UNIÃO EDUCACIONAL MINAS GERAIS S/C LTDA
FACULDADE DE CIÊNCIAS APLICADAS DE MINAS
Autorizada pela Portaria no 577/2000 – MEC, de 03/05/2000
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

GERENCIAMENTO DE REGRAS DE FIREWALL IPTABLES

EM AMBIENTE LINUX

CLAUDYSON JONATHAS ESQUIVEL

Uberlândia
2006

CLAUDYSON JONATHAS ESQUIVEL

GERENCIAMENTO DE REGRAS DE FIREWALL IPTABLES EM AMBIENTE LINUX

Trabalho de Final de curso submetido à UNIMINAS como parte dos requisitos para a obtenção do grau de Bacharel em Sistemas de Informação.

Orientador:
Prof. Msc. Sílvio Bacalá Júnior

Co-orientador:
Esp. Flamaryon Guerim Gomes Borges

Uberlândia

2006

CLAUDYSON JONATHAS ESQUIVEL

GERENCIAMENTO DE REGRAS DE FIREWALL IPTABLES EM AMBIENTE LINUX

Trabalho de Final de curso submetido à UNIMINAS como parte dos requisitos para a obtenção do grau de Bacharel em Sistemas de Informação.

Orientador
Prof. Msc. Sílvio Bacalá Júnior

Co-orientador
Esp. Flamaryon Guerim Gomes Borges

Banca Examinadora:

Uberlândia, 21 de Dezembro de 2006.

Prof. Msc. Sílvio Bacalá Júnior (Orientador)

Esp. Flamaryon Guerim Gomes Borges(Co-orientador)

Prof. Msc Luiz Leonardo Siqueira

Uberlândia

2006

AGRADECIMENTOS

Agradeço a Deus, a minha família pelo apoio e compreensão, aos meus amigos que estiveram presentes durante essa jornada. Obrigado a todos que fizeram parte dessa conquista.

RESUMO

A segurança da informação tem sido um dos principais objetos de preocupação das grandes corporações. O *Firewall* do ambiente Linux, denominado IPTABLES, é um dos recursos mais utilizados por administradores de redes e sistemas computacionais para garantir a proteção e o filtro dos dados trafegados via rede entre os computadores. Entretanto, a criação e configuração de regras para este tipo de *Firewall* é ainda uma tarefa que exige de tais profissionais um esforço e conhecimento avançados. Além de apresentar os conceitos relacionados a esta tecnologia, este trabalho pretende facilitar a criação e manutenção das regras com a construção de uma ferramenta gráfica de interface amigável e de fácil utilização. Ao final, é apresentado um estudo de caso onde será demonstrada a utilização da ferramenta.

Palavras-chave: segurança, *Firewall*, IPTABLES, Linux.

ESQUIVEL, C.J. **Gerenciamento de regras de *Firewall* IPTABLES em ambiente Linux.** 2006. Trabalho de Conclusão do Curso de Bacharelado em Sistemas de Informação, UNIMINAS, 2006.

ABSTRACT

The information security has been the main concerned object of big corporations. The *Firewall* of Linux environment, called IPTABLES, is one of the most used resources used by network administrators and system administrators to guarantee the protection and data traffic filter by network between computers. However the creation and configuration of rules for this type of *Firewall* is still a task that requires of these professionals an effort and advanced knowledge. Besides of showing the concepts related to this technology, this work intends to facilitate the creation and maintance of rules with a graphic tool building with friendly interface of easy utilization. At the end, is shown a case study where will be shown the utilization of this tool.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SÍMBOLOS

FWC – *Firewall Constructor*

NAT - *Network Address Translation*

UML - *Unified Modeling Language*

FBI - *Federal Bureau of Investigation*

DOS – *Denied of Service*

DDOS - *Distributed Denied of Service*

HTTP - *Hypertext Transfer Protocol*

HTTPS – *Secure HyperText Transfer Protocol*

TCP/IP - *(Transfer Control Protocol / Internet Protocol).*

NETBIOS - *Network Basic Input/Output System*

PING - *Packet Internet Groper*

ARP - *Address Resolution Protocol*

ICMP - *Internet Control Message Protocol*

IGMP- *Internet Group Management Protocol*

UDP - *User Datagram Protocol*

SSH - *Secure Shell*

RFC - *Request for Comments*

MAC - *Media Access Control*

SUMÁRIO

1 INTRODUÇÃO

1.1 Cenário Atual

Nas últimas décadas é crescente a preocupação, principalmente das empresas, com a segurança das informações, exigindo uma grande evolução das tecnologias de proteção de dados trafegados pelas redes de computadores. Freqüentemente são noticiados ataques a empresas de diferentes portes, causando, além de prejuízos financeiros, queda da reputação destas.

Um dos principais mecanismos de segurança existentes para minimizar os riscos de ataques é o *Firewall*, um programa que detém autonomia concedida pelo próprio sistema para pré-determinar e disciplinar todo o tipo de tráfego existente entre o mesmo e outros *host*/redes.

Em meados de 80, sob encomenda da AT&T, o primeiro *Firewall* do mundo foi desenvolvido com o intuito de "filtrar" todos os pacotes que saíssem e entrassem na rede corporativa, de modo a manipulá-los de acordo com as especificações das regras previamente definidas. A seleção de tais pacotes em um *Firewall* é realizada através de comandos que definem as regras para entrada e saída dos dados pela rede.

Mesmo diante da evolução dos meios tecnológicos, hoje um *Firewall* continua a possuir e empregar os mesmos conceitos, apenas com alguns aprimoramentos e implementações de novas funcionalidades.

1.2 Identificação do Problema

No Linux, as funções de *Firewall* são agregadas à própria arquitetura do *Kernel*, que é o núcleo do sistema operacional. Enquanto a maioria dos "produtos" *Firewall* pode ser definida como um subsistema, o Linux possui a capacidade de transformar o *Firewall* no próprio sistema. O módulo do *Kernel* no Linux responsável por realizar a função de um *Firewall* é chamado IPTABLES.

A configuração de um *Firewall* IPTABLES é realizada por uma série de comandos que são interpretados pelo *Kernel* do sistema operacional. Tais comandos podem ser executados via *scripts* (arquivos-texto) ou serem inseridos diretamente no *shell*.

Entretanto, esse cenário torna difícil a compreensão das regras e exige maior conhecimento do administrador do sistema. Existem algumas ferramentas que possibilitam configurações em algumas distribuições do Linux, mas são restritas e não oferecem muitos recursos.

1.3 Objetivos do Trabalho

Este trabalho tem por objetivo o desenvolvimento de uma ferramenta que possibilite a administração das regras do *Firewall* IPTABLES de maneira rápida, segura e através de uma interface amigável.

Para tal, pretende-se realizar um estudo sobre o *Firewall* IPTABLES, estudando a sua configuração em ambiente Linux de modo a criar um banco de dados com as regras, facilitando a configuração do ambiente que se deseja proteger.

Além disso, pretende-se demonstrar a eficácia da ferramenta utilizando-a em um ambiente real, para configurar o *Firewall* IPTABLES.

1.4 Justificativa para a Pesquisa

O grande número de regras necessárias para a configuração de um *Firewall* IPTABLES dificulta a compreensão e administração do sistema. A principal justificativa para a realização deste trabalho é que a inserção de tais regras no *Firewall* IPTABLES é hoje realizada manualmente no *prompt* de comando ou com agrupamento em arquivos-texto de difícil compreensão. Este trabalho irá contribuir com a construção de uma ferramenta que facilite essa configuração.

1.5 Organização do Trabalho

O Capítulo 2 apresenta os principais conceitos relacionados a este trabalho: *Firewall*, IPTABLES, ataques, invasões e protocolos.

No Capítulo 3 é apresentada a especificação dos requisitos do sistema proposto, utilizando diagramas da UML (*Unified Modeling Language*).

No Capítulo 4 são apresentados o esquema de utilização da ferramenta e um estudo de caso.

Por fim, o Capítulo 5 expõe mostra as conclusões obtidas no trabalho, as limitações do estudo realizado e sugestões de trabalhos futuros.

2 CONCEITOS RELACIONADOS

2.1 A Segurança em Redes

A necessidade de segurança é um fato que vem transcendendo o limite da produtividade e da funcionalidade. Enquanto a velocidade e a eficiência em todos os processos de negócios significam uma vantagem competitiva, a falta de segurança nos meios que habilitam a velocidade e a eficiência pode resultar em grandes prejuízos e falta de oportunidades de negócios. (NAKAMURA ; GEUS, 2003, p.9).

O mundo da segurança é marcado pela evolução contínua, no qual novos ataques têm como resposta novas formas de proteção que levam ao desenvolvimento de novas técnicas de ataques e assim sucessivamente. Esse mesmo comportamento pode ser observado no mundo da informação, onde também se deve ter em mente que a segurança deve ser contínua e evolutiva. (NAKAMURA ; GEUS, 2003, p.9).

Os seguintes fatores justificam a preocupação com a segurança contínua: a natureza dos ataques, as novas vulnerabilidades das novas tecnologias, a criação de novas formas de ataques, o aumento da conectividade, a complexidade da defesa, o aumento dos crimes digitais e os grandes prejuízos ocasionados pela falta de segurança. (NAKAMURA ; GEUS, 2003, p.10).

O ambiente corporativo é um ambiente que integra diversos sistemas de diferentes organizações. A rede é a tecnologia utilizada para realizar essa integração, permitindo conexões entre todos os seus elementos. A confiabilidade, integridade e disponibilidade da rede são essenciais para o próprio negócio da organização, justificando a preocupação com a segurança das informações.

Para segurança da informação devem ser considerados, além do aspecto tecnológico, os aspectos humanos, processuais, jurídicos e de negócios da organização. No aspecto tecnológico, a segurança de redes é parte essencial. Com o crescimento do comércio eletrônico, a segurança de redes passou a ser mais do que a proteção contra ataques, maus funcionários ou vírus, representando hoje um elemento habilitador dos negócios da organização.

Entretanto, ainda hoje a segurança é tratada de maneira superficial por grande parte das organizações. Não recebendo a devida importância e sem a definição

de uma boa estratégia de segurança, são utilizadas técnicas parciais ou incompletas que podem aumentar a vulnerabilidade da organização. (NAKAMURA; GEUS, 2003, p.10).

2.2 O que Proteger

Segundo ZWICKY (apud PALU, 2005, p.18) há basicamente três itens que devem ser protegidos por uma organização:

- Dados: é a informação propriamente dita, mantida nos computadores. Em relação aos dados, deve-se preocupar com o sigilo, integridade e disponibilidade;
- Recursos: são os computadores que formam a rede da organização;
- Reputação: é considerado o fator principal ao implantar segurança, já que uma invasão pode ocasionar grandes prejuízos relacionados à reputação da corporação.

2.2.1 Medindo as Perdas

Os tipos de perdas que as empresas podem experimentar por causa de lapsos na segurança dos computadores podem ser contabilizados das seguintes maneiras: (BURNETT,STEVE, 2002, p.266).

- Dados ou segredos: Perda de números de cartão de crédito do usuário, comprometimento de relatórios financeiros e acesso não-autorizado às informações.
- Perda de reputação: Às vezes uma avaliação negativa do analista pode causar um impacto tão grande quanto à própria invasão. Isso pode ser uma das principais razões pela qual as empresas raramente informam invasões e roubo de dados.
- Perdas financeiras: Além dos roubos financeiros diretos, a perda de dados e a perda de reputação resultarão em perdas financeiras.

2.3 Tipos de Invasores

Todos os dias, invasores indesejáveis realizam entradas não-autorizadas em sistemas de computador e redes. Esses indivíduos podem estar classificados em grupos: (BURNETT, STEVE, 2002, p.271).

2.3.1 Pessoal interno

A maioria das empresas quer acreditar que seus funcionários são confiáveis e nunca violaram a segurança corporativa. Entretanto, na realidade alguns funcionários não são o que eles aparentam ser. As pessoas que cometem crimes de segurança contra seus empregados são motivadas por diversas razões. O pessoal interno descontente é a principal fonte de crimes informatizados em várias empresas.

A pesquisa feita pelo FBI e pelo *Computer Security Institute* em 2000 informa que 71% dos entrevistados detectaram acessos não-autorizados aos sistemas pelo pessoal interno.

2.3.2 *Hackers*

Conforme NAKAMURA (2003, p.51), O termo *hacker* é utilizado genericamente para identificar quem realiza um ataque em um sistema computacional. Entretanto, há uma classificação específica com base no perfil, na experiência e no tipo de ataque que é realizado:

- *Script Kids*: iniciantes, inexperientes e novatos que conseguem ferramentas que podem ser encontradas prontas na internet;
- *Cyberpunks:* dedicam-se às invasões de sistemas por divertimento e desafio. Geralmente encontram vulnerabilidades em serviços, sistemas ou protocolos, prestando assim, um favor às organizações ao publicar as vulnerabilidades encontradas;
- *Insiders*: empregados insatisfeitos. São os maiores responsáveis pelos incidentes de segurança mais graves nas organizações;
- *Coders*: *hackers* que resolveram compartilhar seus conhecimentos em livros, palestras ou seminários;

- *White hat*: profissionais contratados com o objetivo de descobrir as vulnerabilidades dos sistemas e aplicar as correções necessárias. São conhecidos como *hackers* do bem;
- *Black hat*: utilizam conhecimentos para invadir sistemas ou roubar informações secretas das organizações. Conhecidos como *crackers*;
- *Gray hat*: *hackers* que vivem no limite entre o *white hat* e o *black hat*. Quando contratados para atuarem como *white hat* nem sempre cumprem bem o seu papel, causando danos à organização.

2.3.3 Terroristas

Os grupos terroristas utilizam uma tecnologia de informação e a internet para formular planos, levantar fundos, difundir propagandas e comunicar-se de maneira segura. Além disso, eles são conhecidos por se empenhar em ataques contras sites da WEB de governos estrangeiros e contras servidores de e-mail.

2.3.4 Serviços de Inteligência Estrangeiros

Os serviços de inteligência estrangeiros se adaptaram para utilizar as ferramentas cibernéticas como parte do seu aparato de espionagem. Esses serviços cada vez mais vêem as invasões de computador como ferramenta útil para adquirir informações sigilosas de governos e do setor privado.

2.3.5 Hactivistas

São ataques com motivos políticos contra páginas da WEB publicamente acessíveis ou contra servidores de e-mail e invadem sites da WEB e adulteram suas páginas com mensagens políticas. Embora esses ataques geralmente não adulterem os sistemas operacionais ou redes, eles danificam os serviços e negam o acesso público a sites da WEB que contém informações valiosas.

2.4 Protocolos de Comunicação

Para que possa haver troca de informações entre os pontos de uma rede é necessário que um protocolo de comunicação de dados esteja estabelecido para coordenar o envio e recebimento dos dados. O protocolo agrupa as informações em um pacote de dados lógico que possui um cabeçalho e um corpo. Os cabeçalhos contem informações como origem, destino, tamanho e tipo de pacote. O corpo do pacote contém os dados que se deseja transmitir através da rede.

O protocolo mais utilizado é o TCP/IP (*Transfer Control Protocol* / *Internet Protocol*).

2.4.1 TCP/IP

O TCP/IP é uma família de protocolos para comunicação em redes que surgiu em 1975. Tem especificações públicas e genéricas, permitindo serem implementados por diversos fabricantes. O TCP/IP utiliza um modelo de quatro camadas para a comunicação, são elas:

2.4.1.1 Camada de Aplicação

Esta quarta camada do modelo é responsável pelos aplicativos TCP/IP. Há dois tipos de aplicativos nessa camada: aplicativos baseados em soquete e aplicativos do sistema básico de saída e entrada de rede conhecidos como NETBIOS.

Os aplicativos baseados em soquete existem em todos os clientes que utilizam o TCP/IP. Três elementos são exigidos para os aplicativos baseados em soquete: um endereço IP, uma porta e um tipo de serviço. Cada cliente TCP/IP terá um endereço único de 65.536 pontos de entrada, chamadas portas.

Os aplicativos NETBIOS são comumente vistos em sistemas operacionais da Microsoft. O NETBIOS é um transporte de camada de sessão que fornece uma comunicação virtual aos aplicativos em diferentes clientes. Isso significa que os aplicativos parecem ser capazes de se comunicar com base unicamente em nomes de computador.

2.4.1.2 Camada de Transporte

Esta é a terceira camada do modelo TCP/IP. O propósito da camada de transporte é conectar ou não conectar. Dois protocolos são utilizados nessa camadas, são eles: *Tranmission Control Protocol* (TCP) e *User Datagram Protocol* (UDP). O TCP é uma comunicação orientada a conexão, quando um aplicativo utiliza o TCP um *handshake* de três vias é estabelecido e assegura que os pacotes são entregues livres de erros. O UDP é uma comunicação sem conexão, quando um aplicativo utiliza UDP não estabelece um *handshake* de três vias e não oferece a garantia de entrega de pacote, porém a comunicação é mais rápida na transmissão que o TCP.

2.4.1.3 Camada de Inter-rede

A segunda camada do modelo TCP/IP é a camada de inter-rede. Essa camada é responsável pelo endereçamento, roteamento de rede e fragmentação do pacote. Vários protocolos operam na camada de inter-rede, mas os mais comuns são:

2.4.1.4 Internet Control Message Protocol (ICMP)

Utilizado freqüentemente com o utilitário *Packet Internet Groper* (PING) que na maioria das vezes é útil para solucionar problemas de conectividade. Uma utilização mais avançada do ICMP é a solicitação do roteador. Os clientes podem utilizar o protocolo de descoberta de roteador do ICMP para localizar roteadores em uma rede.

2.4.1.5 *Address* Resolution Protocol (ARP)

É utilizado para determinar endereços IP em endereços MAC, ou seja, determinando o endereço físico da estação. Uma vez que o endereço de MAC é conhecido, o pacote pode ser enviado para o cliente diretamente ao cliente receptor se os clientes estiverem no mesmo segmento. Se os clientes estiverem em segmentos diferentes, o pacote é enviado ao roteador.

2.4.1.6 Internet Group Management Protocol (IGMP)

É utilizado para identificar membros em um grupo que receberam pacotes de dados *multicast*. O IGMP tem várias utilizações em uma rede, mas alguns dos mais comuns incluem: videoconferência, bate-papo na internet e atualizações dinâmicas de roteador.

2.4.1.7 Internet Protocol (IP)

Conforme SCRIMGER, LASALLE, PARIHAR, GUPTA (2002, p.26), É um protocolo sem conexão que fornece seleção de endereçamento e rota. As informações do cabeçalho adicionadas ao pacote de dados contêm os endereços os endereços de origem e destino e a seleção das rotas é feita com base nesses endereços.

O IP é o protocolo responsável por fazer a troca de pacotes de um modo mais simples. Atua na camada de inter-rede provendo as funções necessárias de roteamento dos dados entre as várias redes interconectadas. O cabeçalho do protocolo IP pode ser observado na Figura 1:

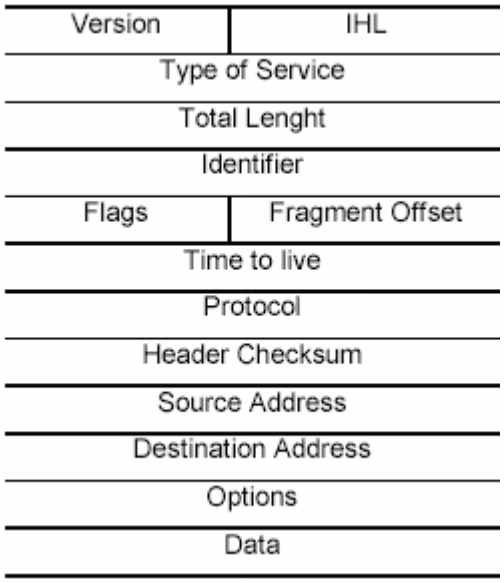

Figura 01. Datagrama IP

A seguir, a descrição de cada um dos campos do datagrama IP:

- *version* – indica a versão do protocolo em uso, atualmente a versão 4;

- *IHL* – Internet *Header* Lenght: informa o comprimento do cabecalho IP em unidades de 32bits;
- *type of service* – tipo de serviço: indica a qualidade de serviço requerida pelo datagrama;
- *total length* – comprimento total: informa o comprimento total do datagrama em bytes, incluindo o cabeçalho do IP;
- *identifier* – identificador: é número único com propósito de orientar a recomposição dos datagramas fragmentados;
- *flags* – indica os atributos relativos à fragmentação dos datagramas;
- f*ragment offset* – indica o deslocamento de blocos no datagrama;
- *time to live* – tempo de vida: indica o tempo em segundos no qual um datagrama permanece válido antes de ser descartado;
- *protocol* – protocolo: indica o protocolo da camada superior para o qual os dados contidos no datagrama devem ser passados;
- *header checksum* – checagem de cabeçalho: indica a integridade do cabecalho IP em nível de bit;
- *source address* – endereço de origem: indica o endereço de origem do *host*;
- *destination address* – endereço de destino: indica o endereço de destino do *host*;
- *options* – opções: indica configurações relacionadas a opções de controle;
- *data* – dados: indica o dados a serem trafegados.

2.4.1.8 Camada de Interface de Rede

A primeira camada do modelo TCP/IP é responsável pelo acesso à rede. A camada da interface de rede se comunica diretamente com a rede. Ela é a ligação entre a topologia de rede e a camada de inter-rede.

2.4.2 Endereçamento IP

O endereço IP é um número composto de 32 bits. Estes bits estão divididos em 4 conjuntos de 8 bits cada. A cada *host* é atribuído um endereço IP único.

Uma primeira parte desses bits é usada para identificar a rede à qual o *host* está conectado e a parte restante é usada para identificar o *host* na rede.

Os endereços IP's foram divididos em cinco classes. A Tabela 01 traz essa divisão:

Tabela 01 – Divisão do IP em classes

Endereço de Rede	Classe	RFC
0	A	1918
10	B	3330
110	C	3330
1110	*Multicast*	3171
1111	Reservado	1700

Existem três faixas de endereços reservados para as redes privadas. Seguem abaixo os endereços que também são conhecidos como IP's não válidos:

- Classe A: de 10.0.0.0 a 10.255.255.255;
- Classe B: de 172.16.0.0 a 172.31.0.0;
- Classe C: de 192.168.0.0 a 192.168.255.255;

2.4.3 Sub-redes

As sub-redes são redes sem classe e podem ser criadas baseadas na informação do endereço de *broadcast* diferentes. Estes endereços são delimitados pelas máscaras de difusão de rede ou máscaras de rede. A máscara de rede define quantos bits são utilizados para o endereço de rede e quantos bits são utilizados para especificar o endereço de *hosts* dentro dessa sub-rede.

2.5 Tipos de Ataques

Embora existam diversos tipos de ataques com características conhecidas, o tipo de ataque mais difícil de conter é o DoS *(Denied of Service)* ou DDoS *(Distributed Denied of Service)*, comumente conhecido como negação de serviço. Sua maior eficiência encontra-se na inundação excessiva de informações ou pacotes direcionados a uma porta de serviço Internet que esteja aberta para o mundo, como o serviço http ou https. Mesmo com um ambiente protegido por *Firewall*, o atacante pode direcionar o tráfego para a porta 80 ou 443, podendo comprometendo a

banda disponível e, principalmente, os recursos de hardware do *Firewall* e do servidor WEB.

Os ataques aos sistemas podem ser executados explorando deficiências na concepção, implementação, configuração ou gerenciamento dos serviços e sistemas. A seguir será demonstrado o funcionamento de um ataque conhecido como SYN FLOODING:

2.5.1 SYN-FLOODING

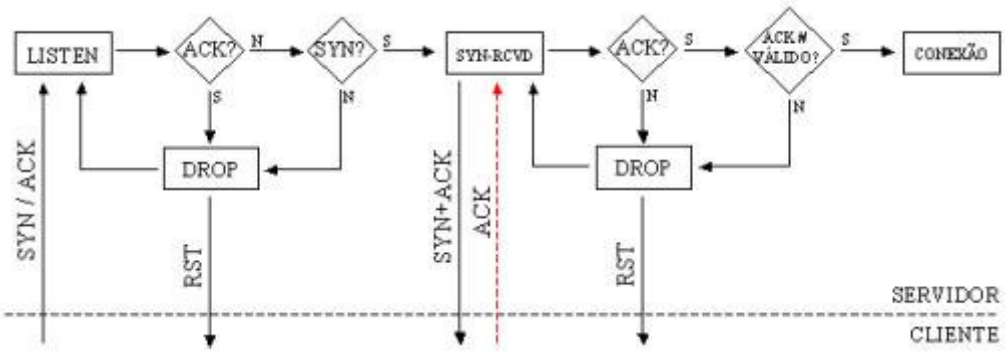

Figura 02. Conexão TCP *3-WAY-HANDSHAKE*

Segundo SANTOS(2006) uma sessão TCP é definida pela sua máquina de estado conforme a Figura 02. Considere um servidor no estado LISTEN em uma determinada porta TCP. Quando um cliente faz uma requisição com um SYN o servidor após confirmar se a flag realmente é SYN, passa para o estado SYN-RCVD e envia um outro segmento com um ACK+SYN de confirmação, e espera determinado tempo para receber um outro segmento de ACK do cliente.

Caso um servidor não receba o último ACK do cliente, ele permanecerá no estado SYN-RCVD por algum tempo, enchendo assim a tabela TCB e preenchendo os recursos do servidor. Quando o segmento ACK do cliente não chega, esta conexão então, é chamada de *half-connection*. Milhares de *half-connection* caracterizam então o SYN flooding.

Existem diversas formas de proteger de um ataque *Syn-Flood*, sendo que uma técnica denominada *Syn-cookies* é extensamente recomendada por diversos especialistas. Para habilitar a proteção através do *Syn-cookies* no linux é necessário editar o arquivo *"/proc/sys/net/ipv4/tcp_syncookies"* e inserir o número "1".

2.6 Tipos de Proteção

Uma organização pode se proteger utilizando diferentes técnicas e mecanismos. O primeiro passo é a definição de uma política de segurança, base para todas as questões relacionadas à proteção da informação. Outro mecanismo importante é a definição de um plano de contingência, prevendo atividades que deverão ser realizadas em momentos críticos, como por exemplo, quando um ataque for detectado. Uma técnica muito utilizada para garantir a proteção das informações é a criptografia, que torna os dados e mensagens não interpretáveis enquanto estão sendo enviados. A criação de um controle de acesso, garantindo que o acesso a um recurso será limitado somente para usuários autorizados também é de fundamental importância para a proteção das informações da organização. Há ainda outro mecanismo de proteção muito utilizado pelas organizações, que será o foco deste trabalho e será tratado mais adiante: o *Firewall*.

2.7 *Firewall*: Classificação

Firewall é um ponto entre duas ou mais redes, que pode ser um componente ou um conjunto de componentes, por onde passa todo o tráfego, permitindo que o controle, a autenticação e os registros de todo o tráfego sejam realizados. Um *Firewall* pode ser formado por um conjunto de software, hardware e política de segurança. Ele detém autonomia concedida pelo próprio sistema para pré-determinar e disciplinar todo o tipo de tráfego existente entre o mesmo e outros *host*/redes.

Em meados de 80, sob encomenda da AT&T, o primeiro *Firewall* do mundo foi desenvolvido com o intuito de “filtrar” todos os pacotes que saíssem e entrassem na rede corporativa, de modo a manipulá-los de acordo com as especificações das regras previamente definidas. Mesmo diante da evolução dos meios tecnológicos, hoje um *Firewall* continua a possuir e empregar os mesmos conceitos, apenas com alguns aprimoramentos e implementações de novas funcionalidades.

2.7.1 Tipos de *Firewall*

Os *Firewalls* podem ser classificados da seguinte forma:

- *Firewall* de filtro de pacotes: é o tipo mais simples em sua arquitetura. Baseia-se apenas em campos básicos do cabeçalho dos protocolos. É um tipo rápido, adequado para grandes volumes de dados. Não faz uma filtragem muito eficiente, pois não armazena uma tabela de estados.
- *Firewall* de inspeção com estado: baseia-se em regras um pouco mais complexas que o *Firewall* de filtro de pacotes, pois pode tratar todo e qualquer campo do cabeçalho dos protocolos. Mantém uma tabela de estados podendo controlar pacotes nos dois sentidos. É mais complexo e, portanto, mais lento.
- *Firewall* de aplicação: não permite que nenhum pacote passe diretamente entre as redes a ele conectadas. Todos os pacotes são enviados a um processo de proxy que determina quando se deve ou não estabelecer a conexão, e fica, por todo o tempo, intermediando a comunicação. Devido às limitações de flexibilidade e de performance, raramente é utilizado sem a composição com os outros tipos de *Firewall*. É mais lento que os tipos descritos anteriormente.
- *Firewall* pessoal: controla e registra as portas ou aplicativos habilitados a entrarem ou saírem de um sistema local. É muito simples e sempre em forma de um software a ser instalado no sistema operacional.

2.8 O *Firewall* IPTABLES

Segundo NETO (2004), o *Firewall* IPTABLES do linux é do tipo inspeção com estado. No Linux, as funções de *Firewall* são agregadas à própria arquitetura do *Kernel*, que é o núcleo do sistema operacional. Enquanto a maioria dos “produtos” *Firewall* pode ser definida como um subsistema, o Linux possui a capacidade de transformar o *Firewall* no próprio. O módulo do *Kernel* no Linux responsável por realizar a função de um *Firewall* é chamado IPTABLES.

O IPTABLES começou a ser desenvolvido em meados de 1999 e faz parte do *kernel* 2.4 do Linux. Ele funciona com base no endereço/porta de origem/destino do pacote, prioridade, e outras informações contidas no cabeçalho IP.

O *Firewall* IPTABLES é um sistema que tenta resolver os problemas ocasionados pela complexidade da criação e implementação das regras de filtragem.

Em sistemas que utilizam o IPTABLES, o *kernel* é inicializado com três listas de regras-padrão, que são também chamadas de *chains* ou cadeias. Estas são:

- *INPUT*: define os pacotes de entrada na rede;
- *OUTPUT*: define os pacotes de saída da rede;
- *FORWARD*: define os pacotes a serem encaminhados.

O IPTABLES possui tabelas que armazenam as cadeias. São elas:

- tabela *filter*: filtragem padrão contendo as três *chains* (*Input, Output e Forward*);
- tabela nat: usada para tradução de endereços possuindo outras *chains* como *Prerouting, Output e Postrouting*;
- tabela *mangle*: usada para alterações especiais como, por exemplo, modificar algum tipo de serviço.

2.8.1 O funcionamento de um *Firewall* IPTABLES

Segundo NAKAMURA e GEUS (2003), cada cadeia (I*NPUT, OUTPUT e FORWARD*) possui seu próprio conjunto de regras de filtragem. Quando o pacote atinge uma das cadeias é examinado pelas regras dessa cadeia. Se a cadeia tiver uma regra que define que o pacote deve ser descartado, ele será descartado nesse ponto. Caso a cadeia tenha uma regra que aceite o pacote, ele continua percorrendo o caminho até chegar à próxima cadeia ou ao destino final. A figura 03, a seguir, ilustra o funcionamento da tabela *filter* do IPTABLES:

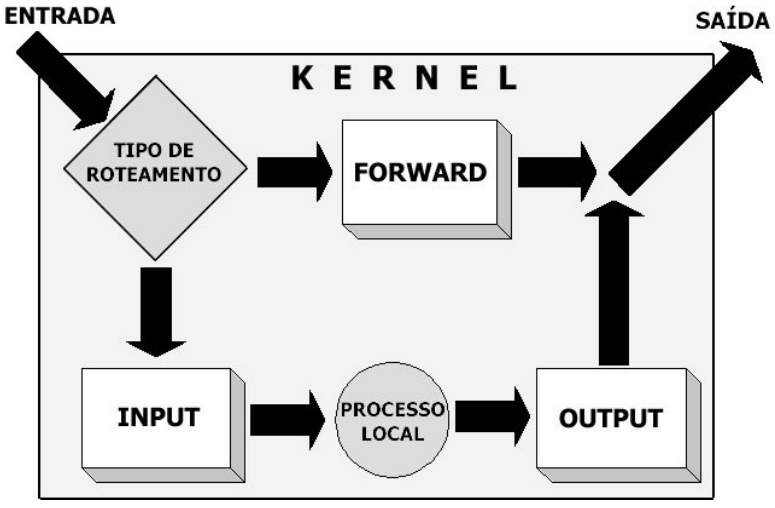

Figura 03. Tabela filter do IPTABLES

O modo de funcionamento do IPTABLES pode ser então resumido da seguinte maneira:

- Quando um pacote é recebido pela placa de rede, o *kernel* primeiramente verifica qual é o seu destino;
- Se o destino for o próprio equipamento, o pacote é passado para a cadeira *INPUT*. Se ele passar pelas regras dessa cadeia, ele será repassado para o processo de destino local, que está esperando pelo pacote.
- Se o *kernel* não tiver o *forwarding* habilitado ou se não souber como encaminhar esse pacote, este será descartado. Se o *forwarding* estiver habilitado para outra interface de rede, o pacote irá para a cadeira FORWARD. Se o pacote passar pelas regras dessa cadeia, ele será aceito e repassado adiante. Normalmente, essa é a cadeia utilizada quando o Linux funciona como um *Firewall*.
- Um programa sendo executado no equipamento pode enviar pacotes à rede, que são enviados à cadeia *OUTPUT*. Se esses pacotes forem aceitos pelas regras existentes nessa cadeia, serão enviados por meio da interface.

2.8.2 Principais comandos do IPTABLES

Existem alguns comandos que são empregados para a sua construção das regras de um *firewal* IPTABLES. As regras são formadas de políticas, comandos e parâmetros.

Políticas IPTABLES:

- *accept*: aceita o pacote recebido;
- *drop*: nega pacote e não o retorna de volta;
- *reject*: nega o pacote e retorna um aviso de erro ao emissor;

Comandos utilizados pelo IPTABLES:

- -A : adiciona ou atualiza uma regra no fim;
- -I : apenas adiciona uma nova regra no início;
- -D : exclui uma regra específica;
- -P : define a regra padrão;
- -L : lista todas as regras armazenadas;

- -F : exclui todas as regras armazenadas;
- -R : substitui uma regra armazenada;
- -C : efetua uma checagem das regras básicas;
- -N : cria uma regra com nome específico;
- -X : exclui uma regra com nome específico.

Parâmetros padrões:

- -p : define qual o protocolo deve ser tratado (TCP, UDP e ICMP);
- -s ou -d : define o endereço de origem ou de destino em que a regra irá atuar;
- –i : define a interface de rede por onde os pacotes são recebidos e enviados;
- -j : define a direção de uma ação baseada em regras similares.

Exemplo de regra:

```
# aceita conexão SSH (22/tcp) a partir do IP do Administrador remoto.
  $IPTABLES -A INPUT -p tcp -s 192.168.1.6 --dport 22 -j ACCEPT
```

2.8.3 A configuração de um *Firewall* IPTABLES

A complexidade das regras de filtragem cresce cada vez mais na medida em que serviços e aplicações são adicionados no ambiente corporativo. Dessa forma, o gerenciamento se torna um fator importante para que erros na criação e implementação de regras sejam minimizados. Além do fator complexidade, existe ainda o fator desempenho, que também é prejudicado quando há um grande número de regras.

A configuração de um *Firewall* IPTABLES é realizada por uma série de comandos (regras) que são interpretados pelo *Kernel* do sistema operacional. Tais comandos podem ser executados via scripts (arquivos-texto) ou serem inseridos diretamente no shell (núcleo do sistema).

O grande problema é que a interface utilizada pela maioria dos administradores torna difícil a compreensão das regras e exige treinamento rigoroso dos usuários para interagir com ela. A inserção e manutenção das regras de filtragem em linhas de comando dificultam o trabalho do administrador da rede, ocasionando falhas que podem aumentar a vulnerabilidade do sistema e ameaçar a segurança da rede e da organização.

Neste trabalho, espera-se resolver o problema descrito, desenvolvendo uma ferramenta com interface gráfica que permita ao administrador da rede manipular as regras de maneira amigável. A utilização de uma interface gráfica para criação e gerenciamento das regras permitirá uma melhor interação entre o administrador e a ferramenta de segurança IPTABLES, facilitando a gerência do *Firewall e* minimizando os riscos de falhas de segurança.

3 ESPECIFICAÇÃO DO SISTEMA

Este capítulo apresenta a especificação do sistema a ser construído para resolver os problemas descritos anteriormente, relacionados à configuração do *Firewall* IPTABLES.

3.1 Visão Geral do Sistema

O FWC – *Firewall Constructor* tem como objetivo facilitar a configuração dos *Firewalls* IPTABLES do sistema operacional Linux. O sistema permitirá que seus usuários gerenciem vários servidores, cadastrando ou escolhendo as regras para configuração do *Firewall* em um ambiente gráfico e amigável.

3.2 Descrição dos Usuários

O FWC tem como foco os usuários que atuam como administradores de redes em empresas que se preocupam com a segurança das informações que trafegam na rede. Para utilizá-lo é necessário que o usuário tenha conhecimento em redes de computadores, protocolos de comunicação, administração de servidores Linux e conceitos de segurança da informação.

3.3 Definição de Escopo

As seções a seguir definem o escopo do FWC, descrevendo seus requisitos funcionais, não funcionais e os não requisitos, deixando bem claro o que não fará parte do escopo desta primeira versão do programa.

3.3.1 Requisitos Funcionais

[RF001]. O sistema deve permitir cadastro e manutenção de usuários.

[RF002]. O sistema deve possuir um controle de acesso, exigindo *login* e senha dos usuários.

[RF003]. O sistema deve permitir cadastro e manutenção de interfaces e portas correspondentes.

[RF004]. O sistema deve possibilitar o gerenciamento de configuração sistema.

[RF005]. O sistema deve permitir o cadastro e manutenção de regras IPTABLES.

[RF006]. O sistema deve permitir que as regras criadas sejam exportadas em arquivos texto para serem aplicadas no servidor Linux correspondente.

[RF007]. O sistema deve permitir que as regras sejam aplicadas *on-line* através de uma conexão SSH com o servidor.

3.3.2 Modelo de Casos de Uso

Os diagramas a seguir retratam as funcionalidades do sistema FWC, que foram organizadas em quatro pacotes: gerenciamento de usuários, gerenciamento de configuração, gerenciamento de interface e gerenciamento de regras.

Pacote 01: Gerenciamento de Usuários

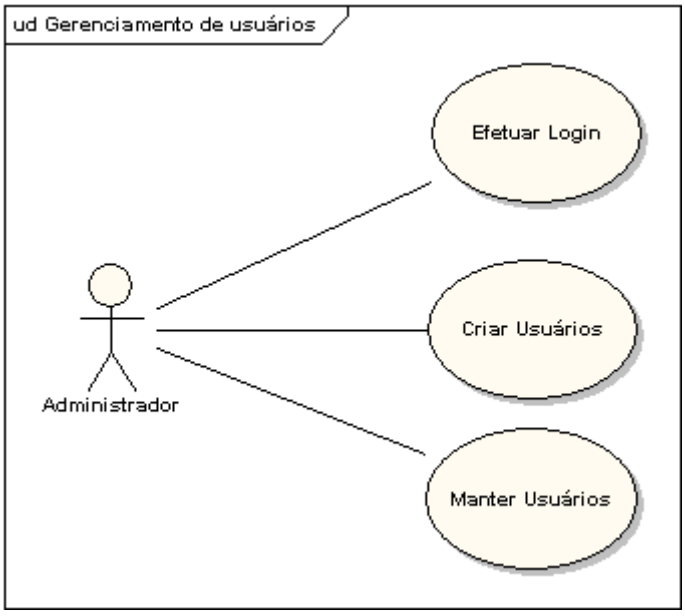

Figura 04. Diagrama de caso de uso Usuários.

Caso de Uso 01: Criar Usuários

Objetivo: Criar um novo usuário de acesso ao sistema

Ator: Administrador

Prioridade: Essencial

Interface associada:

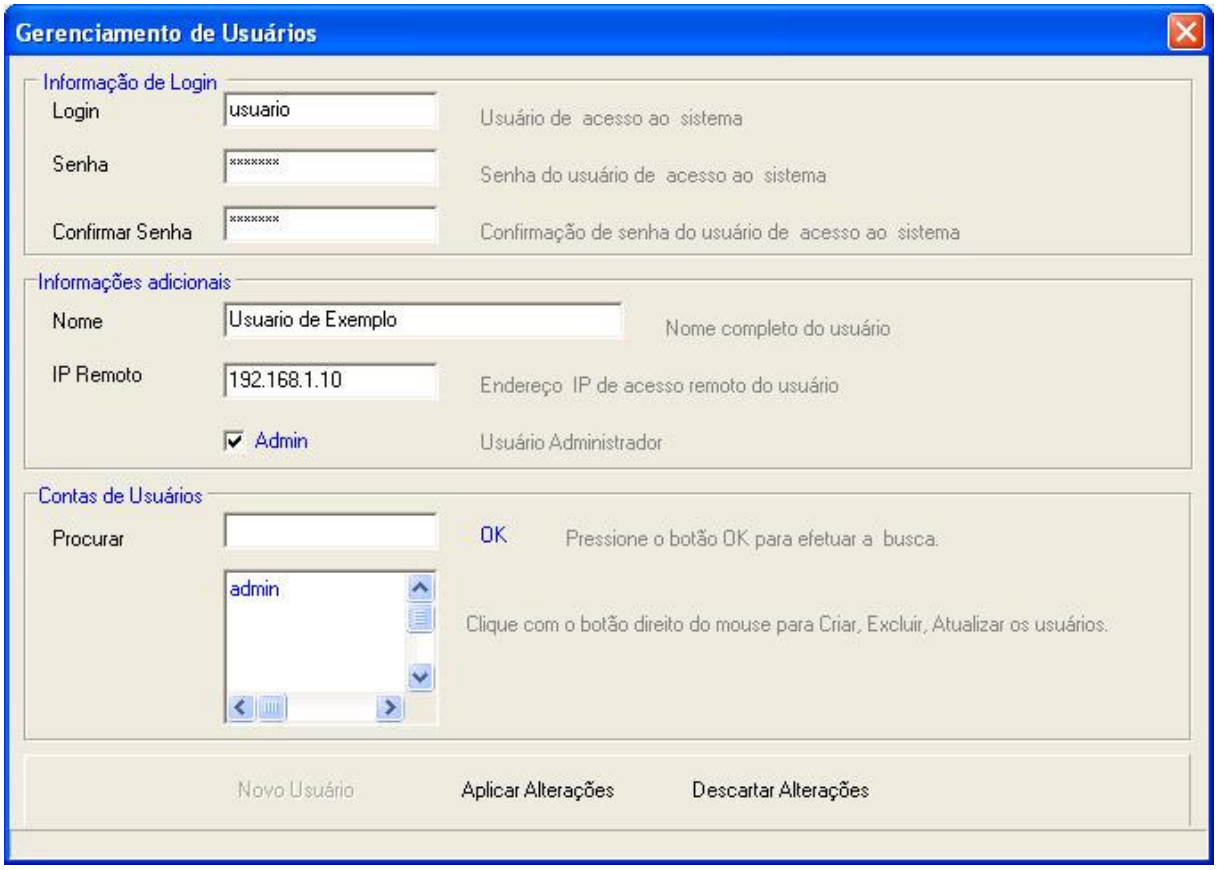

Figura 05. Interface de Gerenciamento de Usuários.

Pré-condições: Usuário com perfil de administrador.

Resultados: Usuário cadastrado no sistema.

Fluxo de eventos Principal:

Criar Usuários

1) Administrador solicita criação de novo Usuário
2) Administrador preenche informações solicitadas
3) Administrador solicita gravação dos dados
4) Sistema valida dados
5) Sistema grava informações

Fluxos Secundários:

Usuário existente

4.1) Sistema exibe mensagem de erro “Usuário já cadastrado”

4.2) Voltar para o passo 2 do fluxo principal

Caso de Uso 02: Efetuar login

Objetivo: Logar no sistema

Ator: Usuário

Prioridade: Essencial

Interface associada:

Figura 06. Interface de Login de Usuários.

Pré-condições: Usuário cadastrado.

Resultados: Usuário Logado no sistema.

Fluxo de eventos Principal:

Efetuar Login

1) Usuário solicita entrada no sistema

2) Usuário preenche informações solicitadas

3) Sistema valida dados

4) Sistema permite entrada no sistema

Fluxos Secundários:

Usuário existente

3.1) Sistema exibe mensagem de erro “Credenciais inválidas”

3.2) Voltar para o passo 2 do fluxo principal

Senha incorreta

3.3) Sistema exibe mensagem de erro “Credenciais inválidas”

3.4) Voltar para o passo 2 do fluxo principal

Pacote 02: Gerenciamento de interfaces

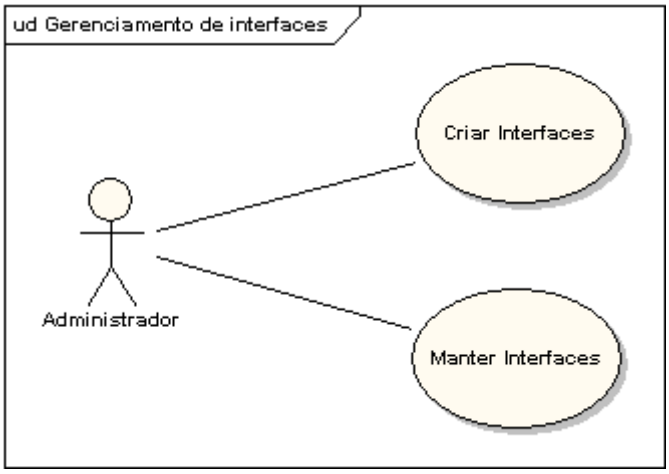

Figura 07. Diagrama de caso de uso de Interfaces.

Caso de Uso 03: Criar Interfaces

Objetivo: Cadastrar informações das interfaces.

Ator: Administrador

Prioridade: Essencial

Interface associada:

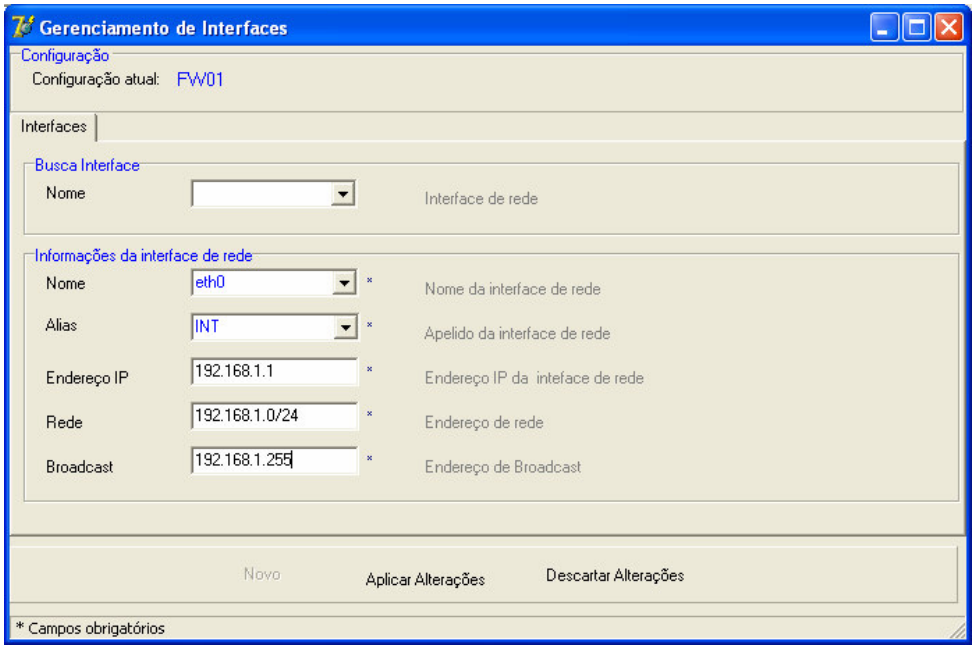

Figura 08. Interface de Gerenciamento de Interfaces.

Pré-condições: Usuário cadastrado, Configuração do sistema cadastrada.

Resultados: Interface cadastrada e associada a uma configuração de um usuário.

Fluxo de eventos Principal:

Criar Interface

1) Administrador solicita criação de nova Interface
2) Administrador preenche informações solicitadas
3) Administrador solicita gravação dos dados
4) Sistema valida dados
5) Sistema grava informações

Fluxos Secundários:

Servidor existente

5.1) Sistema exibe mensagem de erro “Interface já cadastrada”
5.2) Voltar para o passo 3 do fluxo principal

Alias existente

5.3) Sistema exibe mensagem de erro “Alias já cadastrada”
5.4) Voltar para o passo 3 do fluxo principal

Pacote 03: Gerenciamento de Configurações

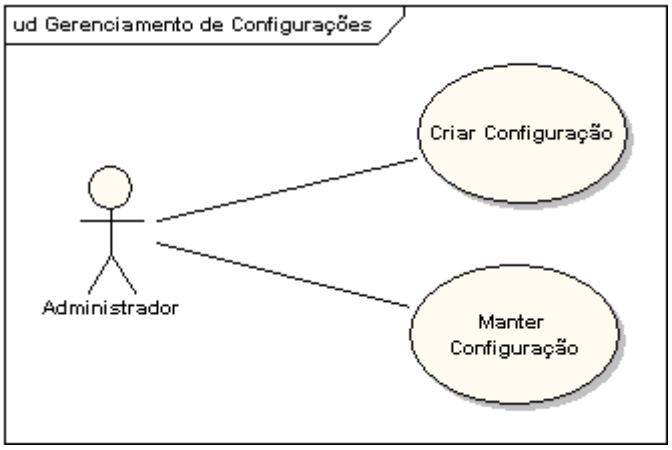

Figura 09. Diagrama de caso de uso de Configuração.

Caso de Uso 04: Criar Configuração

Objetivo: Criar uma configuração do sistema.

Ator: Administrador

Prioridade: Essencial

Interface associada:

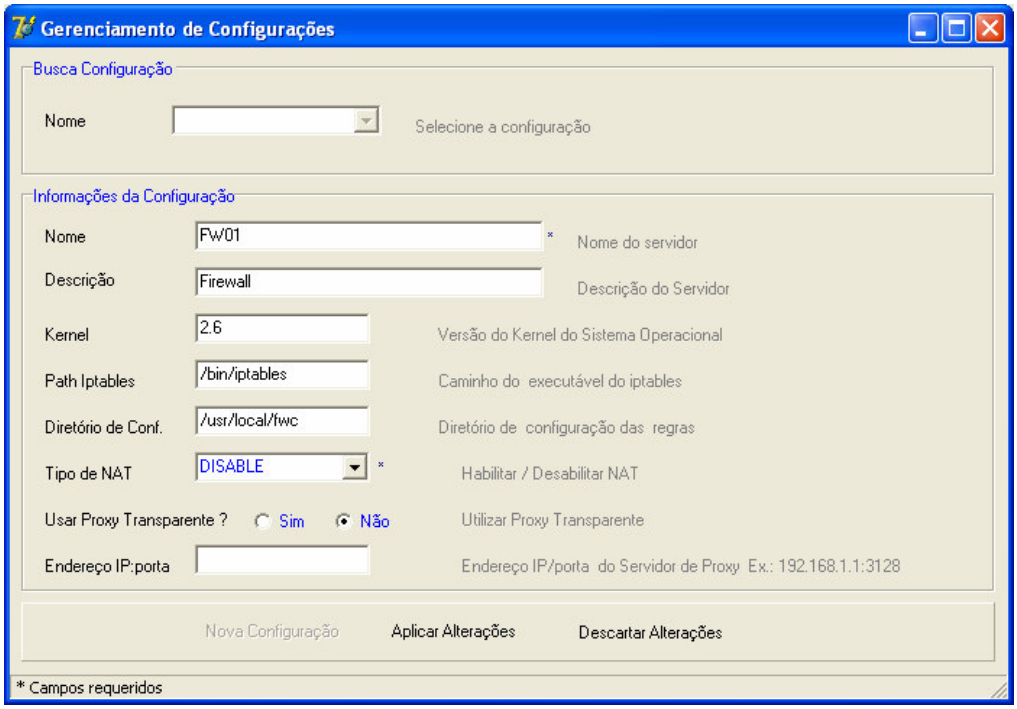

Figura 10. Interface de Gerenciamento de Configurações.

Pré-condições: Usuário cadastrado.

Resultados: Configuração do sistema cadastrada.

Fluxo de eventos Principal:

Criar Configuração

1) Administrador solicita criação de nova Configuração
2) Administrador preenche informações solicitadas
3) Administrador solicita gravação dos dados
4) Sistema valida dados
5) Sistema grava informações

Fluxos Secundários:

Configuração existente

4.1) Sistema exibe mensagem de erro “Configuração já cadastrada”
4.2) Voltar para o passo 1 do fluxo principal

Pacote 04: Gerenciamento de Regras

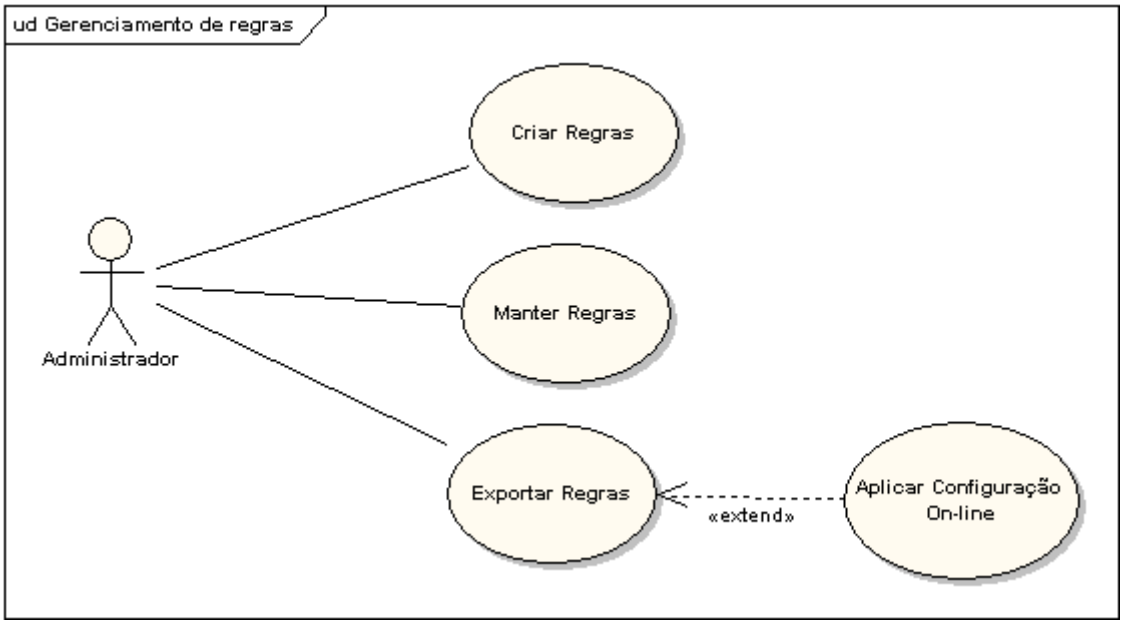

Figura 11. Diagrama de caso de uso de Regras.

Caso de Uso 05: Criar Regras

Objetivo: Cadastrar regras.

Ator: Administrador

Prioridade: Essencial

Interface associada:

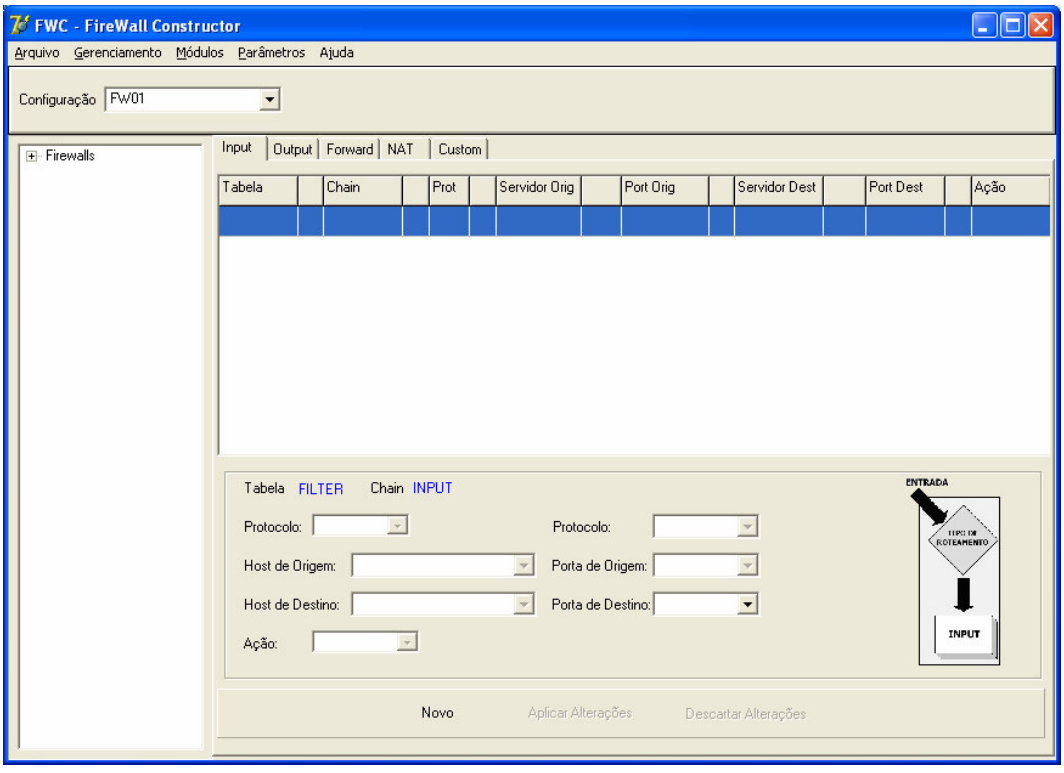

Figura 12. Interface de Gerenciamento de Regras.

Pré-condições: Usuário cadastrado, Interfaces configuradas e configuração selecionada.

Resultados: Regra cadastrada em uma configuração.

Fluxo de eventos Principal:

Criar Regras

1) Administrador seleciona uma configuração
2) Administrador seleciona uma chain
3) Administrador solicita criação de nova regra
4) Administrador preenche informações solicitadas
5) Administrador solicita gravação dos dados
6) Sistema valida dados
7) Sistema grava informações

Fluxos Secundários:

Regras Existentes

6.1) Exibir mensagem: "a regras já existem"
6.2) Volta para o passo 4 do fluxo principal

Caso de Uso 06: Exportar Regras

Objetivo: Permitir que o usuário exporte as regras para um arquivo texto.

Ator: Usuário

Prioridade: Importante

Pré-condições: Configuração selecionada

Resultados: Arquivo de regras

Fluxo de eventos Principal:

Exportar regras

1) Administrador seleciona uma configuração
2) Administrador solicita geração de arquivo
3) Administrador preenche informações solicitadas
4) Sistema valida configuração selecionada
5) Sistema gera arquivo de regras

Fluxos Secundários:

Configuração não selecionada

4.1) Exibir mensagem: "É necessário selecionar uma configuração"

4.2) Volta para o passo 1 do fluxo principal

Regras inexistentes

4.3) Exibir mensagem: "Regras inválidas"

4.4) Volta para o passo 1 do fluxo principal

Erro ao gerar o arquivo

4.5) Exibir mensagem: "Erro ao gerar o arquivo"

4.6) Volta para o passo 3 do fluxo principal

Caso de Uso 07: Aplicar Configuração On-line

Objetivo: Permite que o usuário aplique as configurações on-line em um servidor.

Ator: Usuário

Prioridade: Desejável

Pré-condições: Configuração selecionada, arquivo de regras gerado.

Resultados: Configuração aplicada no servidor

Fluxo de eventos Principal:

Aplicar configuração

1) Executar caso de uso “Exportar Regras”

2) Administrador preenche informações solicitadas

3) Sistema valida informações

4) Sistema aplica regras no servidor

Fluxos Secundários:

Informações incorretas

3.1) Exibir mensagem: "informações incorretas”

3.2) Volta para o passo 2 do fluxo principal

Servidor não responde

3.3) Exibir mensagem: "Servidor não responde"

3.4) Volta para o passo 2 do fluxo principal

3.3.3 Requisitos Não Funcionais

Usabilidade:

[NF001]. O sistema deve apresentar uma interface gráfica amigável ao usuário.

Segurança:

[NF002]. O sistema deve permitir o acesso apenas de usuários cadastrados.

[NF003]. O sistema deverá possuir uma conexão SSH com o servidor que está sendo configurado.

Hardware e Software:

[NF004]. O sistema deve utilizar o banco de dados MySQL versão 5.0.

[NF005]. O sistema deve ser construído na tecnologia Delphi.

[NF006]. O sistema deve ser construído para funcionar no sistema operacional Windows 98 ou superior.

[NF007]. O sistema deve funcionar para ambientes com hardware superior a Pentium III, 500MHz e memória de 256MB.

3.3.4 Não Requisitos

[NR001]. O sistema não permitirá o gerenciamento de *Firewalls* com versões de IPTABLES inferiores a 2.4 ou outros tipos de *Firewalls*.

[NR002]. O sistema não será construído para funcionar no sistema operacional Linux.

3.4 Modelagem de Dados

A Figura 13 retrata o relacionamento das entidades da base de dados do FWC. Os scripts de criação da base de dados do FWC estão no Anexo B.

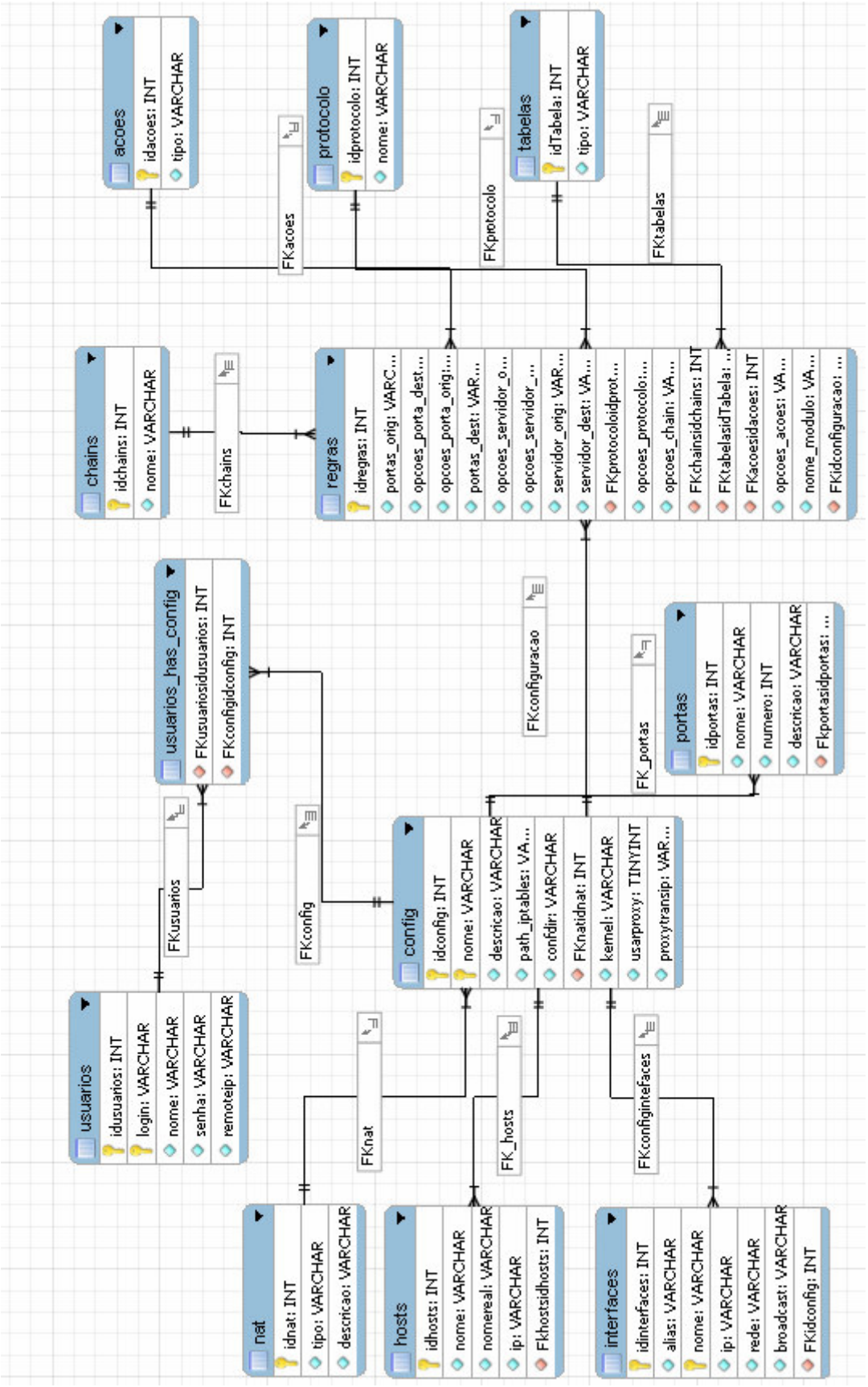

Figura 13. Diagrama de Entidade-Relacionamento.

4 Fluxo de Utilização da Ferramenta

Para utilizar o FWC é necessário seguir alguns passos, que podem ser vistos na Figura 14.

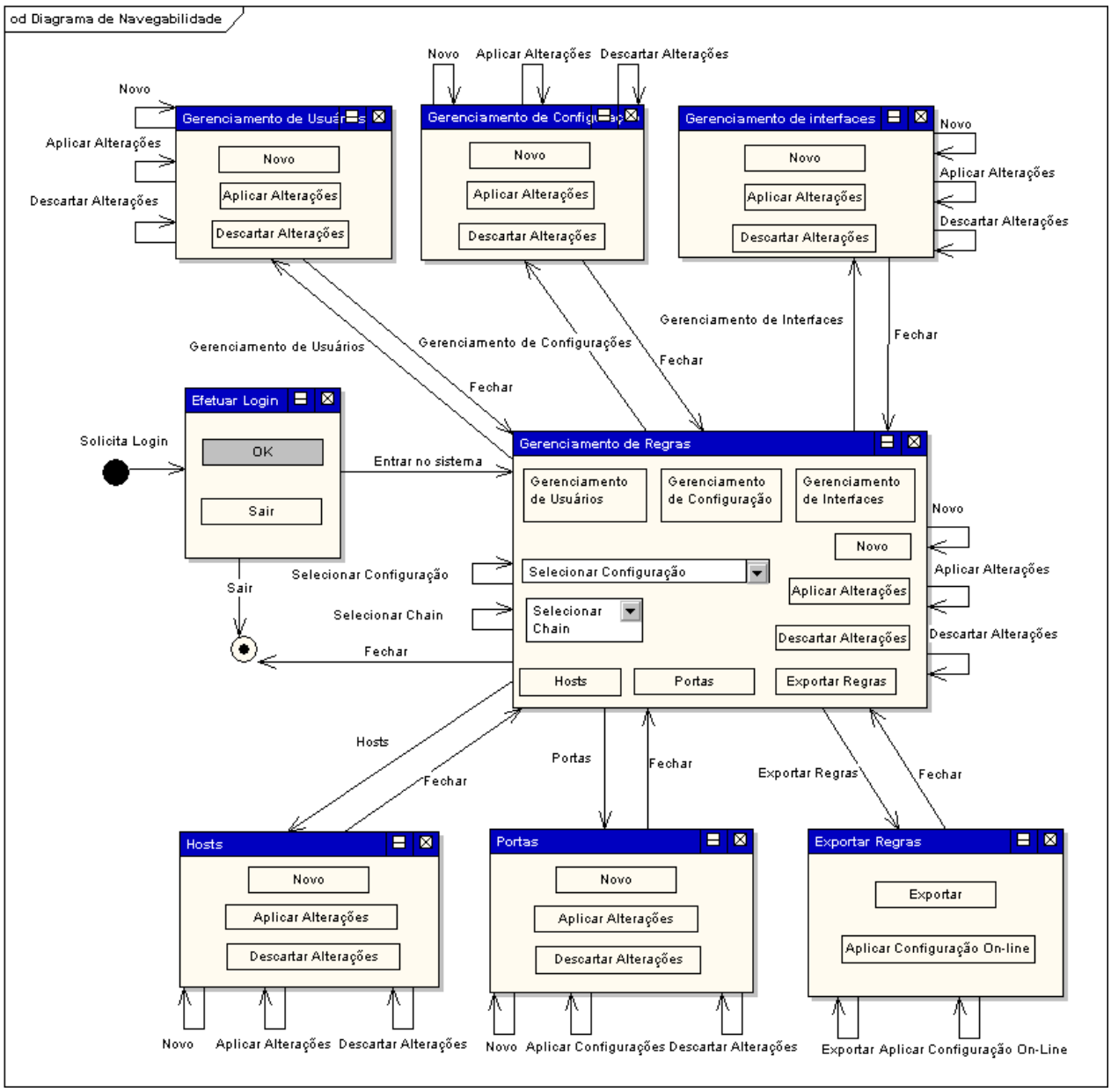

Figura 14. Modelo de navegabilidade

Antes da utilização da ferramenta pelos usuários, é necessário que o administrador cadastre e configure algumas informações no sistema (usuários, protocolos, portas e *hosts*). Para cadastro de usuários, o administrador deve informar: *Login*, Senha, Nome Completo e IP Remoto. As opções de alterar e excluir usuários

estão disponíveis na mesma tela do aplicativo. Entretanto, não é permitida a exclusão de um usuário que tenha alguma configuração associada.

Após os cadastros básicos realizados pelo administrador, pode-se iniciar o fluxo de atividades do usuário FWC. O objetivo principal é a criação de configuração e a aplicação de regras. As atividades iniciam com o *login* no aplicativo. Em seguida, é necessário criar uma configuração para o usuário. Nesta configuração, deve ser informado se a configuração suporta NAT (*Network Adress Translation*).

O passo seguinte é a configuração da interface de rede. Para tal, o usuário deve informar:

- Nome da interface de rede;
- *Alias* (INT - interna, EXT - Externa, DMZ – Zona Desmilitarizada ou VPN – Rede Privada);
- Endereço IP;
- Rede;
- e *Broadcast* da rede.

Em seguida, o usuário deverá selecionar a configuração criada e iniciar a elaboração das regras. Para cadastro de regras, estão disponíveis:

a. Escolha do tipo de *chain* (*INPUT, OUTPUT, FOWARD* – da tabela *Filter*, e NAT, da tabela NAT);
b. Escolha do protocolo;
c. Escolha do *Host* e porta de origem;
d. Escolha do *Host* e porta destino;
e. Escolha da ação;
f. Escolha da interface.

O usuário pode criar quantas regras forem necessárias. Ao final, para que as regras sejam aplicadas, após selecionar a configuração correspondente, pode solicitar a exportação das regras para arquivo ou a aplicação diretamente no servidor.

4.1 Estudo de Caso do FWC

Será demonstrado um exemplo de *Firewall* que servirá para filtrar o tráfego de *hosts* de uma rede interna com a internet. Serão exibidos os passos para geração de regras através do FWC e aplicação de configuração On-line no servidor de *Firewall*. A Figura 15 retrata uma topologia típica que será abordada no estudo de caso.

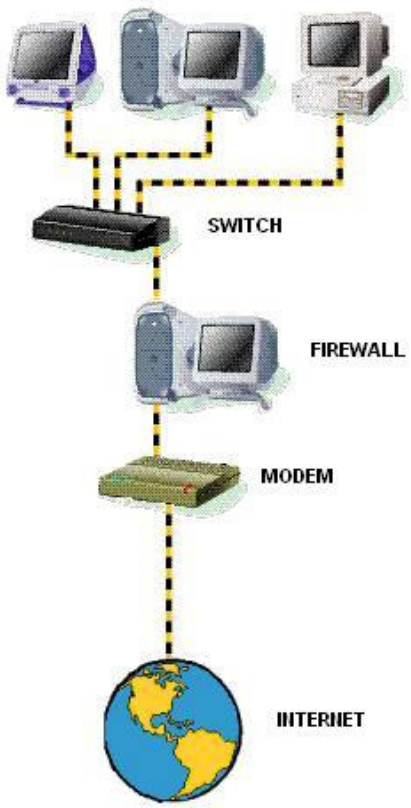

Figura 15. Topologia da rede exemplo

4.1.1 Configuração

- Máquina do *Firewall* com 2 interfaces de rede, uma é eth0 com o IP 192.168.1.1 que faz parte da rede interna e a outra interface é eth1 com o IP xxx.xxx.xxx.249 que faz parte da rede da internet.
- Qualquer acesso externo a máquinas da rede interna é bloqueado.
- Usuários da rede local têm acesso ao servidor *Firewall*.
- Qualquer acesso à máquina do *Firewall* é bloqueado, exceto conexões para acesso a porta 80.

- Todos os usuários têm acesso livre à internet.

4.1.2 Passos para criação das regras.

A seguir serão demonstrados os passos para criação das regras através da ferramenta FWC:

4.1.3 *Login* do administrador

A Figura 16 demonstra a tela de *login* do FWC. Será feito o *login* com um usuário administrador para criação de outra conta de usuário sem perfil de administrador.

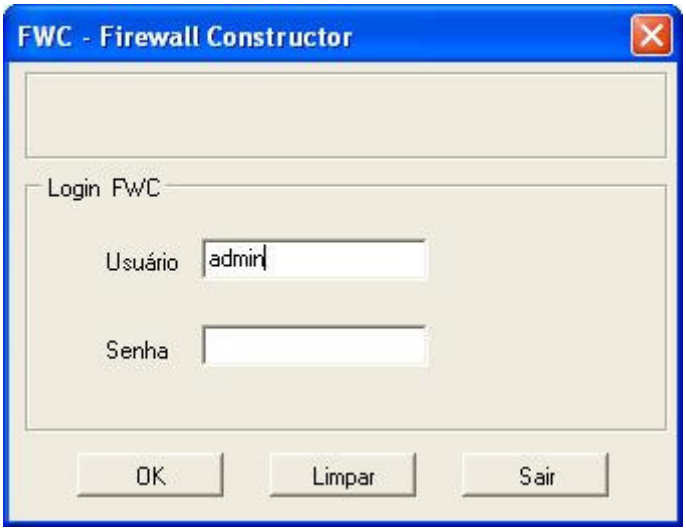

Figura 16. *Login* do FWC - Estudo de Caso.

4.1.4 Criação de usuário

A Figura 17 demonstra a criação do um novo usuário, que posteriormente será utilizado para o gerenciamento de regras. O administrador irá cadastrar os dados do novo usuário que serão solicitados.

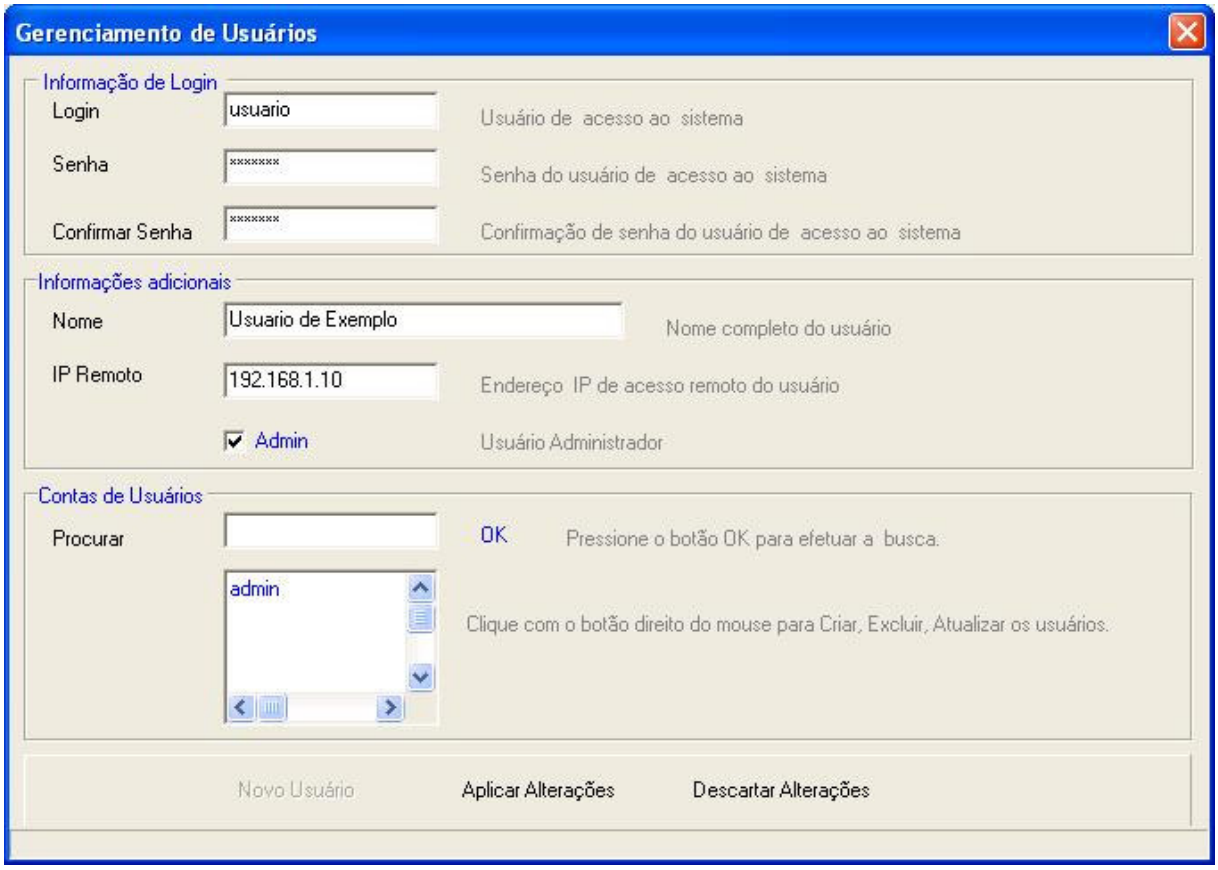

Figura 17. Gerenciamento de Usuários - Estudo de Caso.

4.1.5 *Login* do novo usuário

A Figura 18 exibe a tela de *login*, que será utilizado pelo usuário criado anteriormente para a criação das regras.

Figura 18. *Login* do FWC - Estudo de Caso.

4.1.6 Criação de Configuração

A Figura 19 demonstra a criação uma configuração para o novo usuário. Essas configurações são referentes ao servidor de *Firewall*.

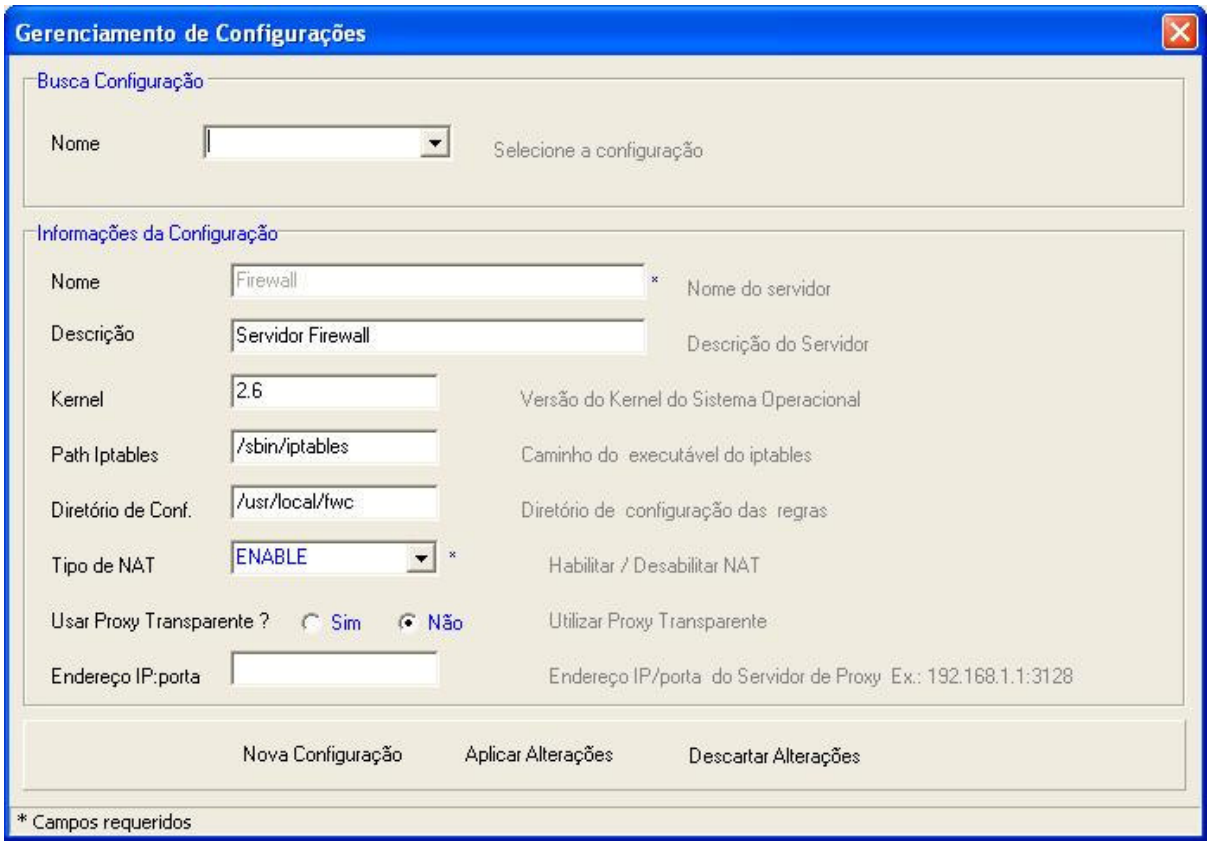

Figura 19. Gerenciamento de Configurações - Estudo de Caso.

4.1.7 Criação de Interfaces

As figuras 19 e 20 demonstram a criação de interfaces de rede do servidor *Firewall* vinculadas à configuração "*Firewall*". Essas interfaces serão configuradas de acordo com a topologia exemplificada anteriormente.

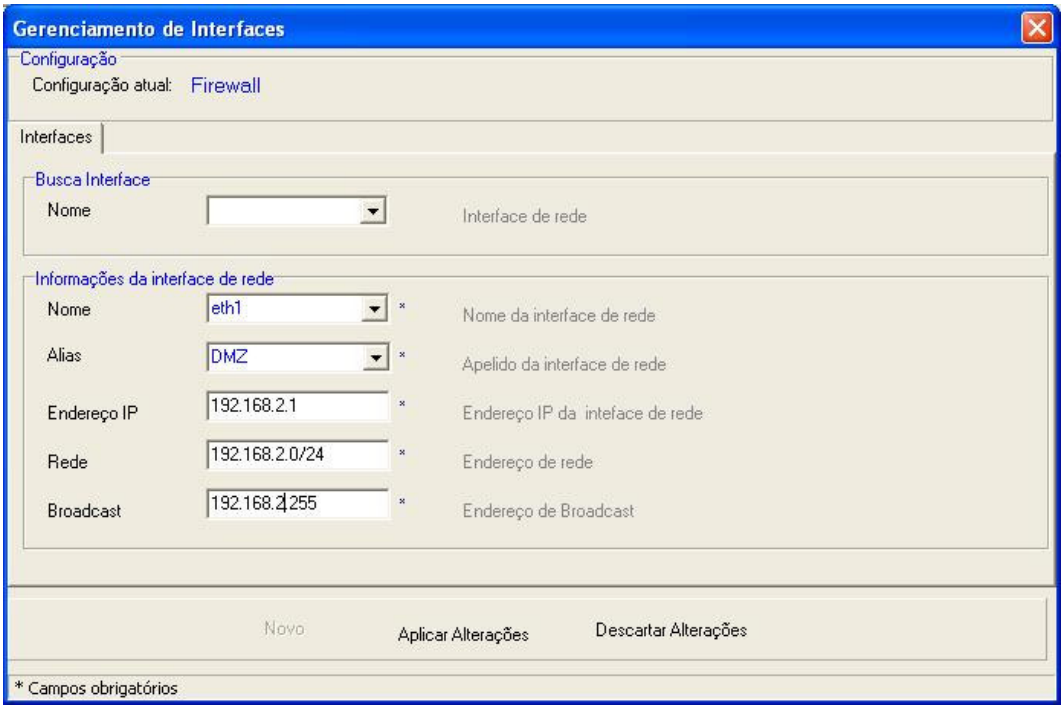

Figura 19. Gerenciamento de Interfaces - Rede Interna.

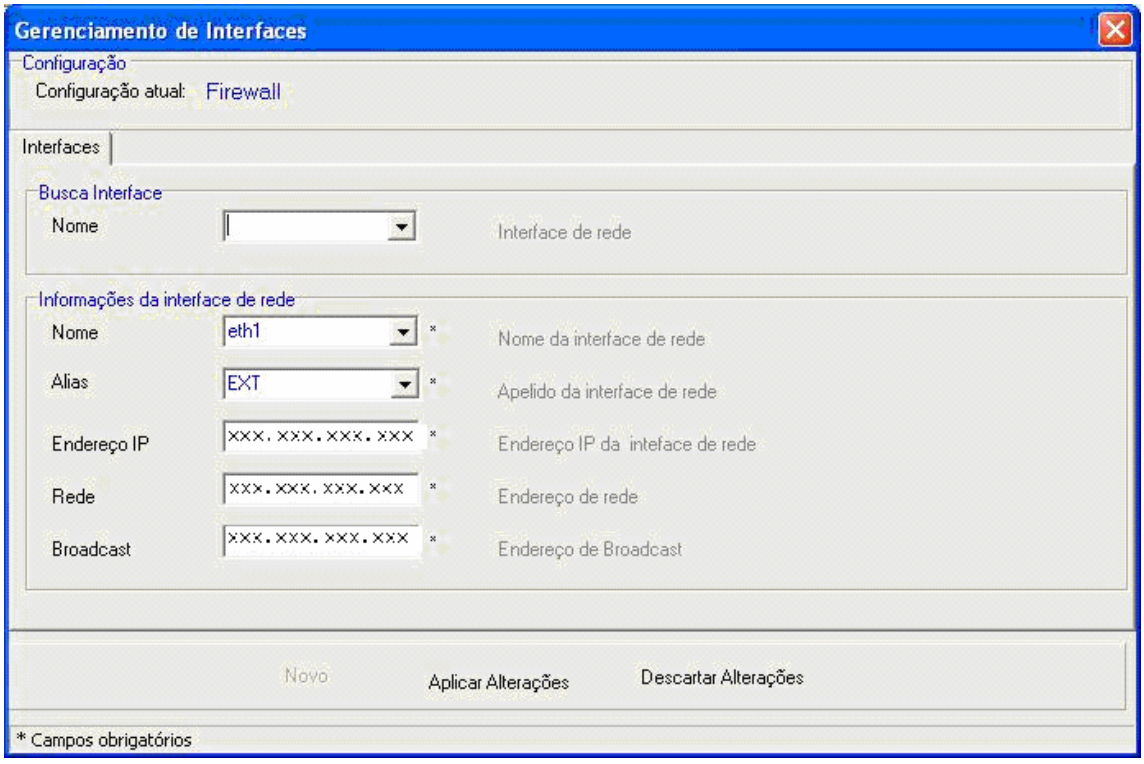

Figura 20. Gerenciamento de Interfaces – Rede Externa.

4.1.8 Criação de regras

As figuras a seguir demonstram a criação de regras de *Firewall*:

4.1.8.1 Regras de *INPUT*

As regras a seguir são utilizadas para permitir conexões destinadas as interfaces lo – *loopback*, eth0 – rede interna, e também permitindo conexões destinadas a interface eth1- rede externa somente conexões para a porta 80.

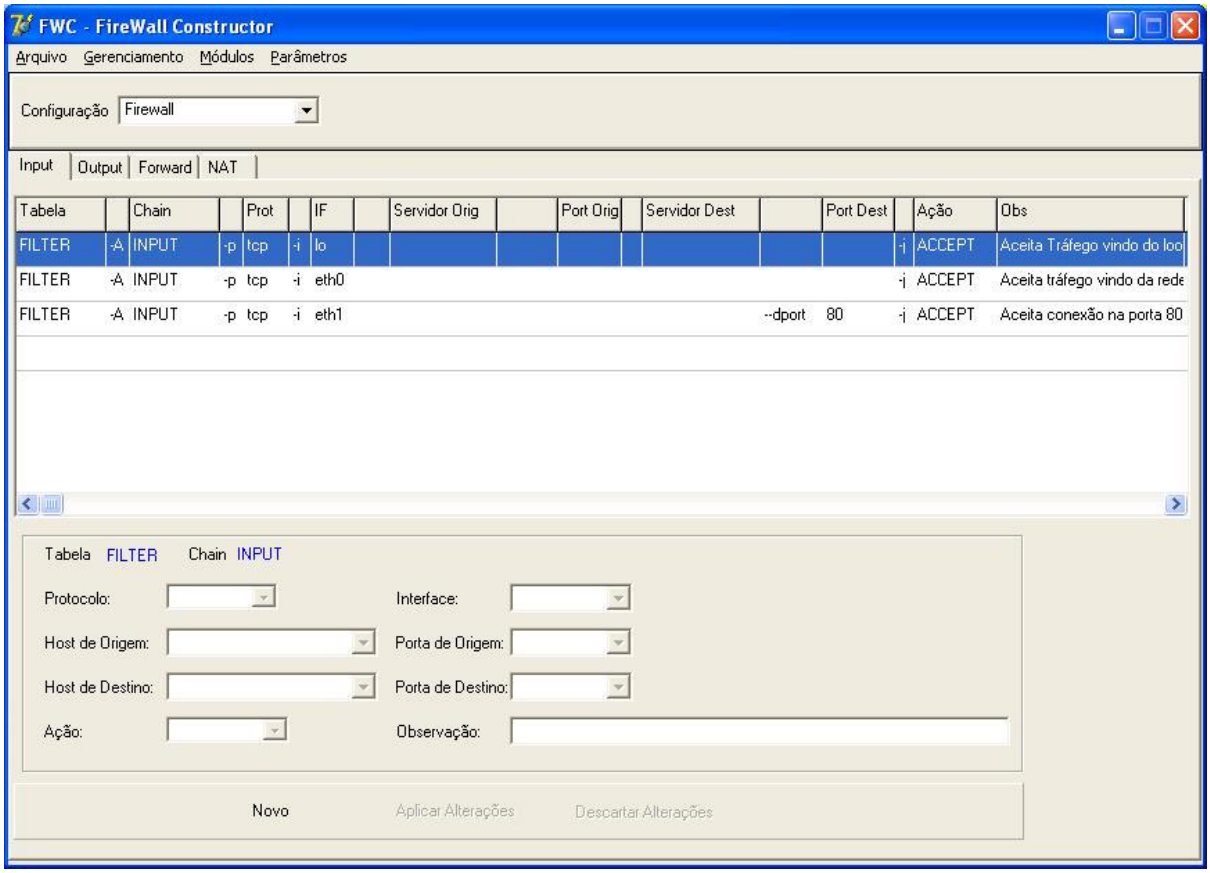

Figura 21. Gerenciamento de Regras de *INPUT*.

4.1.8.2 Regras de *OUTPUT*

As regras a seguir exibidas através da Figura 22 são utilizadas para permitir a saída de pacotes oriundas do próprio *Firewall* e que sairão através das interfaces de rede eth0 e eth1.

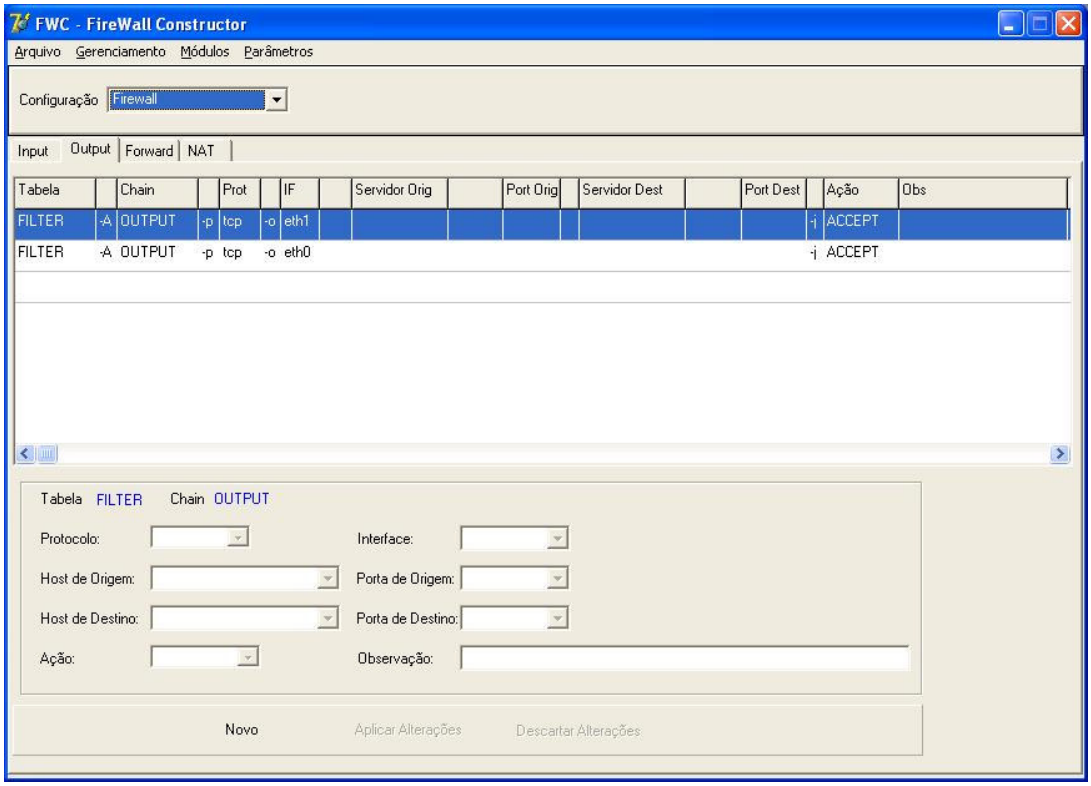

Figura 22. Gerenciamento de Regras de *OUTPUT*.

4.1.8.3 Regras de FORWARD

As regras exibidas na Figura 23 a seguir são utilizadas para permitir o redirecionamento de conexões oriundas da rede interna.

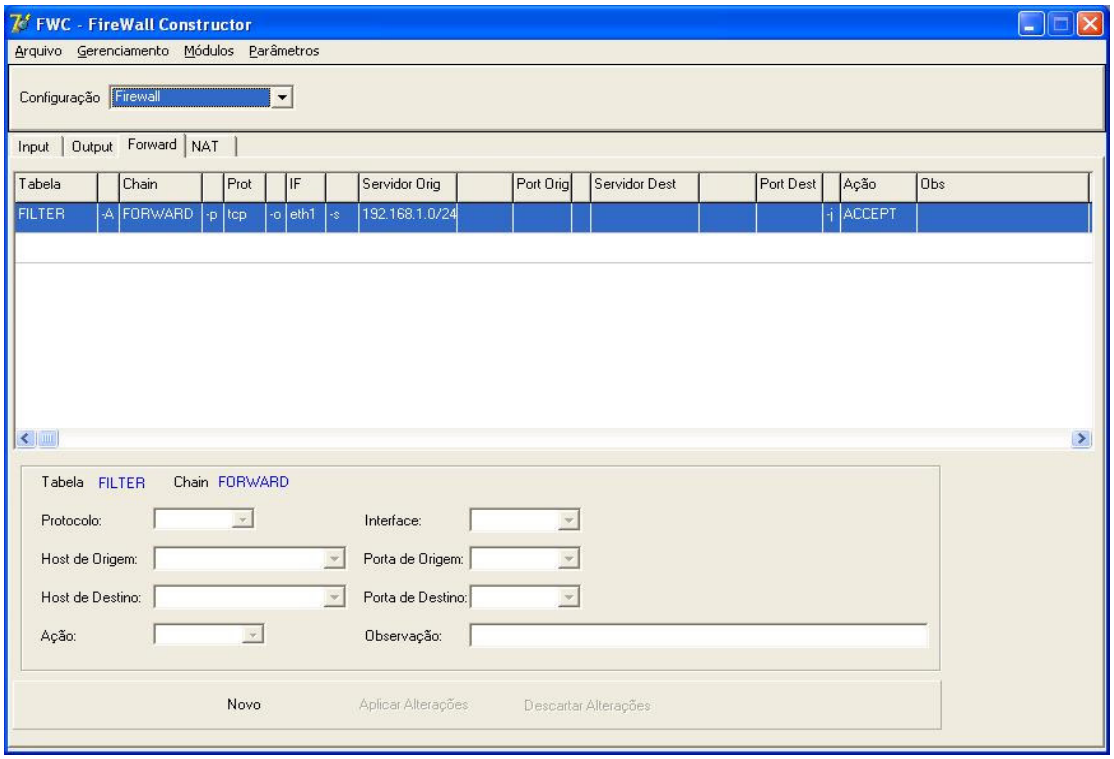

Figura 23. Gerenciamento de Regras de *FORWARD*.

4.1.8.4 Regras de NAT

As regras exibidas na Figura 24 a seguir são utilizadas para permitir as conexões com destino à rede local e para fazer o mascaramento na saída dos dados.

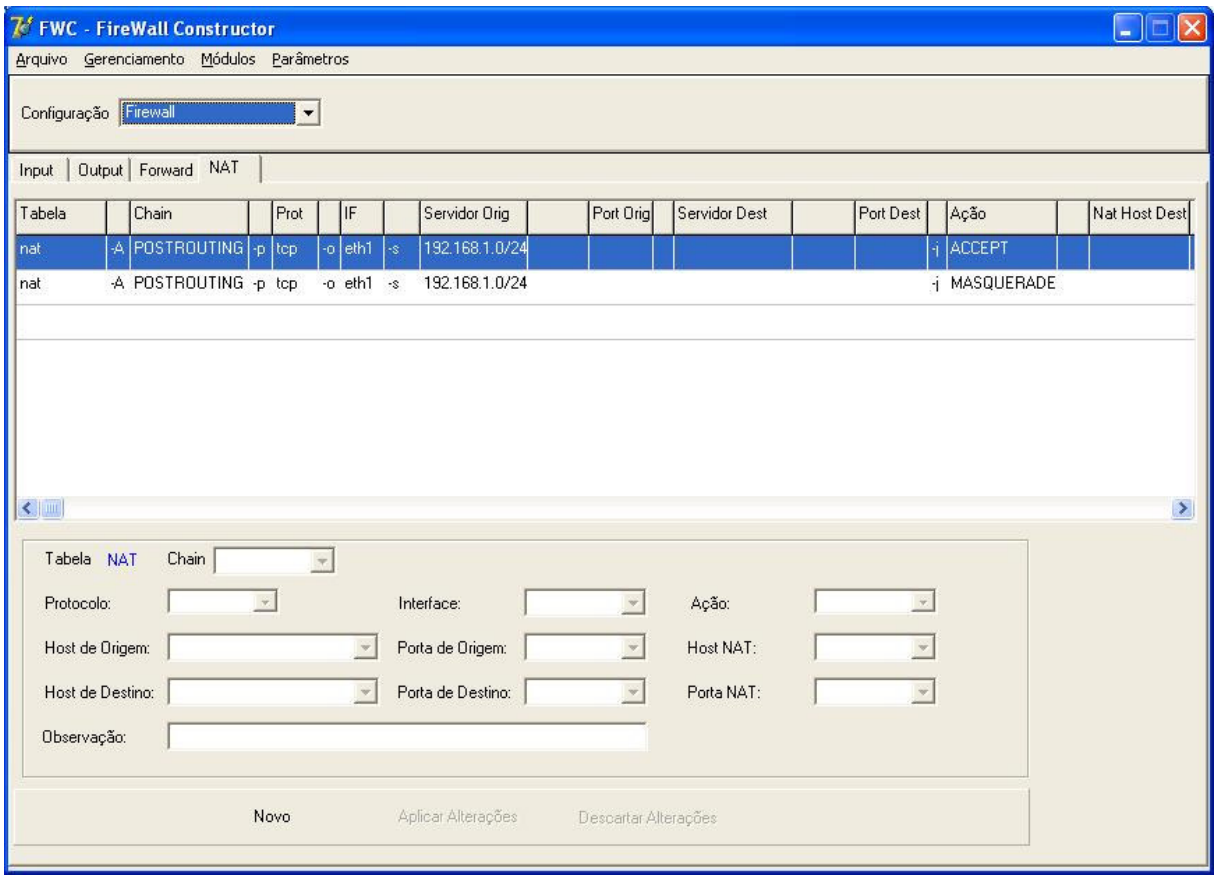

Figura 24. Gerenciamento de Regras de *NAT*.

4.1.9 Exportar Regras

A Figura 25 a seguir é mostra que as configurações serão exportadas e logo serão aplicadas no servidor. Os scripts gerados estão no ANEXO A.

Figura 25. Exportar Regras.

4.1.10 Aplicar Configuração On-Line:

A Figura 26 a seguir mostra como é realizada a conexão com o servidor de *Firewall*, onde serão informados os dados referentes a conexão. Também possibilita a escolha da regra ao qual será enviada por meio da conexão SSH e utilização do shell do *Firewall*.

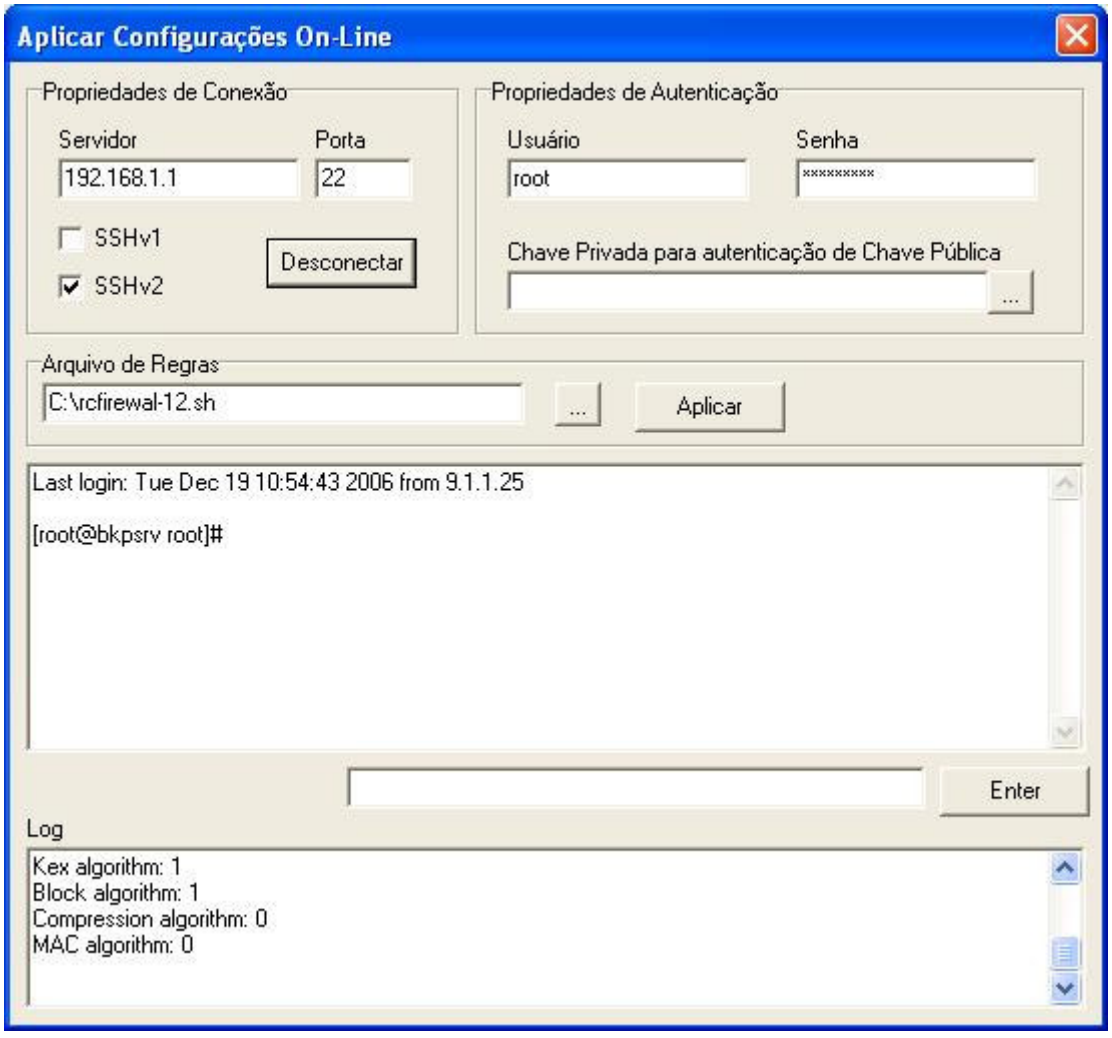

Figura 26. Aplicar Configurações On-Line.

5 CONCLUSÃO

No ambiente Linux, o *Firewall* IPTABLES é hoje um dos principais meios de se garantir a segurança dos dados trafegados em uma rede de computadores. O IPTABLES funciona selecionando ou filtrando os pacotes de dados de acordo com regras e configurações previamente estabelecidas e aplicadas.

A criação e configuração das regras de um *Firewall* IPTABLES é uma tarefa trabalhosa, realizada, em sua maioria, em ambiente texto, com linhas de comando, exigindo uma atenção adicional dos administradores de rede. O objetivo deste trabalho foi apresentar uma alternativa para facilitar esta tarefa de criação e configuração de regras, através de uma ferramenta com interface gráfica amigável para os administradores de redes.

Como foi demonstrado no capítulo 4 deste trabalho, todo o processo de criação e configuração das regras de um *Firewall* IPTABLES de uma rede de uma empresa de médio porte pode ser realizado através da interface gráfica do FWC. Com a ferramenta proposta, o administrador pode utilizar os recursos da interface (clicar e selecionar) para criar configurações, interfaces e regras de maneira mais ágil e segura do que o permitido em ambiente com linhas de comando. Além disso, a funcionalidade de exportar as regras criadas para um arquivo texto substitui a necessidade de criação e manutenção de scripts para aplicação das regras, já que estas ficam armazenadas no banco de dados e, sempre que necessário, poderão ser editadas e exportadas novamente.

Até o momento, o FWC suporta apenas a criação de regras para IPTABLES, em ambientes com no máximo 4 (quatro) redes. Como trabalho futuro, pretende-se expandir o FWC para que atenda a outros tipos de *Firewall* e seja capaz de configurar ambientes com um número maior de rede, facilitando ainda mais o trabalho de configuração de regras e contribuindo para o aumento no nível de segurança das redes corporativas.

6 REFERÊNCIAS BIBLIOGRÁFICAS

- SILVA, A. M.; PINHEIRO, M.S.F.; FREITAS, N.E.F. **Guia para Normalização de Trabalhos Técnico-Científicos**: projetos de pesquisa, monografias, dissertações e teses. 5. ed. Uberlândia:EDUFU, 2004.
- NAKAMURA, E. T. ;GEUS, P. C. **Segurança de Redes em Ambientes Cooperativos.** 2ª. ed. São Paulo: Futura, 2003.
- NETO, U.; **Dominando Linux *Firewall* IPTABLES.** Rio de Janeiro: Ciência Moderna, 2004.
- Burnett, S.; Paine, S. **Criptografia e Segurança:** O Guia Oficial RSA. Rio de Janeiro:Elsevier, 2002.
- SCRIMGER, R.; LASALLE, P.; PARIHAR, M.; GUPTA, M.; **TCP/IP a Bíblia.** Rio de Janeiro: Campus, 2002.
- ZWICKY, ELIZABETH D.; **Construindo *Firewall*s para a Internet**. 2ª ed. Rio de Janeiro, Editora Campus,2001.
- MOTA, J. E.; ***Firewall* com IPTABLES:** Disponível em: <http://www.eriberto.pro.br/IPTABLES/3.html>. Acesso em: 20 set. 2006.
- SiLVA, G. M.; ***Firewall* IPTABLES**: Disponível em: <http://focalinux.cipsga.org.br/guia/avancado/ch-fw-iptables.htm>. Acesso em: 10 out. 2006.
- SANTOS, A.; ***Tópicos avançados em TCP***: Disponível em: < www.secforum.com.br/textos/tcp1.htm>. Acesso em: 10 out. 2006.

7 ANEXOS

7.1 ANEXO A

Segue abaixo o script de exemplo para criação de regras IPTABLES no Linux.

```
####### CONFIGURACAO FIREWALL #########

#Criado por: usuario

####### ATIVACAO DE MODULOS ########
echo "1" > /proc/sys/net/ipv4/ip_forward
echo "1" > /proc/sys/net/ipv4/tcp_syncookies

####### CONFIG ########

#Nome do Servidor=Firewall
#Descricao do Servidor= Servidor Firewall
#kernel=2.6
#path=/sbin/iptables
#confdir=/usr/local/fwc
#proxytransp=

####### Limpando Regras ########
iptables -F
iptables -X
iptables -t nat -F
iptables -t nat -X

####### DEFINICAO DE POLICIAMENTO ########
#Tabela filter

iptables -t filter -P INPUT DROP
iptables -t filter -P OUTPUT DROP
iptables -t filter -P FORWARD DROP
#Tabela nat

iptables -t nat -P PREROUTING DROP
iptables -t nat -P OUTPUT DROP
iptables -t nat -P POSTROUTING DROP


####### INTERFACE  eth0  #########

INT_IFACE=eth0
INT_IP=192.168.1.1
INT_NET=192.168.1.0/24
INT_BRO=192.168.1.255


####### INTERFACE  eth1  #########

EXT_IFACE=eth1
EXT_IP=200.225.xxx.xxx
EXT_NET=200.225.xxx.xxx/30
EXT_BRO=200.225.xxx.255


####### INTERFACE  eth2  #########
```

```
####### INTERFACE  eth3  #########


####### REGRAS INPUT #########

iptables -t FILTER -A INPUT -p tcp -i lo -j ACCEPT # Aceita Tráfego vindo
do loopback
iptables -t FILTER -A INPUT -p tcp -i eth0 -j ACCEPT # Aceita tráfego vindo
da rede interna
iptables -t FILTER -A INPUT -p tcp -i eth1 --dport 80 -j ACCEPT # Aceita
conexão na porta 80

####### REGRAS OUTPUT #########

iptables -t FILTER -A OUTPUT -p tcp -o eth1 -j ACCEPT #
iptables -t FILTER -A OUTPUT -p tcp -o eth0 -j ACCEPT #

####### REGRAS FORWARD #########

iptables -t FILTER -A FORWARD -p tcp -o eth1 -s 192.168.1.0/24 -j ACCEPT #

####### REGRAS nat #########

iptables -t nat -A POSTROUTING -p tcp -o eth1 -s 192.168.1.0/24 -j ACCEPT #
iptables -t nat -A POSTROUTING -p tcp -o eth1 -s 192.168.1.0/24 -j
MASQUERADE #
```

7.2 ANEXO B

Segue abaixo o script de criação da base de dados da aplicação.

```
CREATE DATABASE /*!32312 IF NOT EXISTS*/ fwc;
USE fwc;
CREATE TABLE `acoes` (
  `idacoes` int(10) unsigned NOT NULL auto_increment,
  `tipo` varchar(45) NOT NULL,
  PRIMARY KEY  (`idacoes`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 ROW_FORMAT=COMPACT;

CREATE TABLE `chains` (
  `idchains` int(10) unsigned NOT NULL auto_increment,
  `nome` varchar(45) NOT NULL,
  PRIMARY KEY  (`idchains`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 ROW_FORMAT=COMPACT;

CREATE TABLE `config` (
  `idconfig` int(10) unsigned NOT NULL auto_increment,
  `nome` varchar(45) NOT NULL,
  `descricao` varchar(45) default NULL,
  `path_iptables` varchar(45) NOT NULL,
  `confdir` varchar(45) default NULL,
  `FKnatidnat` int(10) unsigned NOT NULL,
  `kernel` varchar(20) NOT NULL default '',
  `usarproxy` tinyint(1) NOT NULL default '0',
  `proxytransip` varchar(30) NOT NULL default '',
  PRIMARY KEY  (`idconfig`,`nome`),
  KEY `FKnat` (`FKnatidnat`),
```

```
  CONSTRAINT `FKnat` FOREIGN KEY (`FKnatidnat`) REFERENCES `nat` (`idnat`)
ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=latin1 ROW_FORMAT=COMPACT;

CREATE TABLE `hosts` (
  `idhosts` int(10) unsigned NOT NULL auto_increment,
  `nome` varchar(45) NOT NULL default '',
  `nomereal` varchar(45) NOT NULL default '',
  `ip` varchar(45) NOT NULL default '',
  `Fkhostsidhosts` int(10) unsigned NOT NULL default '0',
  PRIMARY KEY  (`idhosts`),
  KEY `FKhostsidhosts` (`idhosts`,`nome`),
  KEY `FK_hosts` (`Fkhostsidhosts`),
  CONSTRAINT `FK_hosts` FOREIGN KEY (`Fkhostsidhosts`) REFERENCES `config`
(`idconfig`) ON DELETE CASCADE ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `interfaces` (
  `idinterfaces` int(10) unsigned NOT NULL auto_increment,
  `alias` varchar(25) NOT NULL default '',
  `nome` varchar(25) NOT NULL default '',
  `ip` varchar(25) NOT NULL default '',
  `rede` varchar(25) NOT NULL default '',
  `broadcast` varchar(25) NOT NULL default '',
  `FKidconfig` int(10) unsigned NOT NULL,
  PRIMARY KEY  (`idinterfaces`,`nome`),
  KEY `FKconfig` (`FKidconfig`),
  CONSTRAINT `FKconfigintefaces` FOREIGN KEY (`FKidconfig`) REFERENCES
`config` (`idconfig`) ON DELETE CASCADE ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=latin1 ROW_FORMAT=COMPACT;

CREATE TABLE `nat` (
  `idnat` int(10) unsigned NOT NULL,
  `tipo` varchar(20) NOT NULL,
  `descricao` varchar(45) default NULL,
  PRIMARY KEY  (`idnat`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 ROW_FORMAT=COMPACT;

CREATE TABLE `portas` (
  `idportas` int(10) unsigned NOT NULL auto_increment,
  `nome` varchar(15) NOT NULL default '',
  `numero` int(10) unsigned NOT NULL default '0',
  `descricao` varchar(45) NOT NULL default '',
  `Fkportasidportas` int(10) unsigned NOT NULL default '0',
  PRIMARY KEY  (`idportas`),
  KEY `FKportasidportas` (`idportas`,`nome`),
  KEY `FK_portas` (`Fkportasidportas`),
  CONSTRAINT `FK_portas` FOREIGN KEY (`Fkportasidportas`) REFERENCES
`config` (`idconfig`) ON DELETE CASCADE ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `protocolo` (
  `idprotocolo` int(10) unsigned NOT NULL auto_increment,
  `nome` varchar(10) default NULL,
  PRIMARY KEY  (`idprotocolo`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 ROW_FORMAT=COMPACT;


CREATE TABLE `regras` (
  `idregras` int(10) unsigned NOT NULL auto_increment,
  `portas_orig` varchar(15) default NULL,
```

```
  `opcoes_porta_dest` varchar(15) default NULL,
  `opcoes_porta_orig` varchar(15) default NULL,
  `portas_dest` varchar(15) default NULL,
  `opcoes_servidor_orig` varchar(15) default NULL,
  `opcoes_servidor_dest` varchar(15) default NULL,
  `servidor_orig` varchar(30) default NULL,
  `servidor_dest` varchar(30) default NULL,
  `FKprotocoloidprotocolo` int(10) unsigned NOT NULL,
  `opcoes_protocolo` varchar(15) default NULL,
  `opcoes_chain` varchar(15) default NULL,
  `FKchainsidchains` int(10) unsigned NOT NULL,
  `FKtabelasidTabela` int(10) unsigned NOT NULL,
  `FKacoesidacoes` int(10) unsigned NOT NULL,
  `opcoes_acoes` varchar(15) default NULL,
  `nome_modulo` varchar(45) NOT NULL default '',
  `FKidconfiguracao` int(10) unsigned NOT NULL,
  `opcoes_interface` varchar(15) NOT NULL default '',
  `interface` varchar(45) NOT NULL default '',
  `observacao` varchar(45) NOT NULL default '',
  `opcoes_host_nat` varchar(45) NOT NULL default '',
  `host_nat` varchar(45) NOT NULL default '',
  `opcoes_porta_nat` varchar(45) NOT NULL default '',
  `porta_nat` varchar(45) NOT NULL default '',
  `precedencia` int(10) unsigned default NULL,
  PRIMARY KEY  (`idregras`),
  KEY `FKconfiguracao` (`FKidconfiguracao`),
  KEY `FKprotocolo` (`FKprotocoloidprotocolo`),
  KEY `FKchains` (`FKchainsidchains`),
  KEY `FKtabelas` (`FKtabelasidTabela`),
  KEY `FKacoes` (`FKacoesidacoes`),
  CONSTRAINT `FKacoes` FOREIGN KEY (`FKacoesidacoes`) REFERENCES `acoes`
(`idacoes`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `FKchains` FOREIGN KEY (`FKchainsidchains`) REFERENCES
`chains` (`idchains`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `FKconfiguracao` FOREIGN KEY (`FKidconfiguracao`) REFERENCES
`config` (`idconfig`) ON DELETE CASCADE ON UPDATE NO ACTION,
  CONSTRAINT `FKprotocolo` FOREIGN KEY (`FKprotocoloidprotocolo`)
REFERENCES `protocolo` (`idprotocolo`) ON DELETE NO ACTION ON UPDATE NO
ACTION,
  CONSTRAINT `FKtabelas` FOREIGN KEY (`FKtabelasidTabela`) REFERENCES
`tabelas` (`idTabela`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=latin1 ROW_FORMAT=COMPACT;

CREATE TABLE `tabelas` (
  `idTabela` int(10) unsigned NOT NULL auto_increment,
  `tipo` varchar(45) default NULL,
  PRIMARY KEY  (`idTabela`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 ROW_FORMAT=COMPACT;

/*!40000 ALTER TABLE `tabelas` DISABLE KEYS */;
INSERT INTO `tabelas` VALUES  (1,'FILTER'),
 (2,'nat');
/*!40000 ALTER TABLE `tabelas` ENABLE KEYS */;

CREATE TABLE `usuarios` (
  `idusuarios` int(10) unsigned NOT NULL auto_increment,
  `login` varchar(15) NOT NULL,
  `nome` varchar(45) default NULL,
  `senha` varchar(20) NOT NULL,
  `remoteip` varchar(45) default NULL,
  `admin` tinyint(1) NOT NULL default '0',
```

```
  PRIMARY KEY  (`idusuarios`,`login`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 ROW_FORMAT=COMPACT;

CREATE TABLE `usuarios_has_config` (
  `FKusuariosidusuarios` int(10) unsigned NOT NULL,
  `FKconfigidconfig` int(10) unsigned NOT NULL,
  KEY `FKusuarios` (`FKusuariosidusuarios`),
  KEY `FKconfig` (`FKconfigidconfig`),
  CONSTRAINT `FKconfig` FOREIGN KEY (`FKconfigidconfig`) REFERENCES
`config` (`idconfig`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `FKusuarios` FOREIGN KEY (`FKusuariosidusuarios`) REFERENCES
`usuarios` (`idusuarios`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=latin1 ROW_FORMAT=COMPACT;
```