de ataques. O documento que define a política de segurança deve deixar de fora todos os aspetos técnicos de implementação dos mecanismos de segurança, pois essa implementação pode variar ao longo do tempo. Deve ser também um documento de fácil leitura e compreensão, além de resumido (WIKIPEDIA, 2005).

A política normalmente contém princípios legais e éticos a serem atendidos no que diz respeito à informática: direitos de propriedade de produção intelectual; direitos sobre softwares e normas legais correlatas aos sistemas desenvolvidos; princípios de implementação da segurança de informações; políticas de controle de acesso a recursos e sistemas computacionais; e princípios de supervisão constante das tentativas de violação da segurança da informação. Além disso, a política pode conter ainda os princípios de continuidade de negócios, procedimentos a serem adotados após a violação de normas de segurança estabelecidas na política, como investigações, julgamento e punições aos infratores da política e plano de treinamento em segurança de informações. É importante que a política estabeleça responsabilidades das funções relacionadas com a segurança e discrimine as principais ameaças, riscos e impactos envolvidos (DIAS, 2000).

De acordo com Silva (2004), a política de segurança deve ir além dos aspectos de

sistemas de informação e recursos computacionais, integrando as políticas institucionais relativas à segurança em geral, às metas de negócios da organização e ao plano estratégico de informática. O objetivo da política de segurança é atingido quando o relacionamento da estratégia da organização, o plano estratégico de informática e demais projetos estiverem sincronizados (Figura 1).



Fonte: adaptado de Dias (2000).

Figura 1 - Política de segurança e seus relacionamentos.

Conforme NBSO (2003), antes que a política de segurança seja escrita, é necessário definir a informação a ser protegida. Usualmente, isso é feito através de uma análise de riscos que identifica:

a) recursos protegidos pela política;

b) ameaças às quais estes recursos estão sujeitos;

c) vulnerabilidades que podem viabilizar a concretização destas ameaças, analisando-as individualmente.

Cabe ressaltar que cada organização possui um ambiente distinto e os seus próprios requisitos de segurança, e deve, portanto, desenvolver uma política de segurança que se molde a essas peculiaridades.

3 FIREWALL

O *Firewall* é um sistema que impõe uma política de controle de acesso entre duas redes, tendo as seguintes propriedades (CHESWICK, BELLOVIN e RUBIN, 2003):

a) todo tráfego de dentro para fora de uma rede, e vice-versa, deve passar pelo *Firewall*;

b) apenas tráfego autorizado, como definido pela política de segurança local, terá permissão de passar;

c) o próprio *Firewall* é imune a penetrações.

Conforme Neto (2004), um *Firewall* atuando como um ponto de indução, ou seja, sendo o único computador diretamente conectado a internet, poderá de forma segura levar serviços de interconectividade à sua rede local, evitando assim que cada *host* de sua *Local Area Network* (LAN) seja responsável por sua segurança.

De acordo com a NBSO (2003), um *Firewall* é um instrumento importante para implantar a política de segurança da sua rede. Ele pode reduzir a informação disponível externamente sobre a sua rede, ou, em alguns casos, até mesmo barrar ataques a vulnerabilidades ainda não divulgadas publicamente e para as quais correções não estão disponíveis. Por outro lado, *Firewalls* não são infalíveis. A simples instalação de um *Firewall* não garante que sua rede esteja segura contra invasores. Um *Firewall* não pode ser a sua única linha de defesa, ele é mais um dentre os diversos mecanismos e procedimentos que aumentam a segurança de uma rede. Outra limitação dos *Firewalls* é que eles protegem apenas contra ataques externos ao *Firewall*, nada podendo fazer contra ataques que partem de dentro da rede por ele protegida.

3.1 TIPOS DE FIREWALL

De acordo com Roger (2005), atualmente temos três tipos de *Firewalls*: filtro de pacotes, filtro de pacotes com base no estado da conexão e filtros de pacotes na camada de aplicação.

Os filtros de pacotes funcionam permitindo ou eliminando pacotes com base em seus endereços de origem ou destino, ou nos números de porta. As decisões são tomadas com base no conteúdo do pacote que o *Firewall* está recebendo ou enviando. Por exemplo, todos os computadores de uma rede local podem acessar páginas na Internet (origem): rede LAN, (destino): Internet, aceita: protocolo *HiperText Transfer Protocol* porta 80 (HTTP), onde os roteadores são exemplos de *Firewalls* de filtro de pacotes.

O *Firewall* com filtro de pacotes com base no estado da conexão (*stateful packet Filter*) baseia suas ações utilizando dois elementos: dados contidos no cabeçalho do pacote e na tabela de estados, que armazena informações do estado de todas as conexões que estão trafegando através do *Firewall* e usa estas informações, em conjunto com as regras definidas pelo administrador, na hora de tomar a decisão de permitir ou não a passagem de um determinado pacote. Como exemplo podemos citar o protocolo *File Transfer Protocol* (FTP): o *Firewall* com filtro de pacotes com base no estado da conexão consegue analisar todo o tráfego da conexão FTP, identificando qual o tipo de transferência que será utilizada (ativa ou passiva) e quais as portas que serão utilizadas para estabelecer a conexão. Sendo assim, todas as vezes que o *Firewall* identifica que uma transferência de arquivos estará sendo realizada, é acrescentado uma entrada na tabela de estados, permitindo que a conexão seja estabelecida. As informações ficam armazenadas na tabela somente enquanto a transferência do arquivo é realizada.

Os *Firewalls* com filtros na camada de aplicação são mais complexos, pois utilizam

um código especial para filtrar a aplicação desejada. Por exemplo, os *Firewalls* com filtros na camada de aplicação podem identificar vírus anexos às mensagens (*e-mails*) que estão chegando ou saindo do seu ambiente computacional. Outro recurso disponível neste tipo de *Firewall* são os registros de todo o conteúdo do tráfego enviado ou recebido.

3.2 ESCOLHA DE UM FIREWALL

De acordo NBSO (2003), a escolha de uma solução de *Firewall* está atrelada a fatores como custo, recursos desejados e flexibilidade, mas um ponto essencial é a familiaridade com a plataforma operacional do *Firewall*. A maioria dos *Firewalls* está disponível para um conjunto reduzido de plataformas operacionais, e a sua escolha deve se restringir a um dos produtos que roda sobre uma plataforma com a qual os administradores da rede tenham experiência. Existem, basicamente, duas razões para esta recomendação. A primeira delas é que você deve estar familiarizado o suficiente com o sistema onde o *Firewall* será executado para configurá-lo de forma segura. A existência de um *Firewall* instalado em um sistema inseguro pode ser até mais perigosa do que a ausência do *Firewall* na rede. A segunda razão é que os produtos tendem a seguir a filosofia da plataforma onde rodam; por exemplo, a maioria dos *Firewalls* para Windows é configurada através de menus e janelas, ao passo que muitos *Firewalls* para Unix são configurados por meio de arquivos texto.

3.3 LOCALIZAÇAO DE UM FIREWALL

De acordo com Santos (2005) a localização dos *Firewalls* na rede depende normalmente da sua política de segurança, entretanto, existem algumas regras que se aplicam à grande maioria dos casos:

a) todo o tráfego deve passar pelo *Firewall*. Um *Firewall* só pode atuar sobre o tráfego que passa por ele. A eficácia pode ser severamente comprometida se existirem rotas alternativas para dentro da rede (modems, por exemplo). Caso não seja possível eliminar todos esses caminhos, eles devem ser documentados e fortemente vigiados através de outros mecanismos de segurança;

b) deve-se ter um filtro de pacotes no perímetro da rede. Esse filtro pode estar localizado entre o roteador de borda e o interior da rede ou no próprio roteador, se ele tiver esta capacidade. O filtro de pacotes de borda é importante para tarefas como bloqueio global de alguns tipos de tráfego e bloqueio rápido de serviços durante a implantação de correções após a descoberta de uma nova vulnerabilidade;

c) deve-se colocar os servidores externos em uma *DeMilitarized Zone* (DMZ). É recomendável colocar os servidores acessíveis externamente (W*eb*, FTP, correio eletrônico, etc) em um segmento de rede separado e com acesso altamente restrito, conhecido como DMZ. A principal importância disso é proteger a rede interna contra ataques provenientes dos servidores externos. Por exemplo, suponha que um atacante invada o servidor *Web* e instale um *sniffer* na rede. Se este servidor *Web* estiver na rede interna, a probabilidade dele conseguir capturar dados importantes (tais como senhas ou informações confidenciais) é muito maior do que se ele estiver em uma rede isolada;

d) considere o uso de *Firewalls* internos. Em alguns casos, é possível identificar na rede interna grupos de sistemas que desempenham determinadas tarefas comuns, tais como desenvolvimento de software, *webdesign* e administração financeira. Nestes casos, recomenda-se o uso de *Firewalls* internos para isolar estas sub-redes umas das outras, com o propósito de aumentar a proteção dos sistemas internos e

conter a propagação de ataques bem-sucedidos.

3.4 FILTRAGEM DE PACOTES DE UM FIREWALL

De acordo com Ribeiro (2004), existem basicamente dois critérios de filtragem que podem ser empregados em *Firewalls*. O primeiro é o de *default deny*, ou seja, todo o tráfego que não for explicitamente permitido é bloqueado. O segundo *default allow*, é o contrário, ou seja, todo o tráfego que não for explicitamente proibido é liberado.

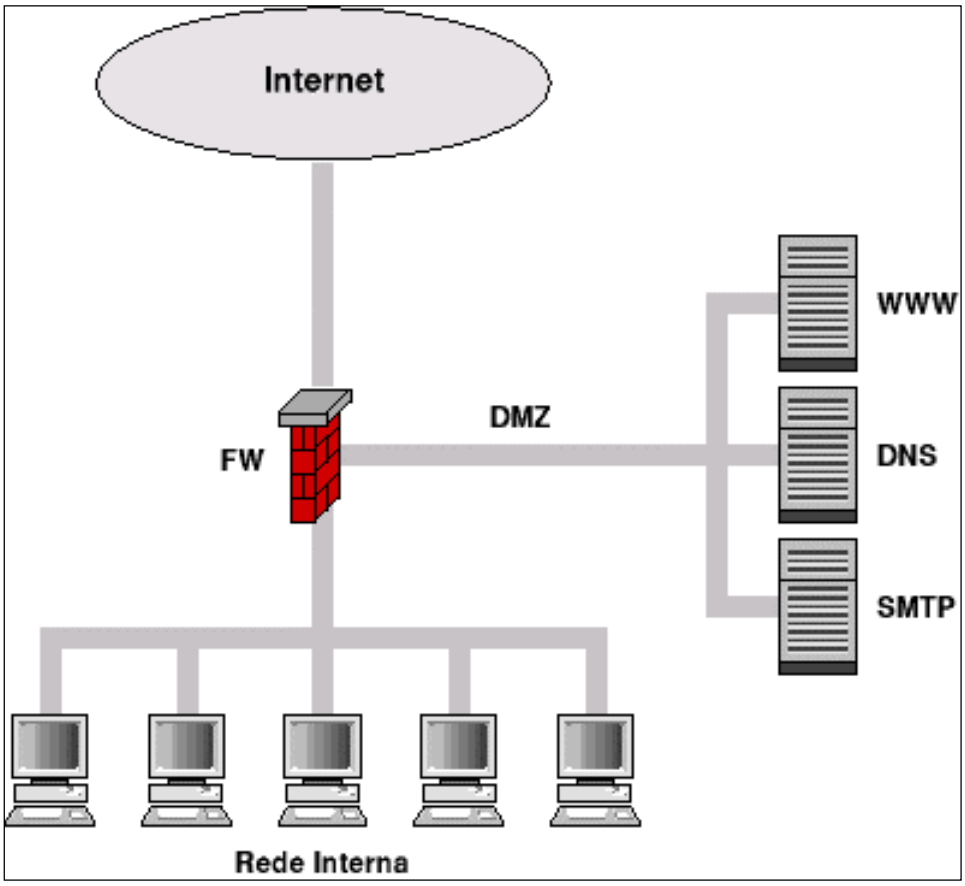
A configuração dos *Firewalls* deve seguir a política de segurança da rede. Se a política permitir, é recomendável adotar uma postura de *default deny*. Esta abordagem é, geralmente, mais segura, pois requer uma intervenção explícita do administrador para liberar o tráfego desejado, o que minimiza o impacto de eventuais erros de configuração na segurança da rede. Além disso, ela tende a simplificar a configuração dos *Firewalls*.

O tráfego para a DMZ deve ser altamente controlado. As únicas conexões permitidas para os sistemas dentro da DMZ devem ser as relativas aos serviços públicos (acessíveis externamente). Conexões partindo da DMZ para a rede interna devem ser na sua maioria, tratadas como conexões oriundas da rede externa, aplicando-se a política de filtragem correspondente.

3.5 ARQUITETURAS DE FIREWALL

Diversas arquiteturas podem ser empregadas para a implantação de *Firewalls* em uma rede. A opção por uma delas obedece a uma série de fatores, incluindo a estrutura lógica da rede a ser protegida, custo, funcionalidades pretendidas e requisitos tecnológicos dos *Firewalls*.

A seguir serão apresentadas duas arquiteturas de *Firewall*, exemplos de arquiteturas que funcionam e que podem eventualmente ser adotados (na sua forma original ou após passarem por adaptações) em situações reais.



Fonte: NBSO (2003).

Figura 2 - Exemplo de um *Firewall* simples.

Na Figura 2, o *Firewall* possui três interfaces de rede: uma para a rede externa, uma para a rede interna e outra para a DMZ. Por default, este *Firewall* bloqueia tudo o que não for explicitamente permitido (*default deny*). Além disso, o *Firewall* usado é do tipo *stateful*, que gera dinamicamente regras que permitam a entrada de respostas das conexões iniciadas na rede interna; portanto, não é preciso incluir na configuração do *Firewall* regras de entrada para estas respostas.

A seguir será apresentado o tráfego liberado em cada interface de *Firewall* da Figura 2:

a) interface externa:

- saída: tudo com exceção de: pacotes com endereços de origem pertencentes a redes reservadas e pacotes com endereços de origem não pertencentes aos blocos da rede interna,

- entrada: apenas os pacotes que obedecem às seguintes combinações de protocolo, endereço e porta de destino: 25 *Transmission Control Protocol* (TCP) para o servidor *Simple Mail Transfer Protocol* (SMTP), 53/TCP e 53 *User Datagram Protocol* (UDP) para o servidor *Domain Name System* (DNS), 80/TCP para o servidor *Word Wide Web* (WWW);

b) interface interna:

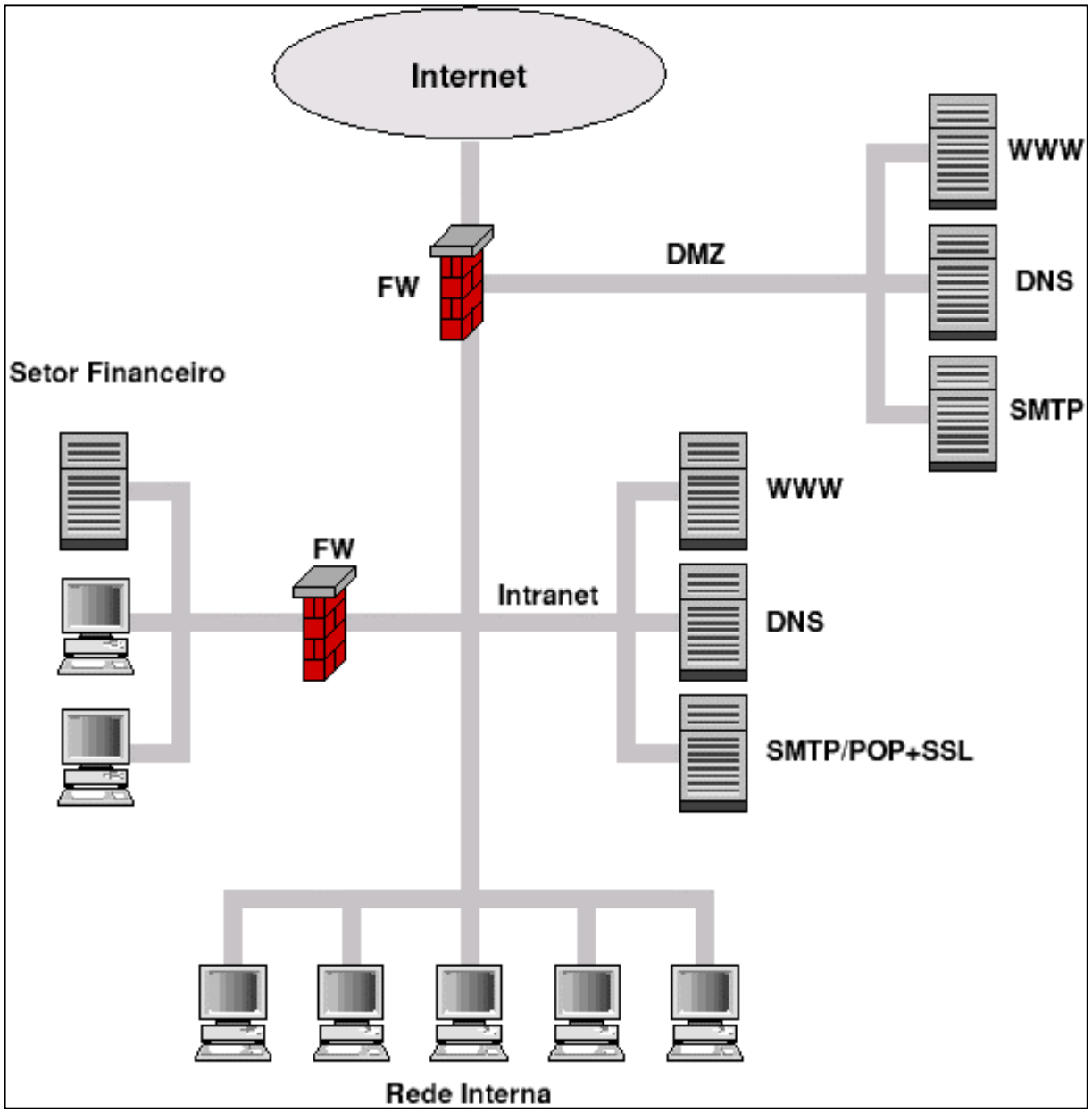
- saída: todo tráfego liberado,

- entrada: apenas os pacotes do protocolo na porta 22 *Secure Shell* (SSH);

c) interface da DMZ:

- saída: portas 25/TCP (SMTP), 53/UDP e 53/TCP (DNS) e 113 *Identification Protocol* (IDENT),

- entrada: além das mesmas regras de entrada da interface externa, também é permitido o tráfego para todos os servidores na com porta de destino 22/TCP (SSH) e endereço de origem na rede interna.



Fonte: NBSO (2003).

Figura 3 - Exemplo de um *Firewall* complexo.

Na Figura 3, além dos servidores externos na DMZ há também servidores na *intranet* e no setor financeiro da organização. Devido à importância das informações mantidas neste setor, a sua rede conta com a proteção adicional de um *Firewall* interno, cujo objetivo principal é evitar que usuários com acesso à rede interna da organização (mas não à rede do setor financeiro) possam comprometer a integridade e/ou o sigilo dessas informações.

A configuração do *Firewall* externo neste segundo exemplo é idêntica ao primeiro. Entretanto, no presente caso supõe-se que o servidor SMTP visível externamente (o da DMZ)

repassa as mensagens recebidas para o servidor SMTP da *intranet*. Para que isso seja possível, é necessário mudar a regra de filtragem para a interface interna, liberando o tráfego do servidor SMTP da DMZ para a porta 25/TCP do servidor SMTP da *intranet*.

A configuração do *Firewall* interno, por sua vez, é bastante simples. O servidor da rede do setor financeiro permite apenas acesso via *HiperText Transfer Protocol over Secure Socker Layer* (HTTPS) para que os funcionários da organização possam fazer suas consultas; outros tipos de acesso não são permitidos. O tráfego liberado por este *Firewall* é o seguinte:

a) interface externa (rede interna):

   - saída: tudo,

   - entrada: apenas pacotes para o servidor do setor financeiro com porta de destino 443/TCP (HTTPS) e endereço de origem na rede interna;

b) interface interna:

   - saída: tudo,

   - entrada: tudo (a filtragem é feita na interface externa).

4 IPTABLES

Segundo Neto (2004), o *Iptables* compõe a quarta geração de sistemas *Firewalls* no Linux, que foi encorporada a versão 2.4 do *Kernel*. Ele é uma versão mais completa e tão estável quanto seus antecessores *Ipfwandm* e *Ipchains*, implementados nos *Kernels* 2.0 e 2.2 respectivamente, onde também podemos destacar a primeira geração de *Firewalls* IPFW do BSD. O *Iptables* é amplamente utilizado devido as funções de *Firewall* estarem agregadas a própria arquitetura do *Kernel*. O Linux utiliza um recurso independente em termos de *Kernel* para controlar e monitorar todo o tipo de fluxo de dados dentro de sua estrutura operacional. A função do *Kernel* é de trabalhar ao lado de processos e tarefas, por esse motivo foi agregado um módulo ao mesmo chamado de *Netfilter* para controlar seu próprio fluxo interno. Criado por Marc Boucher, James Morris, Harald Welte e Rusty Russel, o *Netfilter* é um conjunto de situações agregadas inicialmente ao *Kernel* do Linux e divididos em tabelas. Sob uma ótica mais prática podemos ver o *Netfilter* como um grande banco de dados que contém em sua estrutura 3 tabelas padrões: *Filter*, *Nat* e *Mangle*.

4.1 TABELA FILTER

É a tabela padrão do *Netfilter* e trata das situações implementadas por um *Firewall* de filtro de pacotes. Estas situações são:

a) INPUT: tudo o que entra no *host*;

b) FORWARD: tudo o que chega ao *host* mas deve ser redirecionado a um *host* secundário ou outra interface de rede;

c) OUTPUT: tudo o que sai do *host*.

4.2 TABELA NAT

Esta tabela implementa as funções de NAT no *host* de *Firewall*. O NAT por sua vez, possui diversas utilidades:

a) PREROUTING: utilizada quando há necessidade em fazer alterações em pacotes antes que os mesmos sejam roteados;

b) OUTPUT: trata os pacotes emitidos pelo *Firewall*;

c) POSTROUTING: utilizado quando há necessidade de se fazer alterações em pacotes após o tratamento de roteamento.

4.3 TABELA MANGLE

Implementa alterações especiais em pacotes em um nível mais complexo. A tabela *mangle* pode alterar a prioridade de entrada e saída de um pacote baseado no tipo de serviço (TOS) ao qual o pacote se destinava:

a) PREROUTING: modifica pacotes dando-lhes um tratamento especial antes que os mesmos sejam roteados;

b) OUTPUT: altera os pacotes gerados localmente antes que os mesmos sejam roteados.

O Linux possui funções de *Firewall* agregadas ao *Kernel* pelo módulo *Netfilter*. Tais tabelas possibilitam controlar todas as situações (*chains*) de um *host*. Para que fosse possível moldar o *Netfilter* conforme as necessidades de cada *host*, rede ou subrede, foram desenvolvidas ferramentas de *Front-End*. Essas ferramentas permitem controlar as *chains* contidas nas tabelas agregando regras de tráfego. Historicamente o Linux disponibilizou uma nova ferramenta de manipulação nativa (*Front-End*) a cada nova versão oficial de kernel:

a) KERNEL 2.0 – IPFWAMD;

b) KERNEL 2.2 – IPCHAINS;

c) KERNEL 2.2/2.6 – IPTABLES.

O *Iptables* é uma ferramenta de *Front-End*, desenvolvido por Rust Russel (que participou do projeto de desenvolvimento do *Netfilter*), em colaboração com Michel Neuling e foi incorporado a versão 2.4 do *kernel* em julho de 1999. É composto pelos seguintes aplicativos: *Iptables*, aplicativo principal do pacote *Netfilter* para protocolos ipv4; *Ip6tables*, aplicativo principal do pacote *Netfilter* para protocolos ipv6; *iptables-save*, aplicativo que salva todas as regras inseridas na sessão ativa e ainda em memória em determinado arquivo informado pelo administrador do *Firewall*; *iptables-restore*, aplicativo que restaura todas as regras salvas pelo software *iptables-save*.

Devido a estar incorporado diretamente ao *kernel*, a configuração do *Iptables* não se dá por via de arquivos de configuração, sua manipulação é realizada por síntese digitada em *shell*, ou seja, uma regra na *shell* somente estará valendo para aquela sessão em memória. Uma vez que o computador *Firewall* é resetado ou desligado, tais regras serão perdidas e não mais poderão ser resgatadas. Para resgatar as regras atuais do sistema armazenadas em RAM é utilizado a ferramenta *iptables-save*, que as armazena em arquivo, para restaurar ou reaplicar as regras salvas é utilizado a ferramenta *iptables-restore*.

Suas principais características, além de realizar suas tarefas de forma veloz, segura, eficaz e econômica, apresenta um leque de possibilidades tais como:

a) implementação de filtros de pacotes;

b) desenvolvimento de QOS sobre o tráfego;

c) suporte a *Source Network Address Translation* (SNAT) e DNAT;

d) direcionamento de endereços e portas;

e) mascaramento de tráfego;

f) detecção de fragmentos;

g) monitoração de tráfego;

h) bloqueio de ataque de *Spoofing*, *Syn-Flood*, DOS, *scanners* ocultos, pings da morte, entre muitos outros.

A sintaxe básica do *Iptables* é definida por: “iptables [-t <tabela>] [comando] [ação] [alvo]”.

As tabelas são as mesmas que compõe o *Netfilter*, *Filter*, NAT e *Mangle*. Exemplos utilizando essas opções são: “iptables –t filter”; “iptables –t nat”; “iptables -t mangle”. A tabela *Filter* é a padrão do *Iptables*, se adicionarmos uma regra sem a *flag* –t , o mesmo aplicará situações contidas na tabela *Filter* a tal regra. Já no caso das tabelas NAT e *Mangle*, é necessário especificar sempre.

Os comandos das chains são definidos por:

a) -A: adiciona um nova entrada ao fim da lista de regras;

b) –D: apaga uma regra específica da lista;

c) –L: exibe as regras existentes na lista;

d) –P: altera a política padrão das *chains*. Inicialmente, todas as chains estão setadas como ACCEPT, ou seja, aceitam todo e qualquer tipo de tráfego;

e) –F: este comando remove todas as entradas adicionadas a lista de regras sem alterar a política padrão (-P);

f) –I: insere uma nova regra ao início da lista de regras, contrário do comando –A;

g) –R: substitui uma regra já adicionada por outra;

h) –N: este comando nos permite inserir ou criar uma nova *chain* na tabela específica.

i) –E: renomeia uma *chain*;

j) –X: apaga uma *chain* criada pelo administrador do *Firewall*.

As seguintes ações pode serem configuradas:

a) -p: especifica o protocolo aplicado a regra. Pode ser qualquer valor numérico especificado no arquivo /etc/protocol ou o próprio nome do protocolo (TCP, UDP, ICMP, etc...);

b) –i: especifica a interface de entrada a ser utilizada. Como um Firewall possui mais de uma interface, esta regra acaba sendo muito importante para distinguir a que interface de rede o filtro deve ser aplicado.

c) –o: especifica a interface de saída a ser utilizada e se aplica da mesma forma que a regra –i, porém somente as regras de OUTPUT e FORWARD se aplicam as regras;

d) –s: especifica a origem do pacote ao qual a regra deve ser aplicada. A origem pode ser um *host* ou uma rede;

e) –d: especifica o destino do pacote ao qual a regra deve ser aplicada. Sua utilização se dá da mesma maneira que a a opção –s;

f) !: significa exclusão e é utilizada quando se deseja aplicar um exceção a um regra. É utilizada juntamente com as opções –s, -d, -p. –i. –o;

g) –j: define o alvo (*target*) do pacote caso o mesmo se encaixe em uma regra;

h) --sport: porta de origem do pacote, com está opção é possível aplicar filtros com base na porta de origem do pacote. Somente a aplicados as protocolos TCP ou UDP;

i) --dport: porta de destino, especifica a porta de destino do pacote e funciona da forma similar a regra –sport.

Os seguintes alvos pode ser configurados:

a) ACCEPT: corresponde a aceitar, ou seja, permitir que a entrada e passagem do pacote em questão;

b) DROP: corresponde a descartar, um pacote que é conduzido a este alvo (Target) é

descartado imediatamente. O *Target* DROP não informa ao dispositivo emissor do pacote o que houve;

c) REJECT: corresponde a rejeitar, um pacote conduzido para este alvo (Target) é automáticamente descartado, a diferen;a do REJECT para o DROP é que o mesmo retorna uma mensagem de erro ao *host* emissor do pacote informando o que houve;

d) LOG: cria uma entrada de log no arquivo /var/log/messages sobre a utilização dos demais alvos (Targets), justamente por isso deve ser utilizado antes dos demais alvos.

e) RETURN: retorna o processamento do *chain* anterior sem processar o resto do *chain* atual;

f) QUEUE: encarrega um programa em nível de usuário de administrar o processamento de fluxo atribuído ao mesmo;

g) SNAT: altera o endereço de origem das máquinas clientes antes dos pacotes serem roteados;

h) DNAT: altera o endereço de destino das máquinas clientes;

i) REDIRECT: realiza o redirecionamento de portas em conjunto com a opção --to-port;

j) TOS: prioriza a entrada e saída de pacotes baseados em seu tipo de serviço, informação que esta no *header* do IPV4.

5 DESENVOLVIMENTO DO TRABALHO

Neste capítulo são apresentadas técnicas e ferramentas utilizadas no desenvolvimento do protótipo. O protótipo desenvolvido neste trabalho utiliza uma aplicação *web* para gerenciamento de *Firewall* baseado em *iptables*. A autenticação dos usuários na aplicação é feita utilizando algoritmo *Message Digest* (MD5) recurso do servidor apache, para uma maior segurança os *hosts* que poderão acessar o protótipo deverão ser definidos no servidor httpd. O protótipo permitirá a manutenção nas tabelas *filter* e NAT, onde poderão ser criadas, consultadas, alteradas, backupeadas e restauradas regras. Outra alternativa é a consulta dos *logs* de eventos do *Firewall*.

5.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Através deste protótipo é possível fazer a manutenção das regras de um *firewall iptables* nas tabelas *filter* e NAT. Pode-se citar os requisitos principais, que devem estar presentes no protótipo para realizar a especificação:

a) disponibilizar um mecanismo para autenticação dos usuários que irão realizar manutenção;
b) conectar automaticamente no *firewall* após a autenticação (a aplicação que será desenvolvida para testes do protótipo) e inserir as novas regras no *Iptables;*
c) ter uma interface amigável, onde o usuário terá exemplos de regras de *firewall*;
d) utilizar a linguagem de programação Perl.

5.2 ESPECIFICAÇÃO

Para a especificação do protótipo foi adotada a fluxogramação para demonstrar o fluxo dos principais processos executados nos protótipos.

A figura 4 apresenta a especificação genérica do protótipo.

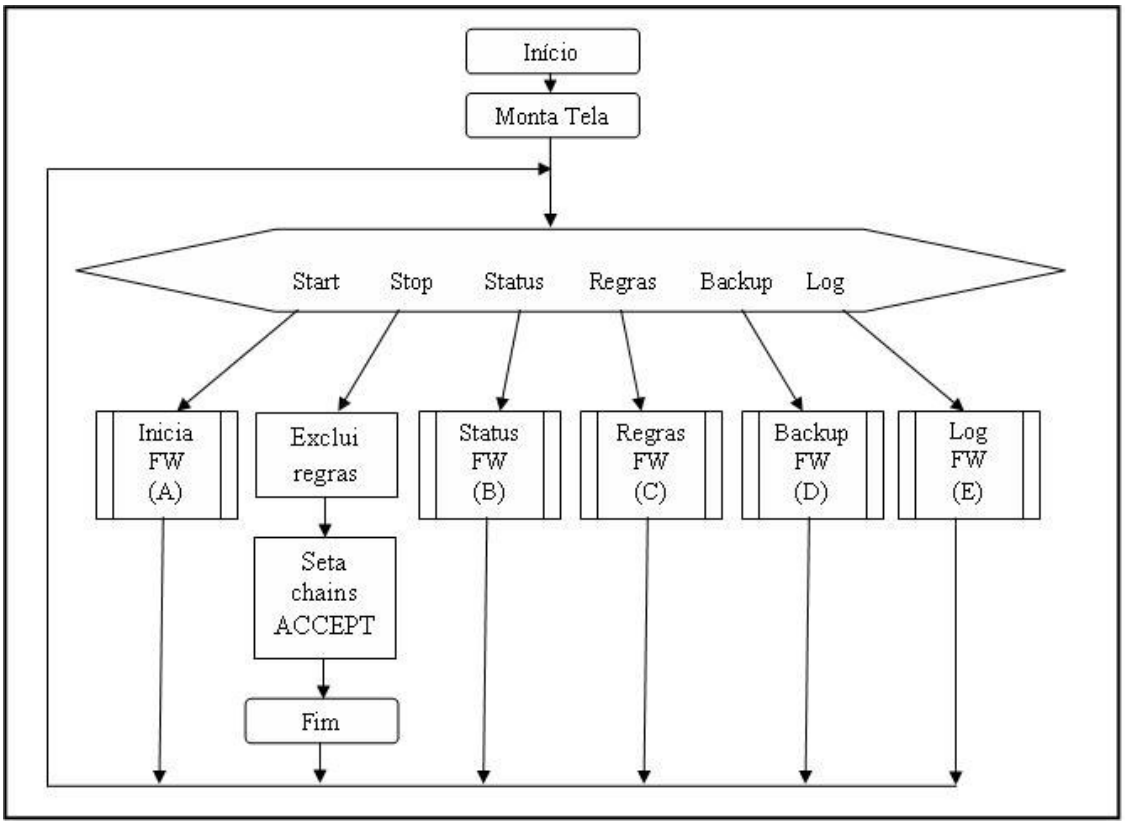


Figura 4 - Especificação genérica do protótipo

Na especificação genérica do protótipo podemos visualizar o menu com as opções de operações do protótipo, onde o detalhamento das opções está divido em processos. A opção "Stop" pára o serviço de *Firewall*, excluindo todas as regras e setando as *chains* para ACCEPT, liberando assim todo o tráfego de dados no *Firewall.*

No Quadro 1 são enumerados os processos e as suas descrições.

Nome	Descrição
Processo A	Inicia Firewall
Processo B	Status Firewall
Processo C	Regras Firewall
Processo D	Backup Firewall
Processo E	Log Firewall

Quadro 1 - processos do protótipo

A seguir são apresentados os diagramas dos processos do protótipo com suas respectivas descrições.

A figura 5 apresenta o detalhamento do processo A, que inicia os scripts de *Firewall,* montando as regras conforme os parâmetros configurados nas tabelas pelo administrador.

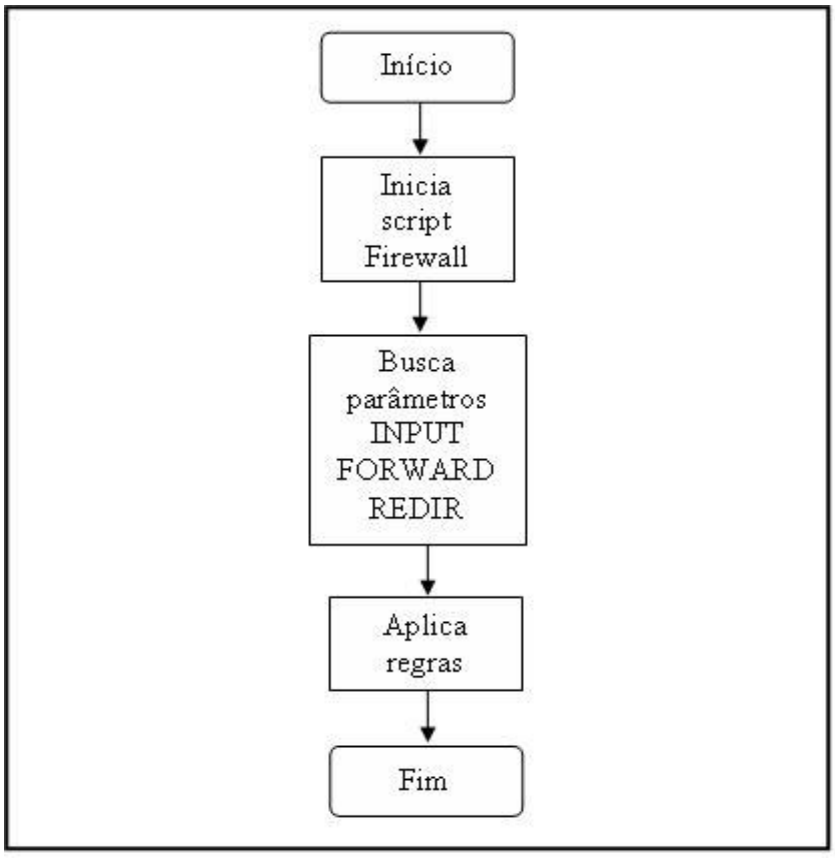


Figura 5 - Fluxograma do processo A

No processo A o *Iptables* é iniciado, neste momento: as *chains* serão setadas para DROP; iniciados os *scripts* para geração de *logs*; carregadas as regras contra ataques conhecidos. Depois disso, busca os parâmetros para aplicar as regras de ACCEPT, para liberação de tráfego permitido.

Na figura 6 observa-se o detalhamento do processo B, que verifica o status do *Iptables*.

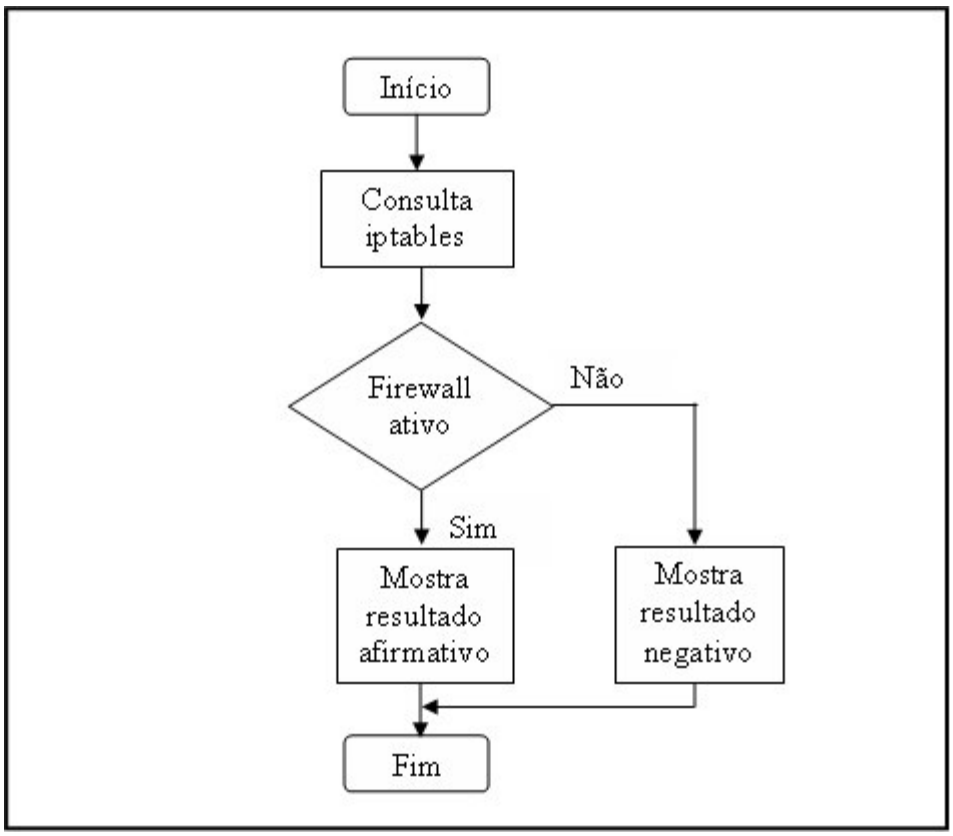


Figura 6 - Fluxograma do processo B

No processo B é feita uma consulta para verificar se o serviço de *Firewall* está ativo, o resultado apresenta o status do *Firewall.*

A figura 7 apresenta o detalhamento do processo C, o qual permite fazer todas as alterações nas *chains*.

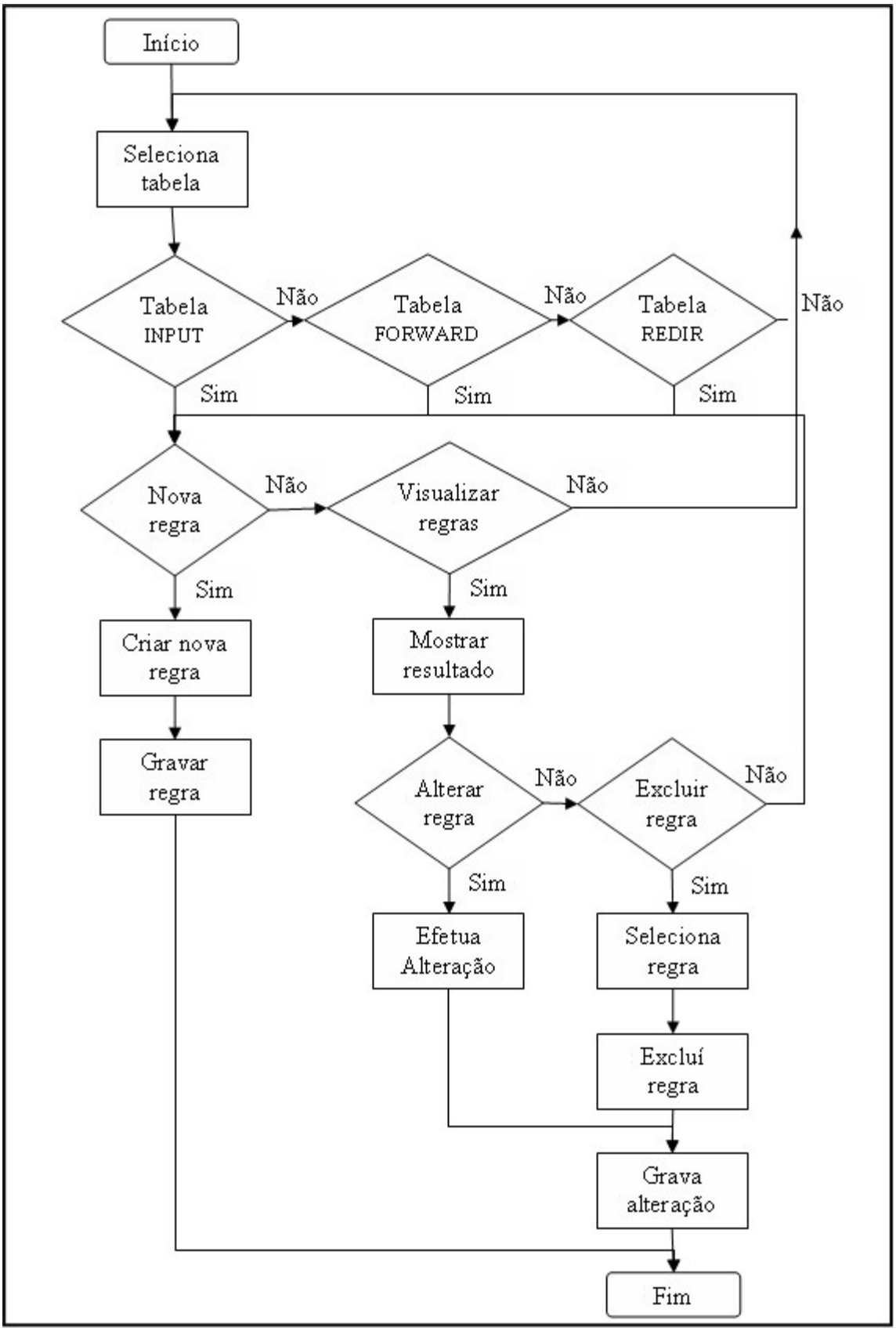


Figura 7 - Fluxograma do processo C

No processo C ocorre toda a manutenção nas regras de liberação de tráfego nas tabelas,

INPUT, FORWARD e REDIR. Este processo é responsável por criar regras, excluir regras e alterar regras de tráfego permitido.

Na figura 8 observa-se o detalhamento do processo D, o qual é responsável por realizar o backup das tabelas. Além do backup, a restauração das tabelas também ocorre nesse processo.

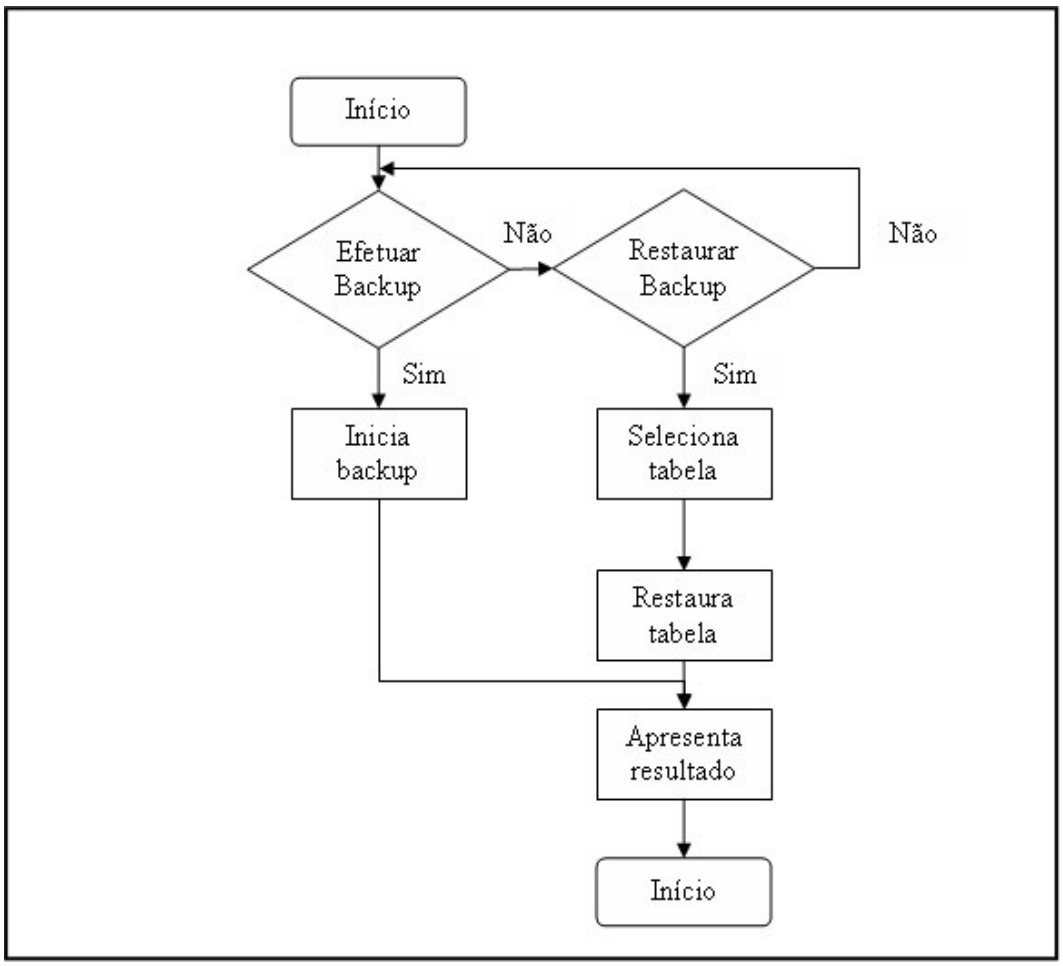


Figura 8 - Fluxograma do processo D

O processo D é responsável por realizar o *backup* das tabelas INPUT, FORWARD e REDIR. A restauração do *backup* de umas das tabelas também ocorre nesse processo.

A figura 9 apresenta o detalhamento do processo E, nele é feita a consulta dos eventos de *log* do *Firewall.*

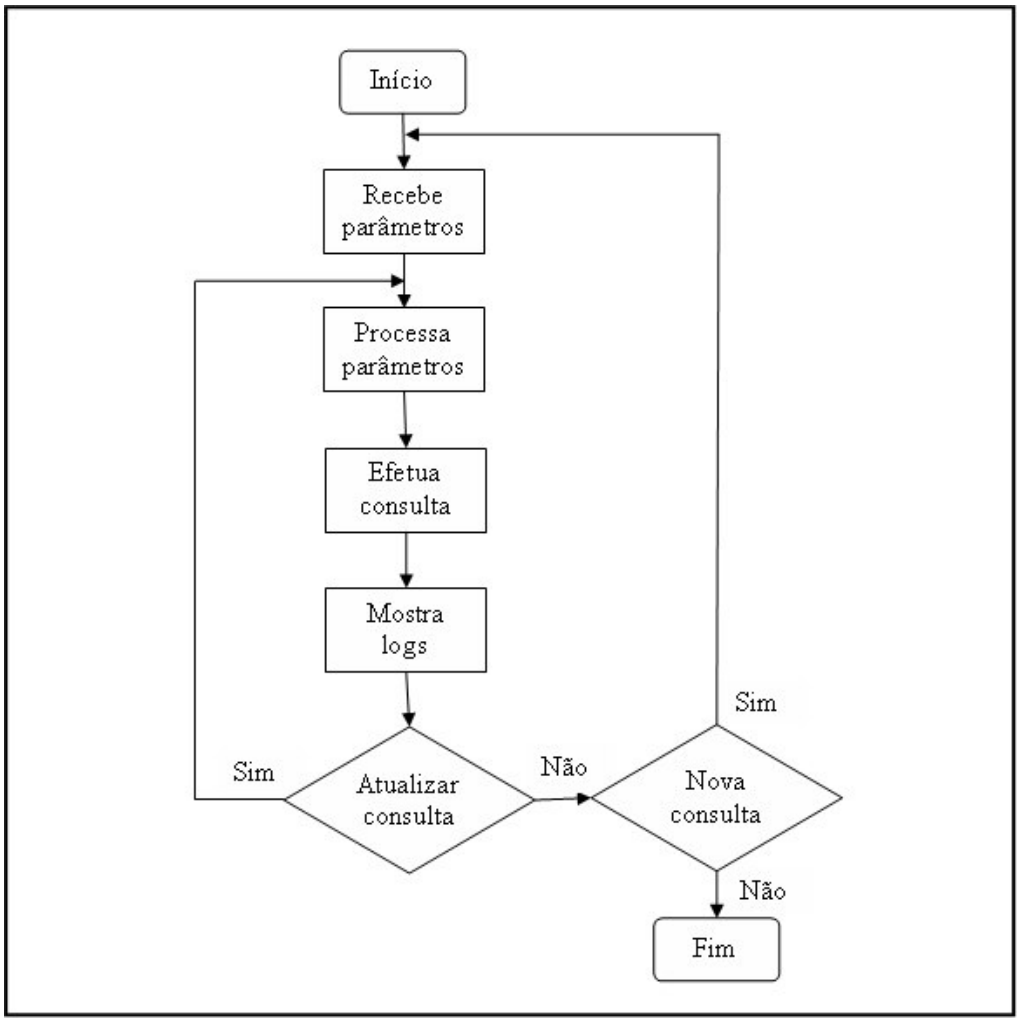


Figura 9 - Fluxograma do processo E

No processo E serão consultados *logs* de *Firewall,* que são gerados pelos scripts de geração de *logs* iniciados no processo A. Este processo faz uma pesquisa no arquivo de *logs*, procurando entradas que satisfazem os parâmetros de pesquisa e apresentando o resultado.

5.3 IMPLEMENTAÇÃO

Neste capítulo serão apresentadas considerações sobre a implementação do protótipo, onde foram utilizadas as seguintes técnicas e ferramentas:

a) linguagem de programação Perl-CGI;

b) programação Shell;

c) servidor Web apache;

d) sistema operacional Linux Fedora 4;

e) *firewall* Iptables.

5.3.1 Configuração do script de *Firewall*

O *script* de *Firewall* foi estruturado seguindo a metodologia: excluir todas as regras caso exista alguma; mudar a política padrão de todas as chains para DROP, ou seja, bloquear tudo; liberar para aceitar o tráfego permitido conforme política de segurança da corporação, com a *chain* ACCEPT; liberar interface de *loopback*, para que haja comunicação entre os processos; bloquear principais tipos de ataques; liberar todo tráfego de OUTPUT, desde que o *Firewall* inicie a conexão; executar a chamada das rotinas que executam os *scripts* que montam as regras que liberam o tráfego de INPUT, FORWARD, SNAT, DNAT e REDIRECT; todo o pacote que tem como destino o *Firewall* ou utiliza o mesmo como roteador, que não se adequa a uma regra previamente definida, será bloqueado e gerado *log* com o prefixo IPTABLES_DROP para que possa ser facilmente identificado nos *logs* do sistema.

```
#=============================================================
# OUTPUT - Liberação de OuTPUT
    iptables -A INPUT -m state --state ESTABLISHED,RELATED  \
          -j ACCEPT
    iptables -A OUTPUT -m state --state NEW,ESTABLISHED,RELATED \
          -j ACCEPT

#=============================================================
# INPUT - entrada de pacotes no firewall - modulo
  /etc/firewall/input.sh /etc/firewall/   fw.input

#=============================================================
# PREROUTING E POSTROUTING - Redirecionamento de portas - modulo
  /etc/firewall/redir.sh /etc/firewall/   fw.redir

#=============================================================
# FORWARD -
  /etc/firewall/forward.sh /etc/firewall/    fw.forward

#=============================================================
# LOG - geracao de logs dos pacotes bloqueados
  iptables -A INPUT   -j LOG --log-prefix "[IPTABLES DROP IPT ] : " \
          --log-level info -m limit   --limit 5/minute
  iptables -A OUTPUT  -j LOG --log-prefix "[IPTABLES DROP OUT ] : " \
          --log-level info -m limit   --limit 5/minute
  iptables -A FORWARD -j LOG --log-prefix "[IPTABLES DROP FWD ] : " \
          --log-level info -m limit   --limit 5/minute
```

Figura 10 - Módulo principal do regerador *chains*

Na Figura 10 é apresentado um trecho do script principal do *Firewall* onde é feita a liberação de todo o tráfego de OUTPUT e são chamados os scripts para as tabelas de INPUT, REDIR e FORWARD. Além disso são setados os parâmetros para geração de logs dos pacotes bloqueados

5.3.2 Módulo Gerador de *Chains*

O módulo gerador de *chains* é iniciado junto com o sistema operacional, sendo responsável pela carga de todas as regras de *Firewall*. É executado como serviço no sistema operacional Linux, em caso de parada desse serviço todas as *chains* retornam ao valor *default*.

Quando iniciado vai executando sequencialmente as linhas do script principal do *Firewall*, e montando cada regra de INPUT, FORWARD, SNAT, DNAT, REDIRECT entre outras, conforme os arquivos com os parâmetros gerados pela Ferramenta Web para gerenciamento de *Firewall* Linux.

```
cmd_input="$bin -A INPUT -s $orig"
cmd_output="$bin -A OUTPUT -d $orig"

if [ "$itin" != "any" ]; then
  cmd_input=$cmd_input" -i $itin -d $dest"
  cmd_output=$cmd_output" -o $itin -s $dest"
else
  cmd_input=$cmd_input" -d $dest"
  cmd_output=$cmd_output" -s $dest"
fi
if [ "$prot" != "all" ]; then
 if [ "$prot" = "icmp" ]; then
    cmd_input=$cmd_input" -p icmp"
    cmd_output=$cmd_output" -p icmp"
 elif [ "$port" = "all" ]; then
     cmd_input=$cmd_input" -p $prot"
     cmd_output=$cmd_output" -p $prot"
  else
     cmd_input=$cmd_input" -p $prot --dport $port"
     cmd_output=$cmd_output" -p $prot --sport $port"
 fi
fi
```

Figura 11 - Trecho do código do módulo gerador de *chains* INPUT

Na Figura 11 é apresentado um trecho do código gerador das regras de INPUT, onde são processados os parâmetros recebidos pelo adminstrador e geradas as regras de *Firewall*.

5.3.3 Página Web para Gerenciamento de *Firewall*

O protótipo *web* oferece as funcionalidades para que o administrador de rede possa fazer as manutenções de regras de *Firewall*, *backup* e restaurações de tabelas de *chains* e consulta de eventos de *log*, através de um navegador Internet.

Ao carregar o protótipo será solicitado um usuário e senha, que foi previamente cadastrado para autenticação no servidor Apache, utilizando o algoritmo MD5. Na tela principal é possível visualizar todos os eventos que podem ser realizados no protótipo, de forma simples e objetiva (Figura 12).

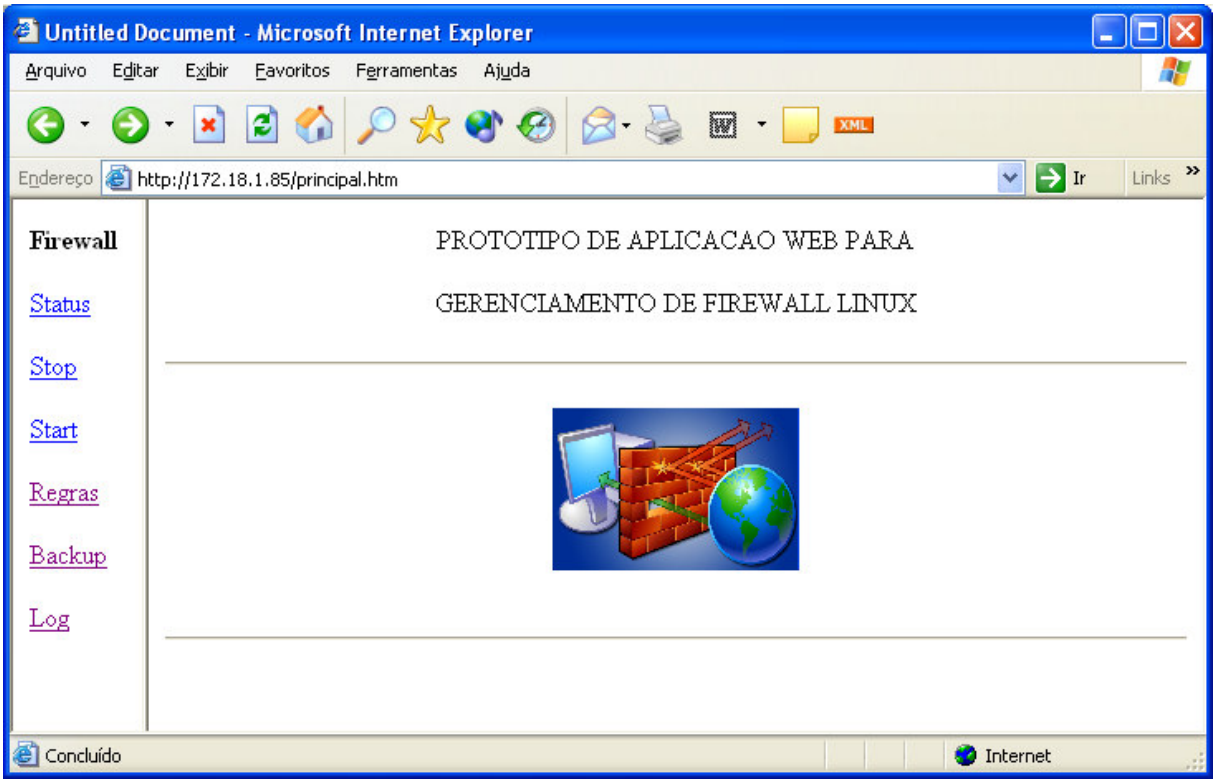


Figura 12 - Tela principal do protótipo.

Ao clicar nos menus serão executadas a funções específicas (*Status*, *Stop*, *Start* do *Firewall*) ou será aberto um novo *frame* com os recursos disponíveis para a opção selecionada. Ao selecionar a opção “Regras” será apresentado um *frame* com as *chains* para manutenção .

Ao Selecionar determinada *Chain*, o administrador poderá visualizar as regras existentes na tabela selecionada, excluir regras, alterar regras ou criar regras de *Firewal* (Figura 13).

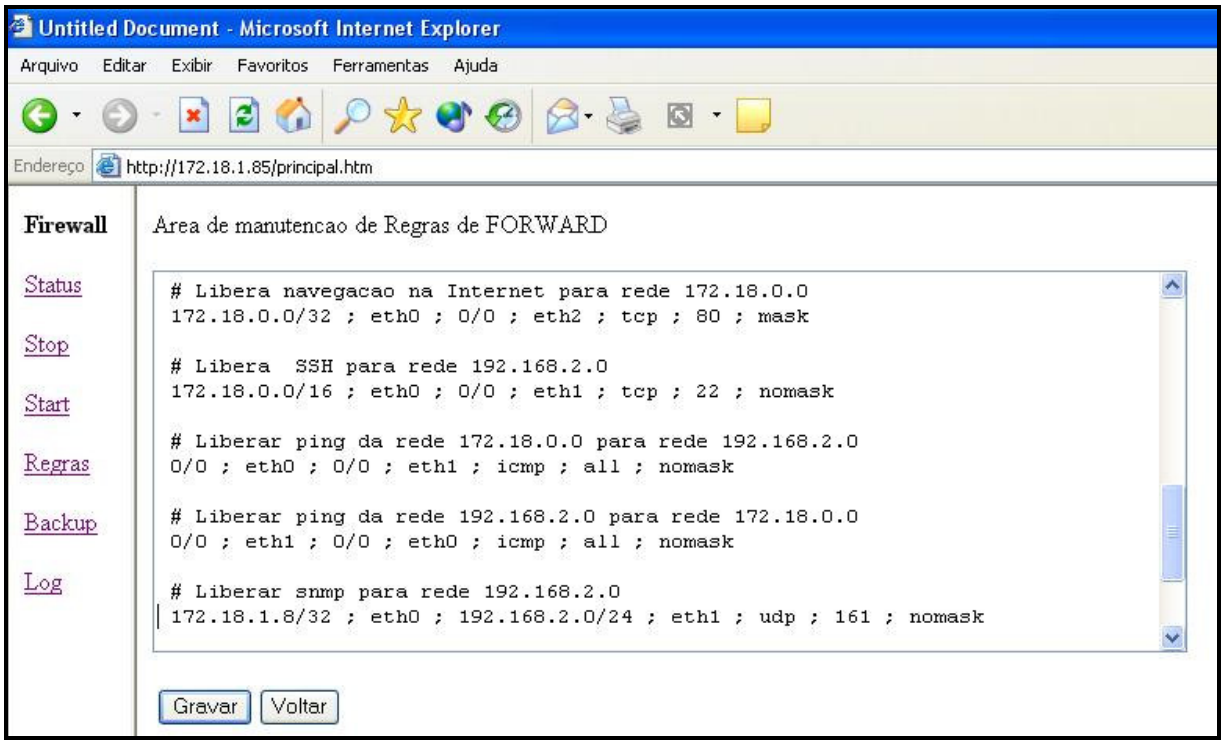


Figura 13 - Visualização regras FORWARD

Dentre as principais funcionalidades do protótipo podemos citar também a área de *backup*, onde se encontram as funcionalidades de efetuar o *backup* das tabelas ou restauração das mesmas (Figura 14).

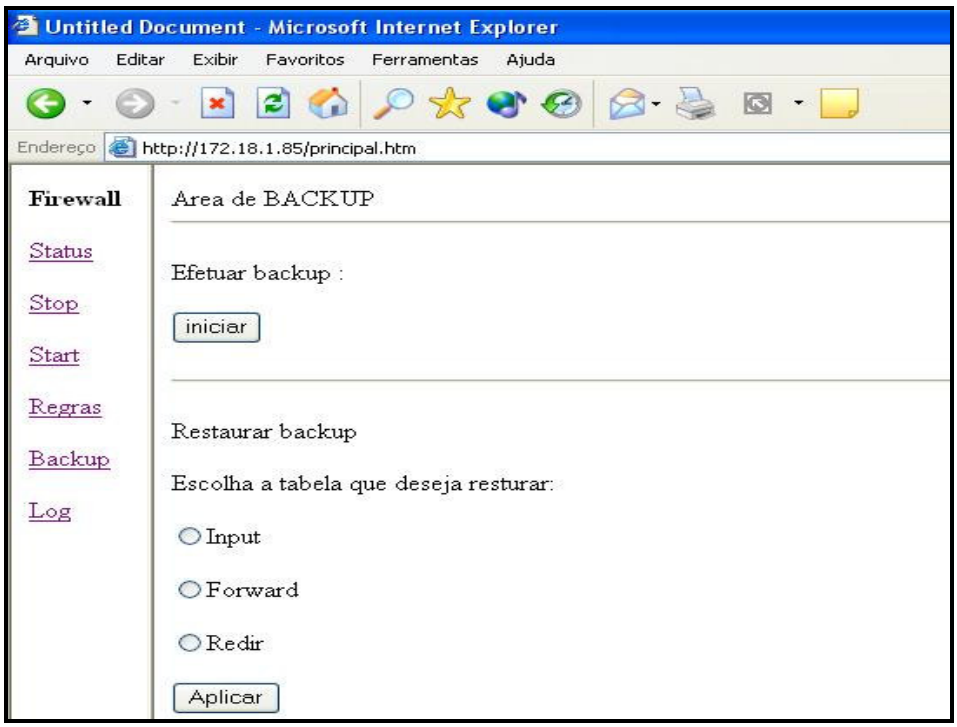


Figura 14 - Área de *backup*

5.3.4 Operacionalidade da implementação

Para ser útil e ter um nível de segurança a mais, a aplicação deverá ser instalada em um diretório onde somente o *root* (usuário administrador do sistema operacional Linux) tem permissão de leitura, escrita e gravação. No servidor Apache deverá ser criado um *alias*, apontando o diretório onde se encontram os programas Perl-CGI, e ainda configurado o método de autenticação e quais os *hosts* podem acessar o protótipo.

5.3.5 Processo manutenção de regra

O caso de uso “manutenção de regras” ocorre quando o administrador de rede acessa o protótipo de aplicação *web* para realizar determina alteração nas regras de *Firewall.*

O primeiro passo é informar “Endereço do protótipo”, “Usuário”, “Senha” para login do usuário. Em seguida, clicar no botão “Regras” para visualizar as opções de tabelas, selecionar a opção “Input” e posteriormente a opção “Criar”. Neste momento será apresentado um exemplo de regra de input que pode ser utilizada para orientar a inserções dos parâmetros da nova regra. Em seguida, entrar com os valores de “Endereço de Origem”, “Máscara de *host* ou rede”, “Interface de entrada do Pacote”, “Endereço de Destino”, “Máscara de *host* ou rede”, “Protocolo” e “Porta de Destino”.

Para confirmar a criação da regras de *Firewall* o administrador deverá usar a opção do menu “gravar”. A opção “limpar” para limpar os parametros e entrar com todas as informações novamente.

5.3.6 Processo aplicar regra

O caso de uso “aplicar regra” ocorre quando o usuário do protótipo parar o serviço de *Firewall* clicando na opção do menu “Stop”, neste momento todas as regras de *Firewall* serão excluídas e aplicadas suas regras *default* (ACCEPT). Para que o *Firewall* seja ativado novamente e a regra criada tenha efeito, deverá ser clicado na opção ”Start”, retornando a mensagem de que o *Firewall* foi iniciado.

Se a regra criada não estiver íntegra, o retorno vai ser a linha com a regra incorreta e uma mensagem de que a regra não está consistente. Neste momento o administrador poderá agir de duas formas, restaurar o *backup* da tabela que apresentou problemas ou excluir a regra e refazê-la.

5.4 RESULTADOS E DISCUSSÃO

Um *Firewall* consistente e seguro depende da implementação de suas regras. Sendo que uma vez definida a política de segurança da empresa e a estrutura de segurança, é necessário bloquear ou liberar portas e protocolos de acordo com as necessidades.

Para ficar mais claro o funcionamento deste protótipo serão discutidos testes e resultados obtidos com o desenvolvimento protótipo.

A política de *Firewall* adotada para esse protótipo é bloquear com DROP todas as *chains* e depois ir liberando o tráfego de acordo com a necessidade. Supondo que fosse criada uma regra de INPUT para liberar o acesso SSH do *host* (172.18.1.100/16) para o *Firewall* (172.18.1.85), seria necessário liberar a entrada do pacote no *Firewall* e depois permitir a saída do pacote para enviar uma resposta a estação que solicitou a requisição.

A figura 15 mostra os parâmetros utilizados no protótipo para criar a regra citada

acima. No campo “EndOrigem” foi informado o endereço de IP do *host* que irá acessar o servidor, no campo “Mask” foi informado a máscara de rede, no campo “Int. Ent” foi informado em qual interface rede o pacote entra para chegar até o *Firewall*. Posteriormente é informado o Destino do pacote no campo “Dest” que é o endereço de IP do *Firewall*, em seguida foram informados o tipo de protocolo e a porta utilizada pelo mesmo, nos campos “Prot” e “Porta” respectivamente.

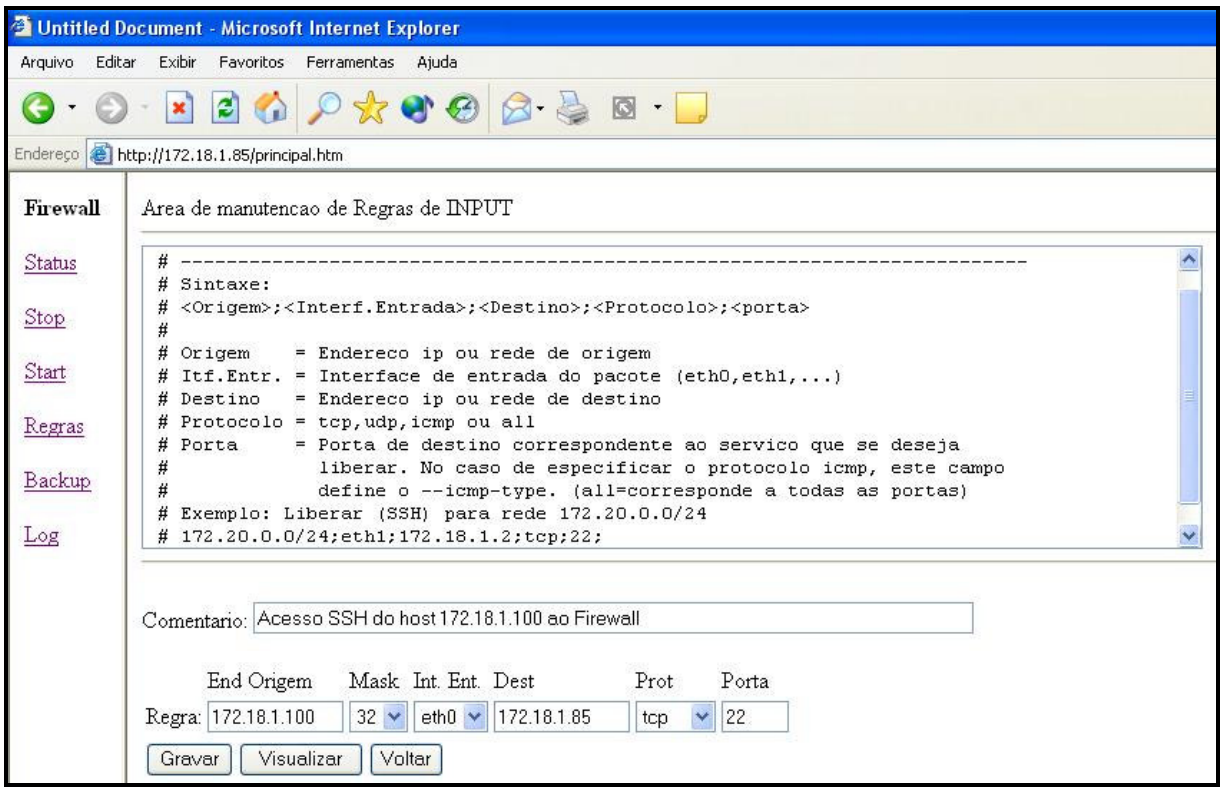


Figura 15 - Criação de regra de INPUT.

O resultado gerado com a entrada dos parâmetros no protótipo pode ser visto no Quadro 2:

Direção do fluxo	Regras Iptables
INPUT	/sbin/iptables –A INPUT –s 172.18.1.100/32 –I eth0 –d 172.18.1.85 –p tcp –dport 22 –j ACCEPT
OUTPUT	/sbin/iptables –A OUTPUT -s 172.18.1.100/32 –o eth0 –s 172.18.1.85 –p tcp –sport 22 –j ACCEPT

Quadro 2 - Regras geradas

Durante a criação das regras para testes foram liberadas diferentes portas e protocolos, sendo que um dos resultados foi o apresentado acima. Resultados semelhantes aos visualizados foram obtidos com as *chains* FORWARD e REDIR.

6 CONCLUSÕES

Com o planejamento das etapas do trabalho, definição do cronograma e ferramentas que seriam utilizadas, a conclusão do trabalho foi conseqüência do cumprimento destas atividades.

É importante ressaltar que para a implementação do protótipo, citar os principais pontos que foram decisivos para o desenvolvimento do trabalho e dentre os quais destacar: a definição do algoritmo de autenticação de usuários, onde foi utilizado o MD5; estruturar os script de *Firewall* para ter um padrão de primeiramente bloquear todo o tráfego; depois apresentar regras para proteção do ambiente contra as principais ameaças e ataques; regras de liberação de tráfego permitido e geração de *logs* de todos os pacotes que foram bloqueados pelo *Firewall.* Além da autenticação do usuário no servidor Apache com o MD5, podem ser definidos os *hosts* que terão permissão de acesso ao protótipo, tendo assim um nível de segurança a mais.

O protótipo foi desenvolvido para realizar manutenção em regras de *Firewall* do *Iptables*. A aplicação oferece uma interface *web* com exemplo de regras de acordo com a *chain* selecionada para criação de regras, permitindo visualizar todas as regras distintas por tabelas, excluir ou modificar regras. Outra funcionalidade importante é o *backup* e a restauração das tabelas. Além disso, oferece a análise dos *logs* o que facilita ao administrador da rede, na hora da criação das regras. Com os resultados obtidos, considera-se que o protótipo atingiu os objetivos propostos.

Uma das deficiências do protótipo é não tratar as *chains* da tabela *mangle*, que é priorizar pacotes que trafegam pelo *Firewall*.

A maior dificuldade encontrada foi no levantamento bibliográfico sobre padrões de implementação da estrutura de montagem das tabelas, com o objetivo de atingir maior

segurança e desempenho.

6.1 EXTENSÕES

Como extensão deste trabalho, propõe-se a implementação de um módulo para configuração de regras para bloqueio de pacotes contra as principais ameaças de redes e ataques. Com geração de *logs* específicos e alertas ao administrador.

REFERÊNCIAS BIBLIOGRÁFICAS

CHESWICK, William R.; BELLOVIN, Steven M.; RUBIN, Aviel D. **Firewall e segurança na Internet**. 2. ed. Tradução Edson Frumankiewicz. São Paulo: Bookman, 2003.

DIAS, Cláudia. **Segurança e auditoria da tecnologia da informação**. Rio de Janeiro: Axcel Books do Brasil, 2000.

NBSO. **Práticas de segurança para administradores de redes Internet**. São Paulo, 2003. Disponível em: <http://www.cert.br/docs/seg-adm-redes/seg-adm-redes.html>. Acesso em: 01 Set. 2005.

NETO, Urubatan. **Dominado Linux Firewall Iptables**. Rio de Janeiro: Ciência Moderna, 2004.

RIBEIRO, Uirá. **Certificação Linux**. Rio de Janeiro: Axcel Books do Brasil, 2004.

ROGER, Denny. **Firewalls:** falsa sensação de segurança. Fortaleza, 2005. Disponível em: <http://www.secforum.com.br/categories.php?op=newindex&catid=5>. Acesso em: 6 Set. 2005.

SANTOS, Luiz C. **Firewall Linux IPTABLES**. Rio de Janeiro, 2004. Disponível em: <http://www.clubedasredes.eti.br/rede0023.htm>. Acesso em: 23 Ago. 2005.

SEGURANÇA máxima para Linux. Rio de Janeiro: Campus, 2000.

SILVA, Antônio Carlos V. **Instalando firewall no Linux de modo fácil**. [S.l.], 2004. Disponível em: <http://br-linux.org/tutoriais/002028.html>. Acesso em: 16 Ago. 2005.

STANGER, James; LANE, Patrick T. **Rede segura Linux**. Rio de Janeiro: Alta Books, 2002.

WIKIPEDIA. **Segurança da informação**. Boston, 2002. Disponível em: <http://pt.wikipedia.org/wiki/Seguran%C3%A7a_da_informa%C3%A7%C3%A3o#Pol.C3.ADticas_de_seguran.C3.A7a>. Acesso em: 01 Set. 2005.