UNIVERSIDADE SAGRADO CORAÇÃO

GLAUBER OLIVEIRA

DESENVOLVIMENTO DE UMA INTERFACE PARA GERENCIAMENTO DE FIREWALL EM IPTABLES

Bauru
2010

UNIVERSIDADE SAGRADO CORAÇÃO

DESENVOLVIMENTO DE UMA INTERFACE PARA GERENCIAMENTO DE FIREWALL EM IPTABLES

Trabalho de conclusão de curso apresentado ao Centro de Ciências Exatas e Naturais como parte dos requisitos para obtenção to título de bacharel em Ciências da Computação, sob orientação do Prof. Dr. Kelton A. Pontara da Costa.

Bauru
2010

GLAUBER ALVES DE OLIVEIRA

DESENVOLVIMENTO DE UMA INTERFACE PARA GERENCIAMENTO DE FIREWALL EM IPTABLES

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Naturais da Universidade Sagrado Coração como parte dos requisitos para obtenção do título de Bacharel em Ciências da Computação sob orientação do Prof. Dr. Kelton A. Pontara da Costa.

Banca examinadora:

_____
Prof. Dr. Kelton A. Pontara da Costa
Universidade Sagrado Coração

_____
Prof. Esp. André Luiz F. Castro
Universidade Sagrado Coração

_____
Prof. Esp. Henrique P. Martins
Universidade Sagrado Coração

Bauru, 14 de Dezembro de 2010.

Dedico esse trabalho a todas as pessoas que me ajudaram diretamente na realização deste e especialmente a meus pais, minha esposa e minhas filhas.

AGRADECIMENTOS

Agradeço primeiramente a Deus, pelo imenso amor e graça.

A minha família que me apóia sempre nas minhas decisões.

Aos meus amigos da Universidade e especialmente ao meu orientador Kelton Costa que acreditou na conclusão deste trabalho.

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

“O especialista é um homem que sabe cada vez mais sobre cada vez menos, e por fim acaba sabendo tudo sobre nada.”
(Bernard Shaw)

RESUMO

Firewall é um quesito de segurança com cada vez mais importância no mu da computação. À medida que o uso de informações e sistemas é cada vez mai proteção destes requer a aplicação de ferramentas e conceitos de segura eficientes, logo o *Firewall* é uma opção praticamente imprescindível.

O *Firewall* é um mecanismo que atua como "defesa" de um computador ou de uma rede, controlando o acesso ao sistema por meio de regras e a filtragem de dados.

Este trabalho apresenta o desenvolvimento de uma interface intuitiva para gerenciamento de *Firewall L*inux via *web* a fim de facilitar a criação e manutenção das regras de segurança da rede. Apresenta também formas de implementação de *Firewalls* e assuntos relacionados a segurança. Utilizou-se o *Firewall Iptables* como sendo o *Firewall* do projeto utilizando a linguagem PHP para o desenvolvimento da interface e o recurso de autenticação de diretórios do *Apache* para restringir o acesso à interface.

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

ABSTRACT

Firewall security is a question more and more importance in the comput world. As the use of information and systems is increasing, their protection requi the application of tools and concepts of effective security, then the Firewall option virtually indispensable.

A firewall is a mechanism that acts as a "defense" of a computer or a network by controlling access to the system through rules and filtering.

This paper presents the development of an intuitive interface for managing the Linux firewall via web to facilitate the creation and maintenance of rules network security. It also presents ways to implement firewalls and security issues. It was used as the Iptables Firewall Project using PHP for development of the interface and the use of Apache's directory authentication to restrict access to the interface.

Key-words: Internet. Security. Firewall management.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

BSD – Berkeley Software Distribution

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*

QOS – *Quality of Service*

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 externas. Conseqüentemente, os recursos de *hardware* e *software* das empresas ficam expostas 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*.

O software de *Firewall* é um dispositivo que fica instalado no microcomputador 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 ele. Baseados nessas análises podem 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).

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

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

b) ocultar informações sobre a rede interna, fazendo com que pareça que todo o tráfego 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, consequentemente 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*[1] e o *Ipchains*[2] respectivamente (STANGER; LANE, 2002). 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

[1] Software responsável por administrar os recursos do IPFW que é um filtro de pacotes nativo no kernel de alguns sistemas operacionais.
2 Software responsável por administrar os filtros de pacotes das versões 2.2 do Kernel do Linux.

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 com pouco experientes na área.

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

1.1 OBJETIVO GERAL

O objetivo deste estudo foi desenvolver uma interface intuitiva via *web* para gerenciamento de um *Firewall* no sistema operacional Linux baseado em *Iptables*.

1.2 OBJETIVOS ESPECÍFICOS

A aplicação desenvolvida visa disponibilizar um mecanismo de fácil compreensão para gerenciamento do *Firewall* em uma rede, não requerendo ao operador saber comandos específicos do *Iptables* e assim podendo gerenciar as regras do *Firewall* com maior rapidez entendendo o que cada uma delas faz.

1.3 JUSTIFICATIVA

O motivo para o desenvolvimento desta aplicação de gerenciamento de *Firewall* através da *web*, utilizando um navegador para a administração, é a grande vantagem que o administrador de rede terá ao gerenciar suas rotinas de segurança além de possibilitar a realização de manutenção nas regras de *Firewall* de forma mais simples e intuitiva permitindo acessibilidade ao aplicativo a partir de plataforma Windows e Linux, independente do navegador utilizado e o fato de poder aplicar métodos de segurança na autenticação de usuários possibilitando uma maior confiabilidade de quem estará realizando as manutenções do *Firewall*.

1.4 ESTRUTURA DO TRABALHO

A estrutura deste estudo está dividida em capítulos que serão explicados a seguir.

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

O segundo capítulo é todo levantamento bibliográfico do estudo onde aborda segurança de redes e conceitos como os objetivos de segurança da informação e políticas de segurança, os conceitos de *Firewall* contendo os tipos, escolha, arquitetura e filtragens além do estudo do *Iptables* e suas divisões em tabela *Filter*, tabela *NAT* e tabela *Mangle* e os conceitos de *Web* e *Sistema Operacional Linux.*

No terceiro capítulo serão apresentados os materiais e recursos usados para o desenvolvimento do estudo.

O quarto capítulo apresentará o desenvolvimento do projeto.

O quinto capítulo mostrará os resultados obtidos.

O Sexto capítulo será a conclusão final do projeto.

2 LEVANTAMENTO BIBLIOGRÁFICO

Para Dias (2000) historicamente, observa-se a evolução das redes de computadores e dos conceitos de segurança, onde esses conceitos passaram a ser estratégias adotadas para desenvolvimento de políticas de segurança em redes privadas. Dentre os vários assuntos referentes ao tema proposto, serão focados neste trabalho: segurança, *Firewall* e *Iptables*.

Dias (2000) ainda afirma que na sociedade da informação, ao mesmo tempo em 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.

2.1 SEGURANÇA EM REDES

Para NBSO (2005), 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, uma vez que a dependência de atividades essenciais de todos os tipos cresce 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).

2.1.1 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 quesitos devem ser observados (DIAS, 2000):

- 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;
- 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;
- 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;
- consistência – certificar-se de que o sistema atua de acordo com as expectativas dos usuários autorizados;
- isolamento – regular o acesso ao sistema;
- confiabilidade – garantir que, mesmo em condições adversas, o sistema atuará conforme o esperado.

Dias (2000) conclui que 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.

2.1.2 Política de Segurança

Sobral (2005) afirma que 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 se adaptar as 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.

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).

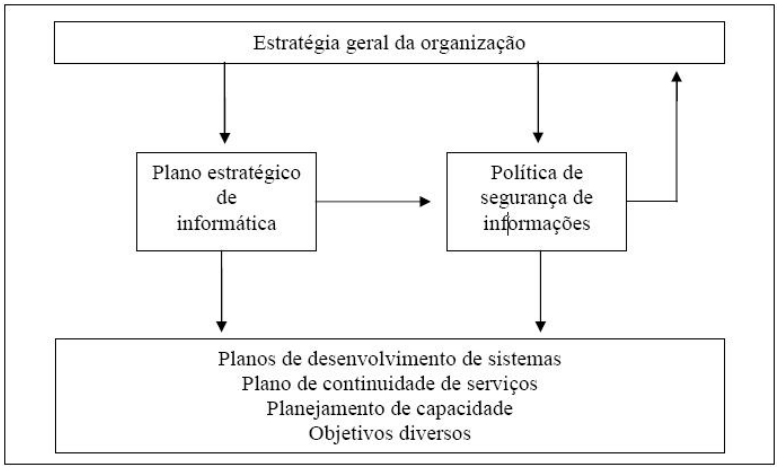

Figura 1 - Política de segurança e seus relacionamentos.
Fonte: adaptada de Dias (2000).

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:

- recursos protegidos pela política;
- ameaças às quais estes recursos estão sujeitos;
- vulnerabilidades que podem viabilizar a concretização destas ameaças, analisando as individualmente.

NBSO (2003) conclui 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.

2.2 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):

- Todo tráfego de dentro para fora de uma rede, e vice-versa, deve passar pelo *Firewall*;
- Apenas tráfego autorizado, como definido pela política de segurança local, terá permissão de passar;
- 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*[3] 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.

2.2.1 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.

Roger (2005) explica que 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.

Roger (2005) ainda ressalta que 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

[3] É qualquer máquina ou computador conectado a uma rede.

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 acrescentada 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.

Para concluir, Roger (2005) explica que 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.

2.2.2 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 Linux ou Unix são configurados por meio de arquivos texto.

2.2.3 Localização de um *Firewall*

Para Santos (2004) 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:

- 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 (modem, por exemplo). Caso não seja possível eliminar todos esses caminhos, eles devem ser documentados e fortemente vigiados através de outros mecanismos de segurança;
- 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.

É recomendável colocar os servidores acessíveis externamente (*Web*, *FTP*, correio eletrônico, etc.) em um segmento de rede separado e com acesso altamente restrito, conhecido como *DMZ*[4]. 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*[5] 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 (STANGER, LANE, 2002).

Segundo Hunt (2000), em alguns casos, é possível identificar na rede interna grupos de sistemas que desempenham determinadas tarefas comuns, tais como desenvolvimento de software, *web design* e administração financeira. Nestes casos, recomenda-se o uso de *Firewalls* internos para isolar estas sub-redes umas das

[4] É uma pequena rede situada entre uma rede confiável e uma não confiável, geralmente entre a rede local e a Internet.
[5] Ferramenta capaz de interceptar e registrar o tráfego de dados em uma rede de computadores.

outras, com o propósito de aumentar a proteção dos sistemas internos e conter a propagação de ataques bem-sucedidos.

2.2.4 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.

Para La-Roque (2004), 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* isolada.

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 (STANGER, LANE, 2002).

2.2.5 Arquiteturas de um *Firewall*

De acordo com NBSO (2000), 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 (NBSO, 2000).

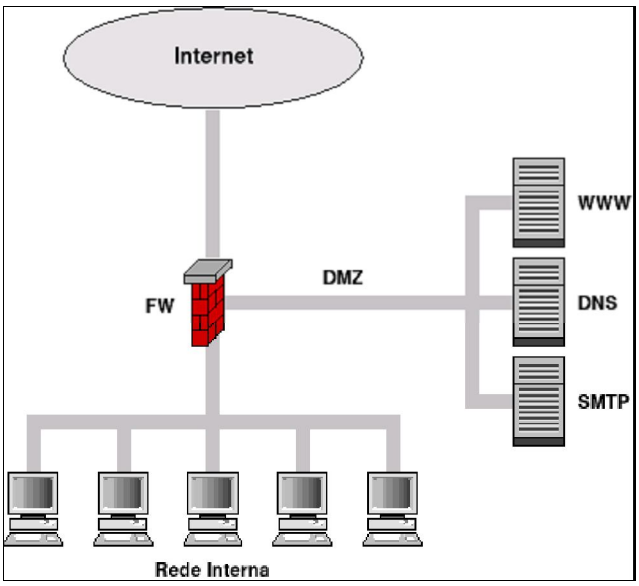

Figura 2 - Exemplo de um *Firewall* simples.

Fonte: NBSO (2003).

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*[6], 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 (NBSO, 2000).

A seguir será apresentado o tráfego liberado em cada interface de *Firewall* da Figura 2 (NBSO, 2000):

- 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*).

[6] Capacidade para identificar o protocolo dos pacotes transitados e "prever" as respostas legítimas.

- Interface interna:
  - saída: todo tráfego liberado,
  - entrada: apenas os pacotes do protocolo na porta 22 *Secure Shell* (*SSH*).
- 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.

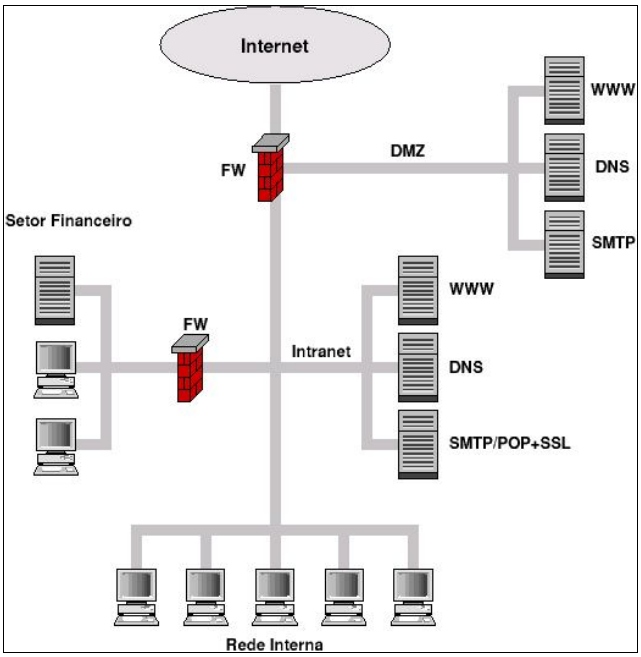

Figura 3 - Exemplo de um *Firewall* complexo.

Fonte: NBSO (2003).

Na Figura 3, além dos servidores externos na *DMZ* há também servidores na *intranet*[7] 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

[7] É uma rede de computadores privada que assenta sobre a suite de protocolos da Internet.

organização (mas não à rede do setor financeiro) possam comprometer a integridade e/ou o sigilo dessas informações (NBSO, 2000).

NBSO (2000) explica que 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* (*HTTP*S) 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 (NBSO, 2000):

- 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;
- interface interna:
  - saída: tudo,
  - entrada: tudo (a filtragem é feita na interface externa).

2.3 IPTABLES

Segundo Santos (2004) *Iptables* é um *Firewall* em nível de pacotes e funciona baseado no endereço de rede e da porta de origem ou destino do pacote. Sua lógica de funcionamento baseia-se na comparação de regras para verificar se um pacote pode ou não trafegar pela rede. Essa filtragem é realizada com o *Netfilter*, módulo que é implementado no *Iptables*. Os principais recursos do *Netfilter* são: modificar e monitorar o tráfego da rede, redirecionar e modificar a prioridade de pacotes que chegam e saem, dividir o tráfego entre as máquinas e o *Network*

Address Translator (*NAT*). O *NAT*[8] realiza a conversão de endereços de rede, onde uma rede privada pode acessar uma rede pública (*Internet*) através de um único endereço público válido.

Santos (2004) ainda explica que o *Netfilter* trabalha com o conceito de tabelas, onde podem ser observadas três cadeias de regras (INPUT, OUTPUT e FORWARD) que funcionam da seguinte forma (figura 4):

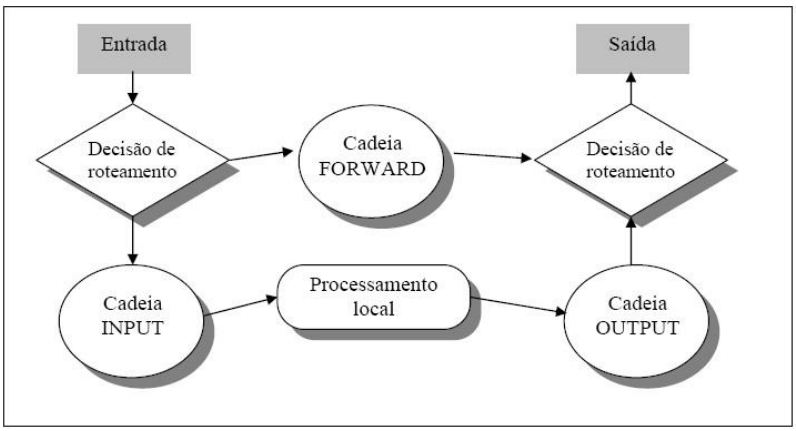

FIGURA 4 – Filtragem utilizando *netfilter* do *Iptables*

Fonte: adaptada de SANTOS (2004)

Para Santos (2004), o cabeçalho do pacote é comparado com cada regra até que encontre uma que case com seu cabeçalho, onde o alvo da regra é aplicado no destino do pacote (DROP, REJECT ou ACCEPT).

Para Silva (2004) a sintaxe básica do *Iptables* é definida por: “iptables [-t <tabela>] [comando] [ação] [alvo]”.

Silva (2004) explica que 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.

As sintaxes dos comandos do *Iptables* são:

- -A: adiciona uma nova entrada ao fim da lista de regras;
- –D: apaga uma regra específica da lista;
- –L: exibe as regras existentes na lista;

[8] É uma técnica que consiste em reescrever os endereços IP de origem de um pacote que passam por um roteador ou firewall de maneira que um computador de uma rede interna tenha acesso ao exterior (rede pública).

- –P: altera a política padrão das *chains*. Inicialmente, todas as chains estão setadas como ACCEPT, ou seja, aceitam todo e qualquer tipo de tráfego;
- –F: este comando remove todas as entradas adicionadas à lista de regras sem alterar a política padrão (-P);
- –I: insere uma nova regra ao início da lista de regras, contrário do comando –A;
- –R: substitui uma regra já adicionada por outra;
- –N: este comando nos permite inserir ou criar uma nova *chain* na tabela específica.
- –E: renomeia uma *chain*;
- –X: apaga uma *chain* criada pelo administrador do *Firewall*.
- As seguintes ações podem ser configuradas:
- -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...);
- –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.
- –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 às regras;
- –s: especifica a origem do pacote ao qual a regra deve ser aplicada. A origem pode ser um *host* ou uma rede;
- –d: especifica o destino do pacote ao qual a regra deve ser aplicada. Sua utilização se dá da mesma maneira que a opção –s;
- !: significa exclusão e é utilizada quando se deseja aplicar uma exceção a uma regra. É utilizada juntamente com as opções –s, -d, -p, –i, –o;
- –j: define o alvo (*target*) do pacote caso o mesmo se encaixe em uma regra;
- --sport: porta de origem do pacote, com está opção é possível aplicar filtros com base na porta de origem do pacote. Somente aplicados aos protocolos TCP ou UDP;

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

Silva (2004) ainda ressalta que os seguintes alvos podem ser configurados:

- ACCEPT: corresponde a aceitar, ou seja, permitir que a entrada e passagem do pacote em questão;
- 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;
- REJECT: corresponde a rejeitar, um pacote conduzido para este alvo (Target) é automaticamente descartado. A diferença do REJECT para o DROP é que o mesmo retorna uma mensagem de erro ao *host* emissor do pacote informando o que houve;
- LOG: cria uma entrada de log no arquivo /var/log/messages sobre a utilização dos demais alvos (Targets), justamente por isso devem ser utilizados antes dos demais alvos.
- RETURN: retorna o processamento do *chain* anterior sem processar o resto do *chain* atual;
- QUEUE: encarrega um programa em nível de usuário de administrar o processamento de fluxo atribuído ao mesmo;
- SNAT: altera o endereço de origem das máquinas clientes antes dos pacotes serem roteados;
- DNAT: altera o endereço de destino das máquinas clientes;
- REDIRECT: realiza o redirecionamento de portas em conjunto com a opção --toport;

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

De acordo com Neto (2004), o *Iptables* compõe a quarta geração de sistemas *Firewalls* no Linux, que foi incorporada a versão 2.4 do Kernel. Ele é uma versão mais completa e tão estável quanto seus antecessores *Ipfwadm* e *Ipchains*, implementados nos Kernels 2.0 e 2.2 respectivamente, onde também podemos destacar a primeira geração de *Firewalls IPFW* do *BSD*[9]. O *Iptables* é amplamente

[9] É um Sistema Operacional UNIX desenvolvido pela Universidade de Berkeley, na Califórnia, durante os anos 70 e 80.

utilizado devido as funções de *Firewall* estarem agregadas a própria arquitetura do *Kernel*.

Neto (2004) ainda afirma que 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.

Neto (2004) conclui que sob uma ótica mais prática podemos ver o *Netfilter* como um grande banco de dados que contêm em sua estrutura 3 tabelas padrões: *Filter*, *Nat* e *Mangle*.

2.3.1 Tabela Filter

Segundo Neto (2004), é a tabela padrão do *Netfilter* e trata das situações implementadas por um *Firewall* de filtro de pacotes. Estas situações são:

- INPUT: tudo o que entra no *host*;
- FORWARD: tudo o que chega ao *host* mas deve ser redirecionado a um *host* secundário ou outra interface de rede;
- OUTPUT: tudo o que sai do *host*.

2.3.2 Tabela NAT

Para Neto (2004), esta tabela implementa as funções de *NAT* no *host* de *Firewall*. O *NAT* por sua vez, possui diversas utilidades:

- PREROUTING: utilizada quando há necessidade em fazer alterações em pacotes antes que os mesmos sejam roteados;
- OUTPUT: trata os pacotes emitidos pelo *Firewall*;
- POSTROUTING: utilizado quando há necessidade de se fazer alterações em pacotes após o tratamento de roteamento.

2.3.3 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 (NETO, 2004):

- PREROUTING: modifica pacotes dando-lhes um tratamento especial antes que os mesmos sejam roteados;
- OUTPUT: altera os pacotes gerados localmente antes que os mesmos sejam roteados.

Segundo La-Roque (2004), o Linux possui funções de *Firewall* agregadas ao *Kernel*[10] pelo módulo *Netfilter*[11]. Tais tabelas possibilitam controlar todas as situações (*chains*[12]) 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:

- KERNEL 2.0 – IPFWAMD;
- KERNEL 2.2 – IPCHAINS;
- KERNEL 2.2/2.6 – IPTABLES.

Para Santos (2004), 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*.

Santos (2004) explica que devido estar incorporado diretamente ao *kernel*, a configuração do *Iptables* não se dá por via de arquivos de configuração, sua

[10] Núcleo de um Sistema Operacional. Local onde estão as rotinas de instruções básicas para operação de um computador.
[11] Módulo que fornece ao sistema operacional Linux as funções de *Firewall*, NAT e log dos dados que trafegam por rede de computadores.
[12] Cadeias para criação das regras do *Firewall*

manipulação é realizada por síntese digitada em *shell*, ou seja, uma regra na *shell* somente estará valendo para aquela sessão em memória.

Santos (2004) ainda ressalta que 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 utilizada 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 (SANTOS, 2004):

- implementação de filtros de pacotes;
- desenvolvimento de *QOS*[13] sobre o tráfego;
- suporte a *Source Network Address Translation* (*SNAT*) e *DNAT*;
- direcionamento de endereços e portas;
- mascaramento de tráfego;
- detecção de fragmentos;
- monitoração de tráfego;
- bloqueio de ataque de *Spoofing*[14], *Syn-Flood* ou *DoS*[15], *scanners ocultos*, *pings da morte*[16], entre muitos outros.

2.4 CONCEITOS DE WEB

Segundo Dias (2003), *WEB* é um conjunto de páginas web, isto é, de hipertextos acessíveis geralmente pelo protocolo *HTTP* na Internet. O conjunto de todos os sites públicos existentes compõe a *World Wide Web*. As páginas num site são organizadas a partir de um *URL* básico, ou sítio onde fica a página principal, e geralmente residem no mesmo diretório de um servidor. As páginas são organizadas dentro do site numa hierarquia observável no *URL*, embora as hiperligações entre

[13] Aptidão de garantir um nível aceitável de perda de pacotes, definido contratualmente, para um uso dado (vozes em IP, videoconferência, etc.).
[14] Técnica sofisticada utilizada por hackers e crackers que os permite acessar sistemas controlados passando-se por pessoa autorizada a fazê-lo.
[15] É uma forma de ataque de negação de serviço em sistemas computadorizados.
[16] É uma forma de ataque a um computador que consiste em enviar um ping malformado e malicioso.

elas controlem o modo como o leitor se apercebe da estrutura global, modo esse que pode ter pouco a ver com a estrutura hierárquica dos arquivos do site.

A *Internet* surgiu na década de 60, no período da guerra fria, como um projeto militar norte americano chamado de *ARPANet*. O principal objetivo era a criação de uma rede estável para no caso de uma eventual catástrofe, garantir a comunicação entre diferentes áreas geográficas. Mais tarde foi se expandindo para os departamentos científicos das universidades. Nos anos 90 começou a ser usada nos meios comerciais como uma poderosa ferramenta de comunicação em massa, onde empresas privadas tiveram grande influência no crescimento da Internet (PALMA; PRATES, 2003).

Desse projeto, surgiram o protocolo Transmission Control Protocol (*TCP*) e o Internet Protocol (*IP*) com o objetivo de interligar computadores do governo americano de diferentes fabricantes, que foi a base para a construção da rede que hoje é chamada de Internet (PALMA; PRATES, 2003).

2.5 CONCEITOS DE PROTOCOLOS TCP/IP E UDP

Do projeto da Internet, surgiram o protocolo Transmission Control Protocol (*TCP*) e o Internet Protocol (*IP*) com o objetivo de interligar computadores do governo americano de diferentes fabricantes, que foi a base para a construção da rede que hoje é chamada de Internet. O *TCP/IP* é um conjunto de protocolos usados em redes de computadores, onde *TCP* e *IP* são dois protocolos dessa família e por serem os mais conhecidos, tornou-se comum usar o termo TCP/IP para se referir à família inteira (PALMA; PRATES, 2003).

Palma e Prates (2003) explicam que o *TCP* e o *IP* são protocolos da camada de transporte e rede respectivamente, onde o *TCP* é um protocolo orientado à conexão e com garantia de entrega do pacote. O protocolo *IP* é responsável por traçar um caminho por onde os dados serão enviados. Em uma rede de computadores, além dos protocolos que são responsáveis pela comunicação, existem os dispositivos de rede que são as portas de entrada e de saída dos dados de cada equipamento de rede. A esses dispositivos são atribuídos endereços *IP*, que são os endereços lógicos de cada dispositivo, responsáveis por identificar cada equipamento na rede.

Protocolo é a "linguagem" usada pelos dispositivos de uma rede de modo que eles consigam se entender, isto é, trocar informações entre si. Para que todos os dispositivos de uma rede consigam conversar entre si, todos eles deverão estar usando uma mesma linguagem, isto é, um mesmo protocolo (SANTOS, 2004).

Santos (2004) conclui que além dos protocolos mencionados, pode-se destacar o User Datagram Protocol (*UDP*) que é um serviço de entrega de datagramas sem conexão, que não garante a entrega nem a integridade das informações. Na comunicação utilizando *UDP*, o computador que envia o datagrama não pede confirmação de entrega ao computador de destino, sendo assim um serviço não confiável. Esse protocolo é muito utilizado na transmissão de sinais onde a velocidade de transmissão é mais importante que a confiabilidade dos dados.

2.6 CONCEITOS DO SISTEMA OPERACIONAL LINUX

Segundo Hunt (2000) o Linux é um clone *UNIX* de distribuição livre para computadores baseados em processadores 386/486/Pentium. A implementação independente da especificação *POSIX*, com a qual todas as versões do *UNIX* padrão *(true UNIX*) estão convencionadas. Foi primeiramente desenvolvido para computadores baseados em 386/486/Pentium, mas atualmente também roda em computadores Alpha da *DEC*, Sparcs da *SUN*, máquinas M68000 (semelhantes a Atari e Amiga), *MIPS* e *PowerPCs*. Escrito inteiramente do nada, não há código proprietário em seu interior e está disponível na forma de código objeto, bem como em código fonte, por isso pode ser livremente distribuído nos termos da *GNU* General Public License (Licença Pública Geral).

Hunt (2000) afirma que o Linux possui todas as características que você pode esperar de um *UNIX* moderno, incluindo:

- Multitarefa real;
- Memória virtual;
- Biblioteca compartilhada;
- "Demand loading";
- Gerenciamento de memória próprio;;

• Executáveis "copy-on-write" compartilhados;;

• Rede TCP/IP;;

• X Windows (interface gráfica).

Segundo Hunt (2000) o *Kernel* do Linux foi, originalmente, escrito por Linus Torvalds do Departamento de Ciência da Computação da Universidade de Helsinki, Finlândia, com a ajuda de vários programadores voluntários através da Internet. No dia 5 de outubro de 1991 Linus Torvalds anunciou a primeira versão "oficial" do Linux, versão 0.02. Desde então muitos programadores têm respondido ao seu chamado, e têm ajudado a fazer do Linux o Sistema Operacional que é hoje.

Ainda para Hunt (2000) a maioria dos programas rodando em Linux *são freeware* (grátis) genéricos para UNIX, muitos provenientes do projeto GNU. Muitas pessoas têm executado *benchmarks* (programas que executam comparações de velocidade de *softwares)* em sistemas Linux rodando em 80486, e tem achado o Linux comparável com *workstations* (estações de trabalho) médias da Sun e da Digital.

Hunt (2000) conclui que o Linux está disponível através da Internet por meio de centenas de sites FTP onde é usado por centenas e centenas de pessoas pelo mundo para desenvolvimento de *softwares, networking* (intra-office e Internet), e como plataforma de usuário final. O Linux tem se tornado uma alternativa efetiva de custo em relação aos caros sistemas UNIX existentes.

2.7 CONCEITOS DA LINGUAGEM PHP 5

Segundo Converse e Park (2001), “...o PHP destaca-se pela sua capacidade, confiabilidade e facilidade de uso. Além disso, oferece o melhor tipo de conectividade para todos os servidores back-end.”

Conforme PHP Group (2004), a versão 5 do PHP traz inúmeras vantagens em relação a seu precursor. Entre elas, é citada uma nova modelagem do gerenciamento de memória e também um completo sistema de suporte á orientações a objetos. Na Figura 5 abaixo, é citada uma das características desta nova modelagem de suporte a orientação a objetos:

```
class Person {
    var $name;
    function getName() {
        return $this->name;
    }
    function setName($name) {
        $this->name = $name;
    }
    function Person($name) {
        $this->setName($name);
    }
}

function changeName($person, $name) {
    $person->setName($name);
}

$person = new Person("Andi");
changeName($person, "Stig");
print $person->getName();
```

Figura 5 –Modelagem do suporte a orientação a objetos no PHP 5.

Fonte: adaptada de PHP Group (2004).

Este pedaço de código, quando executado através do PHP 4, irá ter como resultado de saída a palavra “Andi”. Isto ocorre porque é passado o objeto $person para a função changeName() por valor, e assim a função changeName() trabalha com um clone de $person (PHP Group, 2004).

Segundo Soares (2000), no PHP 5, a infraestrutura da modelagem do objeto foi reescrita para trabalhar com os handles dos objetos. A menos que um objeto seja explicitamente criado clonado um objeto através de um operador, o gerenciador não irá criar clones de objetos.

2.8 CONCEITOS DO SERVIÇO APACHE

Para Marcelo (2005), falar do serviço Web Apache é a mesma coisa que falar sobre um dos softwares mais utilizados nos dias de hoje na internet. O Apache, sem sombra de dúvida, é um dos mais robustos e seguros programas desenvolvidos para ambientes TCP/IP e quem mantém em operação mais de 60% das *homepages*/*sites* disponíveis no mundo.

Marcelo (2005) ainda ressalta que a história do Apache e de seu desenvolvimento começa em 1995 quando a *NCSA* (National Center for Computer Aplications) criou o antigo *NSCA Web Server*, que naqueles tempos veio a tornar-se

o servidor de *http* mais popular existente. Põem a *NSCA* não foi muito adiante com esse projeto tornado-o estagnado, porém, alguns desenvolvedores do projeto continuaram a melhorá-lo e a desenvolver novas atualizações nascendo assim o mais conhecido *Web Server*, o *Apache* (derivação de *Apatchy,* um trocadilho devido ao enorme número de atualizações que foram criadas pela equipe) onde teve seu lançamento oficial em 1995 sob a versão 0.62.

Marcelo (2005) conclui que com isso surgiu o chamado *Apache Group*, ou mais conhecida, a *Fundação Apache*, famosa hoje por manter o mais popular servidor web do mercado que, sem sombra de dúvida, é o líder absoluto na *web*. Hoje os sites mais populares estão *debaixo* do *Apache*, criando assim uma comunidade de usuários espalhada pelo mundo.

2.8.1 Principais Vantagens

Para MARCELO (2005), as principais vantagens a ser citadas pelo servidor *web Apache* são:

- suporte a HTTP 1.1 para criação de hosts virtuais baseados em DNS (*Domain Name Server*);
- suporte a transações seguras em protocolos *SSL* (*Secured Socket Layer*);
- suporte a lingugens *CGI´s*, *Pearl* e *PHP*;
- suporte a autenticação baseada em HTTP;
- suporte a *Serlets Java*;
- logs Customizáveis;
- configuração rápida e simples.

Marcelo (2005) conclui que o *Apache* possui uma outra vantagem que o torna bastante atraente: é gratuito. Por ser um *software* livre, o código fonte do *Apache* é livre e pode ser instalado em vários servidores diferentes, desde que seja obedecida a licença *GNU Public Licence*. Com isso esse *Web Server* se torna imbatível em muitos aspectos.

2.8.2 Restringir acesso a páginas

Segundo Marcelo (2005) a versão 2.0 do *Apache*, também conhecida como *Apache2*, vem com uma importante ferramenta, essa responsável por limitar o acesso a alguns diretórios do servidor que está hospedado o serviço *web.*

Essa restrição, segundo Marcelo (2005) é muito importante ao se desenvolver programas baseados em plataformas *web* como um nível de segurança a mais.

Marcelo (2005) ainda ressalta que a restrição é feita através de um cadastro de usuários e senhas, essas criptografadas com o algoritmo *MD5*.

Segundo Castillo (2006), o *MD5* (*Message-Digest algorithm 5*) é um algoritmo de *hash*[17] de 128 bits unidirecional desenvolvido pela *RSA Data Security*, Inc., descrito na *RFC 1321*, e muito utilizado por softwares com protocolo ponto-a-ponto (*P2P*, ou *Peer-to-Peer*, em inglês) na verificação de integridade de arquivos e login.

Castillo (2006) ressalta que o algoritmo foi desenvolvido em 1991 por Ronald Rivest para suceder ao *MD4* que tinha alguns problemas de segurança. Por ser um algoritmo unidirecional, uma *hash MD5* não pode ser transformada novamente no texto que lhe deu origem. O método de verificação é, então, feito pela comparação das duas *hash* (uma da mensagem original confiável e outra da mensagem recebida). O *MD5* também é usado para verificar a integridade de um arquivo através, por exemplo, do programa *md5sum*, que cria a *hash* de um arquivo. Isto se pode tornar muito útil para downloads de arquivos grandes, para programas *P2P* que constroem o arquivo através de pedaços e está sujeitos a corrupção dos mesmos. Como autenticação de login é utilizada em vários sistemas operacionais *UNIX* e em muitos sites com autenticação.

[17] sequência de bits geradas por um algoritmo de dispersão, em geral representada em base hexadecimal, que permite a visualização em letras e números (0 a 9 e A a F), representando 1/2 byte cada.

3 METODOLOGIA

Para o presente estudo foi utilizado um computador com o sistema operacional *Linux* devidamente instalado contendo o *NetFilter* já compilado para utilização do *Iptables* e também o serviço *Apache* instalado no S.O. (sistema operacional).

A aplicação foi desenvolvida na linguagem *PHP*.

Foi utilizado no estudo um computador com processador *Intel Dual-Core* da *Intel Inside* com 1024Mb (1Gb) de memória *RAM* e *HD* (disco rígido) de 80Gb.

4 DESENVOLVIMENTO

Neste capítulo são apresentadas técnicas e ferramentas utilizadas no desenvolvimento da interface.

A interface desenvolvida neste trabalho utiliza uma aplicação *web* para gerenciamento do *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 a interface deverão ser definidos no servidor *Apache*.

A interface permitirá a manutenção nas tabelas *Filter*, *Nat* e *Mangle*, onde poderão ser criadas, consultadas, alteradas e restauradas regras.

4.1 DESENVOLVIMENTO DA INTERFACE

A aplicação foi desenvolvida na linguagem *PHP* onde é compilado através do servidor que está instalado o serviço *Apache* e o *Iptables.* Para seu desenvolvimento, foram utilizadas as técnicas de programação estudadas nos levantamentos bibliográficos.

O primeiro passo foi criar a página onde o usuário pudesse escolher qual a opção gostaria de executar.

A figura 6 apresenta a tela inicial do programa logo após a autenticação do usuário.

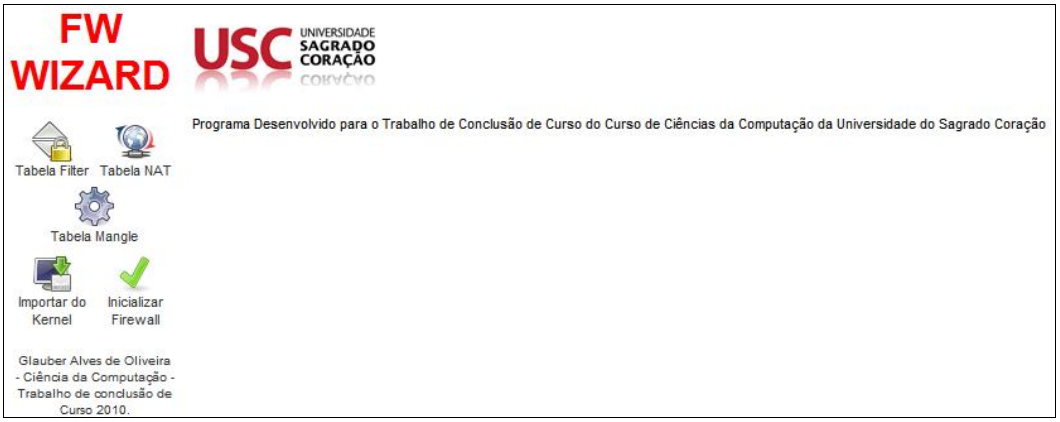

Figura 6 – Tela inicial do gerenciador do *Firewall.*

Do lado esquerdo da tela ficam as opções de controle da interface, nela podemos encontrar as três tabelas (*Filter*, *NAT* e *Mangle*) e logo abaixo a opção de importação da regra do Kernel do Linux para o arquivo de configuração do *Firewall* e a opção de inicializá-lo.

Antes de desenvolver os algoritmos que serão executados no *Firewall*, é muito importante conhecer a fundo todas as funcionalidades das regras do *Iptables.*

Após conhecer os comandos e todas as possíveis situações de regra, pode-se desenhar a interface dividindo-o em três partes:

1. Criação de um módulo para as regras da tabela *Filte*r;
2. Criação de um módulo para as regras da tabela *NAT;*
3. Criação de um módulo para as regras da tabela *Mangle.*

Para o desenvolvimento do módulo da tabela *Filter,* o programa foi desenvolvido e separado em três cadeias:

1. Cadeia *INPUT*;
2. Cadeia *FORWARD*;
3. Cadeia *OUTPUT.*

Cada uma dessas cadeias deve ter uma política de inicialização, ou seja, ou ela bloqueia todos os pacotes que não estão liberados (política BLOQUEAR) ou ela aceitas todos os pacotes que não estão como bloqueados nas regras (política ACEITAR).

A figura 7 mostra as opções que o usuário encontrará ao clicar na opção da tabela *Filter*.

Figura 7 – Opções da tabela *Filter*.

O desenvolvimento do módulo da tabela *NAT,* seguiu os mesmos critérios da tabela *Filter*, porém as cadeias são diferentes.

A figura 8 mostra as opções da tabela *NAT.*

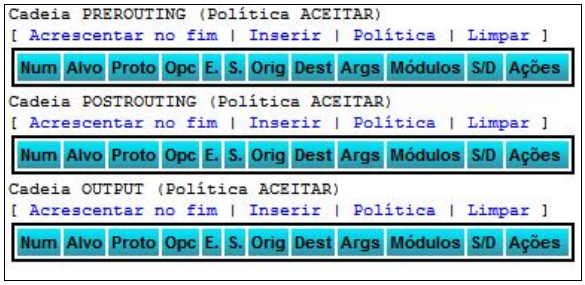

Figura 8 – Opções da tabela *NAT*.

Podemos identificar claramente a diferença entre as tabelas, a tabela *NAT* é utilizada para redirecionamento de pacotes e protocolos, essas podem ser feitas antes da leitura do pacote (PREROUTING) ou após (POSTROUTING), ela também pode redirecionar pacotes que estão saindo do servidor (OUTPUT).

O ultimo módulo a ser desenvolvido foi o da tabela *Mangle.*

Assim como foi exposto no capítulo 2, com a tabela *Mangle* temos um forte aliado para segurança dos pacotes, informações contidas neles e também tratar de maneira diferenciada os pacotes que saem do próprio servidor, dando a ele prioridades diferentes.

A figura 9 mostra as opções da tabela *Mangle.*

Figura 9 – Opções da tabela *Mangle.*

Observando a figura acima, todas as cadeias das tabelas possuem 4 opções básicas:

1. Acrescentar no fim: Essa opção permite que a regra seja acrescentada no fim da lista das regras da cadeia;
2. Inserir: Essa opção permite inserir a regra no topo da cadeia;
3. Política: Essa opção permite alterar a política da cadeia, ou seja, ou ela será de bloquear tudo ou de liberar tudo;
4. Limpar: Essa opção limpa todas as regras da cadeia.

Todas as regras que serão inseridas permanecem na memória do servidor, ou seja, caso o servidor venha a sofrer alguma queda de energia, as regras serão perdidas. Para que isso não se torne um problema, foram desenvolvidos dois módulos onde é possível restaurar as regras e fazer backup que estão na memória para um arquivo.

O primeiro módulo chamado de “Importar do Kernel” faz a leitura das regras que estão na memória do servidor e redireciona a um arquivo de texto do servidor.

O segundo módulo chamado de “Iniciar Firewall” é a restauração, ou seja, com ele é possível restaurar o que está no arquivo para a memória do servidor, tornando novamente ativa todas as regras existentes no momento que foi efetuado o *backup*.

4.1.1 Desenvolvimento dos módulos da interface

O primeiro módulo desenvolvido foi o módulo que contempla as cadeias da tabela *Filter*.

A tabela *Filter* contém em sua estrutura três cadeias básicas como visto no capítulo 2, essas cadeias são para as regras de filtro de entrada, saída e para redirecionamento.

Ao selecionar a opção de inclusão de uma regra em alguma dessas cadeias, foi levado em consideração todas as possibilidades de regras que a cadeia suporta.

A interface verifica em qual tipo de cadeia que está sendo inserida a regra e disponibiliza as opções que podem ser executadas para ela.

A figura 10 é exibida quando o usuário deseja inserir uma regra na cadeia INPUT.

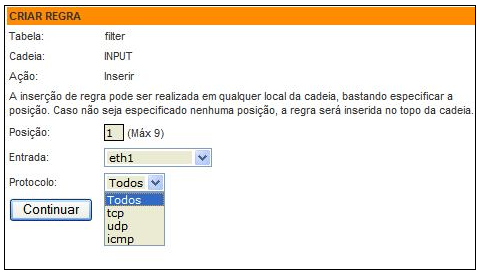

Figura 10 – Criação de regra na tabela *Filter*.

Como verificado na figura 10, o primeiro passo que o usuário precisará informar é qual a posição que a regra irá ocupar (o seu ID), entrada (interface de rede) e o protocolo.

A partir daí ele segue para o segundo passo onde entra com os dados de IP de origem, destino, alvo, porta de origem e destino.

A figura 11 mostra a tela onde o usuário irá inserir sua regra.

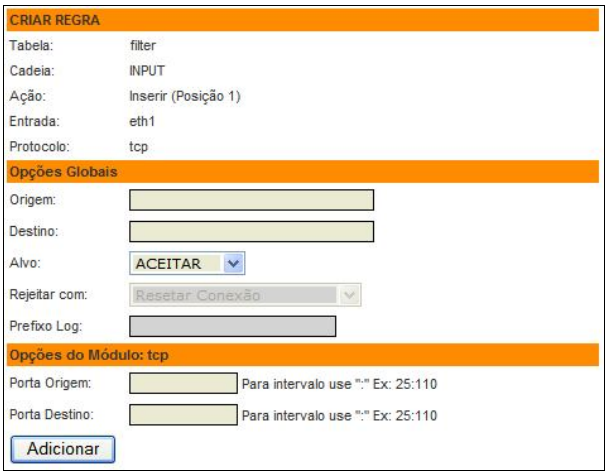

Figura 11 – Finalizar criação de regra na tabela *Filter*.

Ao adicionar a regra, a mesma permanece somente na memória do servidor *Firewall*, portanto é importante após a criação de todas as regras acessar a opção de importar do Kernel para que seja gravado o arquivo com as regras criadas.

Após a criação da regra, é exibido o comando que foi executado no servidor *Firewall*, a figura 12 mostra como ela é exibida.

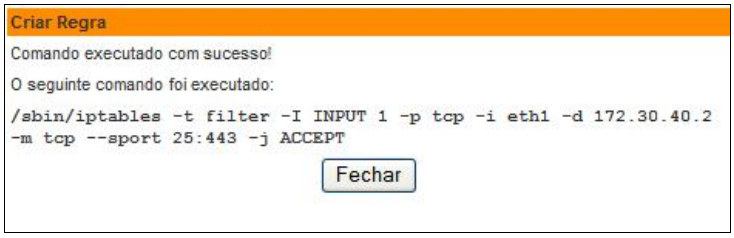

Figura 12 – Exibição do comando executado após criação da regra.

Para o programa identificar as sintaxes, é preciso que o módulo consiga identificar cada situação de entrada. A figura 13 mostra uma parte do código que o programa analisa para gerar o comando.

```
$cmd="$ipt -t $table ";
if ($type=="append")
    $cmd.="-A $chain ";
else
    $cmd.="-I $chain $position ";
if ($protocol!="all")
    $cmd.="-p $protocol ";
    //$cmd.="-p $protocol -m $protocol ";

if (isset($ifacein) and $ifacein!="any")
    $cmd.="-i $ifacein ";

if (isset($ifaceout) and $ifaceout!="any")
    $cmd.="-o $ifaceout ";

if (isset($moduleoption["all"]) and count($moduleoption["all"] > 0)) {
    foreach ($moduleoption["all"] as $option => $value) {
        if ($value!="")
            $cmd.="$option $value ";
    }
}
```

Figura 13 – Trecho do código do módulo gerador de comando

A figura 14 mostra outro trecho do código que complementa o comando com as sintaxes e executa no servidor *Firewall*.

```
$cmd.="-j $target ";
switch($target) {
    case "REJECT"   : $cmd.="--reject-with $rejectwith "; break;
    case "LOG"      : $cmd.="--log-prefix \"$logprefix\" "; break;
    case "REDIRECT" : $cmd.="--to-ports $toports "; break;
    case "SNAT"     : $cmd.="--to-source $snat "; break;
    case "DNAT" : $cmd.="--to-destination $dnat "; break;
    case "TOS"  : $cmd.="--set-tos $tos "; break;
}

$file="/tmp/firewalladmin-".rand();
echo exec("sudo $cmd &> $file",$output,$return);
```

Figura 14 – Trecho do código que executa o comando do servidor.

A criação dos outros módulos foi baseada utilizando essas mesmas funções, porém com suas particularidades de cadeias.

Todas as funções que foram criadas servem para todos os módulos, cada um com as suas sintaxes dependendo da cadeia e tabela.

Para cada comando executado no *Firewall*, o aplicativo exibe o comando para que o usuário se identifique com as sintaxes do *Iptables* e também como forma de didática automática.

Após a criação das regras, as mesmas são exibidas na tela como mostra a figura 15.

Figura 15 – Tela que mostra as regras da tabela selecionada.

Quando uma regra é criada, ele fica disponível para ser visualizada na tela do sistema e pode ser atualizada ou removida.

As telas estão divididas por Tabelas e Cadeias, ou seja, para cada Tabela é exibida de maneira ordenada as regras de cada Cadeia para facilitar a sua identificação.

As opções de Atualização e Exclusão das regras ficam ao lado de cada uma delas, e antes da realização dessas tarefas é exibida uma tela de confirmação, para evitar qualquer tipo de “acidente” quando o usuário clicar sem intenção na opção de excluir uma regra.

A opção de Atualização de regra funciona idêntica a de criação, porém ela cria a regra na mesma posição da regra que será atualizada, ou seja, ela substitui a regra antiga.

5 RESULTADOS OBTIDOS

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 desta interface serão discutidos os testes e resultados obtidos com o desenvolvimento desse aplicativo.

A política adotada para esse *Firewall* será de bloquear tudo que não estiver liberado nas regras estabelecidas. Supondo que fosse criada uma regra para liberar o *SSH* do *host* (172.30.40.51) para o *Firewall* (172.30.40.2), seria necessário liberar a entrada do pacote no Firewall em seguida permitir a saída do pacote para enviar uma resposta à estação que solicitou a requisição.

A figura 16 mostra os parâmetros utilizados para criar a regra citada acima. No campo “Origem” foi informado o endereço de *IP* do *host* que irá acessar o servidor. Posteriormente é informado o Destino do pacote no campo “Destino” que é o endereço de *IP* do *Firewall*, em seguida a porta utilizada pelo mesmo, no campo “Porta Origem”.

Figura 16 – Inserir uma regra de liberação de *SSH*.

Ao inserir essa regra, o seguinte comando é executado: “*iptables -t filter -I INPUT 1 -p tcp -s 172.30.40.51 -d 172.30.40.2 -m tcp --sport 22 -j ACCEPT”*

Nesse momento, o *Firewall* está com essa regra criada, como pode ser visto na figura 17.

Figura 17 – Nova regra inserida.

Nesse momento, tentar conectar no *Firewall* pela porta 22, que é a porta que normalmente representa o *SSH*, ainda não seria possível, pois ainda precisaria ser inserida na cadeia OUTPUT a regra para que o *Firewall* possa emitir a resposta da requisição.

Após inserir a regra da cadeia OUTPUT, cria-se então um laço de confiança entre a estação e o *Firewall* para conexões vindas daquela porta.

A figura 18 mostra que a conexão pôde ser realizada com sucesso.

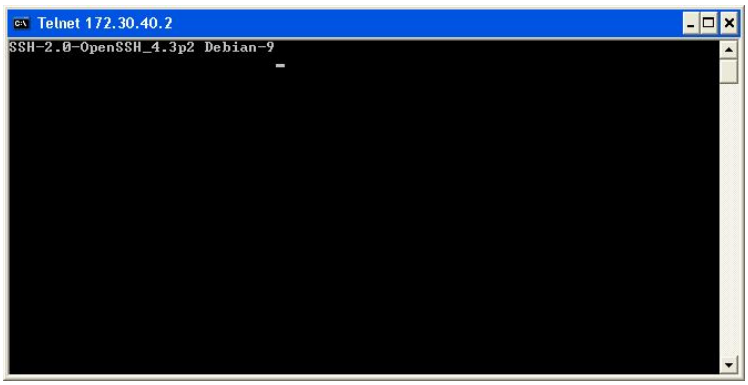

Figura 18 – Conexão estabelecida com o *SSH* do *Firewall*.

Qualquer outro *host* que não estiver liberado nas regras do *Firewall terá* seus pacotes automaticamente bloqueados.

Após inserir algumas regras na tabela INPUT, temos a seguinte situação no *Firewall* como demonstrado na figura 19.

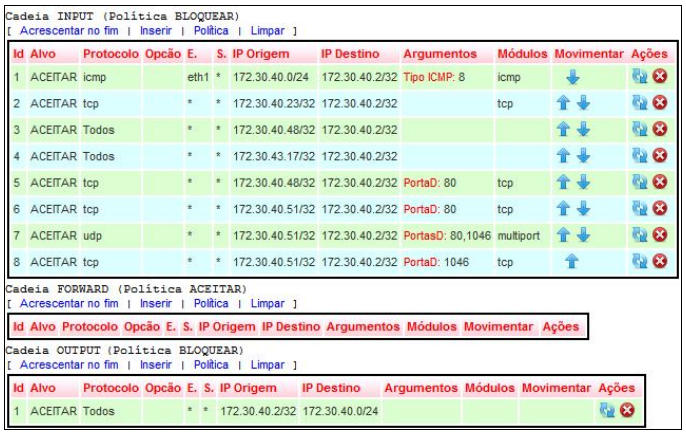

Figura 19 – Regras inseridas no *Firewall*.

Para listar as regras da tabela *Filter* diretamente no *Firewall*, é necessário acessar o servidor via *SSH* e executar o comando “iptables -L”, a figura 20 mostra o resultado do comando.

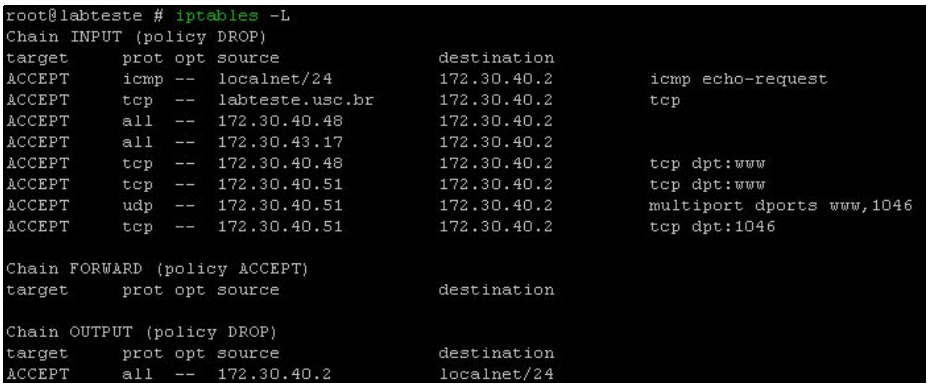

Figura 20 – Listagem das regras inseridas no *Firewall*.

Durante a criação das regras para testes foram liberadas diferentes *hosts,* portas e protocolos, sendo que um dos resultados foi o apresentado acima. Resultados semelhantes aos visualizados foram obtidos nas tabelas *NAT* e *Mangle* com suas respectivas cadeias.

6 CONCLUSÕES

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

É importante ressaltar que para a implementação da interface, citar os principais pontos que foram decisivos para o desenvolvimento do trabalho e dentre os quais podemos destacar: a interface intuitiva para criação das regras sem precisar saber os comandos do *Iptables*; estruturar os módulos do *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 à interface, tendo assim um nível de segurança maior.

A interface foi desenvolvida para realizar criação e manutenção em regras de *Firewall* do *Iptables*. A aplicação oferece uma interface *web* intuitiva de acordo com a tabela e cadeia selecionada para criação de regras, permitindo visualizar todas as regras distintas por tabelas além de excluir ou modificar regras. Outra funcionalidade importante é o *backup* e a restauração das tabelas. Com os resultados obtidos, considera-se que a interface atingiu os objetivos propostos.

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 TRABALHOS FUTUROS

Como extensão desse trabalho, propõe-se a implementação de um módulo de *download* do arquivo das regras, para que o mesmo possa ser salvo em qualquer local. Também se propõe a criação de um módulo de visualização dos *logs* e alertas ao administrador do programa e algumas regras pré configuradas para as principais ameaças de redes e ataques.

REFERÊNCIAS BIBLIOGRÁFICAS

CASTILLO, J. **MD5 Implementation**. Versão 1.1.1.1, 2006. Disponível em <http://www.opencores.org/lgpl.shtml>. Acesso em 15 ago. 2010.

CHESWICK, WILLIAM R.; BELLOVIN, S. M.; RUBIN, AVIEL D. **Firewall e seguranca na Internet**. 2. ed. Tradução Edson Frumankiewicz. São Paulo: Bookman, 2003.

CONVERSE, T; PARK, J. **PHP a bíblia**. Tradução Edson Furmankiewicz. Rio de Janeiro: Campus, 2001.

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

DIAS, C. "**Usabilidade na Web: Criando Portais mais Acessíveis**", Alta Books, Rio de Janeiro, 2003.

HUNT, C. **Servidores de Redes com Linux.** 1ª Ed. São Paulo: Market Books, 2000

LA-ROQUE, E. **Xfwall tutorial 01**: micro stand-alone/acesso discado. [S.l.], 2004. Disponível em: <http://www.starlinux.com.br/tutorial_01>. Acesso em: 26 mar. 2010.

MARCELO, A. **Apache** Configurando o servidor Web Linux. Rio de Janeiro: Brasport, 2005

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 Abr. 2010.

NETO, U. **Dominando Linux Firewall Iptables**. Rio de Janeiro: Ciência Moderna,2004.

PALMA, L.; PRATES, R. **TCP/IP.** São Paulo: Novatec, 2003.
PHP Group. PHP: hypertext preprocessor. [S.l.], 2004. Disponível em: <http://www.php.net/>. Acesso em: 24 mar. 2010.

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

ROGER, D. **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, L. C. **Firewall Linux IPTABLES**. Rio de Janeiro, 2004. Disponível em: <http://www.clubedasredes.eti.br/rede0023.htm>. Acesso em: 25 mar. 2010.

SILVA, A. C. V. **Instalando *Firewall* no Linux de modo fácil.** [S.l.], 2004. Disponível em: <http://br-linux.org/tutoriais/002028.html>. Acesso em: 25 mar. 2010.

SILVA, A. P. **Segurança máxima para Linux**. Rio de Janeiro: Campus, 2000.

SOARES, W. **Programando em PHP**. Rio de Janeiro: Érica, 2000.

SOBRAL, J. B. M. **Segurança em Computação Distribuída**, 2005
Disponível em: <http://www.inf.ufsc.br/~bosco/>. Acesso em: 12 abr. 2010.

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