

Hugo Leonardo Prado Ribeiro
hlribeiro@gmail.com

Vuurmuur: Firewall sem complicações

Maringá
Julho/2005

Hugo Leonardo Prado Ribeiro
hlribeiro@gmail.com

Vuurmuur: Firewall sem complicações

Professor Orientador: Flávio Arnaldo Braga da Silva

CESUMAR - CENTRO UNIVERSITÁRIO DE MARINGÁ

Maringá
Julho/2005



This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/2.5/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.

*Ao meu filho Gustavo, que deu por si o maior presente que um homem pode ganhar.
À minha alma-gêmea Fer, que conseguiu fazer de mim a pessoa mais feliz do mundo.*

Agradecimentos

Primeiramente a Deus, que me deu a capacidade e esperança necessária para chegar até aqui. À Fer, que entendeu todo meu stress durante a produção deste documento. Aos meus pais, por terem me dado a formação responsável pelo que sou hoje. Ao fundão, por termos proporcionado suporte um ao outro para chegarmos até aqui, e pelos encontros diários mais ‘produtivos’ da face da terra. Ao meu professor orientador Flávio, por sugestões e observações vitais para a boa composição deste trabalho. A todos os professores que passaram, com exceção de alguns. À Márcia Pascutti, por ter entendido minha mudança de proposta na última hora. Ao Victor Julien, pelas 2 imagens que eu não tinha, pelo amplo suporte na solução de dúvidas na elaboração deste, e por ter feito o Vuurmuur, possibilitando que eu escrevesse tudo isto. Ao Adi Kriegisch, pelas imagens cedidas para a criação da seção 3.2.7. Finalmente, à *Joseph Schumpeter* (o original), por facilitar tanto a forma de chamar os amigos.

*“Watching the sky while
I try to stay on solid ground.”*
— RAISED FIST

Resumo

Este trabalho é sobre o Vuurmuur, um front-end para Iptables que adiciona inúmeras funcionalidades ao mesmo. Iremos ver toda sua forma de funcionamento e sua filosofia de objetos, além de abordar, de forma fácil e explicativa, todos os passos necessários para se montar um firewall decente, sem precisar dos conhecimentos específicos do Iptables.

Abstract

This study is about Vuurmuur, an Iptables front-end that adds many new functions to it. We will look at the way of Vuurmuur works and your object philosophy. Also, we will see, in an easier and explicative way, all the necessary steps to make a decent firewall, but without need specific Iptables knowledge.

Sumário

Lista de Figuras

Lista de Tabelas

1	Introdução	p. 16
1.1	Objetivos	p. 17
1.1.1	Gerais	p. 17
1.1.2	Específicos	p. 17
1.2	Estrutura	p. 18
1.2.1	Firewall	p. 18
1.2.2	Vuurmuur	p. 18
1.3	Justificativa	p. 18
2	Firewall	p. 19
2.1	Filtro de Pacotes	p. 19
2.2	Iptables	p. 19
2.2.1	Evolução dos mecanismos de filtragem	p. 20
2.2.1.1	A filtragem tradicional	p. 20
2.2.1.2	Novas implementações do iptables	p. 21
2.2.1.3	Extensões de controle de protocolos	p. 21
2.2.1.4	Novos alvos	p. 21
2.2.1.5	Novas associações	p. 22
2.2.2	Regras	p. 22

2.2.2.1	“Forma culta”: Iptables-save, Iptables-restore	p. 23
2.2.2.2	“Forma prática”: script	p. 23
3	Vuurmuur	p. 25
3.1	Conhecendo	p. 25
3.2	Entendendo	p. 27
3.2.1	Interfaces	p. 27
3.2.2	Serviços	p. 28
3.2.3	Hosts	p. 28
3.2.4	Grupos	p. 28
3.2.5	Redes	p. 28
3.2.6	Zonas	p. 29
3.2.7	Juntando tudo isto	p. 29
3.2.8	Pequenos detalhes	p. 31
3.3	Instalando	p. 32
3.3.1	A partir do código fonte	p. 32
3.3.2	Autopackage e binários diversos	p. 36
3.4	Conhecendo menus e opções	p. 37
3.4.1	vuurmuur	p. 37
3.4.2	vuurmuur_log	p. 40
3.4.3	vuurmuur_script	p. 40
3.4.4	vuurmuur_conf	p. 42
3.4.4.1	Vuurmuur Config	p. 48
3.4.4.2	Logview	p. 57
3.4.4.3	Status	p. 59
3.4.4.4	Connections	p. 60
3.4.4.5	Bandwidth	p. 63

3.4.4.6	Vuurmuur_conf Settings	p. 64
3.4.5	Sobre os idiomas	p. 65
3.5	Montando o cenário-exemplo	p. 67
3.5.1	Elaborando o projeto	p. 69
3.5.2	Interfaces	p. 70
3.5.3	Services	p. 72
3.5.4	Zones	p. 76
3.5.5	Blocklist	p. 89
3.5.6	Rules	p. 90
3.6	Limitações e Contribuição	p. 107
4	Conclusão	p. 109
	Referências	p. 111

Lista de Figuras

1	Rede simples	p. 29
2	Rede simples com Zonas	p. 30
3	Rede simples com Zonas e Redes	p. 30
4	Rede simples com Host	p. 31
5	Rede simples com Host, Redes e Zonas	p. 31
6	Descompactando o código-fonte	p. 32
7	Opções do install.sh	p. 33
8	Instalação em andamento	p. 35
9	Binários do Vuurmuur	p. 35
10	Opções do daemon vuurmuur	p. 38
11	Primeira tela: vuurmuur_conf	p. 43
12	Avisos do vuurmuur_conf	p. 44
13	Criação de regras	p. 45
14	Lista de bloqueados	p. 46
15	Criação de zonas, redes, grupos e hosts	p. 46
16	Criação e manipulação de interfaces	p. 47
17	Criação de serviços	p. 47
18	Opções do Vuurmuur	p. 48
19	Vuurmuur Config: General	p. 49
20	Vuurmuur Config: Connections	p. 50
21	Vuurmuur Config: Interfaces	p. 51
22	Vuurmuur Config: System Protection	p. 51

23	Vuurmuur Config: Logging	p. 52
24	Vuurmuur Config: Modules	p. 54
25	Vuurmuur Config: Plugins	p. 55
26	Vuurmuur Config: Aviso sobre módulos não presentes	p. 56
27	Vuurmuur Config: Capabilities	p. 56
28	Logview	p. 57
29	Logview: vuurmuur.log	p. 57
30	Logview: traffic.log	p. 59
31	Vuurmuur_conf: Status	p. 59
32	Connections: Visualização em tempo real	p. 60
33	Connections: in/out/fw	p. 61
34	Connections: connect status	p. 62
35	Connections: group unknown	p. 62
36	Connections: campo 'status' aparecendo	p. 63
37	Traffic Log Volume	p. 64
38	Vuurmuur_conf: Settings	p. 64
39	Vuurmuur_conf em Português do Brasil	p. 66
40	Interfaces: Atribuindo um nome	p. 70
41	Interfaces: Definindo propriedades	p. 71
42	Interfaces criadas	p. 72
43	Serviços: grande quantidade de serviços pré-definidos	p. 72
44	Serviços: Adicionando um novo serviço	p. 73
45	Serviços: Diálogo de propriedades	p. 74
46	Serviços: Definindo portas	p. 74
47	Serviços: Portas de Origem e Destino	p. 75
48	Serviços: Portranges definidos	p. 76

49	Zonas: Definindo um nome	p. 77
50	Zonas: Adicionando	p. 77
51	Zonas: Projeto simples	p. 78
52	Zonas: Redes	p. 78
53	Redes: Adicionando	p. 79
54	Redes: Adicionando rede Internet	p. 80
55	Redes: Atribuindo interfaces	p. 81
56	Redes: Criando a rede 'servers'	p. 81
57	Redes: Criando a rede 'clientes'	p. 82
58	Zonas: Duas redes em uma mesma zona	p. 82
59	Hosts: Adicionando	p. 83
60	Hosts: Amanda.clientes.interno	p. 83
61	Hosts: Lista de hosts	p. 84
62	'Hosts' na Internet	p. 85
63	Grupos	p. 85
64	Grupos: Informatica	p. 86
65	Grupos: Simples e objetivos	p. 87
66	Grupos: Criação	p. 87
67	Grupos: Seleção de membros	p. 88
68	Grupos: Agrupamento de semelhanças	p. 88
69	Blocklist: Endereço IP, host ou grupo	p. 89
70	Blocklist: Bloqueio simples e fácil	p. 90
71	Rules: Criação de regras no Vuurmuur	p. 91
72	Rules: Inserindo uma regra	p. 92
73	Rules: target DROP	p. 93
74	Rules: Hosts, grupos, redes e zonas	p. 94

75	Rules: Primeira regra	p. 95
76	accept any from informatica.clientes.interno to any	p. 95
77	Rules: Duas regras	p. 96
78	Linha separadora	p. 97
79	Regra três	p. 97
80	Regra quatro	p. 98
81	Regra cinco	p. 98
82	Regra seis	p. 99
83	Regra sete	p. 99
84	Regra oito	p. 100
85	Regra nove	p. 100
86	Encaminhamento de portas	p. 102
87	Rules: portforward simples	p. 103
88	Rules: IBM ping	p. 103
89	Rules: SSH	p. 104
90	SNAT	p. 105
91	Liberação de regras para o firewall.	p. 106
92	Snat no topo das regras	p. 107
93	About	p. 108

Lista de Tabelas

1	Cientes: Hosts, Ips e Departamento	p.67
2	Servidores: Hosts e Ips	p.68

1 *Introdução*

Atualmente, o quesito segurança têm-se tornado grande preocupação dos usuários de computadores, no que diz respeito à exposição de seus dados, uma vez que, cada vez mais, o computador se torna ferramenta essencial na produção de documentos que virão a se tornar confidenciais ou importantes. Neste contexto, a Internet se mostra um dos principais pontos desta preocupação, pois é onde se encontram os maiores perigos, e também o maior número de usuários “conectados” entre si. Além da Internet, a segurança de dados também se aplica a diversos outros fatores, como a própria LAN¹, onde os usuários estão sujeitos à más intenções de outros, além de possíveis danos na segurança causados por fatores físicos, que vão desde o bom funcionamento do hardware, até a invasão do local físico onde está situado o equipamento (roubo da máquina, por exemplo).

Com a grande popularização da Internet nos últimos anos, o número de pessoas conectadas à *grande rede* aumentou exponencialmente, fazendo com que cada vez mais fosse possível encontrar de tudo com poucos cliques. Ao mesmo tempo, crescia de forma oculta uma legião de pessoas mal intencionadas, que veio a resultar neste grandiosíssimo problema que temos hoje, o da segurança. Hackers, crackers, curiosos e bisbilhoteiros cada vez mais se interessam pela descoberta de falhas de segurança, falhas de administradores de rede ou qualquer outro fator que possa se tornar uma brecha para qualquer tipo de ação ofensiva, seja ela um simples flood, a exploração de uma falha, o comprometimento de um serviço ou a tomada completa de um sistema. Com isto, na intenção de implementar uma possibilidade de controle sobre o que entra e sai de uma rede/host, surgiram os filtros de pacotes, dentre quais podemos citar os três mais bem-sucedidos: ipfwadm (Linux kernel 2.0.x), ipchains (Linux kernel 2.2.x), e o atual iptables (Linux kernels 2.4.x e 2.6.x).

Filtros de pacotes são, de modo grosseiro, funções de registro implantadas no *kernel*², que permitem-no saber cada pacote que passa por ele, podendo então, com base em uma lista de regras, autorizá-lo ou não a seguir seu destino pré-especificado, bem como alterar

¹*Local Area Network*

²Núcleo principal do sistema Linux

alguns de seus valores. O iptables é uma ótima ferramenta para isto, sendo padrão na grande maioria das implementações de firewall feitas em Linux.

Apesar da grande habilidade do netfilter/iptables em manusear os pacotes de acordo com as regras estabelecidas, ele não fornece nenhuma interface amigável para a criação e manuseio destas regras, bem como o acompanhamento de registros em tempo real do que está sendo processado pelo firewall.

1.1 Objetivos

1.1.1 Gerais

Este trabalho tem como foco a apresentação geral da ferramenta chamada Vuurmuur, que se trata de um *front-end*³ para o iptables, e que implementa várias funcionalidades e facilidades num nível mais alto do que a fornecida pelo mesmo, visando facilitar a vida do administrador de redes, e ao mesmo oferecer-lhe novas possibilidades com suas funções, que serão devidamente apresentadas adiante.

1.1.2 Específicos

Os objetivos específicos deste trabalho se resumem única e exclusivamente à:

- a) Apresentar a ferramenta, com suas funções, capacidades e limitações;
- b) Abordar a sua instalação (a partir de código-fonte), com todas as opções e possibilidades;
- c) Exemplificar sua forma de configuração e demonstrar seus recursos, criando um cenário-exemplo que suporte tal objetivo.

Não é objeto do escopo deste trabalho pôr em prova o sucesso de funcionamento das ferramentas Vuurmuur e Iptables, ficando assim claro o objetivo deste trabalho, que é o de apresentação da ferramenta Vuurmuur.

³Interface que coleta dados, geralmente de forma facilitada em relação ao programa original, formata-os segundo os padrões desse programa, e repassa-os a ele.

1.2 Estrutura

Para que se possa causar um melhor dimensionamento deste trabalho, o mesmo será segmentado em 2 grandes grupos distintos:

1.2.1 Firewall

Neste capítulo, estaremos apresentando um breve descritivo sobre filtros de pacotes e o próprio iptables, necessários para o entendimento da forma de funcionamento de um firewall.

1.2.2 Vuurmuur

Esta parte do trabalho será exclusivamente dedicada ao objeto do trabalho: apresentação, implementação, configuração e criação do cenário-exemplo para o Vuurmuur.

1.3 Justificativa

Atualmente, tantos são os problemas do administrador de redes, tantas são suas preocupações, que acabam fazendo do tempo uma preciosidade, que deve ser muito bem aproveitada. Neste contexto, uma ferramenta que facilita tanto a administração do firewall, especialmente em grandes redes, e que, especialmente, poupa **tempo** do administrador, consta de grande utilidade. Além disto, é uma ferramenta relativamente nova, mas que já faz por merecer destaque dentre as semelhantes em sua categoria.

2 *Firewall*

Firewall, em inglês, significa "Parede de Fogo", e atualmente é um item indispensável de segurança para empresas, redes corporativas e usuários domésticos. Os firewalls usam um filtro de pacotes para tomarem suas decisões, podendo até ter como base de escolha análises complexas de propriedades de protocolos.

2.1 Filtro de Pacotes

Um filtro de pacotes é um software que analisa o cabeçalho (*header*) dos pacotes que processa, e decide o que fazer com ele (1).

De acordo com a análise feita, variadas ações poderão ser tomadas, como descartar completamente o pacote, sem sequer avisar o remetente, rejeitá-lo, enviando uma mensagem ao remetente, deixá-lo passar, entre muitas outras opções, também conhecidas, ao menos nos filtros de pacotes Linux, como *targets*.

No Linux, o filtro de pacotes pode estar disponível de duas formas: compilado diretamente no kernel ou como módulo. Ou, até mesmo, ter parte compilada diretamente no kernel, e algumas funções compiladas como módulos.

2.2 Iptables

Conforme relatado por Russel (1), os kernels Linux têm tido filtros de pacotes desde a série 1.1. A primeira geração, baseada no ipfw do BSD, foi portada por Alan Cox no final de 1994. Essa implementação foi melhorada por Jos Vos e outros para o Linux 2.0, gerando a 'ipfwadm', que controlava as regras de filtragem do kernel. Em meados de 1998, Rusty Russel e Michael Neuling reescrevem novamente linhas de código do kernel, e introduzem a ferramenta 'ipchains', com vários aprimoramentos, para o kernel 2.2. Finalmente, a ferramenta da quarta geração, o 'iptables', após mais um árduo trabalho de reedição do

kernel, é introduzida em meados de 1999 para o Linux 2.4, e consta até hoje, nos kernels 2.6.

O iptables introduziu várias novas funções, como a já mencionada modularização de recursos, a inspeção de estados de pacotes (*Statefull Inspection*), dentre outras opções sofisticadas de filtragem. Para tal, é necessário um kernel que possua a infra-estrutura netfilter¹ implementada. Isso significa que você precisa do kernel 2.3.15 ou posteriores, e responder 'Y (SIM)' para `CONFIG_NETFILTER` na sua configuração do kernel. A ferramenta iptables se comunica diretamente com o kernel (através do netfilter) e indica, segundo suas regras, quais pacotes deverão ser filtrados, e quais ações deverão ser tomadas para eles.

2.2.1 Evolução dos mecanismos de filtragem

Conforme composto no trabalho de Vianna(2), a evolução dos mecanismos de filtragem para os filtros de pacotes do Linux se deu da seguinte forma:

2.2.1.1 A filtragem tradicional

A filtragem de pacotes implementada desde o kernel 2.0 dispõe de três listas de regras na tabela filter (módulo específico do kernel para filtragem de pacotes); tais listas são denominadas *firewall chains* (ou simplesmente chains). As três chains básicas são INPUT, OUTPUT e FORWARD. Quando o pacote atinge uma chain no diagrama, a chain é examinada a fim de que seja decidido o destino do pacote. Se a chain aponta para descartar (DROP) o pacote, ele é descartado, mas se a chain aponta para aceitar o pacote (ACCEPT).

Uma chain é uma lista de regras. Cada regra pode ser interpretada como uma condição do tipo: 'se o cabeçalho do pacote se parece com isso, aqui está o que deve ser feito com o pacote'. Se a regra não se associa com o pacote, então a próxima regra na chain é consultada. Não havendo mais regras a consultar, o kernel analisa a política da chain para decidir o que fazer. Como apresentado anteriormente a chain pode ter 2 tipos de política: 'Tudo o que não for expressamente proibido é permitido' ou 'Tudo o que não for expressamente permitido é proibido'.

Quando o pacote chega ao computador, pela placa ethernet, por exemplo, o kernel analisa o seu destino, num processo denominado roteamento (routing). Segue-se então a decisão de roteamento, onde é realizado o repasse (FORWARD), ou, caso o pacote se destine à própria máquina, o encaminhamento à chain INPUT. Ao chegar nestas chains, o pacote passa por regras específicas que decidirão o seu destino (aceitar ou descartar).

¹Netfilter é um framework dentro do kernel Linux com o qual outras coisas (como o módulo do iptables) podem conectar-se.

2.2.1.2 Novas implementações do iptables

Além dos recursos das soluções de filtragem de pacotes que o antecederam, o iptables (com o suporte do Netfilter) é extensível, isto é, dispõe de suporte a módulos adicionais, proporcionando várias outras funcionalidades. Estes módulos podem ser carregados ‘por demanda’, de acordo com a necessidade de filtragem. Estas extensões são classificadas em três tipos:

- Extensões de controle de protocolos (-p)
- Novos alvos (-j)
- Novas associações (-m)

2.2.1.3 Extensões de controle de protocolos

Extensões TCP

As extensões TCP são automaticamente carregadas quando especificada a opção `-p tcp` e dispõe das seguintes opções:

- `--tcp-flags`: Permite que sejam filtradas flags TCP específicas.
- `--source-port`: Indica uma porta ou conjunto (range) de portas TCP de origem.
- `--destination-port`: Indica uma porta ou conjunto (range) de portas TCP de destino.
- `--tcp-option`: Utiliza-se de números específicos para controlar (e eventualmente descartar) pacotes com a opção TCP igual ao do número informado. Um pacote que não tem um cabeçalho TCP completo é automaticamente descartado se há uma tentativa de examinar suas opções TCP.

Extensões UDP

Essas extensões são automaticamente carregadas se a opção `-p udp` é especificada. Permite as opções `--destination-port` e `--source-port`.

Extensões ICMP

Essas extensões são automaticamente carregadas se a opção `-p icmp` é especificada. Possui uma só opção diferente das demais extensões:

- `--icmp-type`: Controla o tipo de conexão ICMP baseado no nome de tipo ou tipo numérico comumente utilizado em conexões deste tipo.

2.2.1.4 Novos alvos

Os alvos (targets) indicam ‘o que fazer’ com o pacote caso coincida com as condições da regra. Os alvos padrões embutidos no iptables são: DROP (descartar) e ACCEPT (aceitar). Se a regra se associa com o pacote e seu alvo é um desses dois, nenhuma outra regra é consultada: o destino do pacote já foi decidido. Há dois tipos de alvos diferentes dos descritos acima: as chains definidas por usuários e as extensões.

1. Chains definidas por usuários

Uma funcionalidade que o iptables herdou do ipchains é a possibilidade da criação de novas chains, além das três disponíveis (INPUT, FORWARD e OUTPUT). Quando um pacote associa-se com uma regra cujo alvo é uma chain definida pelo usuário, o pacote passa a ser analisado pelas regras dessa nova chain.

2. Extensões ao iptables: Novos alvos (targets)

O outro tipo de alvo é a extensão. Uma extensão-alvo consiste em um módulo do kernel, opcional ao iptables, para prover opções de linha de comando. As extensões na distribuição padrão do netfilter são:

- **LOG**: Esse módulo provê o registro em logs dos pacotes submetidos, possuindo as seguintes opções adicionais:
 - log-level**: Seguido de um número de nível ou nome.
 - log-prefix**: Seguido de uma string de até 29 caracteres, que será adicionada no início da linha de registro de log (syslog), permitindo melhor identificação da mesma.
- **REJECT**: Esse módulo tem o mesmo efeito do alvo ‘DROP’, porém, retorna uma mensagem de erro ao remetente, rejeitando o pacote. O REJECT era um alvo nativo no ipchains. Porém, pela semelhança com o DROP, (além da pouca utilização), foi disponibilizado no iptables como uma extensão opcional

2.2.1.5 Novas associações

Estes tipos de extensões, no pacote netfilter, podem ser habilitadas com a opção **-m**. Estas funcionalidades adicionais permitem um controle mais detalhado dos pacotes, não se detendo ao cabeçalho, analisando informações em seu conteúdo.

- **mac**: Utilizado para associar a regra com o endereço Ethernet (MAC) da máquina de origem do pacote (**--mac-source**).
- **limit**: Utilizado para restringir a taxa de pacotes, e para suprimir mensagens de log.
- **owner**: Associa a regra a várias características do criador do pacote gerado localmente.
- **state**: Interpreta a análise do controle da conexão feita pelo módulo `ip_conntrack`.

2.2.2 Regras

Não será abordado aqui a sintaxe de criação de regras do iptables, uma vez que isto é uma das facilidades oferecidas pelo programa a ser abordado neste trabalho, ou seja, a não-obrigação do usuário em saber exatamente a sintaxe do programa. Entretanto, vale observarmos a forma em que o iptables trabalha com as regras, de forma a entendermos exatamente o que acontece quando ‘informamos’ as regras ao iptables.

Quando você digita um comando do iptables na linha de comando, ele, através do netfilter, ‘implanta’ esta regra diretamente no kernel, que passa a aplicá-la para cada

pacote que passar por ali. Por este motivo, cada vez que a máquina é reiniciada, as regras são automaticamente apagadas.

Para que, ao criar e definir suas regras de firewall, você não necessite fazê-la novamente caso seja preciso reiniciar a máquina, ou até mesmo para que você não fique com sua máquina exposta caso ela, por qualquer motivo, reinicie durante algum período em que você não está por perto, você deve arrumar alguma forma de dizer para o iptables recarregar estas regras no boot da máquina.

Mas como isto é feito?

2.2.2.1 “Forma culta”: Iptables-save, Iptables-restore

A partir do iptables (mas não em seu início), foram introduzidas as ferramentas iptables-save e iptables-restore. Essas duas ferramentas vieram para ‘resolver’ o problema citado anteriormente.

A ferramenta iptables-save, como se auto explica, salva as regras num arquivo texto indicado, usando a seguinte sintaxe:

```
iptables-save > regras.iptables (poderia ser qualquer nome)
```

Este comando gera um arquivo, com uma formatação própria, para ser utilizado com iptables-restore. A sintaxe do iptables-restore é igualmente simples:

```
iptables-restore regras.iptables ou
```

```
iptables-restore < regras.iptables
```

Simple, não? Montamos nossas regras, através da digitação das mesmas diretamente no console, salvamo-as utilizando o iptables-save e, após isso, colocamos o iptables-restore na inicialização do sistema, recuperando o arquivo salvo de regras.

2.2.2.2 “Forma prática”: script

O caso que acabou de ser citado é fácil de se fazer, e as duas ferramentas funcionam muito bem. **Porém**, vamos considerar o seguinte exemplo:

Em uma rede, constantes mudanças podem acontecer em relação às regras de firewall, como inclusão/exclusão de hosts autorizados/proibidos, mudanças de encaminhamento de portas, enfim, alterações de regras diversas.

Neste caso, o que o administrador de redes fará? Se ele quiser utilizar o conjunto `iptables-save + iptables-restore`, ele deverá listar as regras, apagar as que não servirem mais, inserir as novas regras na ordem correta, e salvar o arquivo novamente, para poder ser utilizado no próximo boot. Na prática, quase nenhum administrador se dá a este trabalho. A solução mais comum é desenvolver um script (geralmente em *bash*²) que, quando executado, limpe todas as regras, e aplique as novas, que estarão listadas neste mesmo script.

²Bourne Again Shell, a shell mais utilizada no Linux.

3 *Vuurmuur*

Como foi mencionado, as ferramentas iptables/netfilter formam um excelente conjunto de filtro de pacotes, que podem ser usadas de forma extremamente eficiente como firewall. Entretanto, o iptables não oferece uma forma amigável de criação e manipulação destas regras, no que diz respeito à forma de entrada das mesmas junto ao netfilter/kernel, à praticidade de mudanças, e o estabelecimento de padrões que facilitem a vida do administrador.

Da vontade de se projetar uma forma de criar ambientes mais complexos e práticos de regras, juntamente com a vontade de se oferecer um programa que também fosse acessível à pessoas que não tem conhecimento da complexa linguagem do iptables, surgiu o Vuurmuur, que é o objeto principal deste trabalho.

Tentaremos cobrir ao máximo a explicação de seus recursos, bem como instalação e configuração, de forma a montar um ambiente seguro, porém de forma prática, dinâmica e facilitada.

3.1 Conhecendo

Vuurmuur é uma palavra alemã que significa, *que surpresa*, **firewall**. Foi idealizado e criado pelo holandês Victor Julien, que o mantém até hoje, com a contribuição de pessoas da comunidade. É um programa feito em C para o sistema operacional Linux e utiliza a biblioteca *ncurses* para o GUI¹. É um projeto open-source do portal Sourceforge, e é disponibilizado sob a GNU GPL. No momento em que está sendo escrito este trabalho, ele se encontra em sua versão beta 0.5.67, porém são lançadas novas versões constantemente. Todas as informações sobre o Vuurmuur podem ser encontradas em <http://vuurmuur.sourceforge.net/>.

Segundo as próprias descrições do criador, “o Vuurmuur é um *middle-end/front-end*

¹Graphical User Interface, ou ‘Interface Gráfica para o Usuário’

para iptables para o administrador de redes que precisa de um firewall **decente**, mas não possui os conhecimentos específicos do iptables”.

Na verdade, o Vuurmuur, como já foi mencionado, ultrapassou este objetivo, e conquistou também administradores de rede que têm conhecimentos de iptables, mas viram em suas funcionalidades possibilidades de criação (e também alteração) de ambientes de forma simples, rápida e prática.

Alguns recursos do Vuurmuur:

- Novos conceitos para a manipulação de objetos, como a criação de hosts, grupos, redes e zonas
- Pré-definição de serviços/portas de forma fácil
- Criação de regras de forma intuitiva, simples e organizada
- Visualização de logs e pacotes sendo processados em tempo real
- Fácil mudança de regras, necessárias em eventuais casos onde é preciso agir na hora
- Extensa flexibilidade para personalizações
- Disponível atualmente em 5 diferentes línguas².

O Vuurmuur é dividido atualmente em três partes principais (que são também os binários executáveis):

- **vuurmuur_conf**: o *front-end*, onde é feita a administração do firewall, criação de hosts/grupos/redes/zonas, criação de regras, definição de opções, aplicação de alterações, enfim, a maioria das coisas relacionadas diretamente com o iptables;
- **vuurmuur**: o *middle-end*, que se encarrega de processar todas as inclusões e modificações feitas no `vuurmuur_conf`: converter o esquema de regras/serviços/objetos em regras no formato do iptables e aplicá-los; e carregar arquivos auxiliares ao funcionamento do iptables, como módulos adicionais;
- **vuurmuur_log**: o terceiro componente, que faz a conversão do log do iptables para um formato de fácil leitura, que condiz com os objetos criados no `vuurmuur_conf`.

²Inglês, holandês, russo, alemão e português do Brasil, para o módulo de configuração.

Adicionalmente temos também, ainda em fase de desenvolvimento, o utilitário chamado **vuurmuur_script**, feito para possibilitar alterações através da linha de comando, podendo assim ser usado para automatizar tarefas em outros programas.

3.2 Entendendo

Antes que se possa iniciar o uso do Vuurmuur, é necessário que se entenda sua filosofia, podendo assim aproveitar todos seus recursos, afim de evitar confusões em seus conceitos e na criação de regras, que podem facilmente acontecer caso não se projete seu cenário antes da criação de regras.

O Vuurmuur trabalha com 6 objetos, que são a base para a criação de todas as regras. São eles:

1. Interfaces
2. Serviços
3. Hosts
4. Grupos
5. Redes
6. Zonas.

Antes de explicarmos estes objetos, vale lembrar que, para todos, o nome atribuído à ele na hora da criação, será o mesmo referenciado para a criação da regra. Portanto, é sempre interessante a idéia de se utilizarem nomes intuitivos.

3.2.1 Interfaces

Interfaces: são as interfaces de rede propriamente ditas do computador, as interfaces físicas. Exemplo: eth1, ppp0, tun0, eth0:0.

Note que foi citada uma interface virtual no exemplo, a eth0:0. O Vuurmuur aceita normalmente interfaces virtuais, porém, na hora de sua nomeação, não poderá ser usado o ":" no nome. Veremos isso na montagem do cenário-exemplo.

O Vuurmuur também aceita o uso de interfaces em que o IP é dinâmico; neste caso, o vuurmuur monitorará a interface em um tempo pré-especificado de tempo para possíveis

mudanças de endereço. Um exemplo bastante prático para esta opção são as máquinas que recebem um IP válido dinâmico do modem ADSL.

3.2.2 Serviços

Serviços podem ser daemons específicos rodando em sua própria máquina, como sshd, apache, samba, ou também serviços que estão disponíveis fora de sua rede, como IRC, cvs, jabber, etc.

Serviços nada mais são do que definições de portas e protocolos utilizados para determinado fim. Sendo assim, para cada porta utilizada, esta deverá ter um serviço associado à ela.

3.2.3 Hosts

Hosts são os computadores da rede, ou cada dispositivo que tenha um endereço IP, como impressoras, switches, roteadores. Na configuração dos hosts, informamos apenas um nome, seu endereço IP e, opcionalmente, seu endereço MAC. Com o uso do endereço MAC, aumentamos o nível da segurança, dificultando IP Spoofings.

3.2.4 Grupos

Grupos são uma mera forma de referenciar vários hosts de uma vez. Em sua configuração, apenas informamos seu nome, e seus membros.

Vale a observação de que um host poderá estar em mais de um grupo³.

3.2.5 Redes

Aqui as coisas começam a mudar em relação à forma de agir no iptables.

Na definição de uma rede, você deverá informar o endereço de rede propriamente dito, a máscara de rede e ao menos uma interface. É aqui que está o segredo do Vuurmuur para o controle: a adição de hosts é feito sob alguma rede, ou seja, **todo** host só pode ser cadastrado como **pertencendo** à determinada rede, e então é verificado se o endereço do

³Caso isto aconteça, é importante que se tenha maior atenção na criação das regras, para que não se bloqueie e autorize o mesmo host.

host é compatível com o endereço da rede, que por sua vez, deve ser compatível com o endereço da Interface.

Exemplo: Você adiciona uma rede, `lan`, por exemplo, que tem o endereço `192.168.1.0` e máscara `255.255.255.0`. Você somente poderá atachar à ela uma interface com um endereço que esteja dentro deste intervalo. Da mesma forma, na hora de adicionar os hosts, só serão permitidos hosts que façam parte da rede `192.168.1.0/255.255.255.0`.

Veremos que os endereços fora da sua rede e a Internet formam um caso à parte. Neste caso, a Rede ‘internet’, por exemplo, tem o endereço `0.0.0.0`. Sendo assim, será possível cadastrar qualquer endereço da Internet como sendo um host.

3.2.6 Zonas

Zonas são como recipientes globais para redes. Da mesma forma que os hosts, toda Rede deve ser cadastrada como sendo pertencente à uma Zona. Obviamente, poderemos ter várias Redes dentro de uma mesma zona. Exemplo: em uma empresa, existem 3 segmentos de rede diferentes: uma para os servidores de email, outra para os servidores de arquivo, e uma outra para os usuários. Sendo assim, seria conveniente que você tivesse 2 zonas: uma Zona ‘servidores’ contendo as redes ‘mailservers’ e ‘fileservers’, por exemplo, e uma segunda zona, ‘lan’.

Note que estas combinações de grupos, redes e zonas dão uma enorme flexibilidade ao administrador de redes para a organização de seus hosts, segundo seus ilimitados critérios de separação. Mais à frente, veremos como isso se torna útil e importante no projeto da elaboração de um firewall bem configurado.

3.2.7 Juntando tudo isto

Para que não fiquem dúvidas à respeito da filosofia de objetos do Vuurmuur, vamos analisar o seguinte exemplo, – simples, mas suficiente para que se entenda melhor a idéia – adaptado da própria página do Vuurmuur (3):

Vamos considerar este exemplo da figura 1:

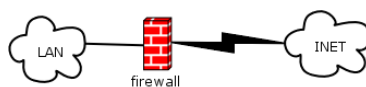


Figura 1: Rede simples

Este é um exemplo simples, onde temos a Internet e a nossa rede separada pelo nosso firewall.

Iremos, agora, definir duas Zonas: ‘int’ e ‘ext’, que poderiam significar ‘interno’ e ‘externo’, respectivamente. Note na figura 2 a representação através das linhas vermelhas. Como mencionado na subseção 3.2.6 (p.29), observe que elas agem como grandes recipientes, sendo a ‘maior entidade’ de nossos objetos:

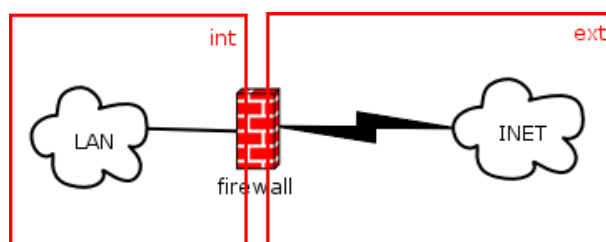


Figura 2: Rede simples com Zonas

Vamos adicionar duas redes, uma em cada zona: iremos chamar estas redes de ‘lan’ e ‘inet’, e serão representadas pelas linhas laranja:

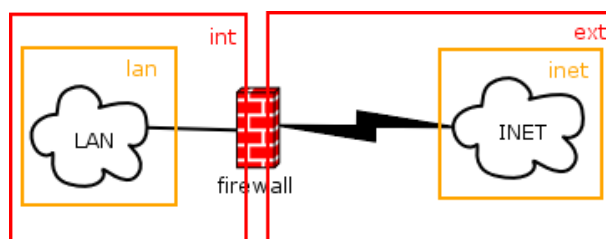


Figura 3: Rede simples com Zonas e Redes

Temos agora, a Rede ‘lan’ dentro da Zona ‘int’, e a rede ‘inet’ dentro da Zona ‘ext’. Podemos imaginar como “minha rede local dentro da zona interna, e a ‘rede’ Internet dentro da zona externa”.

Atenção: é de primordial importância que se entenda a nomenclatura introduzida a partir deste ponto, pois será assim que iremos referenciar os objetos.

O Vuurmuur mostra estes objetos no formato *rede.zona*, portanto, em nosso exemplo, ele seria referenciado como `lan.int` e `inet.ext`. Fácil.

Para exemplificarmos o host, vamos adicionar o ‘Server’ ao nosso primeiro exemplo, como pode ser conferido na figura 4. Na figura 5 podemos ver como ficaria nosso cenário,

com um host, redes e zonas.

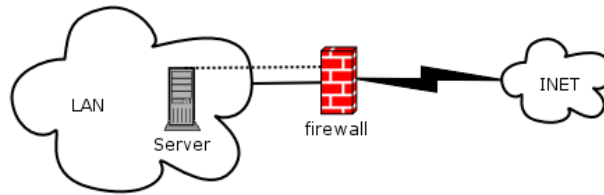


Figura 4: Rede simples com Host

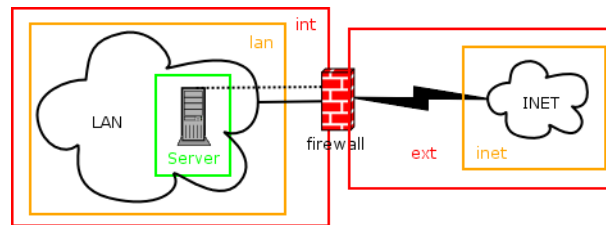


Figura 5: Rede simples com Host, Redes e Zonas

Ficaria a referência, portanto, como `server.lan.int`. É realmente intuitivo este método de referência, e é assim que o Vuurmuur irá mostrar os objetos na hora da criação de regras. Novamente, fica a observação do motivo para o uso de nomes intuitivos.

3.2.8 Pequenos detalhes

Vale observar aqui pequenos detalhes que podem intrigar a cabeça do usuário:

- IP's vs. Nomes: Note que, em todos os exemplos citados até agora, sempre nos referenciamos à IP's, e nunca à nomes. No Vuurmuur, não é possível, por exemplo, cadastrar um host externo como sendo `'www.terra.com.br'`. Ao invés disto, deve-se observar qual é o IP que responde por este endereço, e cadastrá-lo no devido lugar. A explicação para isto é que recomenda-se nunca usar nomes no lugar de endereços IP, devido à grande facilidade de se fazer spoof de DNS.
- Uma questão óbvia, mas que vale a pena ser lembrada: na seção de montagem do cenário-exemplo, teremos a oportunidade de perceber que, no Vuurmuur, temos de autorizar explicitamente **tudo** que poderá entrar/sair/atravessar o firewall. Isto se deve ao fato de o Vuurmuur implementar uma política **séria** de firewall, portanto,

todas as suas políticas padrão⁴ são definidas para DROP, sendo assim necessário fazer a especificação de todos os serviços autorizados.

- Antes de sair cadastrando regras, é importante que se entenda os exemplos da subseção 3.2.7, na página 29, e que se monte uma estrutura organizada e definida. Desta forma, o trabalho na hora da elaboração das regras ficará muito mais simples e intuitivo.

3.3 Instalando

A partir de agora, se dá o início do que poderíamos chamar de ‘mão-na-massa’. Encerra-se o referencial teórico, e passaremos a apresentar, nesta e nas próximas duas seções, *screenshots* e comentários diretamente sobre as opções do programa em si.

3.3.1 A partir do código fonte

Nesta forma de distribuição, o Vuurmuur vem compactado utilizando os utilitários *GNU tar* e *gzip*, resultando no arquivo `Vuurmuur-0.5.67.tar.gz`.

Para descompactarmos o código-fonte do Vuurmuur, podemos utilizar o seguinte comando:

```
# tar xvzf Vuurmuur-0.5.67.tar.gz
```

Teremos algo assim:

```
root@lackware:~# tar xvzf Vuurmuur-0.5.67.tar.gz
Vuurmuur-0.5.67/
Vuurmuur-0.5.67/zones/
Vuurmuur-0.5.67/zones/dnsm/
Vuurmuur-0.5.67/zones/dnsm/zone.config
Vuurmuur-0.5.67/services/
Vuurmuur-0.5.67/services/cvs
Vuurmuur-0.5.67/services/dns
Vuurmuur-0.5.67/services/ftp
Vuurmuur-0.5.67/services/irc
Vuurmuur-0.5.67/services/mem
Vuurmuur-0.5.67/services/ntp
Vuurmuur-0.5.67/services/rdp
Vuurmuur-0.5.67/services/rsh
Vuurmuur-0.5.67/services/svn
Vuurmuur-0.5.67/services/htp
Vuurmuur-0.5.67/services/imap
Vuurmuur-0.5.67/services/ldap
Vuurmuur-0.5.67/services/lisa
Vuurmuur-0.5.67/services/news
Vuurmuur-0.5.67/services/ping
Vuurmuur-0.5.67/services/pop3
Vuurmuur-0.5.67/services/pppt
Vuurmuur-0.5.67/services/sftp
Vuurmuur-0.5.67/services/supp
Vuurmuur-0.5.67/services/ident
Vuurmuur-0.5.67/services/htps
Vuurmuur-0.5.67/services/imap5
Vuurmuur-0.5.67/services/mssql
Vuurmuur-0.5.67/services/pops
Vuurmuur-0.5.67/services/raster
Vuurmuur-0.5.67/services/samba
Vuurmuur-0.5.67/services/rsync
Vuurmuur-0.5.67/services/socks
Vuurmuur-0.5.67/services/ssh
Vuurmuur-0.5.67/services/squid-proxy
Vuurmuur-0.5.67/services/traceroute
Vuurmuur-0.5.67/services/ai rdoscedia
Vuurmuur-0.5.67/services/jabber
Vuurmuur-0.5.67/services/pcanywhere
Vuurmuur-0.5.67/services/telnet
Vuurmuur-0.5.67/services/syslog
Vuurmuur-0.5.67/services/sshwin
Vuurmuur-0.5.67/services/sshwin
Vuurmuur-0.5.67/INSTALL.BEHH
Vuurmuur-0.5.67/INSTALL
Vuurmuur-0.5.67/ChangeLog
root@lackware:~#
```

Figura 6: Descompactando o código-fonte

⁴Default Policies.

Na instalação com o código-fonte, o Vuurmuur vem com um utilitário de instalação chamado `install.sh`. Vejamos quais são suas opções, digitando `# ./install.sh --help`:

```

root@slackware:~# tar xzvf Vuurmuur-0.5.67.tar.gz
Vuurmuur-0.5.67/
Vuurmuur-0.5.67/zones/
Vuurmuur-0.5.67/zones/dmz/
Vuurmuur-0.5.67/zones/dmz/zone.config
Vuurmuur-0.5.67/services/inaps
Vuurmuur-0.5.67/services/mysql
Vuurmuur-0.5.67/services/pop3s
Vuurmuur-0.5.67/services/razor
Vuurmuur-0.5.67/services/samba
Vuurmuur-0.5.67/services/rsync
Vuurmuur-0.5.67/services/socks
Vuurmuur-0.5.67/services/whois
Vuurmuur-0.5.67/services/squid-proxy
Vuurmuur-0.5.67/services/traceroute
Vuurmuur-0.5.67/services/windowsmedia
Vuurmuur-0.5.67/services/jabber
Vuurmuur-0.5.67/services/pcanywhere
Vuurmuur-0.5.67/services/telnet
Vuurmuur-0.5.67/services/syslog
Vuurmuur-0.5.67/services/usermin
Vuurmuur-0.5.67/services/webmin
Vuurmuur-0.5.67/INSTALL.DEBIAN
Vuurmuur-0.5.67/INSTALL
Vuurmuur-0.5.67/ChangeLog
root@slackware:~# cd Vuurmuur-0.5.67
root@slackware:~/Vuurmuur-0.5.67# ./install.sh --help

Help for Vuurmuur-installer.

Commandline options:

Main options:

    --install      install and setup up the config
    --upgrade      install without touching the config
    --uninstall    uninstall but leave the config alone
    --unpack       unpack the archives

Sub options:

    --defaults     use the default values for all questions
    --debug        print some extra info for debugging the install script
    --noupack      don't unpack, use the already unpacked archives

Please read INSTALL for more information.
root@slackware:~/Vuurmuur-0.5.67# █

```

Figura 7: Opções do `install.sh`

Vamos analisar qual é a função de cada opção:

- Main Options: São as opções principais, é obrigatório informar uma delas (e apenas uma.)

`--install`: Esta opção dará início à instalação interativa, e irá perguntar os caminhos em que o usuário deseja instalar os arquivos.

`--upgrade`: Esta opção é utilizada quando estamos fazendo atualização de versões. O setup do Vuurmuur irá instalar todos os arquivos, com exceção dos arquivos de configuração. Após a atualização, o Vuurmuur estará pronto para rodar a versão nova, e já interpretará as regras antigas.

`--uninstall`: Mesmo caso da opção acima, porém utilizado para remover o programa. Todos os arquivos serão removidos, com exceção dos arquivos de

configuração, que permanecerão no local onde estão, para caso seja necessário usá-los posteriormente, ou caso o administrador deseje efetuar um backup disto.

`--unpack`: Esta opção é destinada a quem deseja efetuar a compilação dos pacotes na mão. Após a descompactação do arquivo `Vuurmuur-0.5.67.tar.gz`, são criados mais 3 arquivos com a mesma extensão, são eles:

```
libvuurmuur-0.5.67.tar.gz vuurmuur-0.5.67.tar.gz
vuurmuur_conf-0.5.67.tar.gz
```

Estes arquivos poderiam ser usados para a compilação manual. Em nosso caso, iremos utilizar a facilidade do instalador.

- Sub Options: São as opções auxiliares. Nenhuma delas é obrigatória. Porém, para utilizarmos uma dessas funções, devemos utilizar juntamente alguma das Opções Principais:

`--defaults`: Esta opção fará com que o programa não pergunte pelos locais desejados de instalação. Ao invés disto, ele irá adotar os seguintes locais como padrão: `/etc/vuurmuur` para os arquivos de configuração, `/usr/bin` para os arquivos binários, `/usr/share/vuurmuur` para arquivos diversos e `/var/log/vuurmuur` para os logs.

`--debug`: Esta opção é utilizada em caso de problemas na instalação. Quando ativada, gerará mais detalhes e informações, para o log de instalação.

`--nounpack`: Esta opção diz ao programa para não descompactar os três arquivos citados acima; ao invés disso, utilizar os que já estão descompactados na pasta. É óbvio que deverão existir estes arquivos para que isto funcione.

Conhecidas as devidas opções, iremos iniciar a instalação, utilizando o seguinte comando:

```
# ./install.sh --install --defaults: Isto informa ao instalador para instalar o
Vuurmuur, e não perguntar pelos locais de instalação, utilizando os locais padrão.
```

A instalação se inicia. Dependendo da velocidade do seu computador, a instalação demorará aproximadamente de 5 a 10 minutos. Em casos normais, geralmente o básico do sistema Linux instalado já fornece todas os requisitos do Vuurmuur. Portanto, não costumam ocorrer problemas aqui. Durante a instalação, ele irá informando sobre o status da instalação, como na figura 8:

```

root@slackware:~# tar xuzf Uuurmuur-0.5.67.tar.gz
Uuurmuur-0.5.67/
Uuurmuur-0.5.67/zones/
Uuurmuur-0.5.67/zones/dnz/
Uuurmuur-0.5.67/zones/dnz/zone.config

Help for Uuurmuur-installer.

Commandline options:

Main options:
    --install      install and setup up the config
    --upgrade      install without touching the config
    --uninstall    uninstall but leave the config alone
    --unpack       unpack the archives

Sub options:
    --defaults     use the default values for all questions
    --debug        print some extra info for debugging the install script
    --noupack      don't unpack, use the already unpacked archives

Please read INSTALL for more information.

root@slackware:~/Uuurmuur-0.5.67# ./install.sh --install --defaults

Uuurmuur installation
=====

Welcome to the installation of Uuurmuur. First you will be
asked a couple of questions about the location to install the
various parts of Uuurmuur. It is recommended that you choose
the defaults, by pressing just enter.

Installdir: /usr ...
Using Shareddir: /usr/share/uuurmuur ...
Using Etcdir: /etc/uuurmuur'.
Using Logdir: /var/log/uuurmuur/'.

Ok, thank you. Going to build Uuurmuur now. Depending on your hardware
this process will take about 2 to 10 minutes.

Testing for the installation files...
Going to extract the files...
Extracting the files done...
Going to build libuurmuur... (common code for all parts of Uuurmuur).
█

```

Figura 8: Instalação em andamento

Após o término da instalação, podemos verificar os binários que foram criados:

```

root@slackware:~# cd /usr/bin/
root@slackware:~/usr/bin# ls -l uuurmuur*
-rwxr-xr-x 1 root root 334375 2005-07-12 20:32 uuurmuur*
-rwxr-xr-x 1 root root 977683 2005-07-12 20:42 uuurmuur_conf*
-rwxr-xr-x 1 root root 115944 2005-07-12 20:32 uuurmuur_log*
-rwxr-xr-x 1 root root 65639 2005-07-12 20:32 uuurmuur_script*
root@slackware:~/usr/bin# █

```

Figura 9: Binários do Uuurmuur

O Uuurmuur fornece, junto com sua instalação, um script de inicialização, que poderá

ser usado para iniciar o Vuurmuur durante o boot. Este script está formatado para o estilo ‘Red Hat like’, porém pode ser usado em qualquer distribuição, como o SLACKWARE, com poucas adaptações.

Se houver sido utilizado o caminho padrão durante a instalação, o script estará em `/usr/share/vuurmuur/scripts/vuurmuur-initd.sh`.

Com isso, terminamos o processo de instalação e sua abordagem a partir do código fonte. A partir da próxima seção, tudo o que se for abordado já faz parte da configuração do Vuurmuur em si, em seus diversos módulos.

3.3.2 Autopackage e binários diversos

Como citado na subseção 1.1.2 (p.17), a abordagem de instalação neste trabalho é a feita a partir do código-fonte, e não iremos entrar em detalhes específicos sobre procedimentos de instalação de pacotes binários de distribuições específicas, apesar de o site do Vuurmuur manter pacotes para algumas destas distribuições em seus releases.

Nossa justificativa para tal é a de que a instalação a partir do código-fonte é extensivamente mais abrangente, uma vez que, teoricamente, deverá funcionar em qualquer distribuição Linux, salvo restrições específicas de distribuições.

Entretanto, gostaríamos de citar brevemente uma nova forma em que o Vuurmuur começou a ser distribuído: o Autopackage.

Autopackage é um formato novo que tem, entre outras intenções, prover um pacote que seja independente de distribuições, adaptando-se assim ao ambiente em que estiver sendo executado. É um projeto que está começando a se popularizar agora, e o Vuurmuur já está testando, a nível de versões *alpha*, o *release* de versões utilizando esta forma de empacotamento. Para a próxima versão do Vuurmuur, é provável que já se tenha este suporte de forma oficial.

Maiores informações sobre o Autopackage poderão ser encontradas em <http://www.autopackage.org/>

3.4 Conhecendo menus e opções

Nesta seção, iremos abordar as diversas opções de todos os binários do Vuurmuur. A maior parte estará localizada justamente no módulo `vuurmuur_conf`, a nossa ‘interface gráfica’. Entretanto, veremos úteis opções nos outros módulos também, que poderão ser usados para as mais diversas finalidades, desde ajustes finos até resolução de problemas.

3.4.1 `vuurmuur`

Este é o módulo ‘*core*’⁵ do Vuurmuur. É ele o responsável pela conversão de opções informadas no `vuurmuur_conf` para o formato nativo do iptables, bem como o manuseio de tarefas indiretas ao mesmo. Algumas funções do módulo `vuurmuur`:

- Converter as regras do `vuurmuur_conf` para o formato do iptables, e injetá-las no próprio iptables;
- Em caso de mudanças nas regras, quando clicamos em ‘Aply’ (Aplicar), será ele quem fará também a atualização destas regras no iptables;
- Verificar mudanças em interfaces dinâmicas, para readequar as regras, de acordo com o novo endereço adquirido;
- Carregar módulos adicionais do iptables (caso esta opção esteja ativada).

Esta e outras tarefas são realizadas por este módulo. Na inicialização do `vuurmuur_conf`, se o *daemon* `vuurmuur` não estiver rodando, é informado um status de erro, e as regras não poderão ser aplicadas. Portanto, este módulo deverá estar sempre carregado antes de entrarmos no `vuurmuur_conf`⁶, pois só assim seremos capazes de por em prática todas as configurações feitas lá dentro.

A imagem 10 nos mostra as opções do `vuurmuur`, obtidas com o seguinte comando:

```
# vuurmuur -h
```

⁵Módulo principal, módulo-base.

⁶Na primeira vez em que for executar o Vuurmuur, é aconselhável que se rode primeiramente o `vuurmuur_conf`, definam as regras, e só depois se inicie o `vuurmuur`. Caso isto seja feito em ordem inversa, seu firewall não permitirá mais nenhuma conexão com ele, até que se definam as regras.

```
root@slackware:~# vuurmuur -h
Usage: vuurmuur [OPTION]

Options:
-b gives a bashscript output
-d [1 - 3] enables debugging, 1 low, 3 high
-h gives this help
-U gives the version
-D vuurmuur starts and goes into daemon-mode.
-L specify the loglevel for use with syslog.
-v verbose mode.
-n for use with -D, it goes into the loop without daemonizing.
-C clear all iptables rules and set policy to ACCEPT. Use with care!
-t don't check for iptables capabilities, assume all are supported.

root@slackware:~#
```

Figura 10: Opções do daemon vuurmuur

Vamos às opções:

`vuurmmur -b`: Se este parâmetro for especificado, o `vuurmuur`, ao invés de injetar as regras lidas no `vuurmuur_conf` e convertidas para o iptables, irá mostrar na tela os comandos que utilizaria para fazer isso, imprimindo o *bashscript* que seria executado para tal tarefa. É bastante útil para quem tem curiosidade em saber como o Vuurmuur forma sua lista de regras (que é, por sinal, bastante extensa e complexa, mesmo para os casos mais simples).

`vuurmuur -d [1 - 3]`: Isto habilita e define o nível de informação que será gravado no arquivo `debug.log`. Quanto mais alto o número, mais detalhado será o log. Isto é extremamente útil para resolução de problemas, onde você começa definindo um número baixo, e vai aumentando, até que apareça sua mensagem de erro.

`vuurmuur -h`: Exibe o conteúdo da figura 10, a que estamos analisando neste momento.

`vuurmuur -V`: Exibe a versão do programa.

`vuurmuur -D`: Inicia o `vuurmuur`, e fica em modo *daemon*. Isto quer dizer que ele fará todo seu trabalho junto ao iptables, mas continuará executando em *background*, aguardando pela comunicação do `vuurmuur_conf`. Normalmente, esta será sempre a opção utilizada no *startup*; desta forma, poderemos fazer nossas alterações no `vuurmuur_conf`, e aplicá-las sem mesmo ter de sair do programa.⁷

`vuurmuur -L`: Especifica o nível do log para ser usado nas mensagens enviadas ao `syslog`.⁸

`vuurmuur -v`: Em condições normais, o `vuurmuur` age silenciosamente, gravando apenas suas mensagens nos arquivos de log. Caso esta opção seja especificada, o `vuurmuur` imprimirá também na tela as mensagens de início.

`vuurmuur -n`: Como citado, é utilizado para usar juntamente com o parâmetro ‘- D’. Neste caso, o `vuurmuur` faz o que tem de fazer e entra em loop, mas não passa ao modo *daemon*, e também não vai para *background*, continuando sua execução em primeiro plano.

`vuurmuur -C`: Este comando é utilizado para zerar todas as regras do iptables, e deixá-lo como se não houvesse sido carregado regra alguma para ele. Adicionalmente, seta todas as *default policies* para ACCEPT. Cuidado com este comando, ele deixará seu firewall completamente exposto para qualquer tipo de conexão.

`vuurmuur -t`: Como citado, uma das funções do `vuurmuur` é carregar os módulos adicionais necessários para as regras. Mesmo que esta opção esteja ativada no `vuurmuur_conf`, pode-se mandá-lo ignorar isto com este parâmetro. Desta forma, tentará injetar as regras no iptables, mesmo que este possa não suportar parte delas. Use somente se souber o que está fazendo.

⁷Se o `vuurmuur` for iniciado sem nenhum parâmetro, ele irá simplesmente fazer a conversão das regras, injetá-las no iptables, e finalizar. Seria equivalente ao nosso script citado na subseção 2.2.2.2.

⁸Para maiores informações, consulte `man syslog`.

3.4.2 `vuurmuur_log`

O `vuurmuur_log` é o módulo mais simples do Vuurmuur. Ele apenas lê os logs do iptables (por padrão, em `/var/log/messages`), e os converte para o formato do Vuurmuur. Nesta conversão, o `vuurmuur_log` usará todos os nomes de objetos definidos no `vuurmuur_conf`.

O log gerado por este módulo será muito útil, e amplamente utilizado no `vuurmuur_conf`, para o acompanhamento de logs (inclusive em tempo real). O formato dos logs gerados pelo Vuurmuur é extremamente fácil de se entender e visualizar, pois segue o mesmo padrão ‘humano-intuitivo’ usado na criação de regras. Para iniciá-lo, apenas digite:

```
# vuurmuur_log
```

Obs: o `vuurmuur_log` deverá ser iniciado no processo de *startup* juntamente com o `vuurmuur`.

3.4.3 `vuurmuur_script`

O `vuurmuur_script` é o ‘filho mais novo’ dos módulos do Vuurmuur, e ainda está apenas em fase de testes, inclusive com uma quantidade significativa de bugs. Porém, foi uma das maiores novidades das ultimas versões, e será com certeza, em breve, um recurso muito utilizado pelos administradores de rede para a automatização de funções.

O `vuurmuur_script` é um programa para a linha de comando que têm como função interagir nas configurações do `vuurmuur_conf`, porém sem a necessidade de estar dentro dele. A vantagem disto é que você poderá incluir, por exemplo, comandos em seus próprios programas que façam as alterações ou inclusões necessárias no Vuurmuur e, quando você entrar no `vuurmuur_conf`, encontrará tudo da forma como foi alterado.

Como exemplo, poderíamos citar um script que teria a função de verificar de tempo em tempo as máquinas ativas da rede(através de um simples ping, por exemplo), compará-las com a lista do Vuurmuur e, se fosse necessário, adicionar as máquinas que não tem regras associadas à ela. Isto seria útil, por exemplo, para manter atualizada uma rede onde saem e entram muitos clientes. Este é apenas um exemplo simples. As possibilidades de uso para o `vuurmuur_script` são infinitas, só dependerá dos recursos do programa para o qual ele estiver fazendo interface.

O `vuurmuur_script` trabalha com alguns códigos de retorno, úteis para sabermos se nosso comando foi concluído com sucesso ou se houver erro. Esta característica deixa claro a justa intenção de se utilizar chamadas para o `vuurmuur_script` a partir de outros programas ou scripts. Os códigos de retorno implementados até o momento são:

```
0   ok
1   commandline option error
2   command failed
3   object supplied with -n does not exist
4   object supplied with -n already exists
254 internal program error
```

O código 0 indica que o comando foi concluído com sucesso.

O código 1 indica que há um erro na sintaxe informada.

O código 2 indica uma falha não-específica na execução do comando.

O código 3 indica que um dos objetos informados não existe, no caso de se estar tentando fazer alguma ação com ele.

O código 4 indica que um dos objetos informados já existe, no caso de se estar querendo criar um objeto com o mesmo nome.

O código 254 indica um erro interno do programa.

As possíveis ações definidas até agora são:

`-C --create`: Cria um novo objeto.

`-D --delete`: Apaga um objeto.

`-R --rename`: Renomeia um objeto. O novo nome deve ser informado com a opção `--set`.

`-M --modify`: Modifica uma variável de objeto informada com a opção `--variable`. Use também o `--set` para definir o novo valor.

`-L --list`: Lista objetos.

`-P --print`: Imprime o valor de um determinado objeto. Use a opção `--variable` para imprimir apenas uma variável.

Com este *set* de opções, já conseguimos realizar grande parte das opções disponíveis dentro do próprio `vuurmuur_conf`.

Como objetos, teremos os já conhecidos 6 objetos padrão do Vuurmuur, e mais um

sétimo objeto, que é a regra em si:

```
-o <nome> --host <nome>: Nome de host.  
-g <nome> --group <nome>: Nome de grupo.  
-n <nome> --network <nome>: Nome de rede.  
-z <nome> --zone <nome>: Nome de zona.  
-s <nome> --service <nome>: Nome de serviço.  
-i <nome> --interface <nome>: Nome de interface.  
-r <nome> --rule <nome>: Nome da set de regras.(Para uso futuro) Valor padrão: rules
```

Temos duas opções adicionais:

```
-A --append: Adiciona, ao invés de sobrescrever, quando estiver usando a opção --modify.  
-O --overwrite: Sobrescreve, quando estiver usando a opção --modify (default).
```

3.4.4 vuurmuur_conf

Este é o módulo principal do ponto de vista da interação com o administrador de redes. É nele que normalmente o administrador estará a maior parte do tempo em que estiver ‘dedicando tempo’ ao seu firewall.

Um exemplo atual de cenário típico do Vuurmuur seriam o `vuurmuur` e o `vuurmuur_log` rodando em background e, a partir daí, toda a interação do usuário a partir do `vuurmuur_conf`. Veremos que, sem ao menos sair de sua interface, é possível aplicar todas as regras e mudanças feitas no firewall, facilitando assim imensamente o trabalho.

Para acessarmos este módulo, digitamos na linha de comando:

```
# vuurmuur_conf
```

Obs: Para efeitos de exemplo, iremos iniciar o `vuurmuur_conf` sem o `vuurmuur` e o `vuurmuur_log` estarem rodando em background.

É iniciado o programa e, num breve processo de inicialização, são verificados alguns itens, dentre eles a comunicação com o `vuurmuur` e o `vuurmuur_log`. Após o término deste processo, teremos algo semelhante à isto:

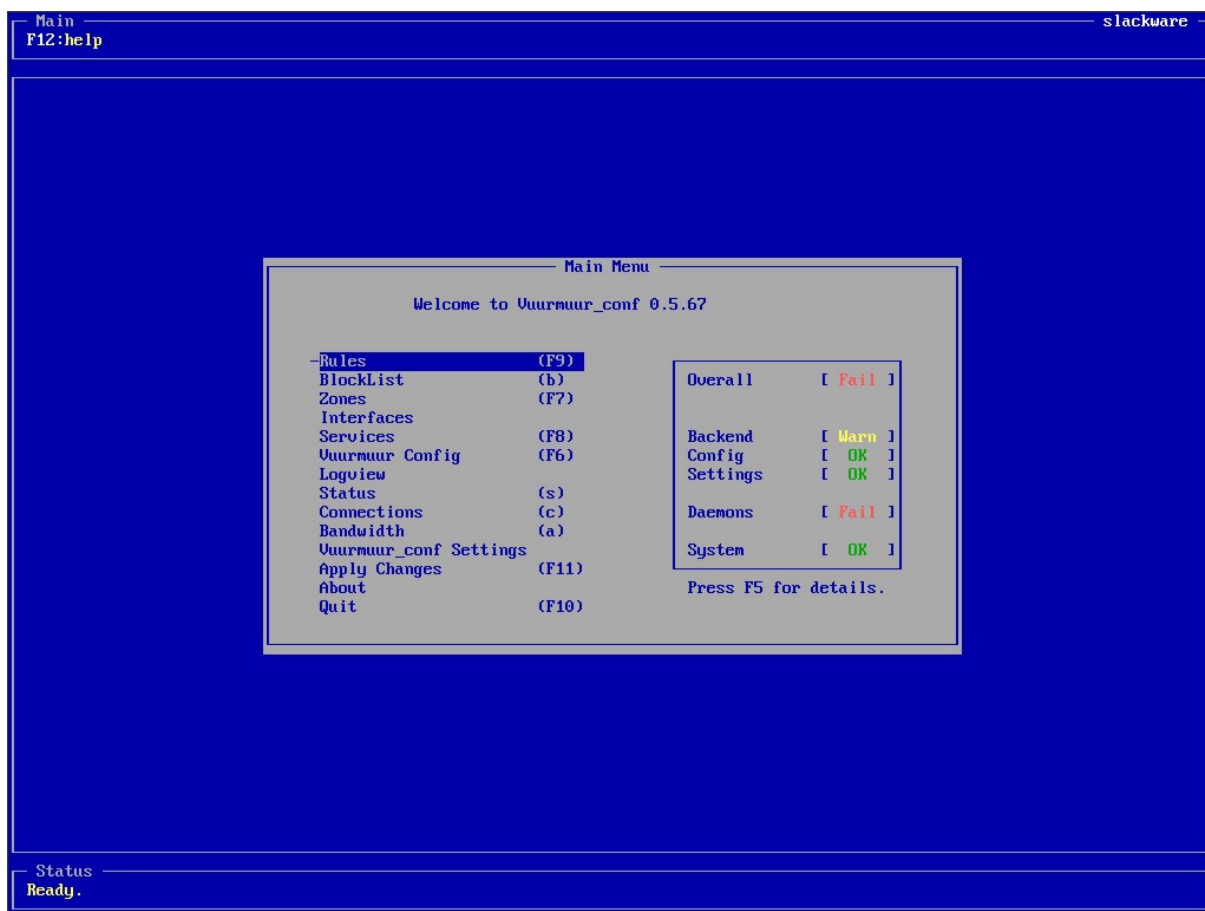


Figura 11: Primeira tela: vuurmuur_conf

Esta é sua ‘cara’; sempre estarão presentes as duas barras de informações: uma na parte superior, e outra na parte inferior. A barra da parte superior nos mostra as opções possíveis para cada menu ou função que estivermos. A barra inferior geralmente nos dá um descritivo de qual a função da opção em que estivermos ‘focados’. Apesar de gráfica, esta interface não faz uso do mouse em momento algum⁹, portanto, todos os comandos, sem exceção, são executados pelo teclado.

Nos próximos *screenshots*, estaremos descrevendo todas as opções encontradas aqui, e suas respectivas finalidades.

No meio, temos a lista vertical de opções e, ao lado dela, temos um quadro de avisos, que nos avisa sobre possíveis erros que tenha verificado. Os avisos mostrados neste quadro podem ser de 3 tipos:

OK: Indica que, para este item, tudo está correndo bem.

⁹Característica comum de aplicações de linha de comando, para facilitar o uso remotamente, através de SSH, por exemplo.

Warn: Algum aviso útil sobre algum item ou configuração que possa ter uma divergência, ou algo que limite determinadas funções do sistema.

Fail: Indica algum item que falhou, e não passou nos testes necessários.

Apertando <F5> temos a lista completa de avisos e erros. Todos os itens desta lista deverão apresentar sempre OK, caso contrário, pode ser necessário descobrir a causa do problema e resolvê-lo, na intenção de não por em risco a segurança e demais serviços dos utilizadores do firewall.

Em nosso caso, temos diversos itens que não estão marcados com OK, e alguns avisos. Vejamos quais são eles:

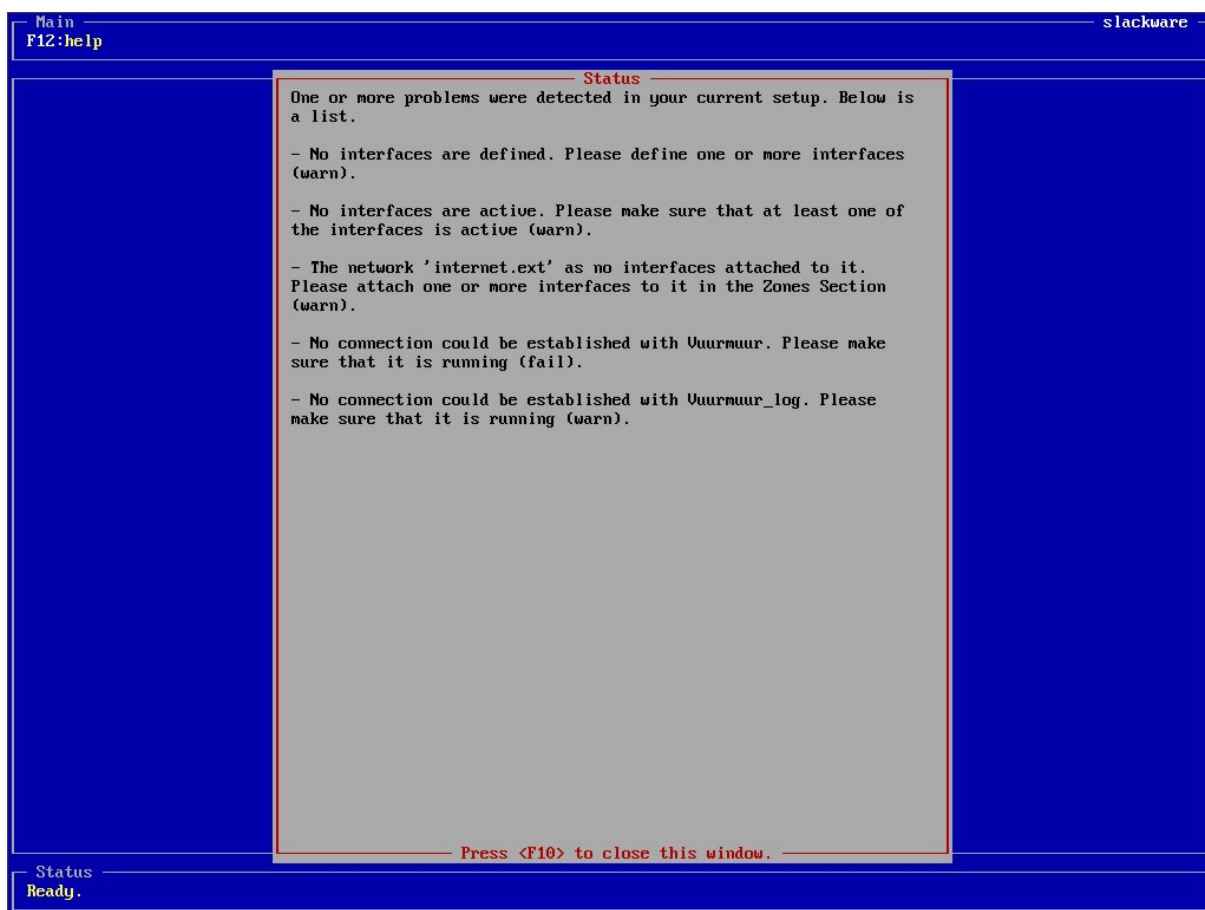


Figura 12: Avisos do vuurmuur_conf

- No primeiro aviso, do tipo WARN, o vuurmuur_conf nos diz que não há interfaces definidas, e pede para definirmos uma ou mais delas.
- O segundo aviso, igualmente do tipo WARN, trata-se também das interfaces, e nos diz que não há interfaces ativas (óbvio, neste caso, pois não existem interfaces definidas).

- O terceiro item (**WARN**) avisa que a rede 'internet.ext'¹⁰ não possui nenhuma interface atachada à ela. Note a nomenclatura apresentada na subseção 3.2.7, na página 29, utilizando o formato *rede.zona*.
- O quarto item trata de uma questão já abordada, e diz que não foi possível estabelecer uma conexão com o **vuurmuur**, e pede para assegurarmos de que ele esteja rodando. É do tipo **FAIL**, e, se ele não estiver rodando, não conseguiremos aplicar as opções escolhidas aqui.
- O último item é semelhante ao anterior, mas avisa sobre o não-sucesso na comunicação com o **vuurmuur_log**. É do tipo **WARN**, porque conseguimos aplicar as regras sem ele, mas ficaremos inaptos a acompanhar os logs em tempo real no formato do **Vuurmuur**.

Vamos conhecer os menus do **Vuurmuur** agora: para voltarmos à tela anterior, a tecla é <F10>. Em todas as partes do **Vuurmuur**, esta é a tecla para sairmos do menu atual e voltarmos para o anterior. Temos também a tecla <F12>, que irá mostrar uma ajuda sobre o item/menu/função focado sempre que disponível.

Estas próximas figuras mostram menus que serão tratados todos na próxima seção (3.5, página 67). Não iremos, portanto, entrar em detalhes sobre eles agora.



Figura 13: Rules: Criação de regras

¹⁰No **Vuurmuur**, 4 zonas já vem previamente cadastradas, juntamente com a rede 'internet', que já vem dentro da zona 'ext'.

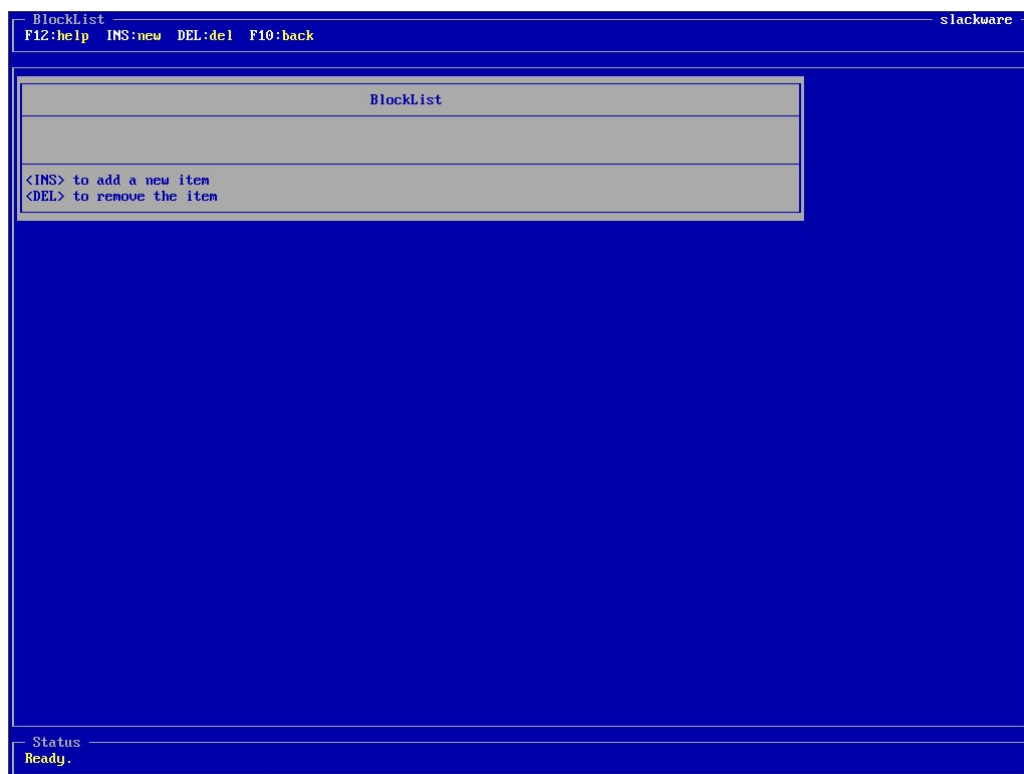


Figura 14: Blocklist: Lista de bloqueados

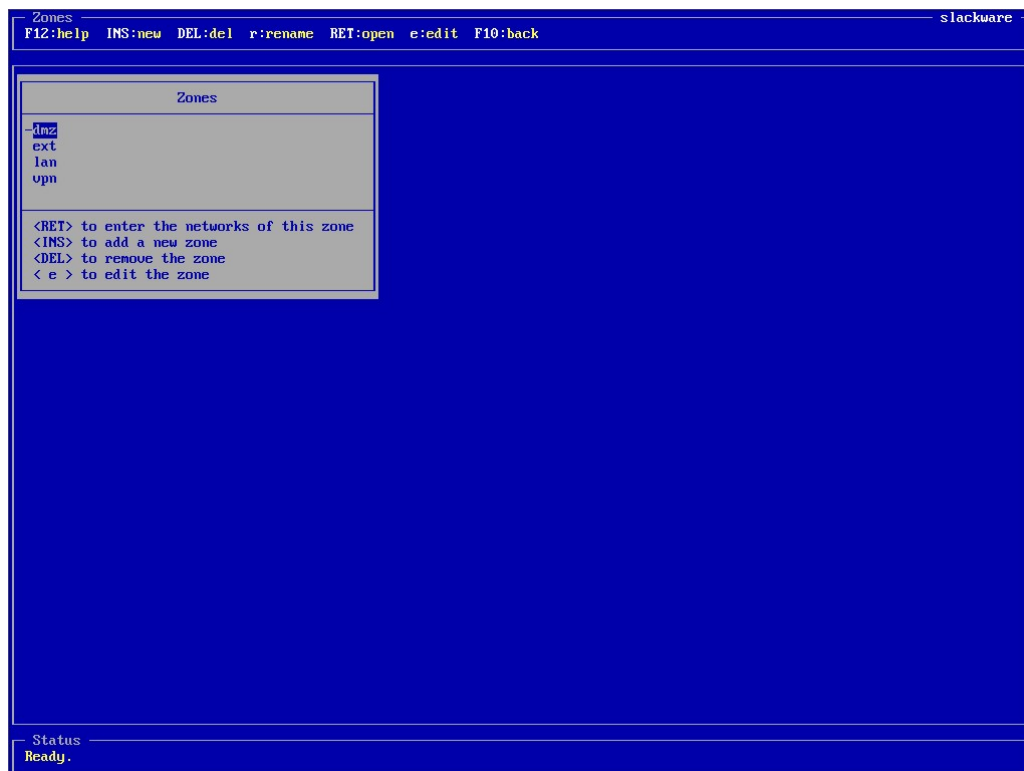


Figura 15: Zones: Criação de zonas, redes, grupos e hosts

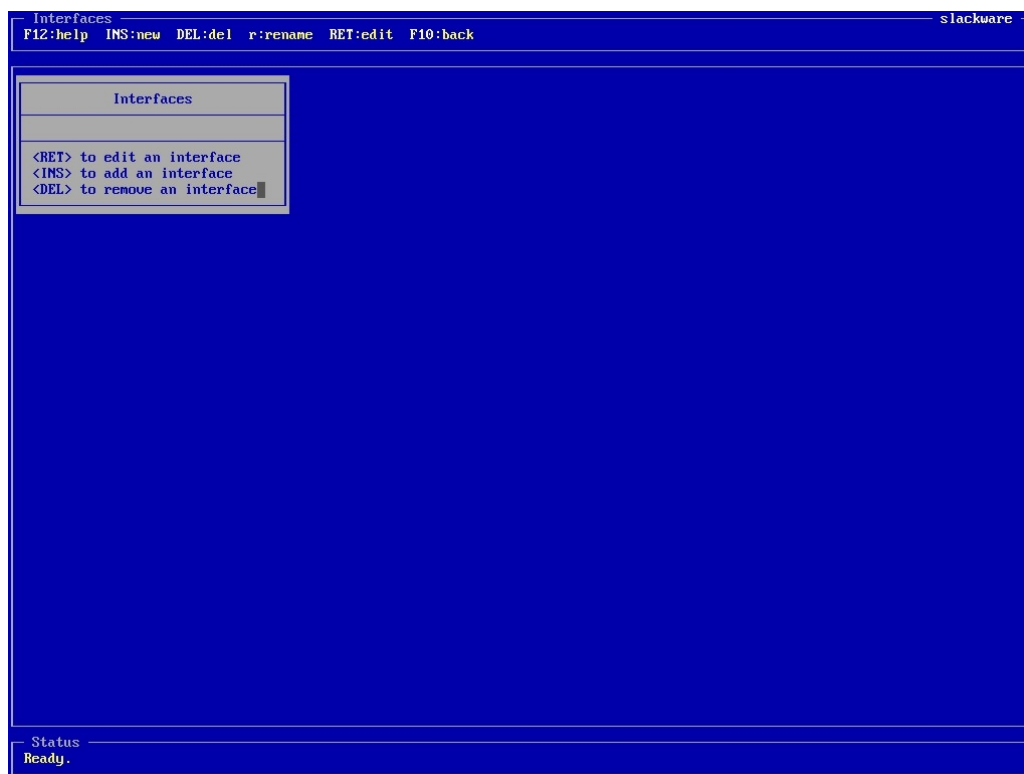


Figura 16: Interfaces: Criação e manipulação de interfaces

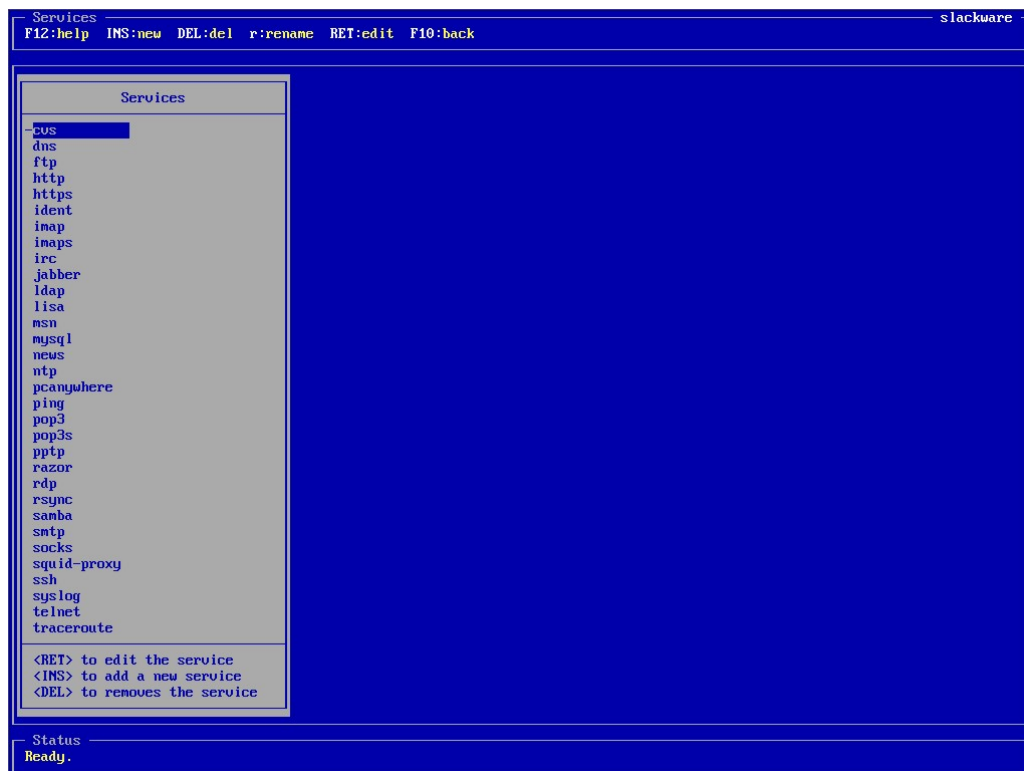


Figura 17: Services: Criação de serviços

3.4.4.1 Vuurmuur Config

O `vuurmuur_conf` tem dois menus distintos de configurações: um para as configurações do Vuurmuur, ou seja, do programa em si, e outro somente para as configurações do `vuurmuur_conf` (onde estamos agora). A figura 18 nos mostra as opções da seção Vuurmuur Config:

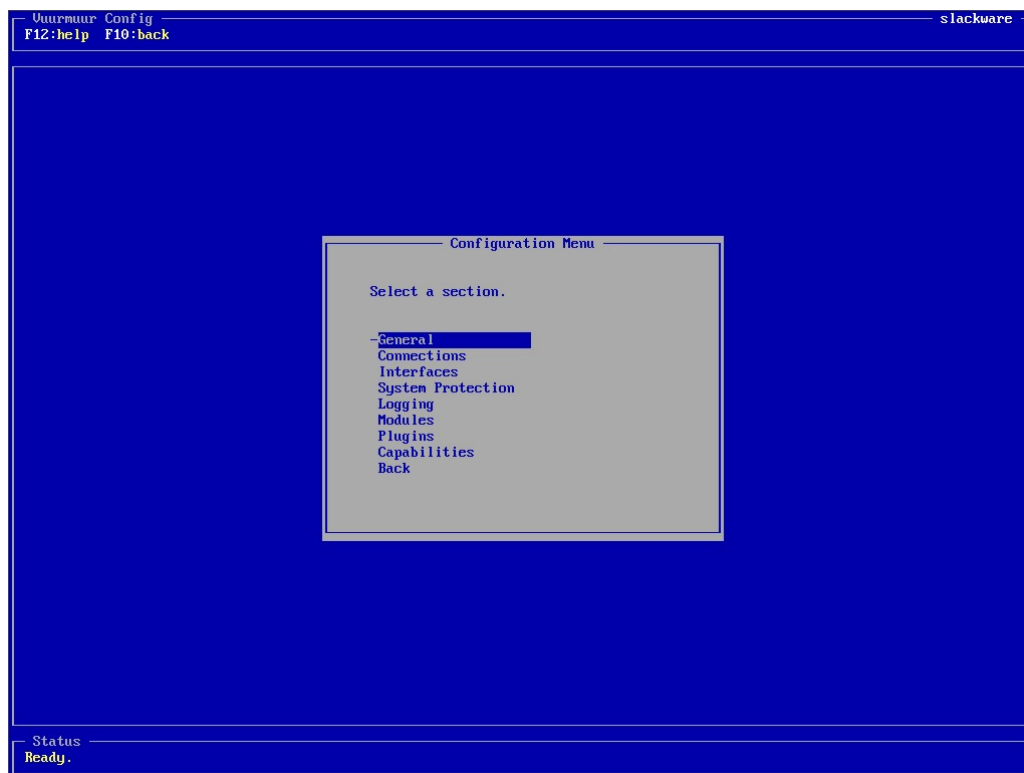


Figura 18: Vuurmuur Config: Opções gerais do Vuurmuur

As opções:

- **General**: Configurações gerais, caminhos do iptables e outros.
- **Connections**: Configurações para limites de conexões, bastante usadas nos scripts manuais de firewall.
- **Interfaces**: Configurações sobre a verificação de interfaces dinâmicas.
- **System Protection**: Algumas proteções...
- **Logging**: Configurações dos logs, caminhos, etc.
- **Modules**: Configurações para o carregamento dos módulos adicionais do iptables, necessários para o funcionamento de algumas funções.

- **Plugins:** O Vuurmuur utiliza plugins para a comunicação e armazenamento de dados. Atualmente só existe um plugin, o `textdir`; portanto é ele que deverá estar nestas configurações.
- **Capabilities:** Neste item, são apontadas os recursos do iptables disponíveis na máquina, de acordo com os módulos carregados ou o suporte compilado no *kernel*.

General

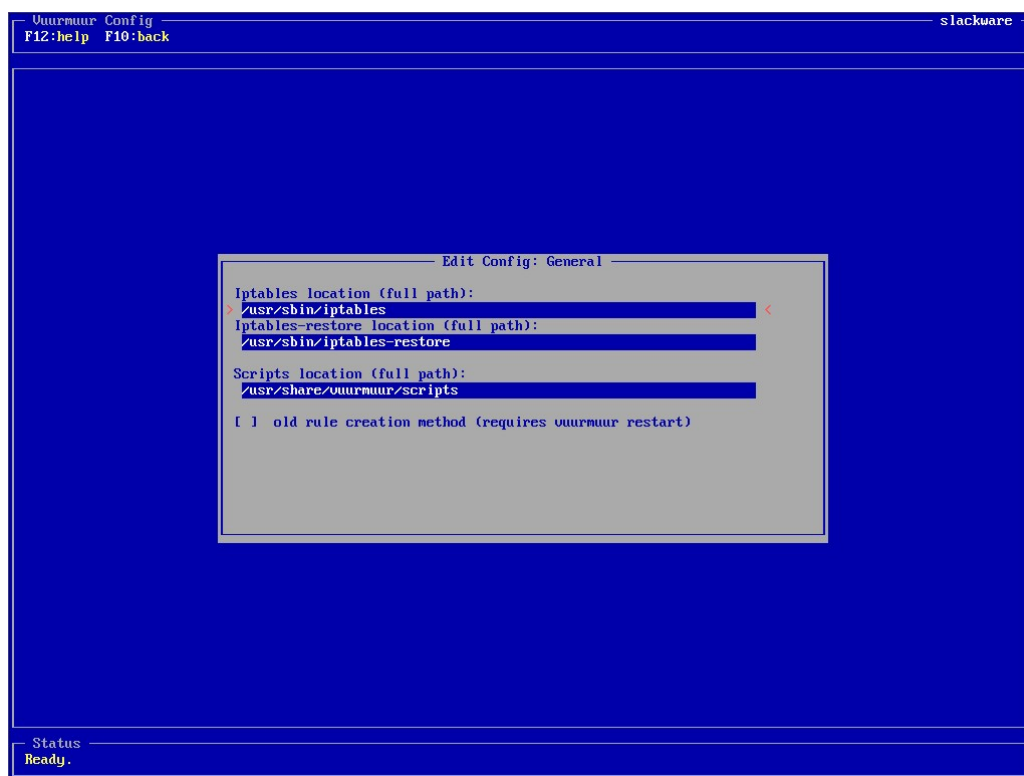


Figura 19: Vuurmuur Config: General

Nesta parte, possíveis configurações que temos são as seguintes:

Iptables location: Aqui deve-se definir o caminho completo para o executável do iptables, usualmente em `/usr/sbin/iptables`.

Iptables-restore location: O mesmo, porém para o iptables-restore. Observe que esta ferramenta tem também sua utilidade com o Vuurmuur ;)

Scripts location: O Vuurmuur utiliza scripts para algumas operações, e aqui deve-se informar o caminho onde estarão armazenados estes scripts.

`Old rule creation method`: A partir da versão 0.5.57, foi introduzida uma nova forma de criação de regras, utilizando justamente o `iptables-restore`. Esta opção deverá ser usada caso surja algum problema com o novo método, ou caso o usuário, por alguma razão, ache-a melhor.

Connections

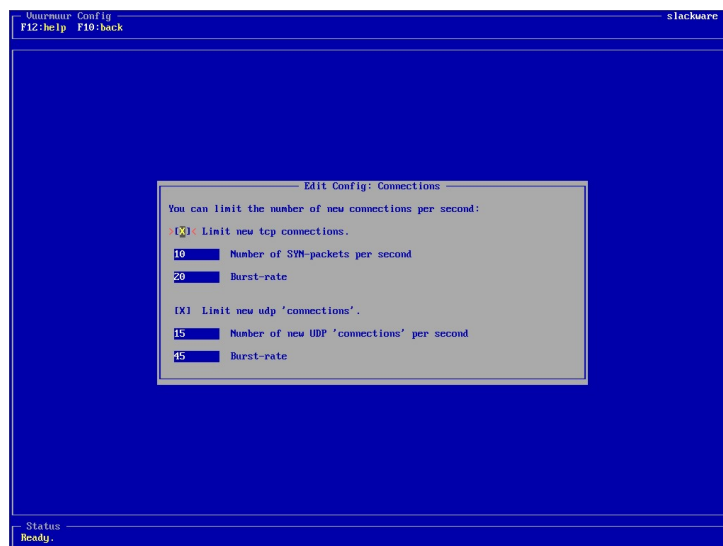


Figura 20: Vuurmuur Config: Connections

Quando usamos scripts com regras para firewall, e também nos scripts prontos que encontramos pela Internet, geralmente existem regras para limitar o número de conexões TCP e UDP por segundo. No Vuurmuur, isto aparece como uma opção, pois a parte de regras se destina estritamente à regras relacionadas aos objetos já explanados.

Você apenas marca para qual tipo de protocolo de conexão você quer seja aplicada a limitação (TCP ou UDP), e coloca os valores para número de pacotes/conexões por segundo, e ‘burst-rate’, para as duas.

Interfaces

Quando temos interfaces com IP’s dinâmicos – interfaces que recebem um IP válido dinâmico da Internet, por exemplo –, determinadas regras podem parar de funcionar caso o IP da interface mude, e as regras não forem atualizadas. O Vuurmuur se encarrega de verificar mudanças em interfaces dinâmicas¹¹, e atualizar as regras que utilizem o endereço desta interface.

¹¹É necessário que a interface esteja marcada como ‘dynamic’ em sua configuração.

Aqui, apenas informamos se queremos que o Vuurmuur cheque atualizações nestas interfaces, e o intervalo que esta checagem deve acontecer.

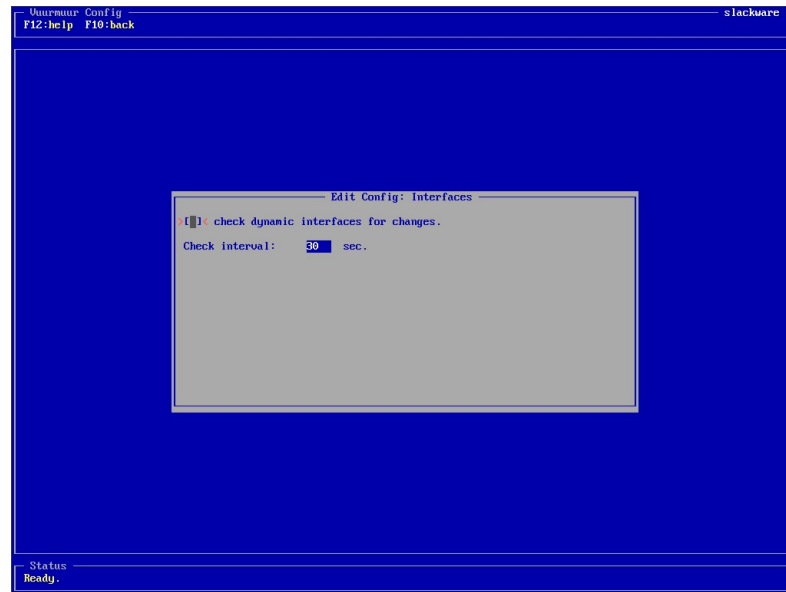


Figura 21: Vuurmuur Config: Interfaces

System Protection

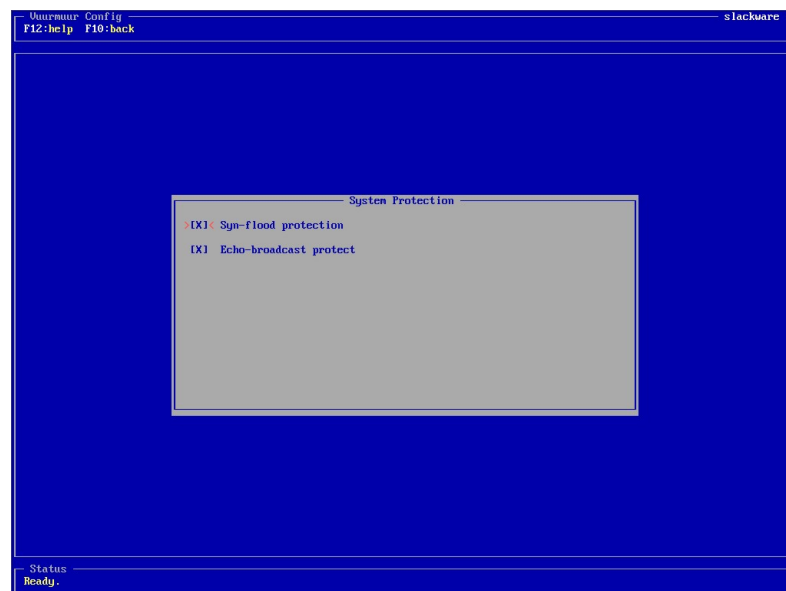


Figura 22: Vuurmuur Config: System Protection

O Vuurmuur oferece proteção automática contra ataques *Syn-flood* e *echo-broadcast*. A proteção contra este tipo de ataque é altamente recomendável. Para ativá-la, apenas marque as caixas.

Logging

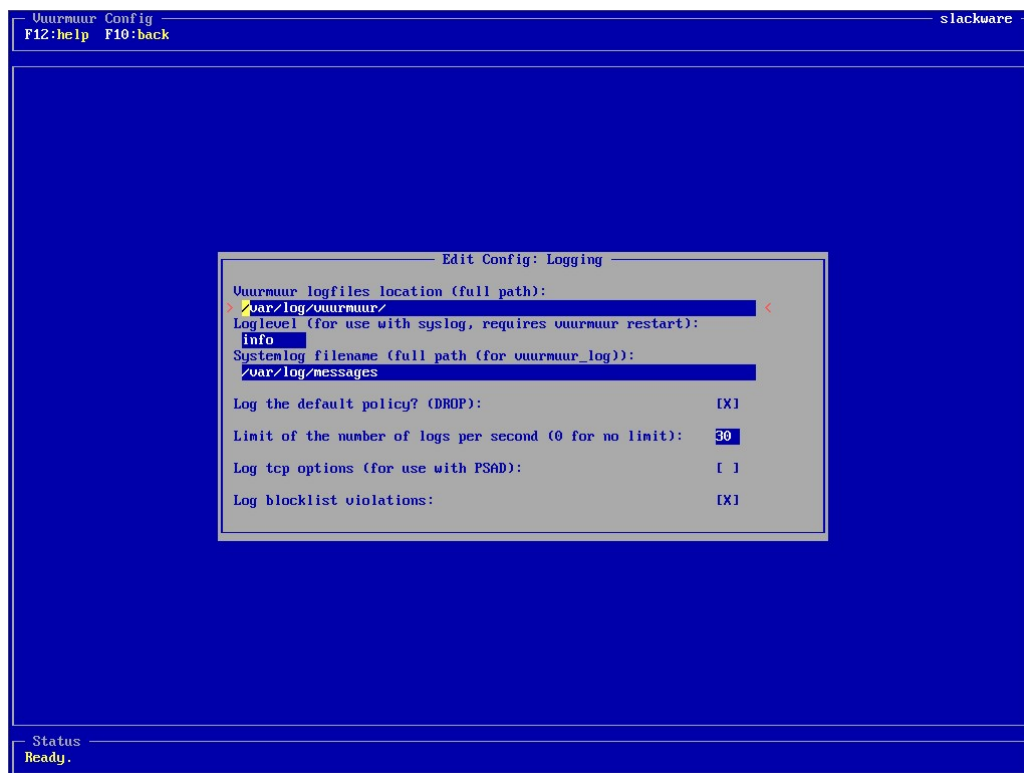


Figura 23: Vuurmuur Config: Logging

Na figura acima, vemos as opções relacionadas à criação de logs em geral. Vejamos o que cada uma faz:

Vuurmuur logfiles location: Indique aqui o caminho onde deverão ser salvos os arquivos de log criados pelo próprio Vuurmuur. São eles:

- `debug.log`: Onde são gravadas informações de depuração geradas pelo Vuurmuur.
- `traffic.log`: Aqui são registradas todas as informações de tráfego convertidas pelo `vuurmuur_log`, geradas a partir do próprio log do iptables. A visualização deste arquivo é bastante freqüente pelos administradores que utilizam o Vuurmuur, pois o tráfego é exibido de uma forma bastante intuitiva, a mesma forma de criação de regras.
- `error.log`: Quando acontecem erros com o Vuurmuur especificamente, sejam eles na inicialização ou durante sua execução, esses erros são gravados neste arquivo.

- `vuurmuur.log`: Aqui são gravadas informações que o Vuurmuur gera em sua inicialização, e também as alterações feitas no `vuurmuur_conf`.

`LogLevel`: Informe o nível de detalhes que o `syslog`¹² irá gerar as mensagens para o log do iptables.

`Systemlog filename`: Especifique aqui onde o `vuurmuur_log` deverá buscar os logs do iptables para fazer a conversão. É aconselhável que se crie um arquivo à parte para os logs do iptables, pois este arquivo pode ficar consideravelmente grande.

`Log the default policy?`: Como já mencionado, a política padrão que o Vuurmuur usa para o iptables é DROP. Marque esta opção se quiser que o Vuurmuur também registre os pacotes que não tenham se encaixado em nenhuma regra. Adicionalmente, a próxima opção, `Limit of the number of logs per second`, permite que se defina um limite de registros por segundo para esses registros.

`Log tcp options`: O Vuurmuur em si não usa isto, mas pode gerar essas informações para serem usados com o PSAD, um *portscan detector* muito eficiente.

`Log blacklist violations`: Este último item deverá ser marcado caso queira que sejam registradas as tentativas de acesso pelos itens cadastrados na Blocklist (veremos à frente).

Cabe aqui uma importante observação, sobre os logs gerados pelo Vuurmuur: dependendo do número de regras que você tenha, e do tráfego em sua rede, os arquivos de log poderão atingir tamanhos muito grandes.

O Vuurmuur fornece, junto com sua instalação padrão, um script para ser utilizado com o `logrotate`, para que seja feita a manutenção destes logs, de modo a evitar que eles cresçam indefinidamente. Se o Vuurmuur tiver sido instalado em seus diretórios padrões, o script poderá ser encontrado em: `/usr/share/vuurmuur/scripts/vuurmuur-logrotate`. Configure-o de acordo com os padrões de sua distribuição.

Modules

Nesta parte, definimos se queremos que o próprio Vuurmuur carregue os módulos necessários (ou apenas os que ainda não tiverem sido carregados, caso já seja feita esta carga durante a inicialização) do iptables para o correto suporte de todas as funções cobertas por ele (Vuurmuur). Vejamos suas opções na figura 24:

¹²Para maiores informações, consulte `man syslog`.

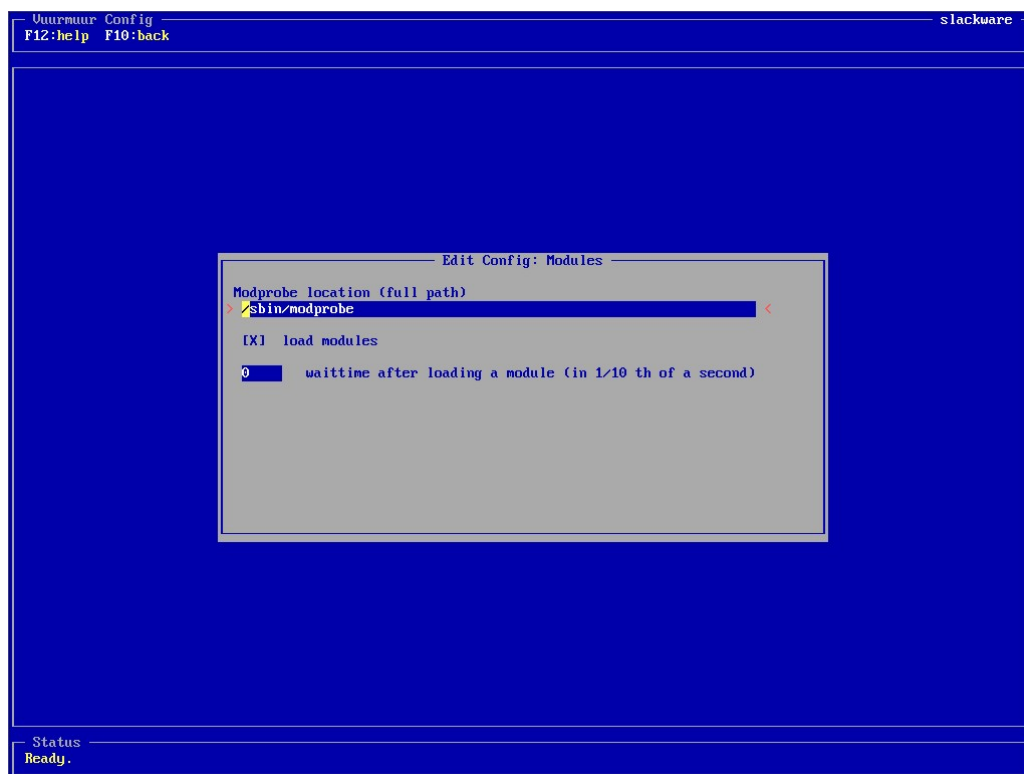


Figura 24: Vuurmuur Config: Modules

A primeira opção, `Modprobe location`, pede o caminho completo para o utilitário `modprobe`, responsável pelo carregamento dos módulos. Logo abaixo, temos a caixa de seleção `load modules`, que serve justamente para dizermos se queremos que o Vuurmuur carregue os módulos ou não.

A opção `waittime after loading a module` serve para informarmos qual será o tempo (em décimos de segundos) entre a carga de um módulo e outro.

Plugins

O Vuurmuur foi projetado de uma forma modularizada. Isto quer dizer que utiliza ‘*backends*’ internos para o armazenamento e manuseio de registros de serviços, zonas, interfaces e regras. Atualmente, só existe apenas um plugin, o `textdir`, do próprio Vuurmuur. Entretanto, o fato de ter sido estruturado dessa forma já possibilita que, no futuro, sejam adicionados e desenvolvidos plugins que possam interagir com o Vuurmuur, de modo a facilitar ainda mais a administração de redes, e fazer a interação com outros programas.

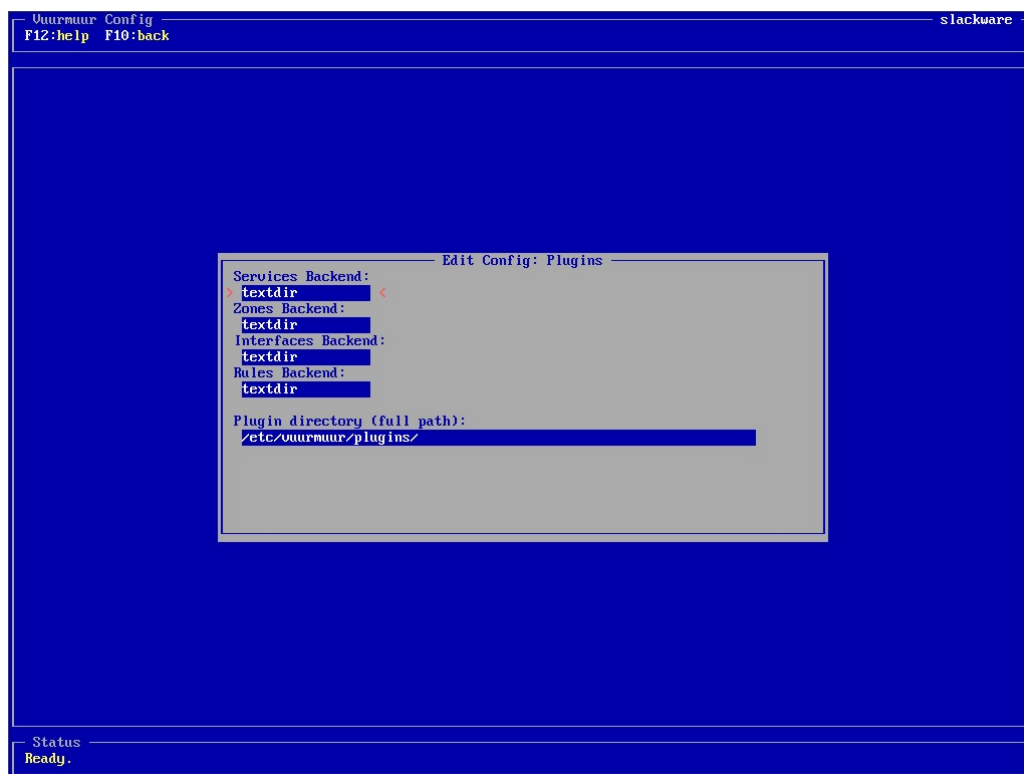


Figura 25: Vuurmuur Config: Plugins

Capabilities

Este item é de extrema importância para o aproveitamento total das funcionalidades do Vuurmuur, e do próprio iptables.

O iptables e seus recursos podem estar disponíveis de duas formas: ou compilados diretamente no kernel, ou como módulos, que podem ser carregados com o *modprobe*¹³, já citado anteriormente.

Normalmente, os recursos mais comuns estão compilados diretamente no kernel, e as funcionalidade ‘extras’ estão disponíveis através de módulos. Para o iptables poder se utilizar destes recursos, os módulos têm de estar carregados quando da utilização do iptables, ou seja, assim que o firewall entrar em funcionamento, em nosso caso.

No caso do Vuurmuur, temos duas formas de fazer isso: ou através do script de inicialização, que tem a opção para informarmos os módulos que queremos que sejam carregados, ou através do Vuurmuur, que pode carregar automaticamente os módulos, como acabamos de ver, na opção ‘Modules’. Esta parte nos mostra quais são as ‘capacidades’ atuais do firewall, ou seja, que tipo de operações definidas pelo iptables ele consegue fazer. Na

¹³Para maiores informações, consulte `man modprobe`.

primeira vez em que entramos nesta tela, pode ser que grande parte dos módulos necessários ainda não tenham sido carregados. Neste caso, o Vuurmuur irá dar uma mensagem dizendo que algumas capacidades não foram carregadas, e perguntará se gostaríamos de tentar carregar esses módulos, conforme mostra a figura 26:

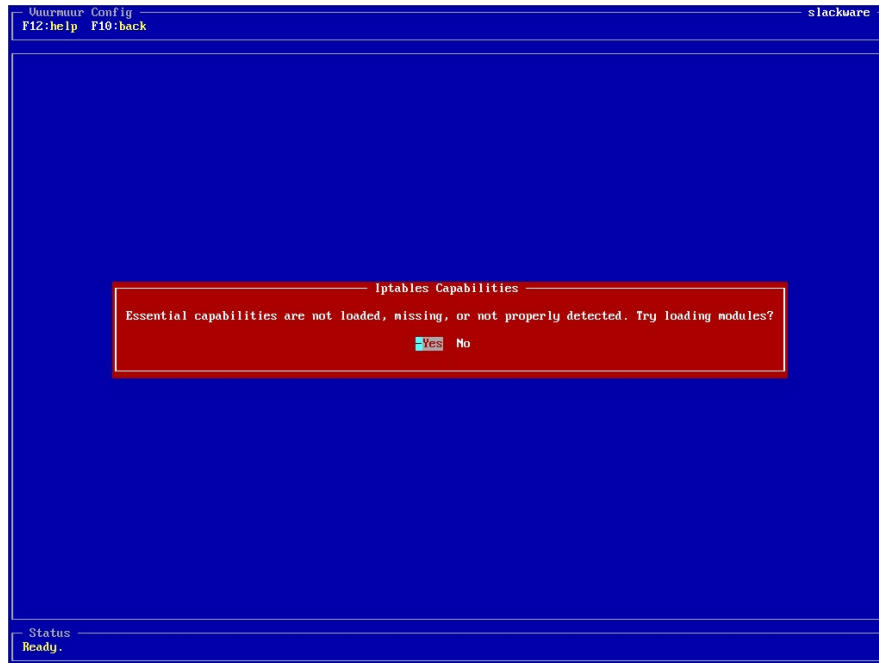


Figura 26: Vuurmuur Config: Aviso sobre módulos não presentes

Após isso, poderemos ver quais funções estão disponíveis em nosso sistema.

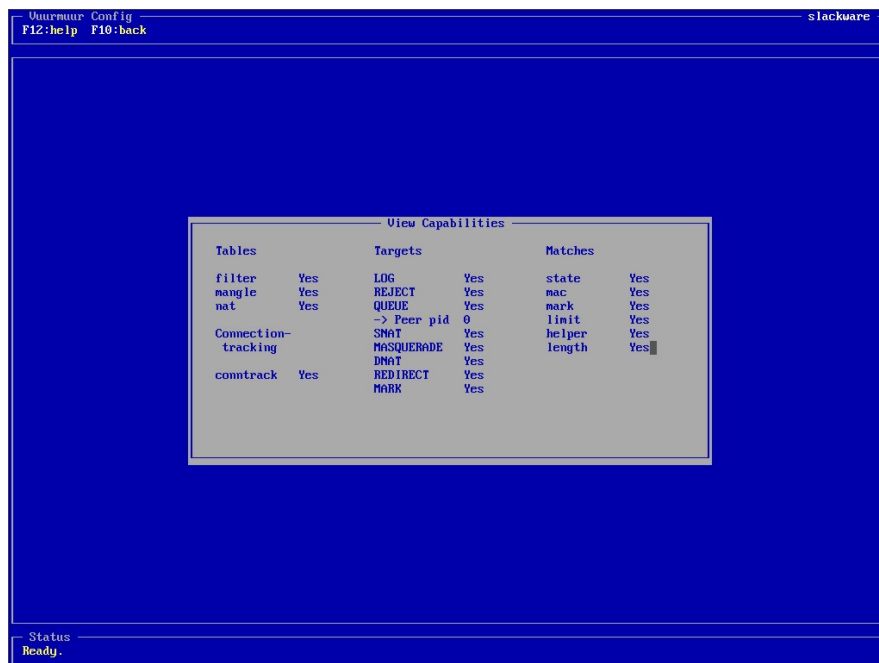


Figura 27: Vuurmuur Config: Capabilities

3.4.4.2 Logview

Esta seção é um dos grandes diferenciais do Vuurmuur, e proporciona uma grande vantagem em relação à utilização de scripts, por exemplo. Aqui podemos visualizar os 4 logs gerados pelo Vuurmuur, `debug.log`, `traffic.log`, `error.log` e `vuurmuur.log`.

Ao entrarmos na seção ‘Logview’, escolhemos qual arquivo desejamos visualizar:

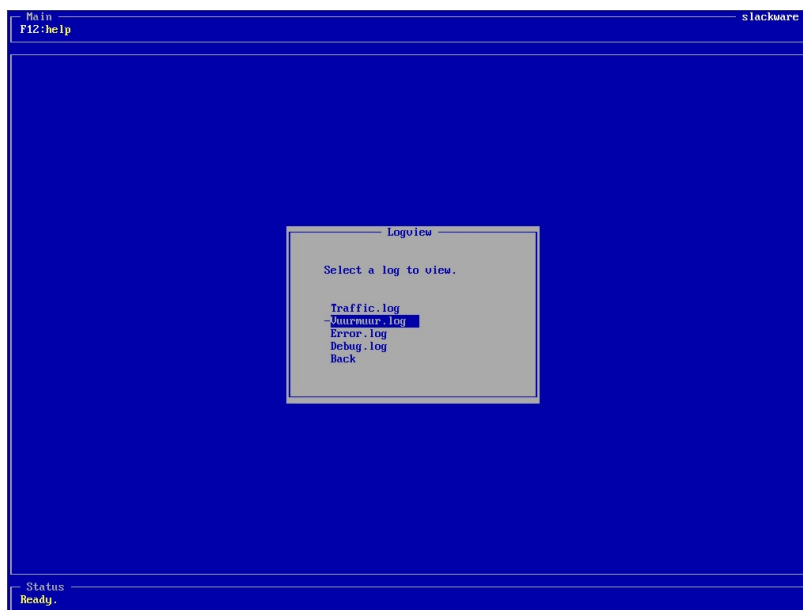


Figura 28: Logview

Vamos pegar, por exemplo, o arquivo `vuurmuur.log`:

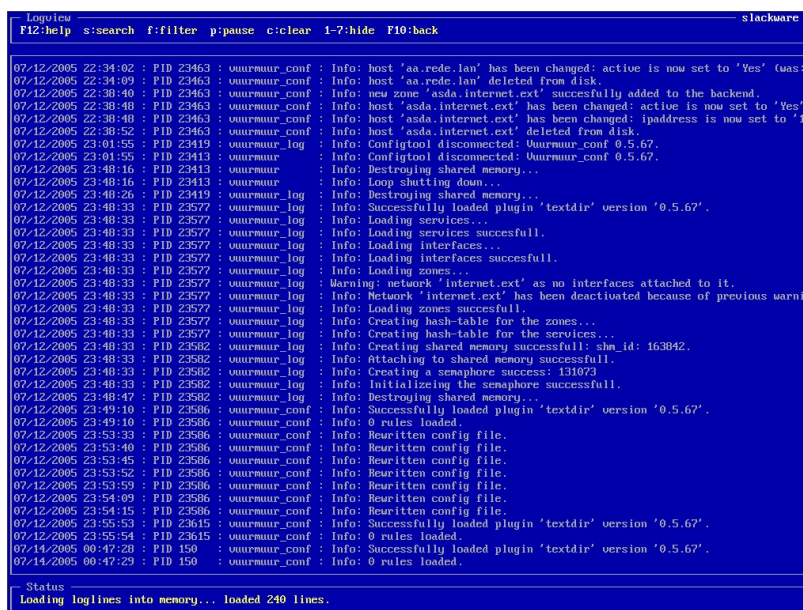


Figura 29: Logview: vuurmuur.log

É exibido o conteúdo do arquivo. Porém, temos, para todos os arquivos visualizados, algumas opções bastante úteis:

`<F12>`: Mostra a ajuda, bastante completa e presente em praticamente todas as partes do programa.

`s:search`: Utilize isto para fazer buscas, dentro do arquivo escolhido ou do resultado que estiver em buffer, no caso de um arquivo que esteja sendo exibido em tempo-real, como o `traffic.log` costuma ser. Na janela de busca, não é permitido o uso de espaços. Após digitado o desejado, o Vuurmuur retorna apenas os resultados que contenham o critério de busca. Após finalizado, aperte `<espaço>` para voltar à visão original.

`f:filter`: Não confundir esta função com a função de busca: a busca é utilizada para resultados estáticos que estão apresentados na tela. O filtro é mais utilizado para o `traffic.log`, que é visualizado em tempo real. Desta forma, só é mostrado o que estiver no filtro. Muitas vezes queremos ver apenas o que determinado host está fazendo, por exemplo, e desta forma temos, em tempo real, toda a atividade. É realmente útil e impressionante o que esta combinação logview + filtro pode fazer.

`p:pause`: Normalmente, quando estamos visualizando o log de tráfego em tempo real, dependendo do tráfego que passa por nosso firewall, as ‘linhas’ passam muito rápido, principalmente se não tivermos nenhum filtro definido. o `pause` serve justamente para ‘congelar’ a rolagem. Tanto o que está a frente do ponto de pausa, como a quantidade de linhas que se pode voltar, é definida pelo tamanho de buffer, que iremos ver na seção [3.4.4.6](#) (p.64).

`c:clear`: Limpa a tela.

`1-7:hide`: Mostra ou esconde as colunas que aparecem na visualização, podendo adequar as suas necessidades.

Para o `error.log` e o `debug.log`, a operação é semelhante, pois o log é geralmente mais estático. O `traffic.log` é um caso à parte: como dito, é a ‘arma-secreta’ do administrador do firewall. Com esta ferramenta, é **muito** fácil acharmos o que quisermos, filtrarmos qualquer host, qualquer serviço ou qualquer evento que esteja passando pelo firewall. Veja um exemplo do Logview em ação com o `traffic.log`:

```

Logview                                     sanctorium
F12:help  s:search  f:filter  p:pause  c:clear  1-7:hide  F10:back

Jul 21 14:15:43: ACCEPT http victor.poort.lan -> 213.239.154.36 (in: eth0 out: ppp0 192.168.1.2:44629
Jul 21 14:15:46: ACCEPT http victor.poort.lan -> 194.126.131.100 (in: eth0 out: ppp0 192.168.1.2:5792
Jul 21 14:15:46: ACCEPT http victor.poort.lan -> 194.109.192.7 (in: eth0 out: ppp0 192.168.1.2:56075
Jul 21 14:15:46: ACCEPT http victor.poort.lan -> 194.126.131.101 (in: eth0 out: ppp0 192.168.1.2:4424
Jul 21 14:15:51: ACCEPT http victor.poort.lan -> 213.239.154.36 (in: eth0 out: ppp0 192.168.1.2:44633
Jul 21 14:15:53: ACCEPT http victor.poort.lan -> 213.239.154.36 (in: eth0 out: ppp0 192.168.1.2:44634
Jul 21 14:15:53: ACCEPT http victor.poort.lan -> 213.239.154.36 (in: eth0 out: ppp0 192.168.1.2:44635
Jul 21 14:15:53: ACCEPT http victor.poort.lan -> 213.239.154.36 (in: eth0 out: ppp0 192.168.1.2:44636
Jul 21 14:15:53: ACCEPT http victor.poort.lan -> 194.126.131.100 (in: eth0 out: ppp0 192.168.1.2:5793
Jul 21 14:15:53: ACCEPT http victor.poort.lan -> 195.154.195.154 (in: eth0 out: ppp0 192.168.1.2:4949
Jul 21 14:15:53: ACCEPT http victor.poort.lan -> 194.126.131.103 (in: eth0 out: ppp0 192.168.1.2:3479
Jul 21 14:15:53: ACCEPT http victor.poort.lan -> 195.154.195.154 (in: eth0 out: ppp0 192.168.1.2:4949
Jul 21 14:15:54: ACCEPT http victor.poort.lan -> 193.79.173.86 (in: eth0 out: ppp0 192.168.1.2:45843
Jul 21 14:15:56: ACCEPT http victor.poort.lan -> 193.79.173.82 (in: eth0 out: ppp0 192.168.1.2:41031
Jul 21 14:16:24: ACCEPT http victor.poort.lan -> 66.35.250.168 (in: eth0 out: ppp0 192.168.1.2:47724
Jul 21 14:16:27: ACCEPT http victor.poort.lan -> 66.150.87.2 (in: eth0 out: ppp0 192.168.1.2:55582 ->
Jul 21 14:16:37: ACCEPT http victor.poort.lan -> frank.internet.ext (in: eth0 out: ppp0 192.168.1.2:4
Jul 21 14:16:37: ACCEPT http victor.poort.lan -> 66.35.250.123 (in: eth0 out: ppp0 192.168.1.2:52034
Jul 21 14:17:12: ACCEPT pop3s victor.poort.lan -> 64.233.171.109 (in: eth0 out: ppp0 192.168.1.2:4330
Jul 21 14:17:32: ACCEPT ssh victor.poort.lan -> firewall(lan-eth0) (in: eth0 192.168.1.2(00:90:27:57:
Jul 21 14:17:58: ACCEPT msn victor.poort.lan -> 207.46.0.185 (in: eth0 out: ppp0 192.168.1.2:34728 ->
Jul 21 14:18:16: DROP 39496->1027(udp) 220.168.143.205 -> firewall(ads1-ppp0) 'internet-in' (in: pp
Jul 21 14:18:43: DROP samba 195.229.199.68 -> firewall(ads1-ppp0) 'internet-in' (in: ppp0 195.229.1
Jul 21 14:19:32: ACCEPT pop3 victor.poort.lan -> frank.internet.ext (in: eth0 out: ppp0 192.168.1.2:4
Jul 21 14:21:41: ACCEPT imap victor.poort.lan -> frank.internet.ext (in: eth0 out: ppp0 192.168.1.2:15
Jul 21 14:21:59: ACCEPT news victor.poort.lan -> 194.109.133.133 (in: eth0 out: ppp0 192.168.1.2:5284
Jul 21 14:22:10: ACCEPT http victor.poort.lan -> 205.156.51.200 (in: eth0 out: ppp0 192.168.1.2:49827
Jul 21 14:22:10: ACCEPT http victor.poort.lan -> 66.35.250.150 (in: eth0 out: ppp0 192.168.1.2:54044
Jul 21 14:22:10: ACCEPT http victor.poort.lan -> 80.232.38.131 (in: eth0 out: ppp0 192.168.1.2:41266
Jul 21 14:22:11: ACCEPT http victor.poort.lan -> 66.35.250.150 (in: eth0 out: ppp0 192.168.1.2:54046
Jul 21 14:22:11: ACCEPT http victor.poort.lan -> 66.35.250.95 (in: eth0 out: ppp0 192.168.1.2:55185 -
Jul 21 14:22:20: ACCEPT http victor.poort.lan -> 152.2.210.81 (in: eth0 out: ppp0 192.168.1.2:37276 -
Jul 21 14:22:31: DROP samba 62.178.228.116 -> firewall(ads1-ppp0) 'internet-in' (in: ppp0 62.178.228

Status
Loading loglines into memory... loaded 3000 lines.

```

Figura 30: Logview: traffic.log

3.4.4.3 Status

```

System Status                               slackware
F12:help  F10:back

----- Status Section -----
Hostname      slackware
Kernel        Linux 2.4.29
Uptime        day h m s
              0 01:39:11

Load          1m 5m 15m
              0.00 0.00 0.00
Memory        Total 91
              Free 53
              Cache 19
              Buffer 11

Connections   Tcp      error
              Udp      error
              Other  error
              Current error
              Maximal error

Interfaces    Speed/s
              Down  Up
              Firewall
              In    Out
              Forwarded
              Recv  Send

-----
Status
Ready.

```

Figura 31: Vuurmuur_conf: Status

A seção **Status** nos mostra diversas informações do sistema:

- Nome da máquina

- Versão do kernel
- *Uptime*
- Carga do processador e memória
- Número de conexões estabelecidas
- Informações relacionadas às interfaces, como a velocidade de banda total passante naquele momento, e o total de tráfego de entrada e saída, bem como a contagem do tráfego encaminhado.

Em alguns casos, especialmente nas informações relacionadas ao tráfego de rede, esta seção pode ser muito útil.

3.4.4.4 Connections

Esta parte é também muito especial, assim como o Logview para o `traffic.log`. Aqui, vemos todas as conexões que estão sendo processadas pelo firewall, em tempo real. Veja este exemplo na figura 32:

```

Connections
F12:help i:in/out/fw s:status c:connect u:grp unknown f:filter
panthro.solabia.com.br

2: pop3      Maquina1. redeinterna. lan  -> firewall (ipl)      DISCONNECTING  INCOMING  44325->110  (6) 192.168.6.100  -> 200.203.148.210
1: 39060 -> 9923  Maquina1. redeinterna. lan  -> firewall (interna)  CONNECTED      INCOMING  39060->9923  (6) 192.168.6.100  -> 192.168.6.253
1: 53808 -> 2869  Maquina3. redeinterna. lan  -> firewall (interna)  DISCONNECTING  INCOMING  53808->2869  (6) 192.168.6.110  -> 192.168.6.253
1: 59793 -> 5000  firewall (interna)         -> Maquina2. redeinterna. lan  DISCONNECTING  OUTGOING  59793->5000  (6) 192.168.6.253  -> 192.168.6.59
1: 59786 -> 5000  firewall (interna)         -> Maquina5. redeinterna. lan  DISCONNECTING  OUTGOING  59786->5000  (6) 192.168.6.253  -> 192.168.6.13
1: 59727 -> 5000  firewall (interna)         -> Maquina2. redeinterna. lan  DISCONNECTING  OUTGOING  59727->5000  (6) 192.168.6.253  -> 192.168.6.59
1: dns      Maquina2. redeinterna. lan  -> firewall (interna)  CONNECTED      INCOMING  4438->53  (17) 192.168.6.59  -> 192.168.6.253
1: dns      Maquina1. redeinterna. lan  -> firewall (interna)  CONNECTED      INCOMING  32975->53  (17) 192.168.6.100  -> 192.168.6.253
1: 59748 -> 5000  firewall (interna)         -> Maquina3. redeinterna. lan  DISCONNECTING  OUTGOING  59748->5000  (6) 192.168.6.253  -> 192.168.6.110
1: 59768 -> 5000  firewall (interna)         -> Maquina5. redeinterna. lan  DISCONNECTING  OUTGOING  59768->5000  (6) 192.168.6.253  -> 192.168.6.13
3: msn      Maquina2. redeinterna. lan  -> internet. ext      CONNECTED      FORWARDING  3597->1863  (6) 192.168.6.59  -> 207.46.6.38
1: 59728 -> 5000  firewall (interna)         -> Maquina5. redeinterna. lan  DISCONNECTING  OUTGOING  59728->5000  (6) 192.168.6.253  -> 192.168.6.13
1: 59787 -> 5000  firewall (interna)         -> Maquina3. redeinterna. lan  DISCONNECTING  OUTGOING  59787->5000  (6) 192.168.6.253  -> 192.168.6.110
1: 59733 -> 2869  firewall (interna)         -> Maquina4. redeinterna. lan  DISCONNECTING  OUTGOING  59733->2869  (6) 192.168.6.253  -> 192.168.6.140
1: 41067 -> 2869  Maquina3. redeinterna. lan  -> firewall (interna)  DISCONNECTING  INCOMING  41067->2869  (6) 192.168.6.110  -> 192.168.6.253
1: pop3      Maquina5. redeinterna. lan  -> firewall (interna)  DISCONNECTING  INCOMING  3308->110  (6) 192.168.6.13  -> 192.168.6.253
1: 32730 -> 2869  Maquina2. redeinterna. lan  -> firewall (interna)  DISCONNECTING  INCOMING  32730->2869  (6) 192.168.6.59  -> 192.168.6.253
1: 13347 -> 2869  Maquina3. redeinterna. lan  -> firewall (interna)  DISCONNECTING  INCOMING  13347->2869  (6) 192.168.6.110  -> 192.168.6.253
1: 52839 -> 2869  Maquina3. redeinterna. lan  -> firewall (interna)  DISCONNECTING  INCOMING  52839->2869  (6) 192.168.6.110  -> 192.168.6.253
1: 27902 -> 2869  Maquina3. redeinterna. lan  -> firewall (interna)  DISCONNECTING  INCOMING  27902->2869  (6) 192.168.6.110  -> 192.168.6.253
1: 59775 -> 5000  firewall (interna)         -> Maquina2. redeinterna. lan  DISCONNECTING  OUTGOING  59775->5000  (6) 192.168.6.253  -> 192.168.6.59
1: 28756 -> 2869  Maquina3. redeinterna. lan  -> firewall (interna)  DISCONNECTING  INCOMING  28756->2869  (6) 192.168.6.110  -> 192.168.6.253
1: 59790 -> 2869  firewall (interna)         -> Maquina4. redeinterna. lan  DISCONNECTING  OUTGOING  59790->2869  (6) 192.168.6.253  -> 192.168.6.140
1: msn      Maquina4. redeinterna. lan  -> internet. ext      CONNECTED      FORWARDING  1032->1863  (6) 192.168.6.140  -> 207.46.0.96
1: 12114 -> 2869  Maquina3. redeinterna. lan  -> firewall (interna)  DISCONNECTING  INCOMING  12114->2869  (6) 192.168.6.110  -> 192.168.6.253
1: squid-proxy Maquina3. redeinterna. lan  -> firewall (interna)  CONNECTED      INCOMING  2680->3128  (6) 192.168.6.110  -> 192.168.6.253
1: 59750 -> 2869  firewall (interna)         -> Maquina4. redeinterna. lan  DISCONNECTING  OUTGOING  59750->2869  (6) 192.168.6.253  -> 192.168.6.140
1: 59746 -> 5000  firewall (interna)         -> Maquina2. redeinterna. lan  DISCONNECTING  OUTGOING  59746->5000  (6) 192.168.6.253  -> 192.168.6.59
1: 59772 -> 2869  firewall (interna)         -> Maquina4. redeinterna. lan  DISCONNECTING  OUTGOING  59772->2869  (6) 192.168.6.253  -> 192.168.6.140

Status
Active filter: 'Maquina' (press 'enter' to clear).

```

Figura 32: Connections: Visualização em tempo real

Temos também algumas opções úteis aqui:

i:in/out/fw: Esta opção faz a segmentação da exibição em três partes: as conexões que estão entrando no firewall (*incoming*), as conexões que estão saindo (*outgoing*), e as conexões que estão sendo encaminhadas (*forwarded*).

s:status: Isto mostra ou esconde o campo de status de conexões, que pode ser ‘conectando’ (*connecting*), ‘desconectando’ (*disconnecting*), ou ‘conectado’ (*connected*).

c:connect: Esta função é semelhante à *in/out/fw*, porém esta separa as conexões em conexões estabelecidas (*established*), conexões inicializando (*initializing*) ou conexões encerrando (*closing*).

u:grp unknow: Esta opção nos diz se deverão ser exibidas aqui linhas que contenham hosts que não pertencem a nenhum grupo. Caso ativada, só aparecerão as linhas em que o host é ‘conhecido’ pelo Vuurmuur. Neste caso, o nome cadastrado para ele é que aparecerá.

Vejamos alguns exemplos das diversas configurações da seção **Connections**:

```

Connections
F12:help i:in/out/fw s:status c:connect u:grp unknow f:filter panthro.solabia.com.br

Connections: Total: 271 Connecting: 4 Established: 79
              Disconnecting: 188

-- Forwarded Connections (29)
15: msn redeinterna.lan -> internet.ext CONNECTED 1049->1863 (6) 192.168.6.200 -> 207.46.0.82
4: msn redeinterna.lan -> internet.ext DISCONNECTING 1401->1863 (6) 192.168.6.178 -> 207.46.0.143
1: msn Maquina4.redeinterna.lan -> internet.ext CONNECTED 1032->1863 (6) 192.168.6.140 -> 207.46.0.96
3: msn Maquina2.redeinterna.lan -> internet.ext CONNECTED 3597->1863 (6) 192.168.6.59 -> 207.46.6.38
2: ftp redeinterna.lan -> internet.ext DISCONNECTING 4631->21 (6) 192.168.6.58 -> 66.98.156.195
1: http Maquina4.redeinterna.lan -> internet.ext CONNECTING 2435->80 (6) 192.168.6.140 -> 200.170.96.148
1: 1025 -> 161 redeinterna.lan -> internet.ext CONNECTING 1025->161 (17) 192.168.6.117 -> 10.0.26.101
1: 2442 -> 554 Maquina4.redeinterna.lan -> internet.ext CONNECTING 2442->554 (6) 192.168.6.140 -> 200.170.96.148
1: msn Maquina2.redeinterna.lan -> internet.ext DISCONNECTING 4499->1863 (6) 192.168.6.59 -> 207.46.0.162

-- Incoming Connections (43)
3: squid-proxy redeinterna.lan -> firewall(interna) CONNECTED 4166->3128 (6) 192.168.6.170 -> 192.168.6.253
1: 29749 -> 2869 Maquina2.redeinterna.lan -> firewall(interna) DISCONNECTING 29749->2869 (6) 192.168.6.59 -> 192.168.6.253
6: squid-proxy Maquina4.redeinterna.lan -> firewall(interna) DISCONNECTING 2426->3128 (6) 192.168.6.140 -> 192.168.6.253
3: squid-proxy redeinterna.lan -> firewall(interna) DISCONNECTING 2024->3128 (6) 192.168.6.145 -> 192.168.6.253
1: dns Maquina1.redeinterna.lan -> firewall(interna) CONNECTED 32975->53 (17) 192.168.6.100 -> 192.168.6.253
1: 11729 -> 2869 redeinterna.lan -> firewall(interna) DISCONNECTING 11729->2869 (6) 192.168.6.178 -> 192.168.6.253
1: 56334 -> 2869 Maquina3.redeinterna.lan -> firewall(interna) DISCONNECTING 56334->2869 (6) 192.168.6.110 -> 192.168.6.253
1: 20413 -> 2869 Maquina3.redeinterna.lan -> firewall(interna) DISCONNECTING 20413->2869 (6) 192.168.6.110 -> 192.168.6.253
4: squid-proxy Maquina4.redeinterna.lan -> firewall(interna) CONNECTED 2440->3128 (6) 192.168.6.140 -> 192.168.6.253
1: dns Maquina4.redeinterna.lan -> firewall(interna) CONNECTED 1027->53 (17) 192.168.6.140 -> 192.168.6.253
1: samba redeinterna.lan -> redeinterna.lan(broadcast) CONNECTING 137->137 (17) 192.168.6.117 -> 192.168.6.255
1: 59783 -> 2869 redeinterna.lan -> firewall(interna) DISCONNECTING 59783->2869 (6) 192.168.6.178 -> 192.168.6.253

-- Outgoing Connections (199)
43: http firewall(ip1) -> internet.ext DISCONNECTING 60197->80 (6) 200.203.148.210 -> 67.138.240.8
34: http firewall(ip1) -> firewall(ip1) CONNECTED 48981->80 (6) 200.203.148.210 -> 200.203.148.210
1: 60258 -> 5000 firewall(interna) -> redeinterna.lan DISCONNECTING 60258->5000 (6) 192.168.6.253 -> 192.168.6.170
8: http firewall(ip1) -> internet.ext CONNECTED 60282->80 (6) 200.203.148.210 -> 200.170.96.147
1: 60173 -> 2869 firewall(interna) -> redeinterna.lan DISCONNECTING 60173->2869 (6) 192.168.6.253 -> 192.168.6.139
1: 60225 -> 5000 firewall(interna) -> redeinterna.lan DISCONNECTING 60225->5000 (6) 192.168.6.253 -> 192.168.6.190
1: 60180 -> 5000 firewall(interna) -> redeinterna.lan DISCONNECTING 60180->5000 (6) 192.168.6.253 -> 192.168.6.170
1: 60230 -> 5000 firewall(interna) -> Maquina3.redeinterna.lan DISCONNECTING 60230->5000 (6) 192.168.6.253 -> 192.168.6.110
1: 60236 -> 5000 firewall(interna) -> redeinterna.lan DISCONNECTING 60236->5000 (6) 192.168.6.253 -> 192.168.6.58
1: 60213 -> 5000 firewall(interna) -> Maquina3.redeinterna.lan DISCONNECTING 60213->5000 (6) 192.168.6.253 -> 192.168.6.110
3: dns firewall(ip1) -> internet.ext CONNECTED 39089->53 (17) 200.203.148.210 -> 200.170.96.148

-- Status
Ready

```

Figura 33: Connections: in/out/fw

```

Connections
F12:help i:in/out/fw s:status c:connect u:grp unknown f:filter panthro.solabia.com.br

Connections: Total: 227 Incoming: 46 Forwarding: 23
              Outgoing: 158

----- Established Connections ----- (65)
1: msn Maquina4.redeinterna.lan -> 207.46.0.96 FORWARDING 1032->1863 (6) 192.168.6.140 -> 207.46.0.96
1: msn Maquina2.redeinterna.lan -> 207.46.6.38 FORWARDING 3597->1863 (6) 192.168.6.59 -> 207.46.6.38
1: squid-proxy Maquina4.redeinterna.lan -> firewall(interna) INCOMING 2412->3128 (6) 192.168.6.140 -> 192.168.6.253
1: squid-proxy Maquina3.redeinterna.lan -> firewall(interna) INCOMING 2693->3128 (6) 192.168.6.110 -> 192.168.6.253
1: dns Maquina2.redeinterna.lan -> firewall(interna) INCOMING 4438->53 (17) 192.168.6.59 -> 192.168.6.253
1: dns Maquina1.redeinterna.lan -> firewall(interna) INCOMING 32975->53 (17) 192.168.6.100 -> 192.168.6.253
1: msn Maquina2.redeinterna.lan -> 207.46.0.77 FORWARDING 3248->1863 (6) 192.168.6.59 -> 207.46.0.77
1: msn Maquina2.redeinterna.lan -> 207.46.0.95 FORWARDING 3032->1863 (6) 192.168.6.59 -> 207.46.0.95
1: msn Maquina2.redeinterna.lan -> 207.46.6.27 FORWARDING 3025->1863 (6) 192.168.6.59 -> 207.46.6.27

----- Connections Initializing ----- (1)

----- Connections Closing ----- (161)
11: squid-proxy Maquina4.redeinterna.lan -> firewall(interna) INCOMING 2411->3128 (6) 192.168.6.140 -> 192.168.6.253
1: 56412 -> 2869 Maquina3.redeinterna.lan -> firewall(interna) INCOMING 56412->2869 (6) 192.168.6.110 -> 192.168.6.253
1: 59903 -> 2869 firewall(interna) -> Maquina4.redeinterna.lan OUTGOING 59903->2869 (6) 192.168.6.253 -> 192.168.6.140
1: 59889 -> 2869 firewall(interna) -> Maquina4.redeinterna.lan OUTGOING 59889->2869 (6) 192.168.6.253 -> 192.168.6.140
1: 59899 -> 5000 firewall(interna) -> Maquina5.redeinterna.lan OUTGOING 59899->5000 (6) 192.168.6.253 -> 192.168.6.13
1: 65230 -> 2869 Maquina3.redeinterna.lan -> firewall(interna) INCOMING 65230->2869 (6) 192.168.6.110 -> 192.168.6.253
1: 17654 -> 2869 Maquina3.redeinterna.lan -> firewall(interna) INCOMING 17654->2869 (6) 192.168.6.110 -> 192.168.6.253
1: 14378 -> 2869 Maquina3.redeinterna.lan -> firewall(interna) INCOMING 14378->2869 (6) 192.168.6.110 -> 192.168.6.253
1: 59920 -> 5000 firewall(interna) -> Maquina3.redeinterna.lan OUTGOING 59920->5000 (6) 192.168.6.253 -> 192.168.6.110

----- Status -----
Active filter: 'Maquina' (press 'enter' to clear).

```

Figura 34: Connections: connect status

```

Connections
F12:help i:in/out/fw s:status c:connect u:grp unknown f:filter panthro.solabia.com.br

31: http firewall(ip1) -> firewall(ip1) CONNECTED OUTGOING 48981->80 (6) 200.203.148.210 -> 200.203.148.210
9: http firewall(ip1) -> 207.46.1.3 DISCONNECTING OUTGOING 60007->80 (6) 200.203.148.210 -> 207.46.1.3
6: pop3 192.168.6.120 -> firewall(interna) DISCONNECTING INCOMING 2015->110 (6) 192.168.6.120 -> 192.168.6.253
5: http firewall(ip1) -> 200.221.6.20 DISCONNECTING OUTGOING 60117->80 (6) 200.203.148.210 -> 200.221.6.20
10: squid-proxy Maquina4.redeinterna.lan -> firewall(interna) DISCONNECTING INCOMING 2426->3128 (6) 192.168.6.140 -> 192.168.6.253
3: squid-proxy 192.168.6.178 -> firewall(interna) DISCONNECTING INCOMING 1721->3128 (6) 192.168.6.178 -> 192.168.6.253
9: http firewall(ip1) -> 207.46.3.10 DISCONNECTING OUTGOING 60125->80 (6) 200.203.148.210 -> 207.46.3.10
1: http firewall(ip1) -> 200.6.42.68 DISCONNECTING OUTGOING 60092->80 (6) 200.203.148.210 -> 200.6.42.68
2: squid-proxy 192.168.6.20 -> firewall(interna) CONNECTED INCOMING 2116->3128 (6) 192.168.6.20 -> 192.168.6.253
1: msn 192.168.6.178 -> 207.46.0.143 DISCONNECTING FORWARDING 1401->1863 (6) 192.168.6.178 -> 207.46.0.143
1: dns firewall(ip1) -> 192.5.6.30 CONNECTED OUTGOING 39089->53 (17) 200.203.148.210 -> 192.5.6.30
1: msn 192.168.6.58 -> 207.46.2.47 CONNECTED FORWARDING 3025->1863 (6) 192.168.6.58 -> 207.46.2.47
1: msn 192.168.6.138 -> 207.46.0.40 CONNECTED FORWARDING 1031->1863 (6) 192.168.6.138 -> 207.46.0.40
1: dns firewall(ip1) -> 81.52.250.134 CONNECTED OUTGOING 39089->53 (17) 200.203.148.210 -> 81.52.250.134
1: 60074 -> 5000 firewall(interna) -> Maquina3.redeinterna.lan DISCONNECTING OUTGOING 60074->5000 (6) 192.168.6.253 -> 192.168.6.110
1: samba 192.168.6.200 -> redeinterna.lan(broadcast) CONNECTING INCOMING 138->138 (17) 192.168.6.200 -> 192.168.6.255
1: msn 192.168.6.44 -> 207.46.0.66 CONNECTED FORWARDING 1215->1863 (6) 192.168.6.44 -> 207.46.0.66
1: 60038 -> 2869 firewall(interna) -> 192.168.6.139 DISCONNECTING OUTGOING 60038->2869 (6) 192.168.6.253 -> 192.168.6.139
1: 60048 -> 5000 firewall(interna) -> 192.168.6.178 DISCONNECTING OUTGOING 60048->5000 (6) 192.168.6.253 -> 192.168.6.178
1: 60011 -> 5000 firewall(interna) -> 192.168.6.120 DISCONNECTING OUTGOING 60011->5000 (6) 192.168.6.253 -> 192.168.6.120
1: 60008 -> 5000 firewall(interna) -> 192.168.6.14 DISCONNECTING OUTGOING 60008->5000 (6) 192.168.6.253 -> 192.168.6.14
1: dns 192.168.6.178 -> firewall(interna) CONNECTING INCOMING 1031->53 (17) 192.168.6.178 -> 192.168.6.253
1: 60028 -> 2869 firewall(interna) -> 192.168.6.145 DISCONNECTING OUTGOING 60028->2869 (6) 192.168.6.253 -> 192.168.6.145
1: 60056 -> 5000 firewall(interna) -> Maquina3.redeinterna.lan DISCONNECTING OUTGOING 60056->5000 (6) 192.168.6.253 -> 192.168.6.110
1: 60023 -> 2869 firewall(interna) -> Maquina4.redeinterna.lan DISCONNECTING OUTGOING 60023->2869 (6) 192.168.6.253 -> 192.168.6.140
1: msn 192.168.6.145 -> 207.46.0.65 CONNECTED FORWARDING 1056->1863 (6) 192.168.6.145 -> 207.46.0.65
1: http firewall(ip1) -> 65.54.239.81 CONNECTED OUTGOING 60136->80 (6) 200.203.148.210 -> 65.54.239.81
1: 24694 -> 2869 192.168.6.170 -> firewall(interna) DISCONNECTING INCOMING 24694->2869 (6) 192.168.6.170 -> 192.168.6.253
1: squid-proxy 192.168.6.170 -> firewall(interna) CONNECTED INCOMING 4166->3128 (6) 192.168.6.170 -> 192.168.6.253
1: 60060 -> 5000 firewall(interna) -> 192.168.6.130 DISCONNECTING OUTGOING 60060->5000 (6) 192.168.6.253 -> 192.168.6.130
1: 45731 -> 2869 Maquina3.redeinterna.lan -> firewall(interna) DISCONNECTING INCOMING 45731->2869 (6) 192.168.6.110 -> 192.168.6.253
1: dns Maquina2.redeinterna.lan -> firewall(interna) CONNECTED INCOMING 4438->53 (17) 192.168.6.59 -> 192.168.6.253
1: dns Maquina1.redeinterna.lan -> firewall(interna) CONNECTED INCOMING 32975->53 (17) 192.168.6.100 -> 192.168.6.253
1: 60032 -> 5000 firewall(interna) -> 192.168.6.20 DISCONNECTING OUTGOING 60032->5000 (6) 192.168.6.253 -> 192.168.6.20
1: http firewall(ip1) -> 207.46.248.112 DISCONNECTING OUTGOING 60128->80 (6) 200.203.148.210 -> 207.46.248.112
1: msn 192.168.6.139 -> 207.46.0.93 CONNECTED FORWARDING 1034->1863 (6) 192.168.6.139 -> 207.46.0.93
1: https 192.168.6.139 -> 200.246.211.106 CONNECTED OUTGOING 46922->443 (6) 200.203.148.210 -> 200.246.211.106
1: http firewall(ip1) -> 200.221.8.71 DISCONNECTING OUTGOING 60114->80 (6) 200.203.148.210 -> 200.221.8.71
1: 60049 -> 5000 firewall(interna) -> 192.168.6.190 DISCONNECTING OUTGOING 60049->5000 (6) 192.168.6.253 -> 192.168.6.190
4: http firewall(ip1) -> 207.46.1.10 DISCONNECTING OUTGOING 59992->80 (6) 200.203.148.210 -> 207.46.1.10
3: http firewall(ip1) -> 200.221.7.75 DISCONNECTING OUTGOING 60098->80 (6) 200.203.148.210 -> 200.221.7.75
1: 60067 -> 5000 firewall(interna) -> 192.168.6.178 DISCONNECTING OUTGOING 60067->5000 (6) 192.168.6.253 -> 192.168.6.178

----- Status -----
Ready.

```

Figura 35: Connections: group unknown

ID	Protocol	Interface	Destination	Status	Direction	Ports	IP Addresses
49:	http	firewall(ip1)	-> internet.ext	DISCONNECTING	OUTGOING	60007->80	(6) 200.203.148.210 -> 207.46.1.3
34:	http	firewall(ip1)	-> firewall(ip1)	CONNECTED	OUTGOING	49981->80	(6) 200.203.148.210 -> 200.203.148.210
16:	msn	redeinterna.lan	-> internet.ext	CONNECTED	FORWARDING	1049->1863	(6) 192.168.6.200 -> 207.46.0.82
10:	squid-proxy	Maquina4.redeinterna.lan	-> firewall(interna)	DISCONNECTING	INCOMING	2426->3128	(6) 192.168.6.140 -> 192.168.6.253
10:	dns	firewall(ip1)	-> internet.ext	CONNECTED	OUTGOING	39089->53	(17) 200.203.148.210 -> 209.63.57.200
7:	squid-proxy	redeinterna.lan	-> firewall(interna)	DISCONNECTING	INCOMING	2034->3128	(6) 192.168.6.145 -> 192.168.6.253
5:	pop3	redeinterna.lan	-> firewall(interna)	DISCONNECTING	INCOMING	2015->110	(6) 192.168.6.120 -> 192.168.6.253
2:	squid-proxy	redeinterna.lan	-> firewall(interna)	CONNECTED	INCOMING	4166->3128	(6) 192.168.6.170 -> 192.168.6.253
1:	39060 -> 9923	Maquina1.redeinterna.lan	-> firewall(interna)	CONNECTED	INCOMING	39060->9923	(6) 192.168.6.100 -> 192.168.6.253
1:	60075 -> 2869	firewall(interna)	-> redeinterna.lan	DISCONNECTING	OUTGOING	60075->2869	(6) 192.168.6.253 -> 192.168.6.139
1:	10021 -> 2869	redeinterna.lan	-> firewall(interna)	DISCONNECTING	INCOMING	10021->2869	(6) 192.168.6.178 -> 192.168.6.253
1:	60172 -> 5000	firewall(interna)	-> Maquina3.redeinterna.lan	DISCONNECTING	OUTGOING	60172->5000	(6) 192.168.6.253 -> 192.168.6.110
1:	60165 -> 2869	firewall(interna)	-> redeinterna.lan	DISCONNECTING	OUTGOING	60165->2869	(6) 192.168.6.253 -> 192.168.6.136
1:	60029 -> 5000	firewall(interna)	-> redeinterna.lan	DISCONNECTING	OUTGOING	60029->5000	(6) 192.168.6.253 -> 192.168.6.120
1:	60076 -> 2869	firewall(interna)	-> redeinterna.lan	DISCONNECTING	OUTGOING	60076->2869	(6) 192.168.6.253 -> 192.168.6.138
1:	60168 -> 5000	firewall(interna)	-> redeinterna.lan	DISCONNECTING	OUTGOING	60168->5000	(6) 192.168.6.253 -> 192.168.6.20
1:	60013 -> 5000	firewall(interna)	-> redeinterna.lan	DISCONNECTING	OUTGOING	60013->5000	(6) 192.168.6.253 -> 192.168.6.178
1:	58380 -> 2869	Maquina3.redeinterna.lan	-> firewall(interna)	DISCONNECTING	INCOMING	58380->2869	(6) 192.168.6.110 -> 192.168.6.253
1:	60049 -> 5000	firewall(interna)	-> redeinterna.lan	DISCONNECTING	OUTGOING	60049->5000	(6) 192.168.6.253 -> 192.168.6.190
1:	60039 -> 2869	firewall(interna)	-> redeinterna.lan	DISCONNECTING	OUTGOING	60039->2869	(6) 192.168.6.253 -> 192.168.6.138
1:	60167 -> 5000	firewall(interna)	-> redeinterna.lan	DISCONNECTING	OUTGOING	60167->5000	(6) 192.168.6.253 -> 192.168.6.160
1:	11729 -> 2869	redeinterna.lan	-> firewall(interna)	DISCONNECTING	INCOMING	11729->2869	(6) 192.168.6.178 -> 192.168.6.253
1:	60178 -> 5000	firewall(interna)	-> redeinterna.lan	DISCONNECTING	OUTGOING	60178->5000	(6) 192.168.6.253 -> 192.168.6.178
1:	60064 -> 2869	firewall(interna)	-> redeinterna.lan	DISCONNECTING	OUTGOING	60064->2869	(6) 192.168.6.253 -> 192.168.6.145
1:	60150 -> 5000	firewall(interna)	-> redeinterna.lan	DISCONNECTING	OUTGOING	60150->5000	(6) 192.168.6.253 -> 192.168.6.160
1:	60009 -> 5000	firewall(interna)	-> redeinterna.lan	DISCONNECTING	OUTGOING	60009->5000	(6) 192.168.6.253 -> 192.168.6.170
3:	msn	redeinterna.lan	-> internet.ext	DISCONNECTING	FORWARDING	1401->1863	(6) 192.168.6.178 -> 207.46.0.143
1:	60173 -> 2869	firewall(interna)	-> redeinterna.lan	DISCONNECTING	OUTGOING	60173->2869	(6) 192.168.6.253 -> 192.168.6.139
1:	54082 -> 2869	redeinterna.lan	-> firewall(interna)	DISCONNECTING	INCOMING	54082->2869	(6) 192.168.6.178 -> 192.168.6.253
1:	60048 -> 5000	firewall(interna)	-> redeinterna.lan	DISCONNECTING	OUTGOING	60048->5000	(6) 192.168.6.253 -> 192.168.6.178
6:	http	firewall(ip1)	-> internet.ext	CONNECTED	OUTGOING	60136->80	(6) 200.203.148.210 -> 65.54.239.81
2:	dns	redeinterna.lan	-> firewall(interna)	CONNECTED	INCOMING	1031->53	(17) 192.168.6.178 -> 192.168.6.253
1:	60068 -> 5000	firewall(interna)	-> redeinterna.lan	DISCONNECTING	OUTGOING	60068->5000	(6) 192.168.6.253 -> 192.168.6.190
1:	60060 -> 5000	firewall(interna)	-> redeinterna.lan	DISCONNECTING	OUTGOING	60060->5000	(6) 192.168.6.253 -> 192.168.6.130
1:	dns	Maquina2.redeinterna.lan	-> firewall(interna)	CONNECTED	INCOMING	4438->53	(17) 192.168.6.99 -> 192.168.6.253
1:	dns	Maquina1.redeinterna.lan	-> firewall(interna)	CONNECTED	INCOMING	32975->53	(17) 192.168.6.100 -> 192.168.6.253
1:	15197 -> 3478	redeinterna.lan	-> internet.ext	CONNECTED	FORWARDING	15197->3478	(17) 192.168.6.178 -> 64.15.206.211
1:	60037 -> 5000	firewall(interna)	-> Maquina3.redeinterna.lan	DISCONNECTING	OUTGOING	60037->5000	(6) 192.168.6.253 -> 192.168.6.110
1:	60144 -> 5000	firewall(interna)	-> redeinterna.lan	DISCONNECTING	OUTGOING	60144->5000	(6) 192.168.6.253 -> 192.168.6.170
3:	pop3	Maquina1.redeinterna.lan	-> firewall(ip1)	DISCONNECTING	INCOMING	44336->110	(6) 192.168.6.100 -> 200.203.148.210
1:	60181 -> 2869	firewall(interna)	-> redeinterna.lan	DISCONNECTING	OUTGOING	60181->2869	(6) 192.168.6.253 -> 192.168.6.145
1:	60195 -> 5000	firewall(interna)	-> redeinterna.lan	DISCONNECTING	OUTGOING	60195->5000	(6) 192.168.6.253 -> 192.168.6.58

Figura 36: Connections: campo ‘status’ aparecendo

3.4.4.5 Bandwidth

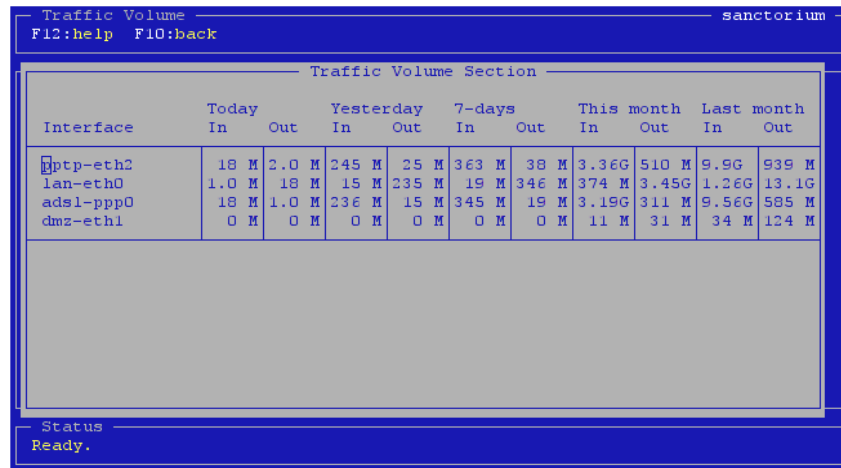
Todo o ‘conteúdo’ desta seção é gerado não pelo Vuurmuur, mas sim por outro programa, o Ip Traffic Volume¹⁴. Esta seção nos mostra dados relacionados ao consumo de banda do firewall, de um modo geral, em diversos períodos de tempo.

Para todas as categorias, é listado o volume de tráfego de todas as interfaces, tanto na entrada, quanto na saída. São apresentadas as seguintes categorias:

- Today: É apresentado o tráfego corrente do dia atual.
- Yesterday: É exibido a contagem do tráfego do dia anterior.
- 7-days: Aqui é exibido a soma do tráfego dos últimos 7 dias, ou seja, de 1 semana atrás em diante.
- This month: Mesma coisa, mas para o mês inteiro.
- Last month: Exibe todo o tráfego do mês anterior. Bastante útil para as comparações com o mês atual.

¹⁴Saiba mais sobre este interessante projeto em <http://iptrafficvolume.sourceforge.net>.

Através do script `iptrafvol.pl` fornecido com o IP Traffic Volume, o Vuurmuur lê os dados gerados pelo programa, e os apresenta de forma organizada para a conferência do usuário. A imagem 37 nos dá uma noção da ferramenta:



The screenshot shows a terminal window titled "Traffic Volume" with the user "sanctorium". The main content is a table titled "Traffic Volume Section" with columns for Interface, Today (In/Out), Yesterday (In/Out), 7-days (In/Out), This month (In/Out), and Last month (In/Out). The data is as follows:

Interface	Today		Yesterday		7-days		This month		Last month	
	In	Out	In	Out	In	Out	In	Out	In	Out
hptp-eth2	18 M	2.0 M	245 M	25 M	363 M	38 M	3.36G	510 M	9.9G	939 M
lan-eth0	1.0 M	18 M	15 M	235 M	19 M	346 M	374 M	3.45G	1.26G	13.1G
adsl-ppp0	18 M	1.0 M	236 M	15 M	345 M	19 M	3.19G	311 M	9.56G	585 M
dmz-eth1	0 M	0 M	0 M	0 M	0 M	0 M	11 M	31 M	34 M	124 M

The status bar at the bottom indicates "Status Ready."

Figura 37: Traffic Log Volume

3.4.4.6 Vuurmuur_conf Settings

Aqui são tratadas as configurações do `vuurmuur_conf` especificamente. Vejamos suas opções:

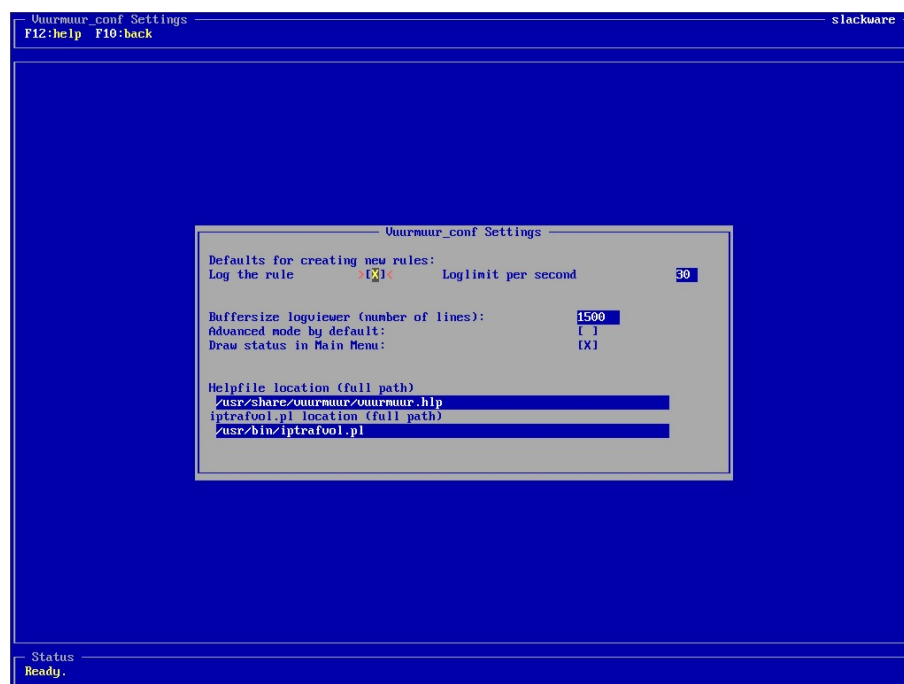


Figura 38: Vuurmuur_conf: Settings

`Defaults for creating new rules`: O que for definido aqui, aparecerá como padrão quando formos criar alguma regra. Temos apenas duas opções:

- `Log the rule`: Indica se queremos que essa regra gere registro. Ativada por padrão.
- `Loglimit per second`: O limite de logs por segundo gerado por esta regra.

`Buffersize logviewer`: Especificamos aqui o número de linhas que deverá servir de buffer no `Logviewer`. Isto representa o quanto poderemos ‘voltar’ a visualização das linhas, ou seja, quantas delas permanecem armazenadas em relação à que estiver sendo processada naquele instante.

`Advanced mode by default`: O `Vuurmuur` possui algumas opções que não são exibidas por padrão, em diversas partes do programa. Segundo o autor do programa, a razão para isto é a de não ‘encher’ o usuário com opções. Entretanto, marcando este campo, teremos todas as opções sendo exibidas. Recomendável.

`Draw status in Main menu`: Se este item estiver desativado, não aparecerá, no menu principal, o quadro lateral direito de avisos sobre as conexões com o `vuurmuur`, avisos sobre inconformidades com objetos, etc.

`Helpfile location+`: o caminho completo para o arquivo de ajuda, que contém todos os textos exibidos nas telas de ajuda (<F12>).

`iptraf.vol location`: Especificamos aqui o caminho completo para o script do `Ip Traffic Volume`, usado para gerar todo o conteúdo da seção `Bandwidth`, como citado.

3.4.5 Sobre os idiomas

Como citado na seção 3.1 (p.25), o `Vuurmuur` tem um ótimo suporte à internacionalização. Ele utiliza a biblioteca `gettext` para disponibilizar este suporte

Atualmente, as traduções aplicam-se ao módulo `vuurmuur_conf`, que é onde as coisas realmente acontecem. Temos suporte às línguas inglesa (padrão), holandesa, russa, alemã e português do Brasil (mantida hoje, não por acaso, por este autor ;)).

A linguagem do `Vuurmuur` é automaticamente selecionada de acordo com a variável de sistema `LANG`. Caso a variável do sistema seja diferente da linguagem desejada, podemos ajustá-la com o seguinte comando:

```
# LANG=<código_da_linguagem>
```

Exemplo:

```
# LANG=pt_BR
```

Devido às dificuldades quanto a flexibilização do layout exibido, algumas traduções (dentre elas a Português do Brasil) ainda apresentam alguns problemas especificamente de layout. Sendo assim, é perfeitamente normal encontrar, em alguma das internacionalizações, algumas divergências visuais. Assim como o programa, as traduções encontram-se em estágio *beta*. Porém, costumam ser altamente atualizadas, assim como as próprias versões do Vuurmuur.

Exemplo do `vuurmuur_conf` com a variável `LANG` ajustada para `pt_BR`:

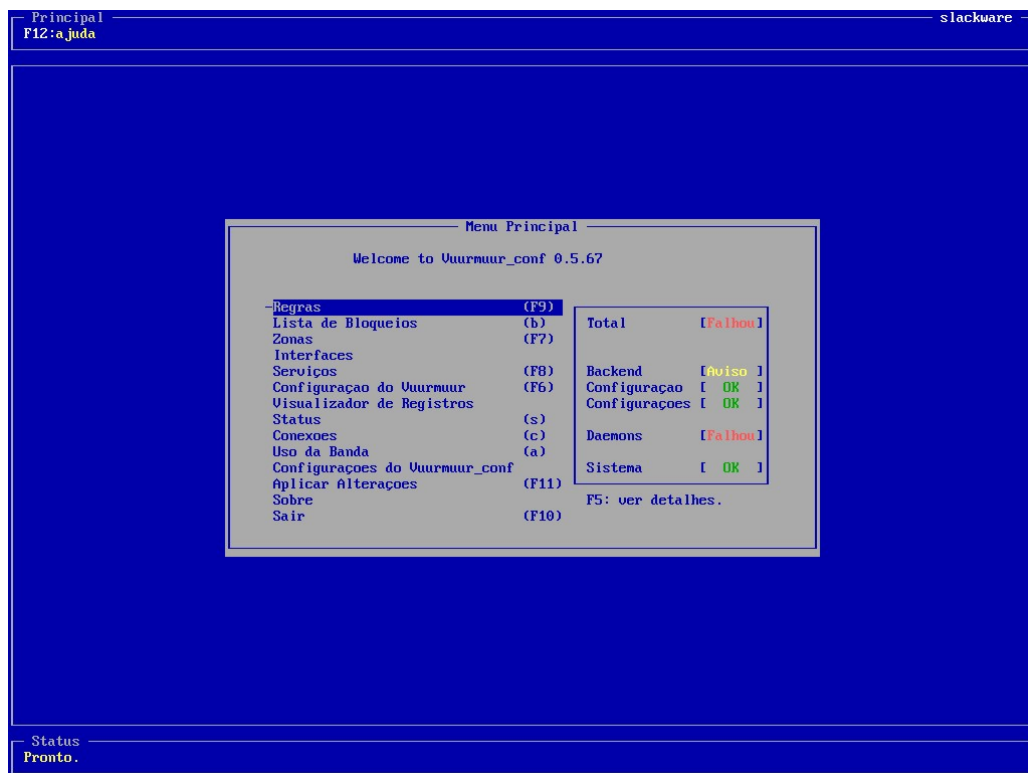


Figura 39: Vuurmuur_conf em Português do Brasil

3.5 Montando o cenário-exemplo

Neste capítulo, iremos propor um cenário simples, com algumas máquinas e condições, e definirmos as zonas, interfaces, redes, hosts, serviços, enfim, criaremos toda a condição necessária para o funcionamento do firewall, de acordo com este cenário. Somente serão abordados aqui as seções **Rules**, **Blocklist**, **Interfaces** e **Services**, uma vez que todos os outros itens já foram cobertos em outras seções.

Em nosso cenário exemplo, teremos uma rede com as seguintes características:

- Número de Servidores: **3** (incluindo o firewall)
- Número de Máquinas Cliente: **10**
- Impressoras com endereço IP: **1**
- Endereço de Rede das máquinas cliente: **192.168.1.0/24**
- Endereço de rede dos servidores: **192.168.5.0/24**
- Endereços IP válidos: somente 1 (**200.200.200.200**)

Em nossa rede, teremos diferentes ‘privilégios’ de acesso, de acordo com cada grupo (departamento), podendo haver exceções dentro do grupo. Na tabela 1, temos a lista de hosts, com seus determinados endereços e departamentos:

Máquinas Cliente			
Nº	Nome do Host	Departamento	Endereço IP
1	Amanda	Informática	192.168.1.1
2	Bruno	Informática	192.168.1.2
3	Carlos	Administrativo	192.168.1.3
4	Fernanda	Administrativo	192.168.1.4
5	Gustavo	Administrativo	192.168.1.5
6	Joaquim	Contabilidade	192.168.1.6
7	Mario	Contabilidade	192.168.1.7
8	Pedro	Financeiro	192.168.1.8
9	Roberta	Compras	192.168.1.9
10	Teodoro	Compras	192.168.1.10
11	Hp 9000dn	-	192.168.1.20

Tabela 1: Clientes: Hosts, Ips e Departamento

Da mesma forma, a dos servidores, incluindo as três interfaces da máquina firewall:

Servidores		
Nº	Nome do Host	Endereço IP
12	FileServer	192.168.5.1
13	MailServer	192.168.5.2
14	Firewall (Rede Servidores)	192.168.5.254
14	Firewall (Rede Clientes)	192.168.1.254
14	Firewall (Internet)	200.200.200.200

Tabela 2: Servidores: Hosts e Ips

Teremos um serviço hipotético próprio da empresa, o **SAPo**, utilizado no sistema ERP, que utilizará as portas 6699 e 9966, protocolo TCP.

Em nosso cenário, o firewall também desempenha o papel de roteador da Internet. Sendo assim, será ele o gateway das máquinas cliente e servidor. Abaixo estão listados os privilégios que cada grupo terá, bem como as exceções, se houver:

- O grupo **Informática** terá acesso livre a qualquer coisa, interna e externamente. Porém, o usuário Bruno não deverá ser capaz de conectar-se via SSH no **FileServer**.
- O grupo **Administrativo** terá acesso à Internet, poderá mandar e receber emails (**MailServer**), acessar o samba (**FileServer**) conectar no serviço MSN e utilizar o **SAPo** (**FileServer**). Entretanto, eles não deverão ser capazes de enviar qualquer coisa para o site do UOL (200.100.200.100¹⁵).
- O grupo **Contabilidade** poderá somente se conectar no serviço MSN, mandar e receber emails, e pingar endereços externos.
- O grupo **Financeiro** poderá utilizar o **SAPo**, conectar-se via ftp no **FileServer**, e acessar o site MoneyWorld (200.150.10.5).
- O grupo **Compras** não poderá fazer nada, com exceção do usuário Teodoro, que poderá mandar e receber emails.

¹⁵Todos os endereços IPs utilizados aqui são hipotéticos. Qualquer coincidência ou referência à qualquer ip externo é meramente ilustrativa.

- A impressora HP 9000dn terá acesso ao samba.

Da mesma forma, temos algumas regras que se aplicam aos servidores, e à Internet em geral:

- Os serviços oferecidos pelo **FileServer** estarão disponíveis somente em nossa rede local.
- Todas as conexões vindas da Internet com destino às portas 25 (pop3) e 110 (stmp) deverão ser encaminhadas para o **MailServer**.
- Somente a IBM (200.200.200.210) poderá nos pingar à partir da Internet.
- Todo e qualquer pacote originado do IP 200.1.5.10 (Crackers) deverá ser completamente bloqueado.
- Somente o Linus (240.120.60.2) poderá conectar-se via SSH em nossa máquina firewall, a partir da Internet.

Dadas as condições de nosso cenário, vamos à montagem do cenário-exemplo no Vuurmuur. É importante observar que a maneira de criação do cenário imposta aqui representa particularmente a forma de pensar do autor, e que existem vários caminhos para se chegar ao resultado desejado, ficando, assim, a critério do administrador de redes, um desenvolvimento para o seu caso específico.

3.5.1 Elaborando o projeto

Como sabemos, é de extrema importância elaborarmos, antes de sairmos criando regras, um bom projeto para nosso firewall. Mas, o que seria um projeto para o firewall?

Um projeto para o firewall define a forma de pensar para a criação das regras, define o que se poderá agrupar, e por qual critério, define as exceções, enfim, define a base teórica da criação das regras.

Em nosso caso, como apresentamos uma diversidade imensa de privilégios, fica difícil estabelecermos padrões de agrupamento. Sendo assim, a melhor maneira que encontramos para possíveis grupos é justamente o critério de serviços.

Em nosso exemplo, estaremos fazendo a definição das três interfaces de rede do firewall, a definição de nosso serviço interno, o cadastramento dos hosts, redes e zonas, a criação dos grupos citados acima, separados por tipo de serviço, com os respectivos hosts, e finalmente a criação das regras.

3.5.2 Interfaces

Em nosso firewall, teremos três interfaces de rede:

1. Interface `eth0` que recebe o IP válido da Internet, com endereço `200.200.200.200`
2. Interface `eth1` que está conectada à rede dos servidores, com endereço `192.168.5.254`
3. Interface `eth2` que está conectada à rede dos clientes, com endereço `192.168.1.254`.

Começaremos o processo fazendo o cadastro das interfaces, pois ele será necessário na criação das redes. Na figura 40, temos o exemplo de cadastro da interface `eth0`:

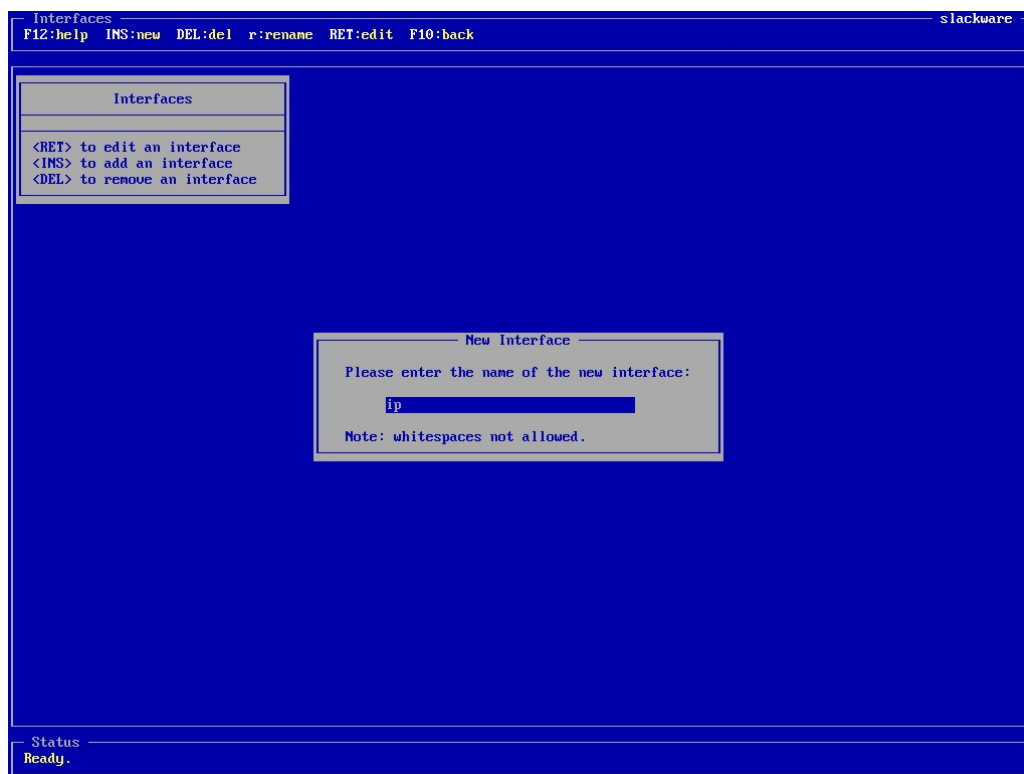


Figura 40: Interfaces: Atribuindo um nome

Quando entramos na seção `Interfaces`, ela encontra-se em branco. Para iniciarmos a adição de uma nova interface, pressionamos `<INS>`. Aparece uma janela pedindo para

digitarmos um nome para esta interface. É importante que coloquemos um nome que referencie no futuro o que ela representa. Em nosso caso, iremos chamar de 'ip', para referenciar nosso link de Internet, que poderia ser um serviço de **Ip** dedicado.

Definimos um nome e apertamos <ENTER>. Aparece uma janela de propriedades, como pode ser visto na figura 41. Temos algumas opções:

Active: Informamos se a interface está ativa ou não. Se estiver definida como 'No', não seremos capazes de fazer qualquer coisa utilizando-a. Deixamos, por padrão, em 'Yes'.

IPAddress: Aqui definimos o endereço IP da interface. Caso ela seja dinâmica, deixamos este campo em branco, e marcamos a caixa **Dynamic**. Como não é nosso caso, preenchemos seu endereço, 200.200.200.200.

Device: Aqui associamos a interface que estamos criando à um dispositivo existente no sistema, em nosso caso, **eth0**.

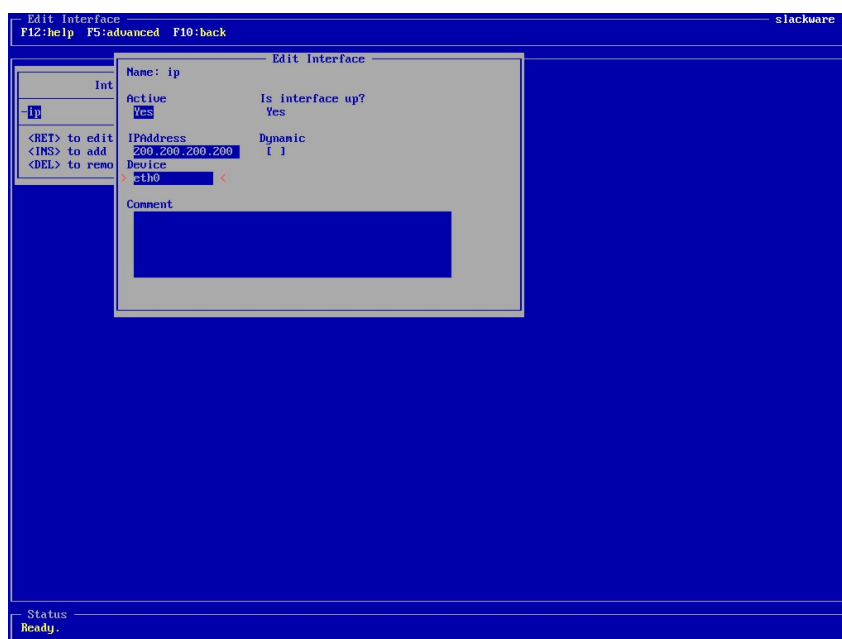


Figura 41: Interfaces: Definindo propriedades

Temos também um campo livre para adicionarmos qualquer comentário que julgemos útil. Confirma-se pressionando <F10>. Repetimos o procedimento, informando os nomes **clientes** e **servers** para as outras 2 interfaces, de acordo com nosso cenário, sempre colocando nomes intuitivos. Para editarmos uma interface, apertamos <ENTER>; para

renomearmos, apertamos <r> e, para apagarmos, apertamos , sempre sobre o nome da interface. Ficamos assim:

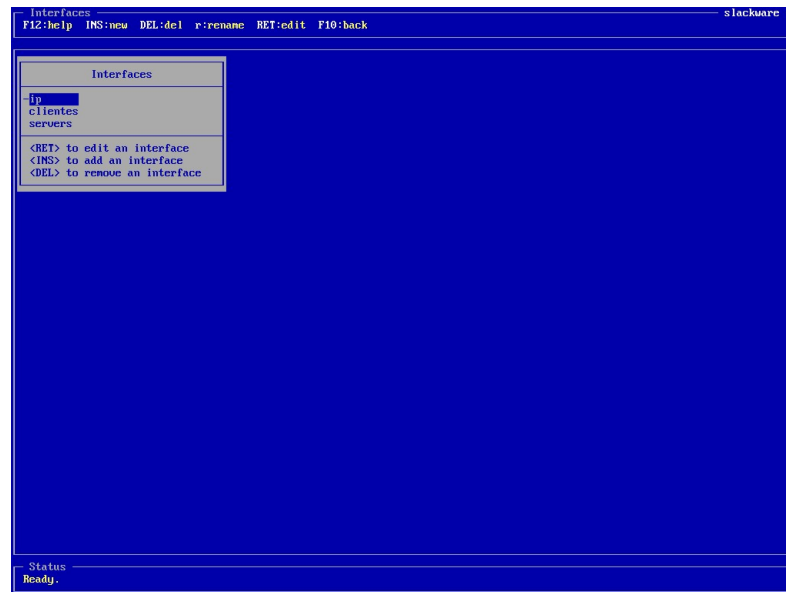


Figura 42: Interfaces criadas

3.5.3 Services

Iremos agora definir nosso serviço interno, o **SAPo**, e conhecermos a seção de serviços. Assim que entramos na seção **Services**, já encontramos uma grandiosa lista de serviços pré-definidos, como pode ser observado na figura 43:

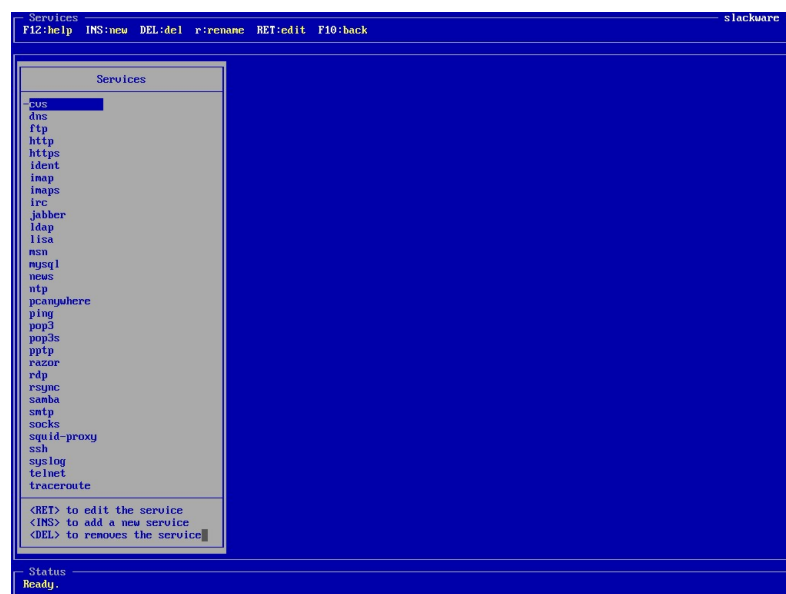


Figura 43: Serviços: grande quantidade de serviços pré-definidos

O Vuurmuur já nos ‘corta caminho’, nos dispensando de cadastrarmos tantos serviços que usamos¹⁶. Iniciamos a adição de um novo serviço pressionando <INS>, e fornecendo um nome para nosso serviço, no caso, **SAPo**:

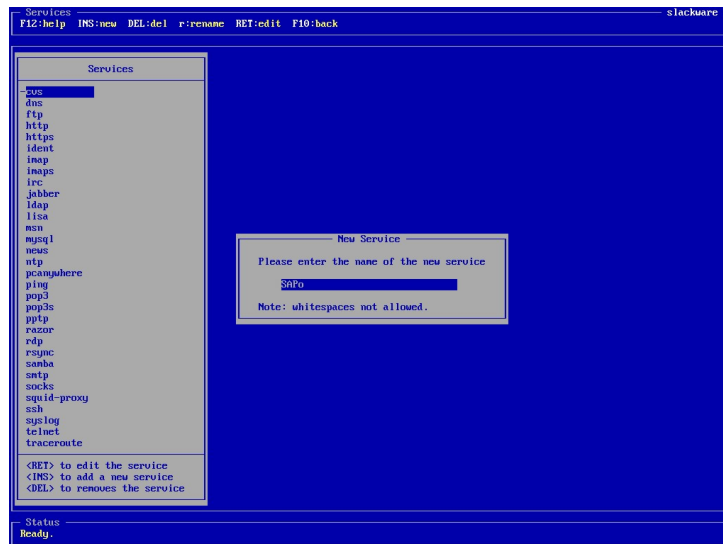


Figura 44: Serviços: Adicionando um novo serviço

Na figura 45, poderemos observar as possíveis opções para a adição de um novo serviço:

Active: Exatamente o mesmo de Interfaces. Este campo aparece em todos os lugares que contenham objetos no Vuurmuur. Em todos os casos, quando definimos ‘No’, nenhuma regra é criada para eles.

Broadcast: Aqui informamos se este serviço utiliza broadcasts de rede, pois o Vuurmuur usará padrões diferentes na criação da regra. ‘No’.

Protocol Helper: Aqui informamos se o serviço utiliza algum módulo auxiliar de protocolo, como o `ip_nat_ftp`, por exemplo.

¹⁶É importante observar novamente a **necessidade** de termos serviços para todo e qualquer tráfego que for passar pelo firewall, desde um simples `ping` até serviços que utilizam várias ranges de portas, como o `samba`, por exemplo.

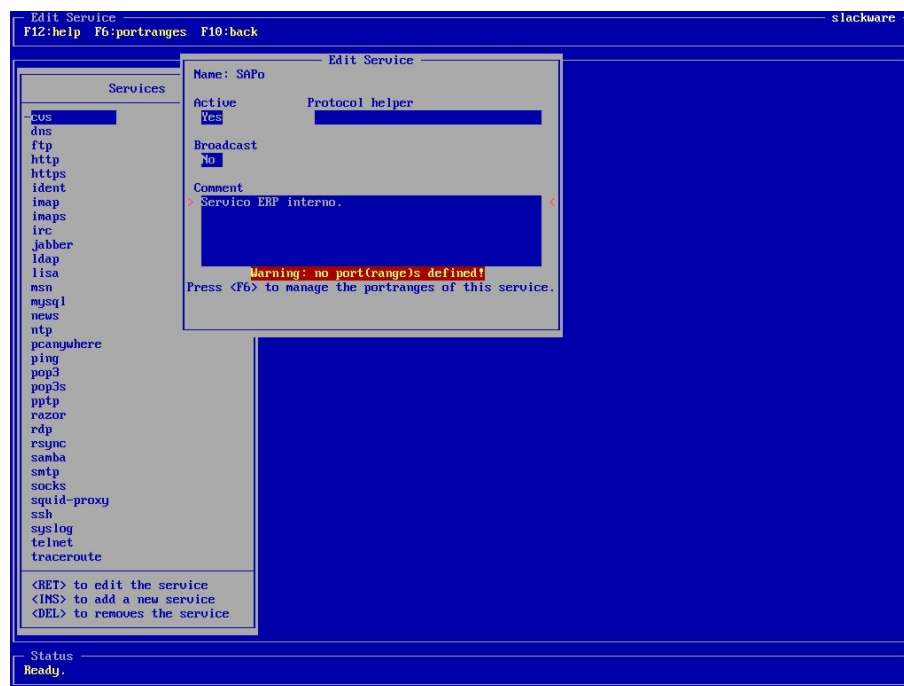


Figura 45: Serviços: Diálogo de propriedades

Defina um comentário, caso queira. Observe a tarja vermelha com o aviso **Warning: no port(range)s defined!**: Ele está nos dizendo que não foram criadas as associações das portas utilizadas para este serviço. Pressionamos <F6> para gerenciar as faixas de porta.

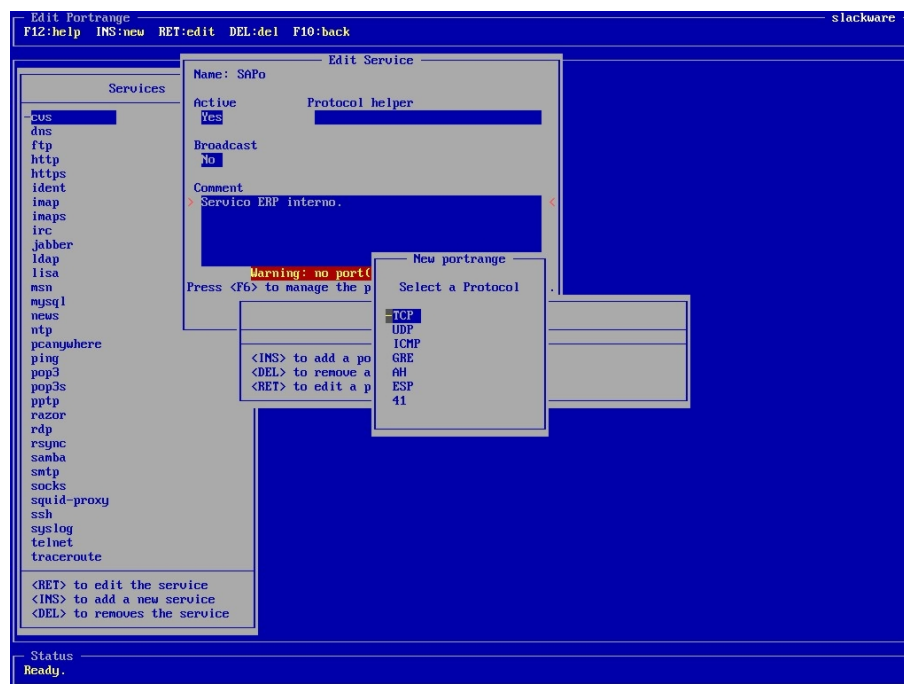


Figura 46: Serviços: Definindo portas

Aparece a lista de *portranges*. Observe que podemos ter várias faixas de porta para um mesmo serviço, bastando para isso apenas cadastrarmos-las aqui. Pressionamos <INS> para inserir uma nova faixa de portas, e selecionamos qual protocolo desejaremos usar com esta faixa. Caso sejam escolhidos os protocolos GRE, AH, ESP ou 41, o Vuurmuur automaticamente os adicionará na lista de *portranges*, junto com a mensagem: “this protocol doesn’t use ports”. Em nosso caso, escolhemos TCP.

Como poderemos observar na figura 47, este diálogo nos dá a opção de preenchermos quatro campos:

Nos campos Low e High, em Source, indicamos a faixa de portas de origem. Geralmente, os serviços utilizam portas randômicas acima de 1024 para estabelecerem serviços em hosts remotos. O Vuurmuur traz já preenchido este intervalo.

Nos campos Low e High, em Destination, definimos o intervalo de portas de destino utilizado para um serviço. Caso este utilize somente uma porta, definimo-la em Low, e deixamos o campo High em branco. Em nosso caso, como são duas portas distintas, adicionaremos duas entradas separadamente.

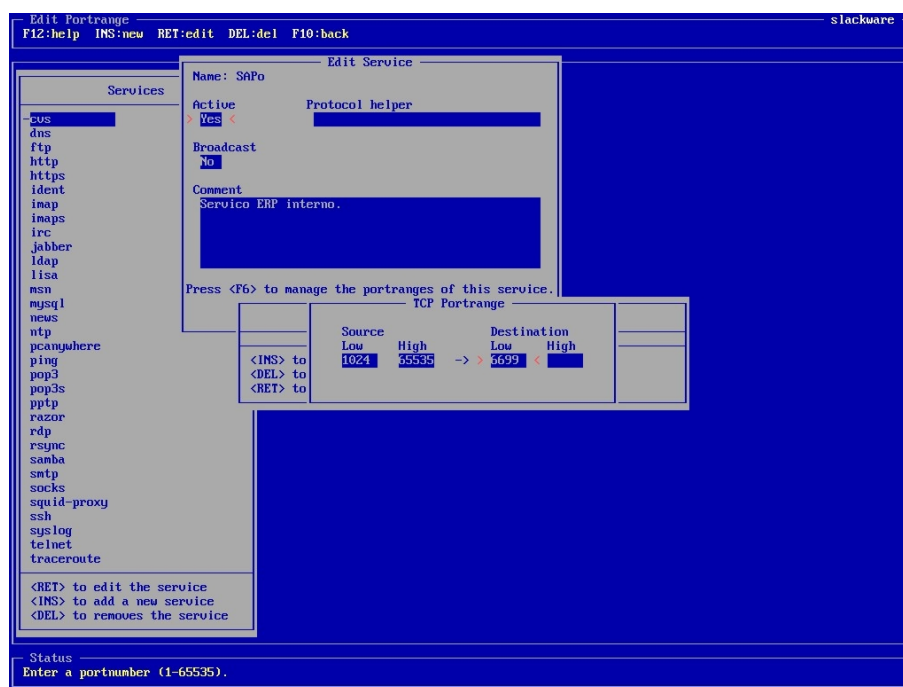


Figura 47: Serviços: Portas de Origem e Destino

Repetimos a operação até termos cadastrado todas as portas utilizadas para o serviço. Pressionando <F10> repetidamente, vamos saindo dos menus, e subindo níveis. Caso fosse necessário, poderíamos adicionar mais serviços, editar serviços existentes, ativá-los,

desativá-los, enfim, toda a operação de manutenção desta natureza.

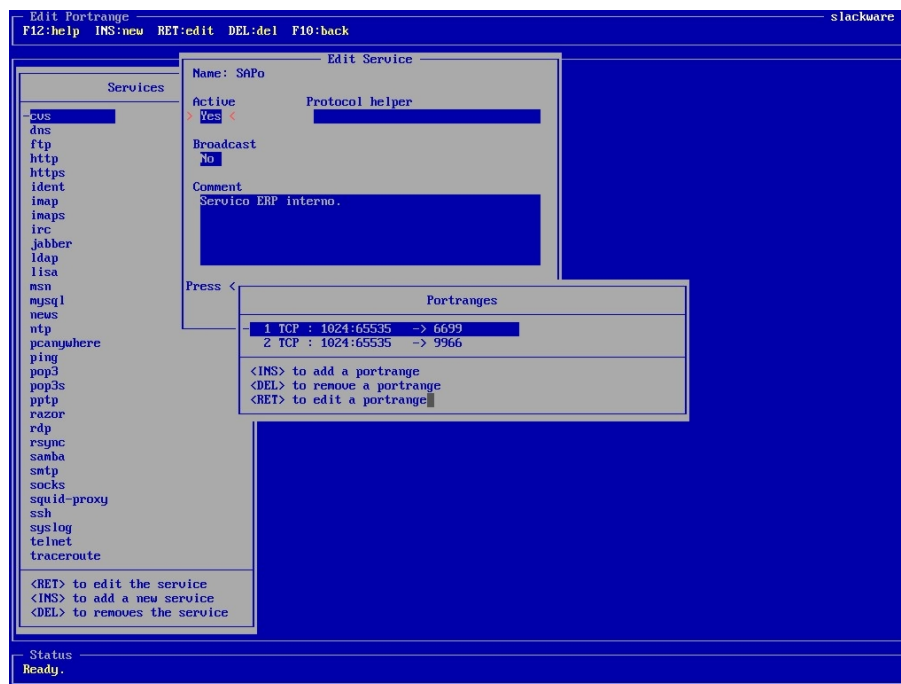


Figura 48: Serviços: Portranges definidos

3.5.4 Zones

Estamos prontos para a definição de nossa visão lógica da rede, como foi exemplificada na subseção 3.2.7 (p.29). Aqui, faremos a definição completa de nossa rede: o cadastro das redes propriamente ditas, definição de zonas, cadastro de hosts e grupos. É importante que se tenha uma grande atenção ao definir estes objetos de forma correta, pois será decisório na elaboração de regras, bem como na conversão feita pelo Vuurmuur para o formato nativo do iptables.

Assim que entramos na seção ‘Zones’, temos três zonas pré-cadastradas, que na maioria das vezes atendem às necessidades dos administradores:

- **int**: Referenciando a parte interna de nossa rede.
- **ext**: Referenciando tudo que está fora de nossa rede. Esta zona já vem com uma rede cadastrada, a ‘internet’. Endereços de páginas na Internet ou outros servidores externos deverão ser cadastrados em uma rede como esta.
- **dmz**: Referente à *Zona Desmilitarizada*, utilizada em técnicas de segurança para isolarmos servidores em uma rede segura.

Em nosso caso, iremos começar do zero, criando nossas próprias redes, mas poderíamos usar as redes pré-definidas sem o menor problema. Iniciamos a criação de uma nova rede pressionando <INS>, e digitando o nome desejado para a zona:

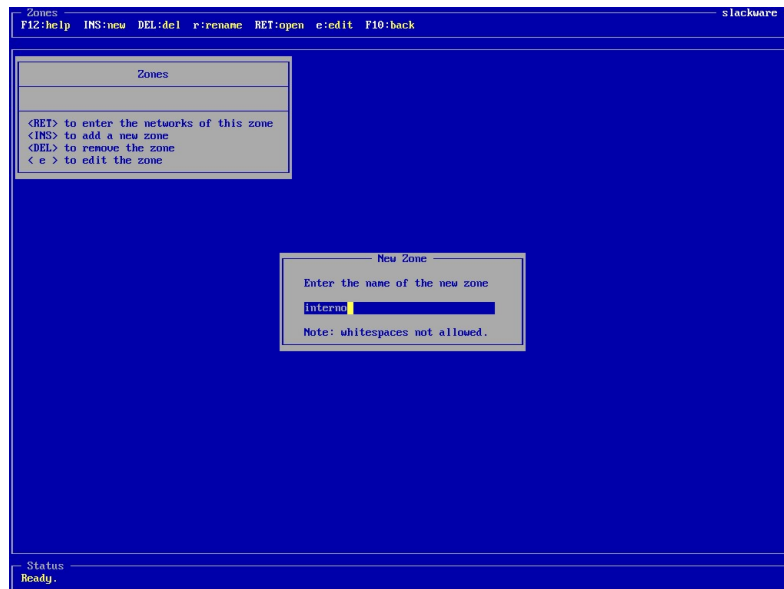


Figura 49: Zonas: Definindo um nome

Em sua configuração, apenas definimos se ela está ativa ou não, e digitamos um comentário, caso queiramos. Observe que ela exibe, à direita, uma contagem sobre o número de redes, hosts e grupos que ela contém. Como acabou de ser criada, o número para todos é 0 (zero).

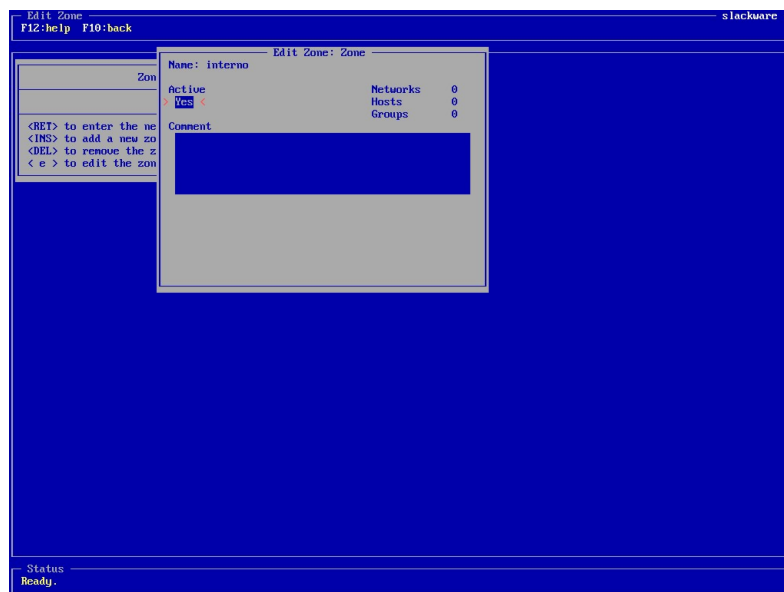


Figura 50: Zonas: Adicionando

Em nosso cenário, criaremos apenas duas zonas: **interno** e **externo**. Em projetos mais complexos de firewall, poderiam haver até 5 zonas ou mais, dependendo da forma de visão do administrador.

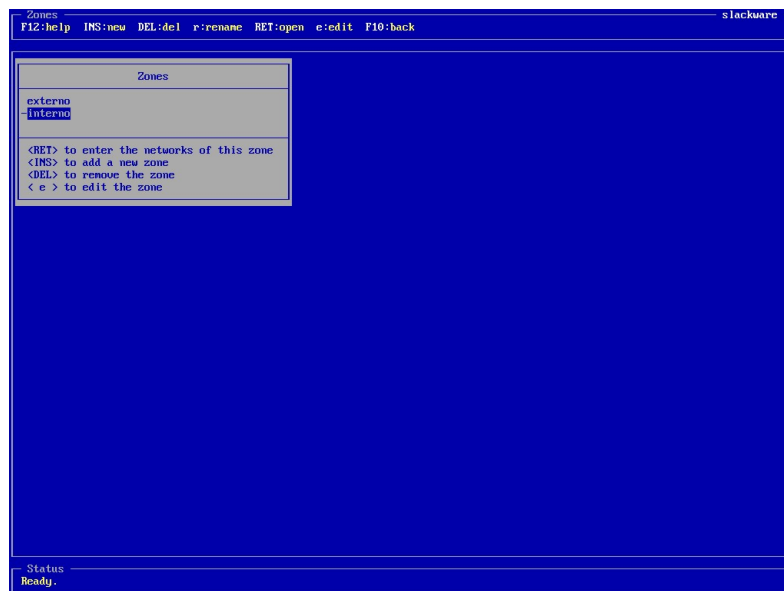


Figura 51: Zonas: Projeto simples

Devemos definir as redes agora: para isso, selecionamos a zona para a qual iremos criar uma rede, e pressionamos **<ENTER>**. Com isso, abre-se a janela de redes. Para adicionarmos uma rede, pressionamos **<INS>**, e digitamos o nome desejado, como de costume. Vamos criar a rede 'internet', para a zona 'externo'.

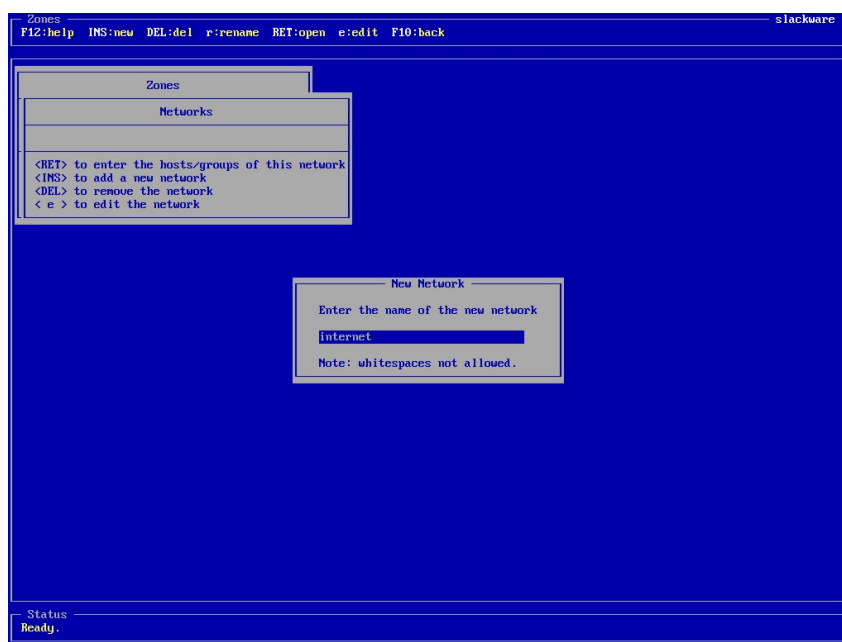


Figura 52: Zonas: Redes

Ao confirmarmos o nome, teremos a janela de edição da rede, essa sim com bastante opções. Veja na figura 53:

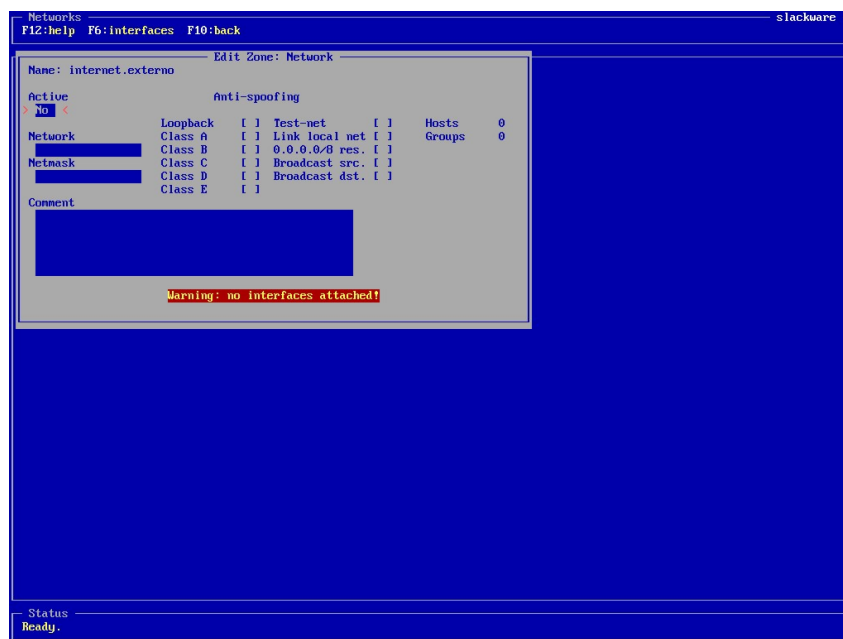


Figura 53: Redes: Adicionando

Repare, no campo Name, a nomenclatura utilizada pelo Vuurmuur: `internet.externo`. É a forma hierárquica dos objetos do Vuurmuur em ação. Vamos às opções:

Active: Nossa já conhecida função. Porém, vale adicionar aqui uma observação extremamente útil: Se, por qualquer motivo, em qualquer momento, precisemos bloquear o acesso de determinados objetos, podemos definir este campo para 'No'. Desta forma, não serão criadas regras para nenhum objeto listado dentre deste grupo. Isto vale para redes, zonas, hosts e grupos. Exemplo: em uma emergência, preciso que, rapidamente, todos meus servidores fiquem inacessíveis. Se eu tiver uma classe somente para os servidores, como será a rede 'servers', em nosso caso, bastaria apenas definir o Active para 'No', e aplicar as alterações. Extremamente simples e rápido.

Network: Aqui colocamos o endereço de rede para a nossa rede especificada. A rede *Internet* é um caso à parte: como deve abrigar qualquer host externo, não pode ter restrições de endereços de redes. Sendo assim, definimos o endereço de rede para 0.0.0.0. Para os outros casos, colocamos o endereço de rede normalmente.

Netmask: Máscara de rede. Tenha a certeza de colocar a máscara certa aqui, pois o

Vuurmuur irá fazer uma verificação do endereço de cada host com ela, para assegurar que os hosts realmente pertencem a esta rede. Para o caso da rede 'internet', a máscara a ser colocada aqui também é 0.0.0.0.

Anti-Spoofing Protections: O Vuurmuur pode criar regras de proteção automáticas contra anti-spoofing. Para isso, basta apenas marcarmos que tipo de proteção desejamos. Obviamente, se tivermos aqui uma rede /24, não marcaremos a opção **Class C**. Para a rede Internet, marcamos todas as opções.

Comment: Digite aqui um comentário opcional.

Veja na figura 54 como ficaria o objeto 'internet.externo'. Repare também no aviso com fundo vermelho, **Warning: no interfaces attached!**, que avisa que não temos nenhuma interface associada a esta rede.

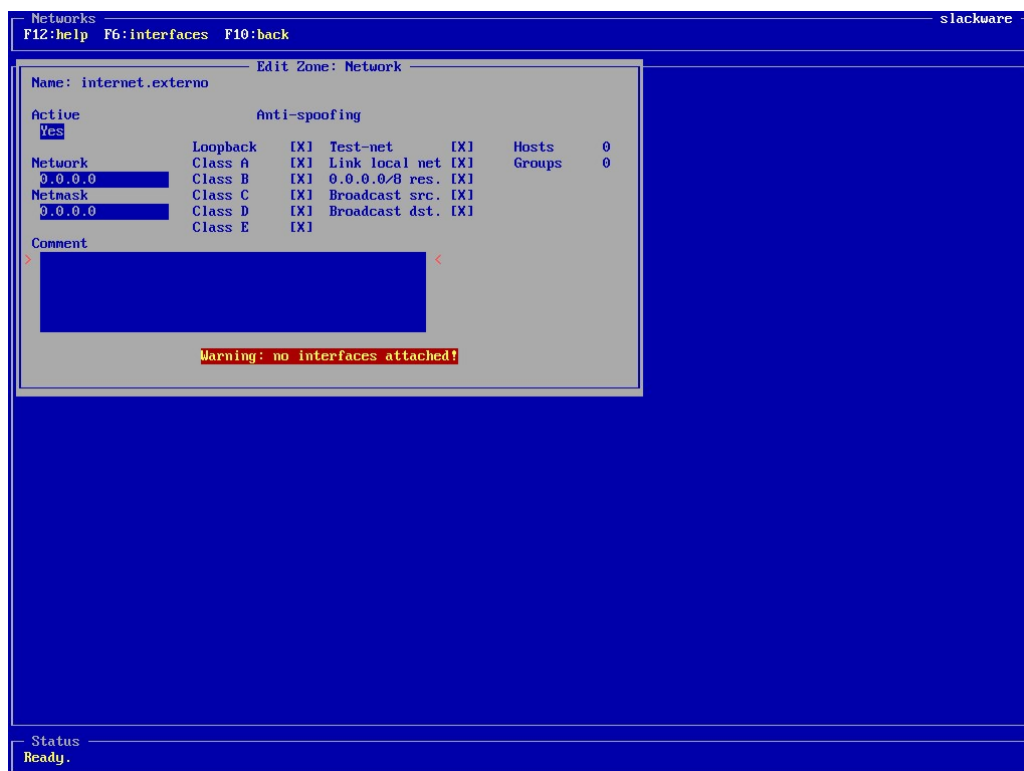


Figura 54: Redes: Adicionando rede Internet

A associação de interfaces às redes é simplesmente vital para o correto funcionamento do Vuurmuur: é através desta associação que o Vuurmuur saberá, na hora de criação de

regras, quem faz parte da rede interna, e quem faz parte da Internet.

Pressione <F6> para o gerenciamento de interfaces. Aparece uma lista em branco. Para adicionarmos uma interface, pressionamos <INS>. Podemos ter mais de uma interface para a mesma rede. Ao pressionar <INS>, aparece a lista das interfaces, que criamos na subseção 3.5.2 (p.70). Confirmamos com <ENTER>, e adicionamos quantas interfaces forem necessárias.

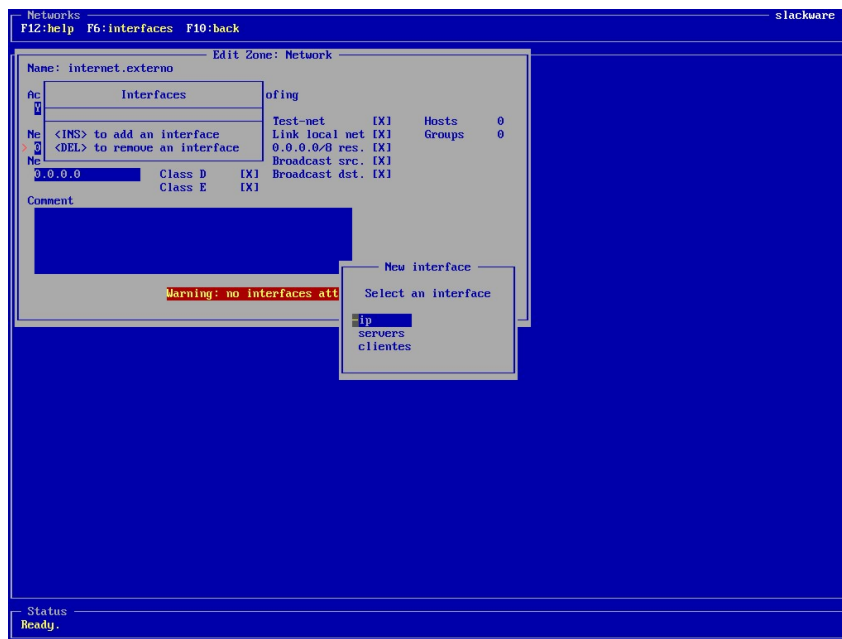


Figura 55: Redes: Atribuindo interfaces

Na zona 'interno', teremos duas redes: a 'servers' e a 'clientes'. O processo de criação é exatamente igual ao que acabamos de ver. Alguns *screenshots*:

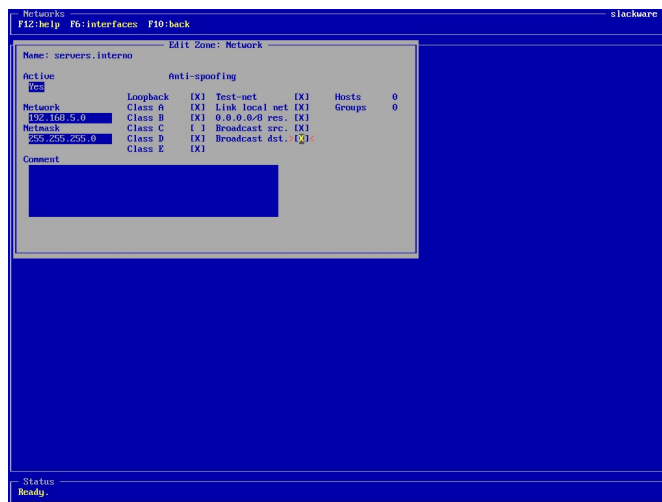


Figura 56: Redes: Criando a rede 'servers'

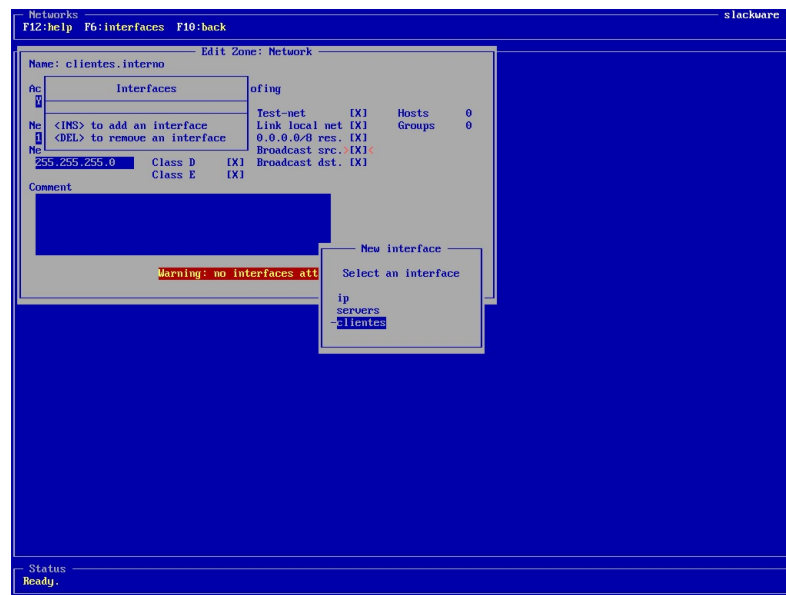


Figura 57: Redes: Criando a rede 'clientes'

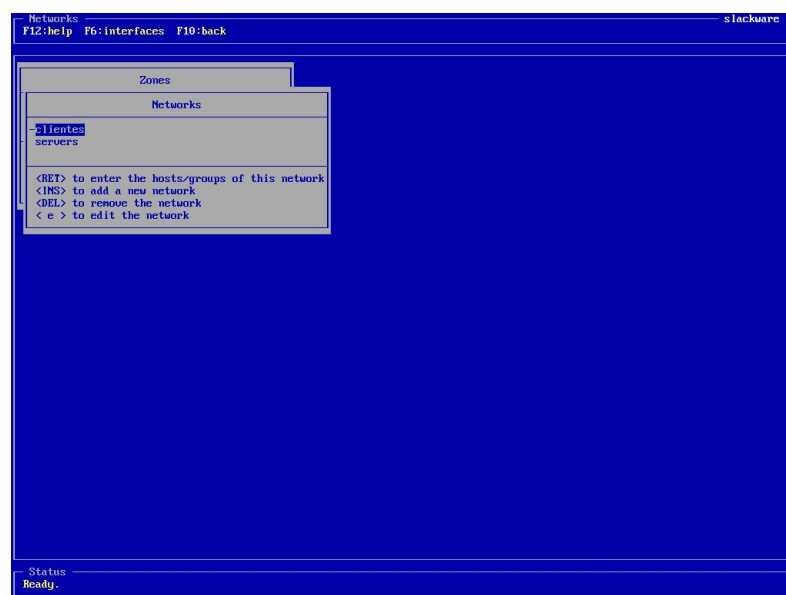


Figura 58: Zonas: Duas redes em uma mesma zona

Após criadas corretamente as redes, devemos partir para a árdua tarefa de criação de grupos e hosts: pressionando-se <ENTER> em cima do nome da rede, para que apareçam as opções **Hosts**, **Groups** ou **Network**. Todas elas são auto-explicativas, e o processo para o início da inclusão de um host já é bastante conhecido. . .

Entramos em **Hosts**, apertamos <INS> para adicionar um novo host, e digitamos um nome para ele:

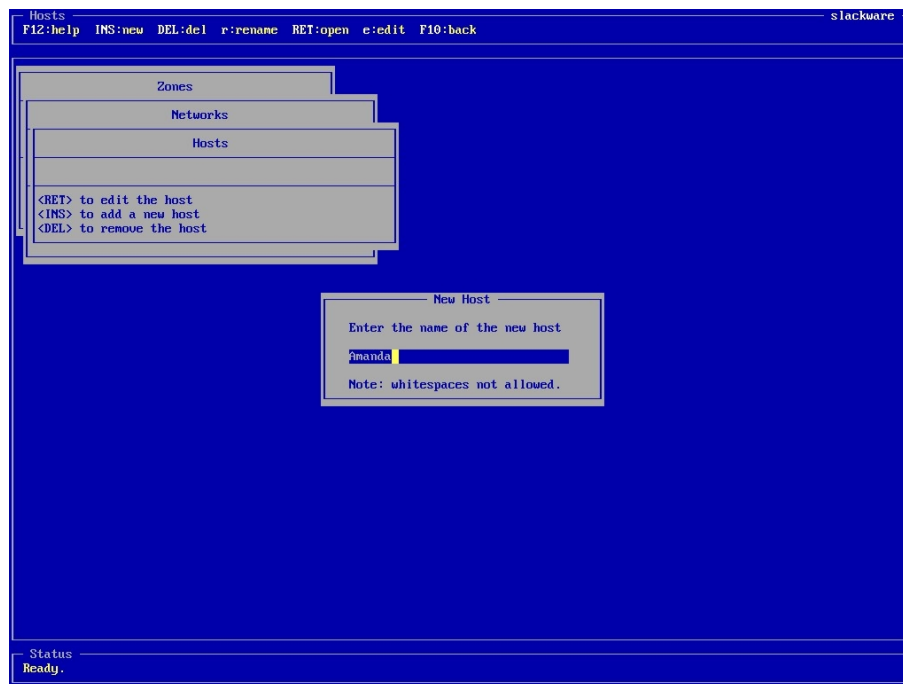


Figura 59: Hosts: Adicionando

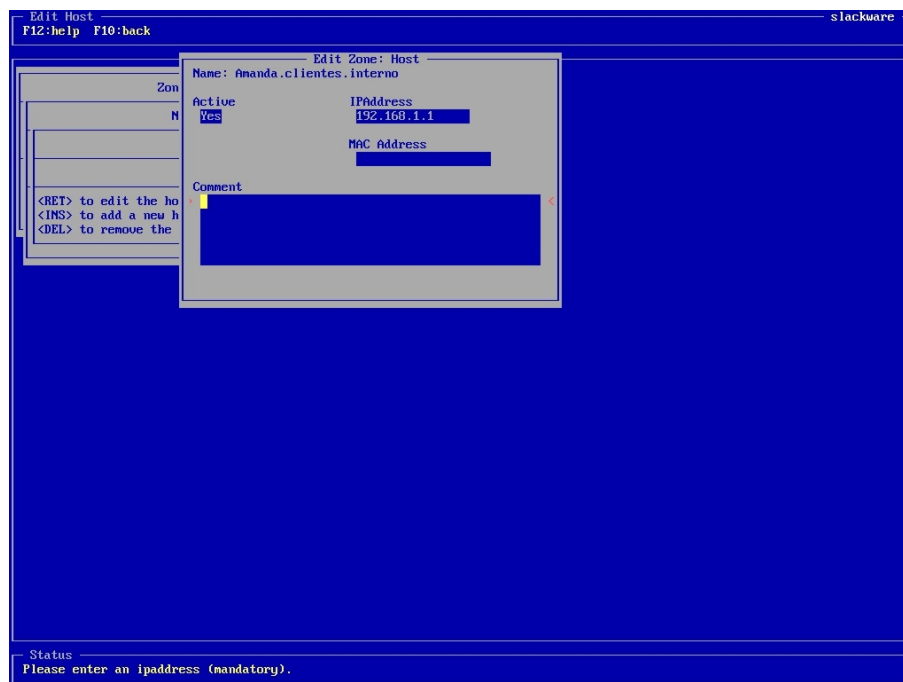


Figura 60: Hosts: Amanda.clientes.interno

A tela para a adição de hosts é simples: você define seu estado (**Active**) e comentário (**Comment**), e fora isso temos apenas duas opções:

IPAddress: Digite o endereço IP deste host. Deve ser condizente com o endereço de

rede e máscara cadastrados na rede qual este host pertence. É obrigatória a digitação deste item.

MAC Address: Item opcional, mas que aumenta um nível em sua segurança, fazendo a amarração do endereço IP com o endereço físico da interface de rede do cliente.

Repare novamente na figura 60 a nomenclatura utilizada: `Amanda.clientes.interno`.

Repetimos o processo, até que tenham sido cadastrados todas as nossas máquinas cliente:

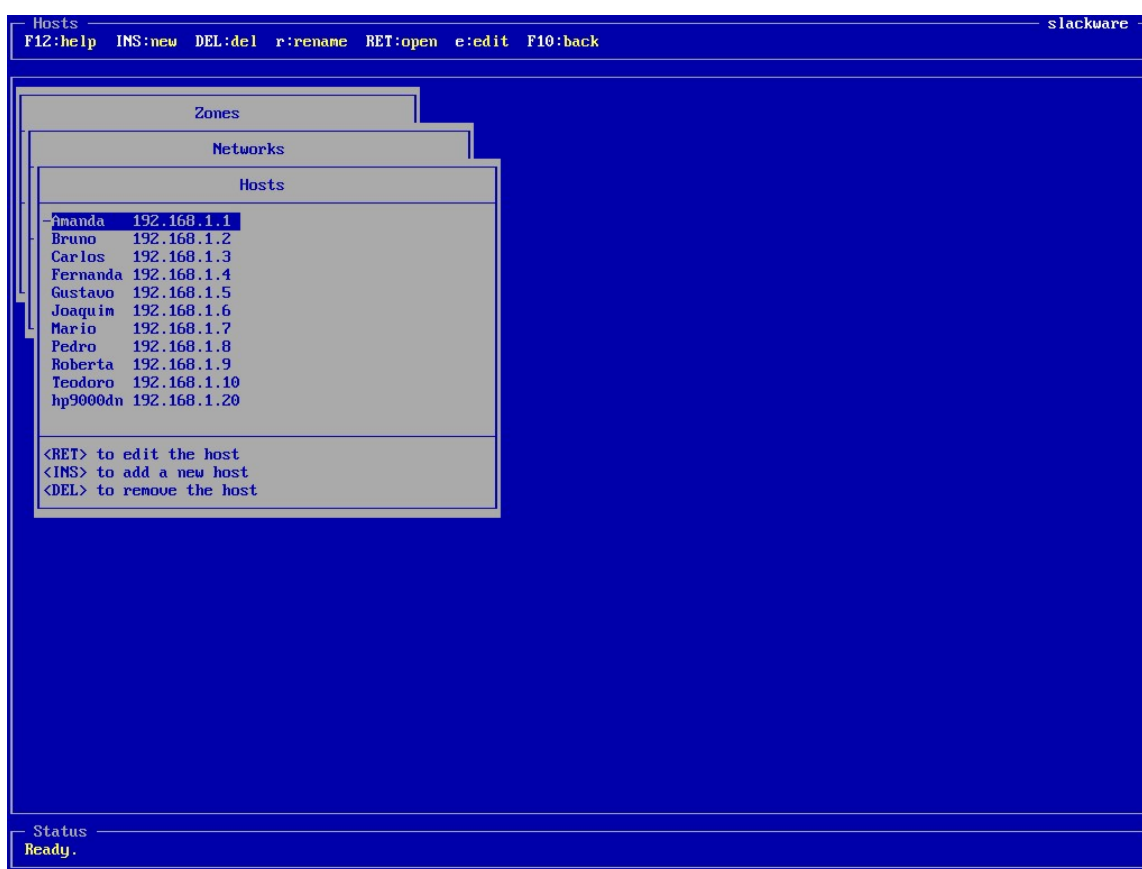


Figura 61: Hosts: Lista de hosts

O processo para a criação dos hosts na rede `servers` é o mesmo, sempre obedecendo o critério de endereços IP relacionados às redes nas quais se encontram. Se observarmos bem, nas especificações exatas na página 68 sobre as considerações e regras para os hosts e grupos, poderemos observar que ficaram vários objetos de fora a serem cadastrados aqui. Quais seriam eles?

Caso ainda não esteja claro: na hora da criação de regras, o que iremos informar são apenas os objetos previamente cadastrados. Sendo assim, os 'hosts externos' **também**

precisam ser cadastrados. Eles não fazem parte de nossa rede local, mas fazem parte de outra rede cadastrada: a `internet.zone`, portanto, deverão também ser cadastrados, exatamente da mesma forma que os outros hosts:

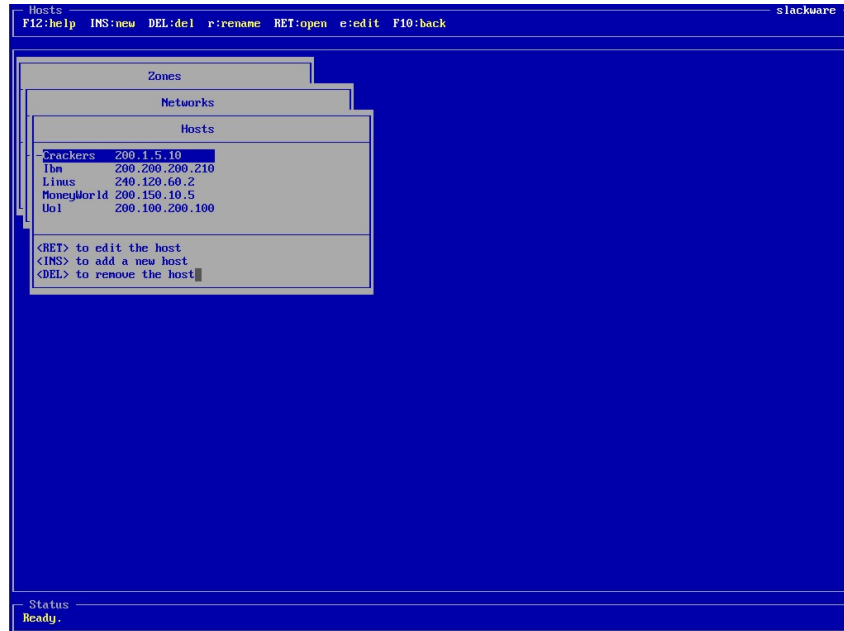


Figura 62: 'Hosts' na Internet

Finalmente, vamos à criação de grupos. De todos os objetos com qual o Vuurmuur trabalha, os grupos são os mais simples. Tudo que temos à fazer é dar um nome ao grupo, e adicionar seus membros.

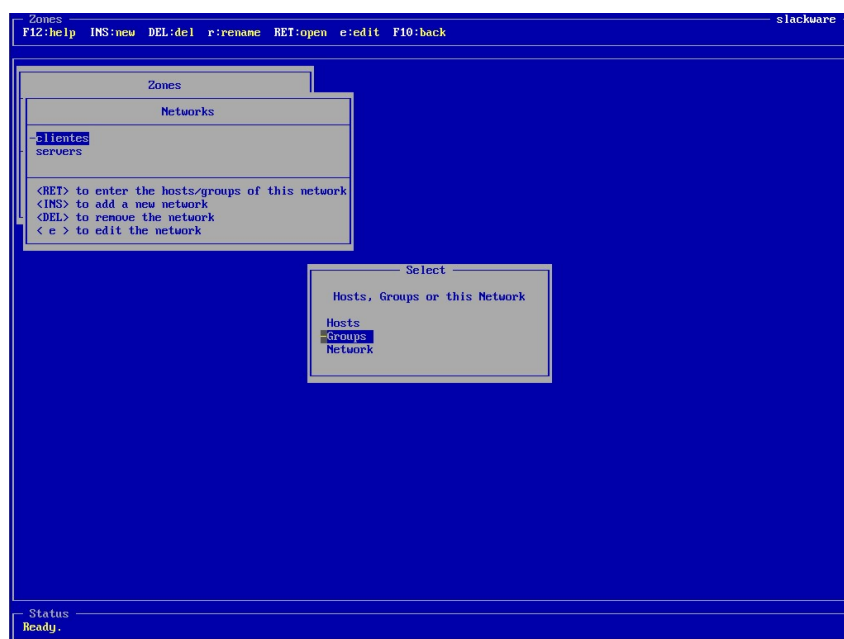


Figura 63: Grupos

É neste local que criamos os grupos. Como pôde ser observado, os grupos também são associados às redes, que por sua vez são associados às zonas. Um <ENTER> para entrarmos na listagem de grupos, e um <INS> para adicionarmos um grupo, informando um nome:

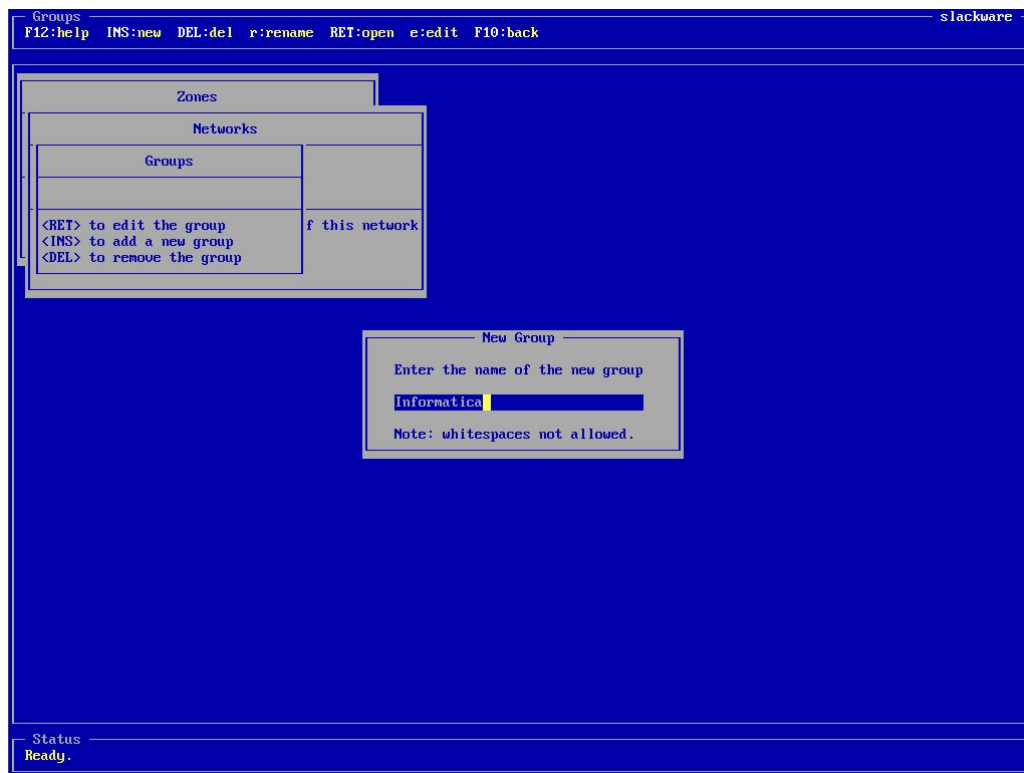


Figura 64: Grupos: Informatica

Como descrito, em nosso caso, os grupos serão associados aos departamentos, pois tivemos um padrão para eles. As exceções mencionadas nas regras serão feitas depois, diretamente nas regras. Tenha em mente os grupos como quaisquer (no mínimo 2) hosts que façam algo em comum, ou que tenham uma característica em comum. Já é o suficiente para se criar um grupo. Como também mencionado, um host pode fazer parte de vários grupos, sendo assim, podemos adequar nossas necessidades. Exemplo: Imaginem o seguinte cenário: Temos 5 hosts, o 1, 2, 3, 4 e 5. Temos 3 serviços: SSH, http e ftp. O 1 e o 5 deveriam acessar SSH, enquanto o 2, 3 e 4 deveriam acessar http. **Porém**, o 1, o 2 e o 5 deveriam acessar ftp. Neste caso, poderíamos criar grupos de serviço, 3 (três) no caso, que iriam conter as pessoas com direitos de acesso a cada serviço.

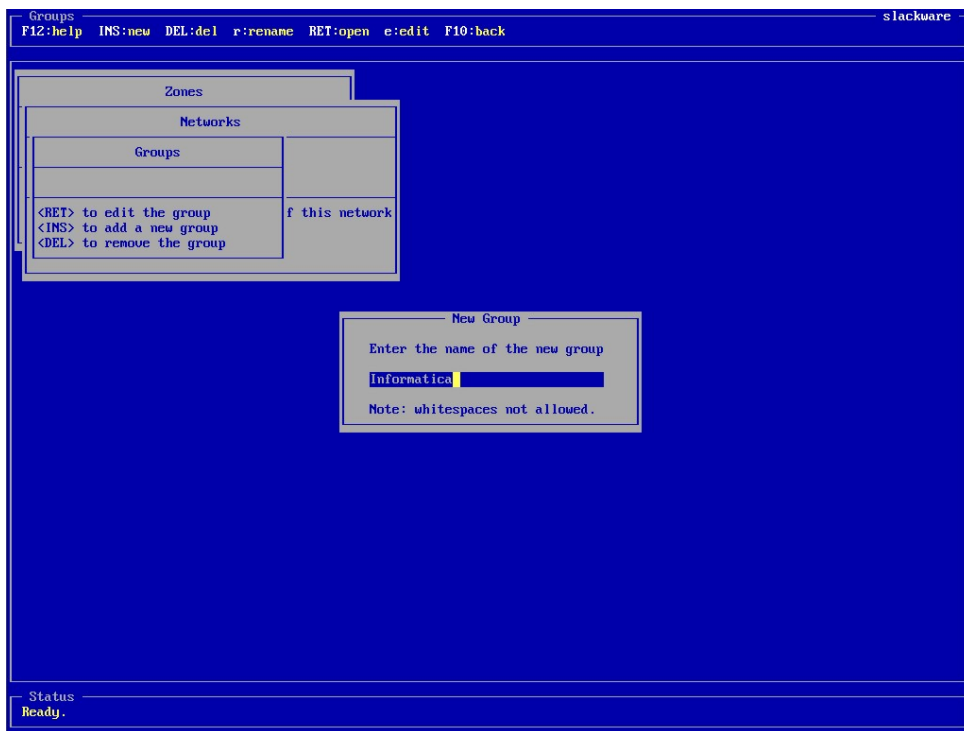


Figura 65: Grupos: Simples e objetivos

Realmente não há o que fazer: só definimo-os como ativos ou não, e gerenciamos seus membros. Note a mensagem: **Warning: no members!**:

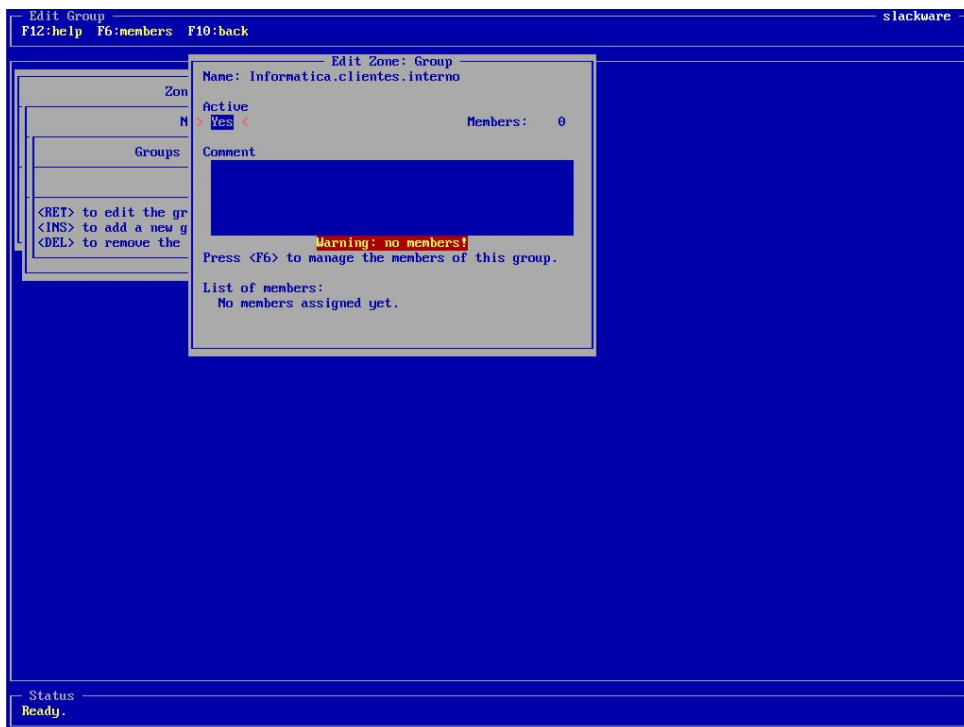


Figura 66: Grupos: Criação

Para gerenciar os membros, aperte <F6>, e a conhecida listagem aparecerá. Para adicionar um novo membro, pressione <INS>, e selecione um nome na lista de membros disponíveis para aquela rede. Veja o exemplo na figura 67:

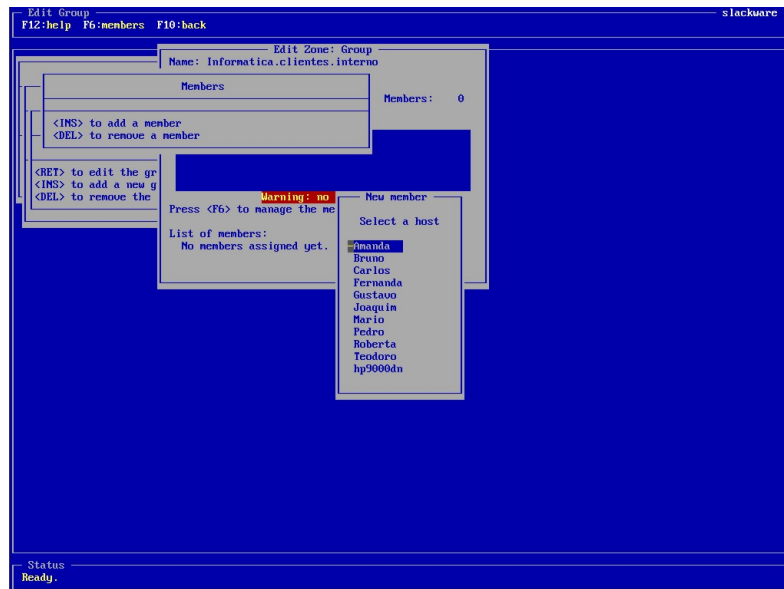


Figura 67: Grupos: Seleção de membros

Repetimos o processo até que tenham sido definidos todos os grupos (departamentos):

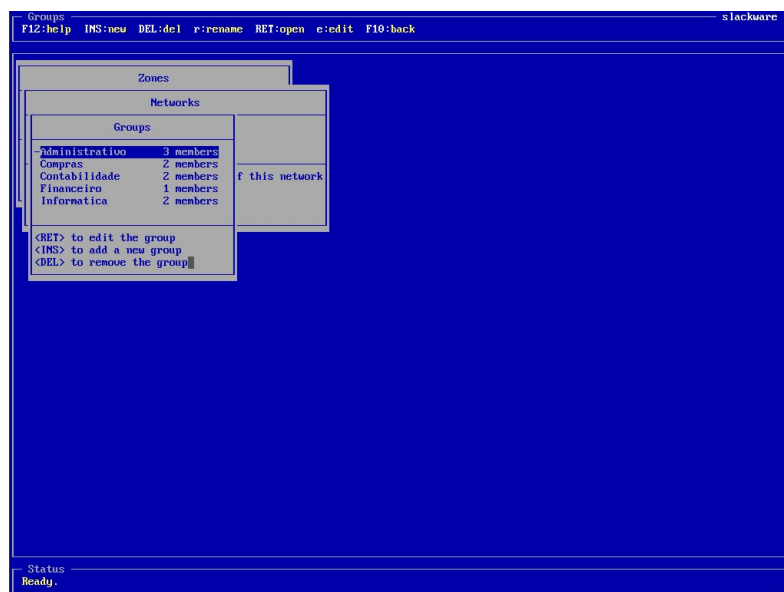


Figura 68: Grupos: Agrupamento de semelhanças

Com isso, fechamos todos os cadastros necessários de objetos, e podemos partir para a fase final, e a que mais interessa: a criação de regras!

3.5.5 Blocklist

Antes de passarmos à criação explícita de regras, devemos passar por este ponto, que é também uma criação de regra, mas é uma etapa infinitamente mais simples, e que economiza muito do administrador de redes.

O conceito de blocklist, ou até mesmo blacklist, como é até mais normalmente encontrado, baseia-se no simples conceito de que tudo que tiver ali será completamente negado, ou bloqueado, ou a ação negativa que seja. O exemplo mais clássico e aparente de blocklist é a de remetentes de email bloqueados. Todo email que estiver em uma determinada blocklist será automaticamente apagado.

No Vuurmuur, o Blocklist segue exatamente o mesmo conceito, dentro de nosso cenário: tudo que estiver listado ali dentro será completamente bloqueado, não sendo capaz de realizar qualquer conexão que esteja sendo controlada pelo firewall.

O recurso Blocklist do Vuurmuur nos permite adicionar um endereço IP, um host pré-cadastrado ou grupo, também pré-cadastrado. Extraordinariamente neste caso, o endereço IP não precisa ser objeto cadastrado nas zonas. Em nosso exemplo, temos o host lá, cadastrado na zona ‘internet.ext’. Porém, caso você tenha muitos problemas com hosts ‘não bem-vindos’, você terá a opção de adicionar diretamente seu ip. A interface, como de costume, segue o mesmo padrão de todas as telas do Vuurmuur. Para adicionarmos um host na blocklist, pressionamos <INS>, e selecionamos nosso alvo:

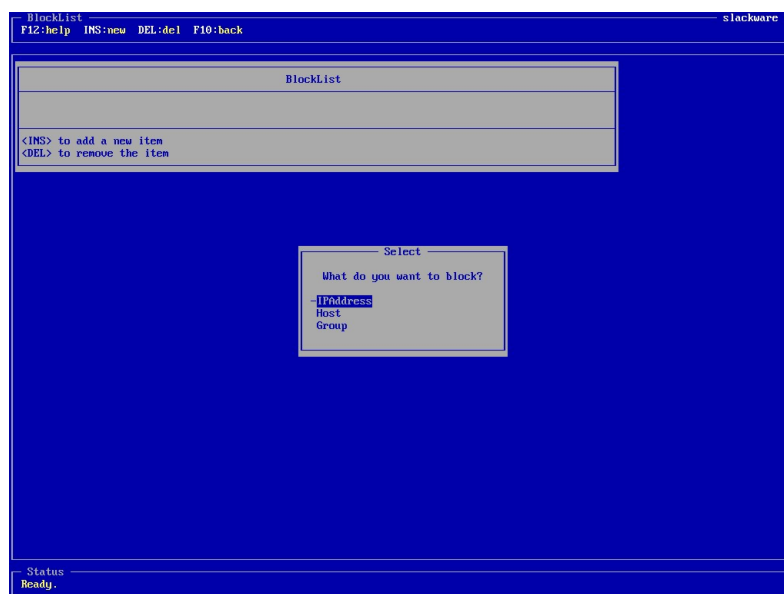


Figura 69: Blocklist: Endereço IP, host ou grupo

Em nosso caso, iremos escolher a opção `host`, pois já temos nosso `host`, `Crackers`, cadastrado na lista de `hosts`. Confirmamos normalmente. Podemos adicionar quantos `hosts` quisermos nesta lista, e eles estarão automaticamente eliminados.

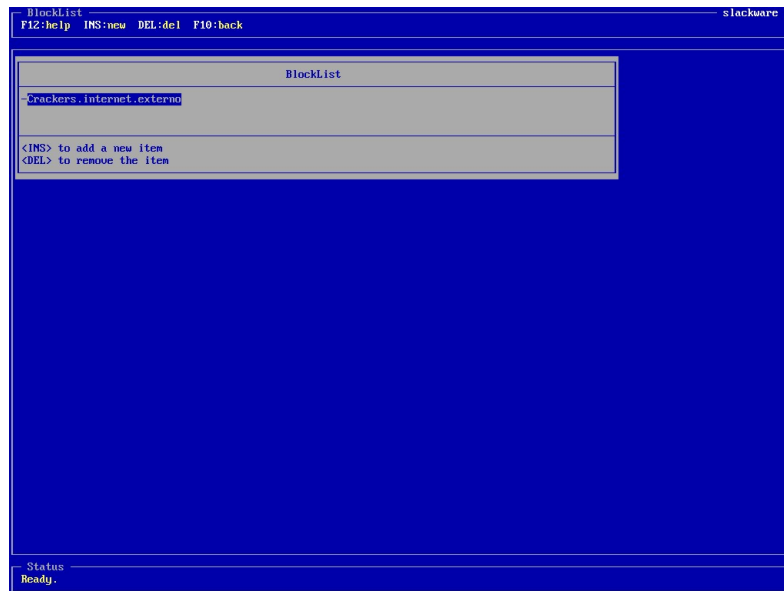


Figura 70: Blocklist: Bloqueio simples e fácil

3.5.6 Rules

Nesta última seção do trabalho, iremos tratar explicitamente da principal referência que temos quando dizemos a palavra `firewall`: **regras**. Todo o foco deste trabalho esteve voltado para a criação de objetos, definição de escopos, parametrização de opções, e tudo para chegarmos a um só resultado principal: as regras. Aqui, iremos mostrar como é realmente fácil a criação de regras utilizando o `Vuurmuur` e seus objetos previamente cadastrados.

É importante que coloquemos novamente a definição do `Vuurmuur` neste ponto, como exibido na seção 3.1 (p.25):

O `Vuurmuur` é um *middle-end/front-end* para `iptables` para o administrador de redes que precisa de um `firewall` decente, mas não possui os conhecimentos específicos do `iptables`.

A intenção é clara, neste caso: o `Vuurmuur` nos poupa de ter o conhecimento avançado da sintaxe do `iptables`, mas é **completamente** indispensável que tenhamos conhecimentos sobre a forma de funcionamento de um `firewall`, a forma de criação de regras, e, ao menos,

os conceitos básicos do próprio iptables. Entretanto, iremos relacionar aqui apenas 2 conceitos elementares, apenas para que as pessoas que nunca viram um firewall na vida possam entender o que está acontecendo:

- As regras são lidas de cima para baixo, uma-a-uma. Caso um pacote se encaixe em determinada regra, então ele não continuará sendo processado.
- Pelo motivo acima, sempre que quisermos fazer uma negação, e quando ela for uma exceção à regra, deveremos fazê-la antes da regra geral, pois senão, o objeto a ser ‘enquadrado’ na exceção cairá na regra comum antes, e não sofrerá a ação desejada.

A primeira imagem que temos quando entramos na seção **Rules** será semelhante à mostrada na figura 71:



Figura 71: Rules: Criação de regras no Vuurmuur

Os atalhos disponíveis são os mesmo que estamos habituados e já conhecemos, com exceção do **m:move**, e alguns outros ‘escondidos’, que iremos ver à frente.

Para iniciarmos a criação de regras, iremos definir uma ordem de criação das regras: primeiramente faremos as regras relacionadas aos clientes, e em seguida aos servidores e à internet em geral. Para facilitar, estarei numerando sequencialmente as regras, para melhor identificação¹⁷.

¹⁷O número da regra neste documento não será igual ao número da regra gerada no Vuurmuur.

Para adicionarmos uma regra, pressionamos <INS>, e teremos nossa janela de criação de regras:

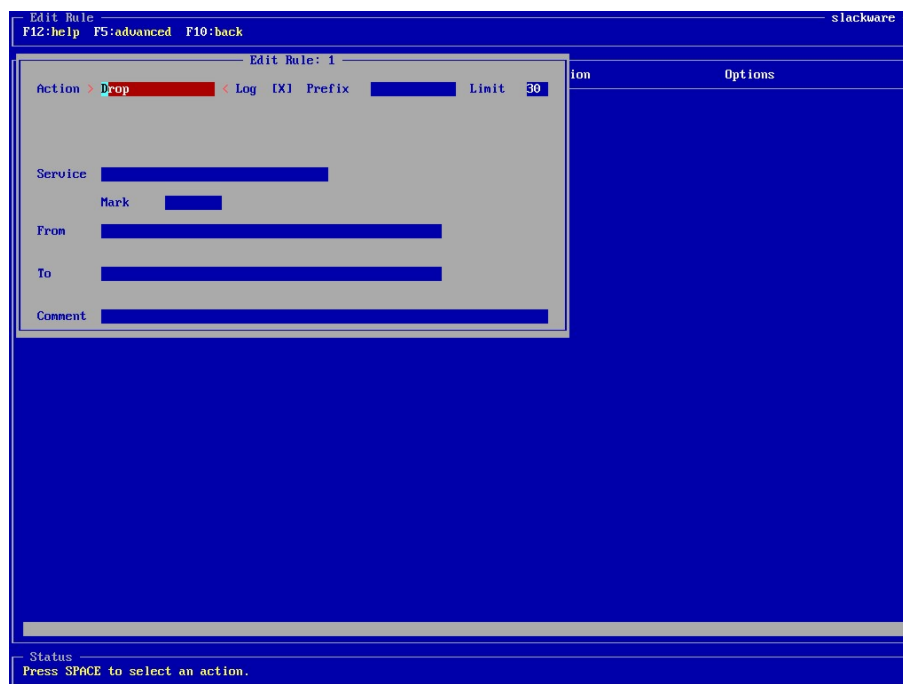


Figura 72: Rules: Inserindo uma regra

Esta é a tela com a qual iremos trabalhar na criação de regras. Observe o campo **Action**: ele é quem define o que fazer com a regra. Devemos escolhê-lo antes de tudo pois, de acordo com este *target*, a tela irá mudar, disponibilizando as opções referente à cada *target*¹⁸. Porém, alguns campos irão sempre aparecer, e iremos listá-los abaixo. Os campos específicos de cada serviço serão usados caso tenhamos um serviço condizente com aquela *target*. Eis os campos padrão:

Action: Como acabamos de comentar, é exatamente quem define os campos adicionais que aparecerão. Para acessar as opções dos menus *drop-down*, coloque o foco no item desejado, e aperte a barra de espaços.

Log []: Marque esta caixa caso você queira que esta regra gere registros. Por *default*, estará marcada.

Prefix: Este é um campo livre, para ser digitado qualquer texto para ser incluso no registro.

¹⁸Caso, em nossos exemplos, estejam aparecendo mais campos do que em seu teste prático, experimente apertar o botão <F5>. Isto terá acontecido quando você não tiver ativado o 'Advanced Mode' por *default*, no `vuurmuur_conf`.

Limit: Este número representa o máximo de registros que essa regra poderá gerar por segundo.

Service: Este é o primeiro grande campo importante na criação de regras. Quando você estiver criando uma regra, sempre será necessário informar um serviço, uma origem, e um destino. Neste campo, você informa para qual serviço a regra deverá se aplicar. Existe também a opção 'any', que significa que a regra se encaixará para qualquer serviço.

From: Aqui, colocaremos a origem da conexão. Aqui, desfrutamos de nosso trabalho anteriormente. Podemos selecionar um host específico, um grupo, uma rede, ou uma zona inteira. Adicionalmente, teremos sempre, aqui e no campo **To**, a opção **firewall**, que se refere justamente às conexões encaminhadas às interfaces do próprio firewall.

To: Aqui informamos o destino da conexão. Os padrões são os mesmos para o campo **From**.

Atenção: o Vuurmuur faz uma severa verificação de consistência das regras. Portanto, se você obtiver um erro ao criá-las, revise-a, pois podem ocorrer diversos erros.

Regra 1: Negar o acesso via SSH no FileServer para o usuário Bruno.

Para o iptables, a ação de negar um pacote pode ser feita de duas formas: descartando-o (**DROP**) ou rejeitando-o (**REJECT**). O mais comum é que descartemos o pacote. Para criarmos nossa regra, iremos definir a *target* **DROP**. Veja:

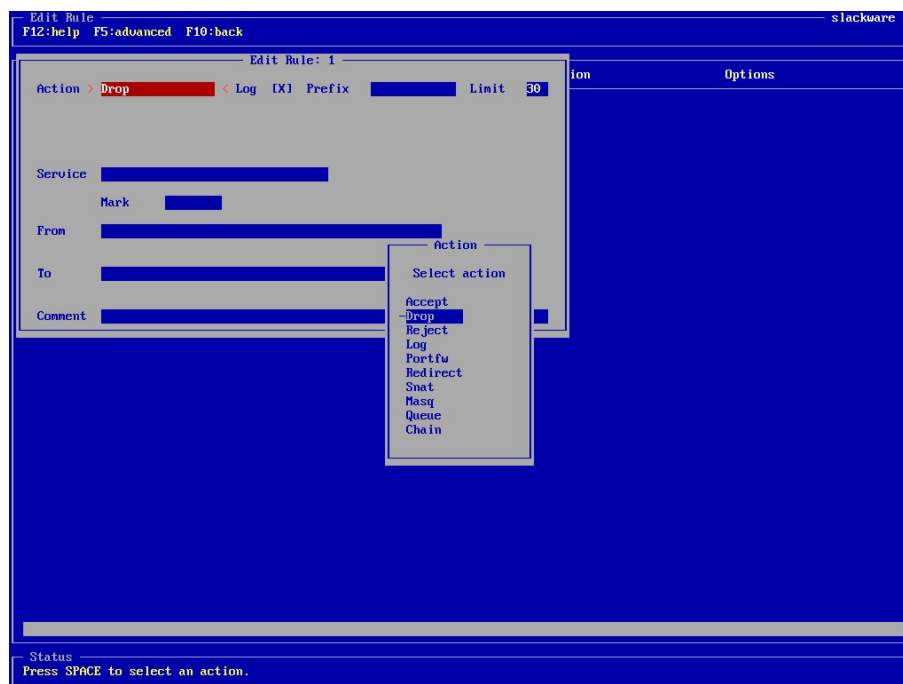


Figura 73: Rules: target DROP

Em seguida, devemos escolher o serviço, que neste caso é o SSH. Simplesmente vá até `service` com a seta, pressione `<ESPAÇO>` para abrir a lista de serviços, e selecione-o. Agora, vá até o campo `from:` como origem, teremos, obviamente o host Bruno. Ao abrir a lista para `From`, damos de cara com o grande diferencial do Vuurmuur. Veja na figura:

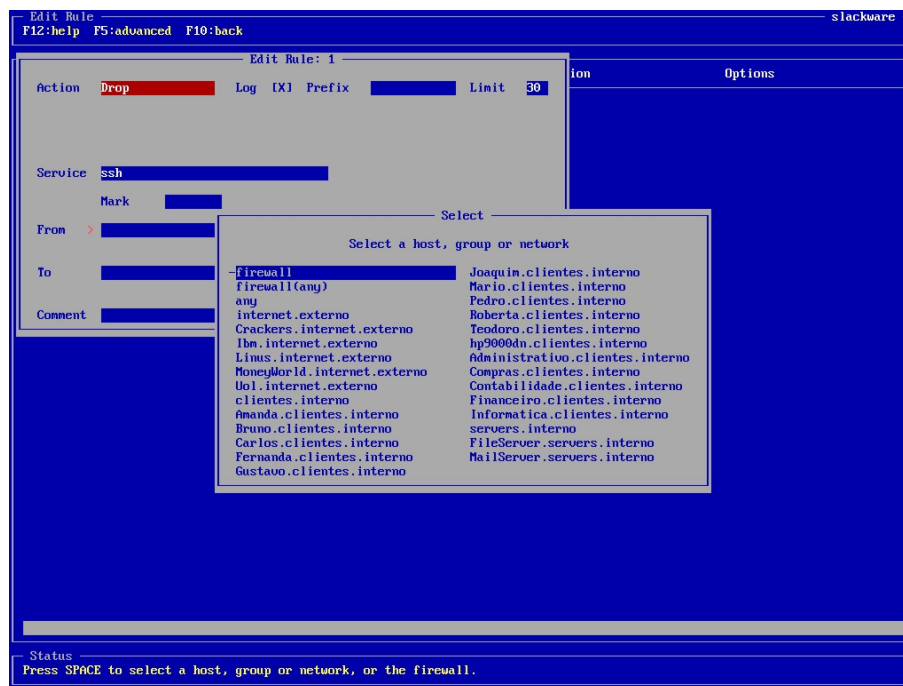


Figura 74: Rules: Hosts, grupos, redes e zonas

Aparecem aqui **todos** os objetos cadastrados por nós: aparecem os hosts, os grupos e as redes, todas interligadas através da nomenclatura utilizada pelo Vuurmuur. Caso você tenha feito uma correta estruturação dos nomes de zonas, redes, hosts e grupos, poderá encontrar facilmente agora qualquer objeto. Selecionamos, portanto `Bruno.clientes.interno`. Bom, já temos que nossa ação será o `DROP`, e já dissemos que será quando ocorrer uma conexão SSH a partir do computador Bruno. Mas, uma conexão para onde? `FileServer`, preenchamos, portanto, o campo `To` com isto.

Caso queira, digite valores para os campos opcionais. Relembrando que, sempre que estivermos em cima de um campo, na lista de regras, irá aparecer uma referência sobre aquela regra no rodapé.

Aperte `<F10>` para confirmar, e nós já teremos criado nossa primeira regra. Sempre que quiser editar uma regra, apenas aperte `<ENTER>` sobre ela. Após criada a regra, a barra de espaços alterna entre manter esta regra ativa ou não. Isto é extremamente útil quando temos alguma regra que não é utilizada sempre. Podemos deixá-la cadastrada, e ativá-la apenas quando necessário.

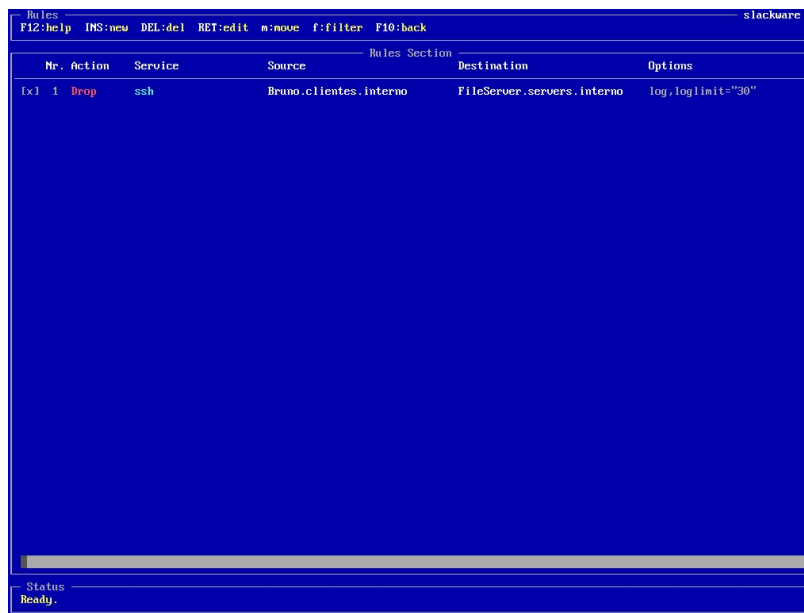


Figura 75: Rules: Primeira regra

Já temos nossa primeira regra, pronta para ser aplicada. Veja como é fácil de se visualizar e entender qual é a ação desenvolvida pela regra. Através das diferentes cores, com poucos dias de uso já conseguimos visualizar as regras de forma bem clara.

Fizemos a exceção, agora vamos à regra:

Regra 2: Liberar toda a comunicação para o grupo Informática.

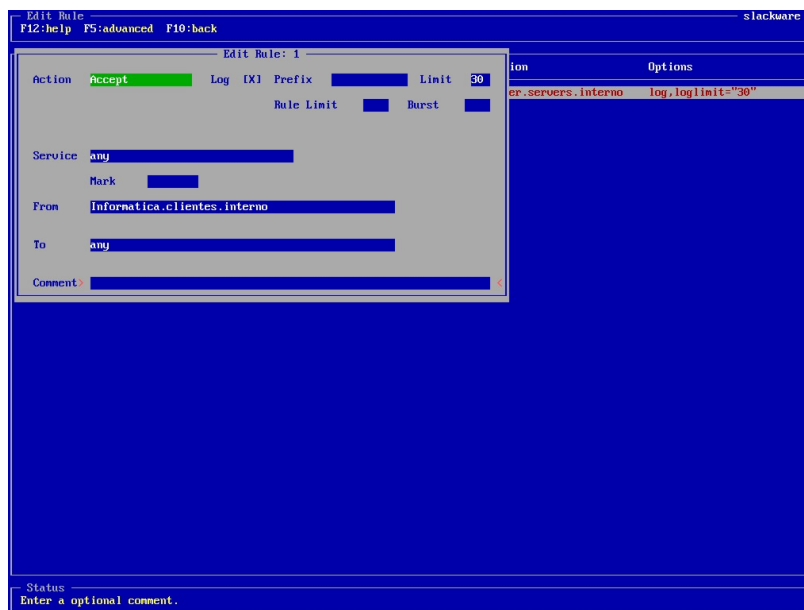


Figura 76: accept any from informatica.clientes.interno to any

Veja o que fizemos nesta simples regra: em **Service**, colocamos **Any**, ou seja, qualquer serviço. Em **From**, utilizamos o grupo **Informática**, previamente criado. E em **To**, colocamos **Any** também. Isso significaria dizer “Aceite qualquer serviço vindo de ‘Informatica.clientes.interno’ com destino à qualquer lugar”. Quando utilizamos o Vuurmuur, nos acostumamos até a pensar nas regras desta forma. Ficamos com nossas duas regras cadastradas:

Nr.	action	Service	Source	Destination	Options
Ex] 1	Accept	any	Informatica.clientes.interno	any	log,loglimit='30'
Ex] 2	Drop	ssh	Bruno.clientes.interno	FileServer.servers.interno	log,loglimit='30'

Figura 77: Rules: Duas regras

Repare que nossa segunda regra apareceu abaixo da que havíamos criado primeiro: vamos aprender mais alguns comandos, antes de passarmos às próximas regras.

Para definir a ordem das regras no Vuurmuur, apenas utilize + (mais) e - (menos). O símbolo de **mais** desce a regra, e o símbolo de **menos** sobe. Pode parecer que está ao contrário, mas pense como se fossem ‘pesos’ nas regras. O + a deixa mais pesada, e a faz descer, e o - a deixa mais leve, fazendo-a subir. :)

Temos também o **m:move**. Aperte **m** sobre uma regra, e informe sua nova posição. Temos agora um recurso extremamente útil e prático: ao apertar <L>, teremos uma linha horizontal separadora, tão somente apenas com esta função.

Quando temos muitas regras, algo que isole as regras, ao menos para te dar um referencial, funciona muito bem. Este é a intenção desta linha. Você pode dar um nome

para ela, apertando, sobre a linha, <ENTER>, e digitando um nome. Vejamos:



Figura 78: Linha separadora

O restante das regras dos clientes são basicamente operações de autorizar e negar. Portanto, como *screenshots* sempre valem mais do que palavras, irei colocar a regra, seguido de imagens, que explicarão muito melhor do que este texto. ;)

Regra 3: Descartar qualquer pacote enviado do grupo ‘Administrativo’ para o UOL através de http.

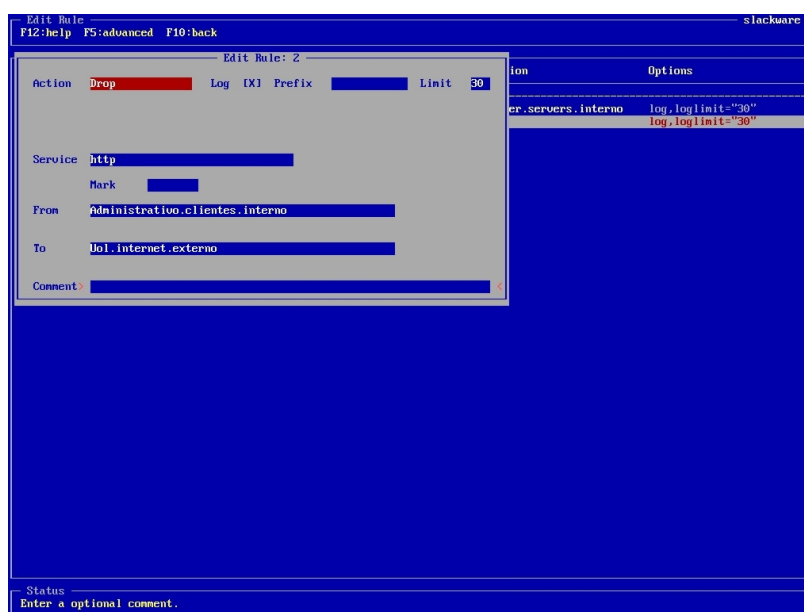


Figura 79: Regra três

Regra 4: Liberar para o grupo ‘Administrativo’ o acesso à Internet, emails, samba, msn e SAPO.

Rules						
F12:help INS:new DEL:del RET:edit n:noue f:filter F10:back						
slackware						
Nr.	Action	Service	Source	Rules Section	Destination	Options
1				[Informatica]		
[x]	2	Drop	ssh	Bruno.clientes.interno	FileServer.servers.interno	log,loglimit="30"
[x]	3	Accept	any	Informatica.clientes.interno	any	log,loglimit="30"
	4			[Administrativo]		
[x]	5	Drop	http	Administrativo.clientes.intern	Uol.internet.externo	log,loglimit="30"
[x]	6	Accept	http	Administrativo.clientes.intern	internet.externo	log,loglimit="30"
[x]	7	Accept	https	Administrativo.clientes.intern	internet.externo	log,loglimit="30"
[x]	8	Accept	pop3	Administrativo.clientes.intern	MailServer.servers.interno	log,loglimit="30"
[x]	9	Accept	smtp	Administrativo.clientes.intern	MailServer.servers.interno	log,loglimit="30"
[x]	10	Accept	samba	Administrativo.clientes.intern	FileServer.servers.interno	log,loglimit="30"
[x]	11	Accept	msn	Administrativo.clientes.intern	internet.externo	log,loglimit="30"
[x]	12	Accept	SAPO	Administrativo.clientes.intern	FileServer.servers.interno	log,loglimit="30"

Status Ready.

Figura 80: Regra quatro

Regra 5: Permitir ao grupo ‘Contabilidade’ conectar no msn, mandar e receber emails e pingar endereços externos.

Rules						
F12:help INS:new DEL:del RET:edit n:noue f:filter F10:back						
slackware						
Nr.	Action	Service	Source	Rules Section	Destination	Options
1				[Informatica]		
[x]	2	Drop	ssh	Bruno.clientes.interno	FileServer.servers.interno	log,loglimit="30"
[x]	3	Accept	any	Informatica.clientes.interno	any	log,loglimit="30"
	4			[Administrativo]		
[x]	5	Drop	http	Administrativo.clientes.intern	Uol.internet.externo	log,loglimit="30"
[x]	6	Accept	http	Administrativo.clientes.intern	internet.externo	log,loglimit="30"
[x]	7	Accept	https	Administrativo.clientes.intern	internet.externo	log,loglimit="30"
[x]	8	Accept	pop3	Administrativo.clientes.intern	MailServer.servers.interno	log,loglimit="30"
[x]	9	Accept	smtp	Administrativo.clientes.intern	MailServer.servers.interno	log,loglimit="30"
[x]	10	Accept	samba	Administrativo.clientes.intern	FileServer.servers.interno	log,loglimit="30"
[x]	11	Accept	msn	Administrativo.clientes.intern	internet.externo	log,loglimit="30"
[x]	12	Accept	SAPO	Administrativo.clientes.intern	FileServer.servers.interno	log,loglimit="30"
	13			[Contabilidade]		
[x]	14	Accept	msn	Contabilidade.clientes.intern	internet.externo	log,loglimit="30"
[x]	15	Accept	pop3	Contabilidade.clientes.intern	MailServer.servers.interno	log,loglimit="30"
[x]	16	Accept	smtp	Contabilidade.clientes.intern	MailServer.servers.interno	log,loglimit="30"
[x]	17	Accept	ping	Contabilidade.clientes.intern	internet.externo	log,loglimit="30"

Status Ready.

Figura 81: Regra cinco

Regra 6: Permitir a utilização do SAPO, conexão ftp para o FileServer e acesso ao site MoneyWorld ao grupo 'Financeiro'.

Nr.	Action	Service	Source	Destination	Options
[Informatica]					
[x] 2	Drop	ssh	Bruno.clientes.interno	FileServer.servers.interno	log,loglimit="30"
[x] 3	Accept	any	Informatica.clientes.interno	any	log,loglimit="30"
[Administrativo]					
[x] 5	Drop	http	Administrativo.clientes.interno	Uol.internet.externo	log,loglimit="30"
[x] 6	Accept	http	Administrativo.clientes.interno	internet.externo	log,loglimit="30"
[x] 7	Accept	https	Administrativo.clientes.interno	internet.externo	log,loglimit="30"
[x] 8	Accept	pop3	Administrativo.clientes.interno	MailServer.servers.interno	log,loglimit="30"
[x] 9	Accept	smtp	Administrativo.clientes.interno	MailServer.servers.interno	log,loglimit="30"
[x] 10	Accept	samba	Administrativo.clientes.interno	FileServer.servers.interno	log,loglimit="30"
[x] 11	Accept	msn	Administrativo.clientes.interno	internet.externo	log,loglimit="30"
[x] 12	Accept	SAPO	Administrativo.clientes.interno	FileServer.servers.interno	log,loglimit="30"
[Contabilidade]					
[x] 14	Accept	msn	Contabilidade.clientes.interno	internet.externo	log,loglimit="30"
[x] 15	Accept	pop3	Contabilidade.clientes.interno	MailServer.servers.interno	log,loglimit="30"
[x] 16	Accept	smtp	Contabilidade.clientes.interno	MailServer.servers.interno	log,loglimit="30"
[x] 17	Accept	ping	Contabilidade.clientes.interno	internet.externo	log,loglimit="30"
[Financeiro]					
[x] 19	Accept	SAPO	Financeiro.clientes.interno	FileServer.servers.interno	log,loglimit="30"
[x] 20	Accept	ftp	Financeiro.clientes.interno	FileServer.servers.interno	log,loglimit="30"
[x] 21	Accept	http	Financeiro.clientes.interno	MoneyWorld.internet.externo	log,loglimit="30"

Status
Showing all rules.

Figura 82: Regra seis

Regra 7: Abrir a exceção para o usuário Teodoro mandar e receber emails.

Action	Service	From	To	Options
Accept	pop3	Teodoro.clientes.interno	MailServer.servers.interno	log,loglimit="30"

Status
Press SPACE to select a host, group or network, or the firewall.

Figura 83: Regra sete

Regra 8: Bloquear todo o resto para o grupo 'Compras'.

Rules						slackware
F12:help INS:new DEL:del RET:edit n:nove f:filter F10:back						
Rules Section						
Nr.	Action	Service	Source	Destination	Options	
1						
-----[Informatica]-----						
[x]	2	Drop	ssh	Bruno.clientes.interno	FileServer.servers.interno	log,loglimit="30"
[x]	3	Accept	any	Informatica.clientes.interno	any	log,loglimit="30"
4						
-----[Administrativo]-----						
[x]	5	Drop	http	Administrativo.clientes.inter	l01.internet.externo	log,loglimit="30"
[x]	6	Accept	http	Administrativo.clientes.inter	internet.externo	log,loglimit="30"
[x]	7	Accept	https	Administrativo.clientes.inter	internet.externo	log,loglimit="30"
[x]	8	Accept	pop3	Administrativo.clientes.inter	MailServer.servers.interno	log,loglimit="30"
[x]	9	Accept	smtp	Administrativo.clientes.inter	MailServer.servers.interno	log,loglimit="30"
[x]	10	Accept	samba	Administrativo.clientes.inter	FileServer.servers.interno	log,loglimit="30"
[x]	11	Accept	nsn	Administrativo.clientes.inter	internet.externo	log,loglimit="30"
[x]	12	Accept	SAPo	Administrativo.clientes.inter	FileServer.servers.interno	log,loglimit="30"
13						
-----[Contabilidade]-----						
[x]	14	Accept	nsn	Contabilidade.clientes.intern	internet.externo	log,loglimit="30"
[x]	15	Accept	pop3	Contabilidade.clientes.intern	MailServer.servers.interno	log,loglimit="30"
[x]	16	Accept	smtp	Contabilidade.clientes.intern	MailServer.servers.interno	log,loglimit="30"
[x]	17	Accept	ping	Contabilidade.clientes.intern	internet.externo	log,loglimit="30"
18						
-----[Financeiro]-----						
[x]	19	Accept	SAPo	Financeiro.clientes.interno	FileServer.servers.interno	log,loglimit="30"
[x]	20	Accept	ftp	Financeiro.clientes.interno	FileServer.servers.interno	log,loglimit="30"
[x]	21	Accept	http	Financeiro.clientes.interno	MoneyWorld.internet.externo	log,loglimit="30"
22						
-----[Compras]-----						
[x]	23	Accept	pop3	Teodoro.clientes.interno	MailServer.servers.interno	log,loglimit="30"
[x]	24	Accept	smtp	Teodoro.clientes.interno	MailServer.servers.interno	log,loglimit="30"
[x]	25	Drop	any	Compras.clientes.interno	any	log,loglimit="30"

Status
Ready.

Figura 84: Regra oito

Regra 9: A impressora terá acesso ao samba.

Rules						slackware
F12:help INS:new DEL:del RET:edit n:nove f:filter F10:back						
Rules Section						
Nr.	Action	Service	Source	Destination	Options	
1						
-----[Informatica]-----						
[x]	2	Drop	ssh	Bruno.clientes.interno	FileServer.servers.interno	log,loglimit="30"
[x]	3	Accept	any	Informatica.clientes.interno	any	log,loglimit="30"
4						
-----[Administrativo]-----						
[x]	5	Drop	http	Administrativo.clientes.inter	l01.internet.externo	log,loglimit="30"
[x]	6	Accept	http	Administrativo.clientes.inter	internet.externo	log,loglimit="30"
[x]	7	Accept	https	Administrativo.clientes.inter	internet.externo	log,loglimit="30"
[x]	8	Accept	pop3	Administrativo.clientes.inter	MailServer.servers.interno	log,loglimit="30"
[x]	9	Accept	smtp	Administrativo.clientes.inter	MailServer.servers.interno	log,loglimit="30"
[x]	10	Accept	samba	Administrativo.clientes.inter	FileServer.servers.interno	log,loglimit="30"
[x]	11	Accept	nsn	Administrativo.clientes.inter	internet.externo	log,loglimit="30"
[x]	12	Accept	SAPo	Administrativo.clientes.inter	FileServer.servers.interno	log,loglimit="30"
13						
-----[Contabilidade]-----						
[x]	14	Accept	nsn	Contabilidade.clientes.intern	internet.externo	log,loglimit="30"
[x]	15	Accept	pop3	Contabilidade.clientes.intern	MailServer.servers.interno	log,loglimit="30"
[x]	16	Accept	smtp	Contabilidade.clientes.intern	MailServer.servers.interno	log,loglimit="30"
[x]	17	Accept	ping	Contabilidade.clientes.intern	internet.externo	log,loglimit="30"
18						
-----[Financeiro]-----						
[x]	19	Accept	SAPo	Financeiro.clientes.interno	FileServer.servers.interno	log,loglimit="30"
[x]	20	Accept	ftp	Financeiro.clientes.interno	FileServer.servers.interno	log,loglimit="30"
[x]	21	Accept	http	Financeiro.clientes.interno	MoneyWorld.internet.externo	log,loglimit="30"
22						
-----[Compras]-----						
[x]	23	Accept	pop3	Teodoro.clientes.interno	MailServer.servers.interno	log,loglimit="30"
[x]	24	Accept	smtp	Teodoro.clientes.interno	MailServer.servers.interno	log,loglimit="30"
[x]	25	Drop	any	Compras.clientes.interno	any	log,loglimit="30"
26						
-----[Impressora]-----						
[x]	27	Accept	samba	hp9000dn.clientes.interno	FileServer.servers.interno	log,loglimit="30"

Status
Ready.

Figura 85: Regra nove

Após tudo isso, nossas regras de clientes estão prontas. Podemos passar às regras para os servidores e a Internet. Aqui temos pouquíssimas regras, em comparação à complexidade que se pode encontrar aqui. De qualquer forma, a assimilação desta forma de criação é extremamente simples e fácil de se pegar.

Servers

Regra 10: Os serviços oferecidos pelo FileServer estarão disponíveis somente na rede local.

Esta regra foi colocada aqui somente para lembrarmos aqui como é vantajoso termos um firewall que implementa corretamente as políticas, onde tudo que não é claramente especificado vai para DROP. A regra 10 **também** poderia ser feita através do próprio samba, mas, em nosso caso, não precisaremos criar regra alguma para isso, visto que já autorizamos todos que utilizam o samba, e qualquer outra conexão irá para DROP.

É muito comum encontrar, principalmente em scripts montados de firewall, políticas-padrão definidas para ACCEPT. Fazer isto é realmente anular todo o poder de seu firewall. Não se há necessidade, por exemplo, de criarmos várias e várias páginas de regras para bloquearmos Backdoors, ataques variados, etc, uma vez que somente será autorizado entrar na rede aquilo que foi explicitamente declarado.

Regra 11: Fazer o encaminhamento de portas dos serviços pop3 e smtp para o MailServer.

Nesta regra, estaremos utilizando uma nova *target*, até então não utilizada aqui: o *portfw*, também conhecido como DNAT. O *portfw* faz o encaminhamento de portas para hosts internos da nossa rede, hosts que não tem um IP público, de forma que eles possam responder diretamente às requisições feitas para tal portas.

Utilizando o DNAT, é possível que tenhamos vários servidores em nossa rede, respondendo por vários serviços, através de um único IP público. Para pequenas e médias empresas, isto é bastante atrativo e é o que normalmente acontece, visto que geralmente essas empresas não estão dispostas ou simplesmente não podem pagar os elevados preços de linhas de dados.

Quando escolhemos a *target portfw* no Vuurmuur, podemos perceber que as opções todas mudam, como na figura 86:

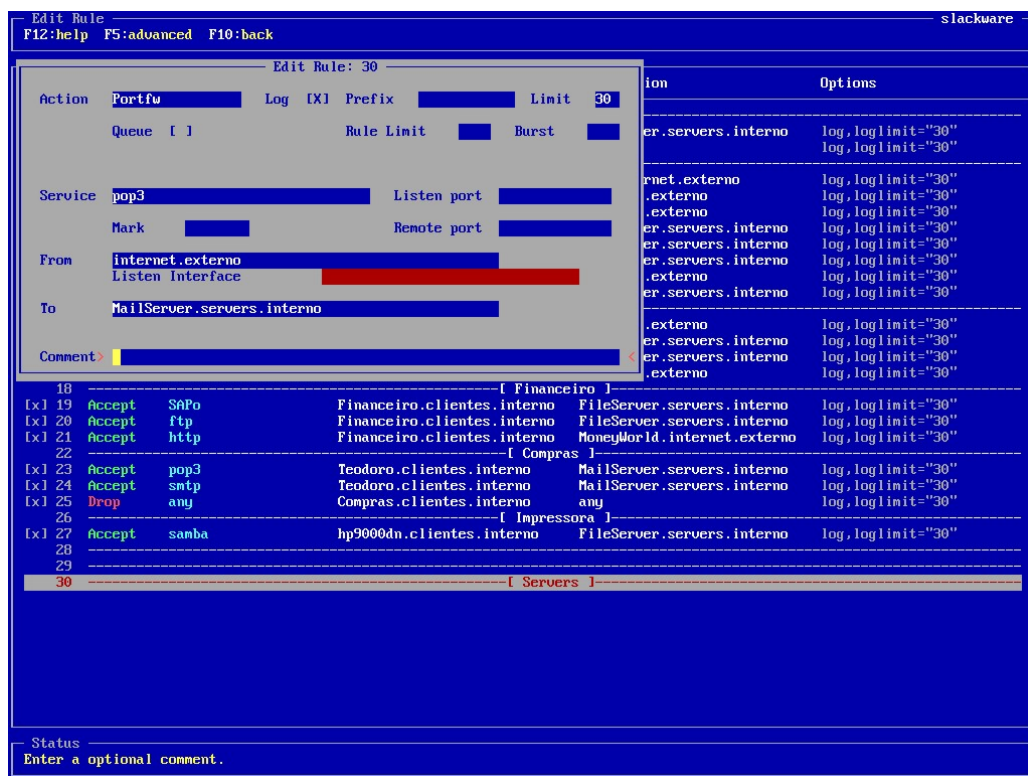


Figura 86: Encaminhamento de portas

Além dos campos Rule Limit e Burst, que também aparecem em outros *targets* e estão relacionados ao controle de Syn-Flood, temos alguns campos particulares:

Listen Port: Aqui podemos adicionar algumas portas adicionais para o firewall escutar. Caso ele receba um requisição em alguma dessas portas, ele re-encaminha para a porta de serviço especificada.

Remote Port: Funciona da mesma forma, mas você informa as portas quais o firewall deverá encaminhar o pacote recém-chegado, além da porta de serviço padrão. Além disso, no *portw*, quando selecionamos um objeto, podemos dizer de qual rede ele está vindo, limitando assim através da interface.

Para preenchermos o *portfw* corretamente, informamos o serviço, seguido de sua origem (no caso, a Internet), e o seu destino. Porém, destino não será como no iptables, em que colocaríamos o endereço do firewall. Aqui, o firewall já sabe que, se você está fazendo um *portforward*, você não pode passar para você mesmo. Portanto, neste caso, o *To* em questão deve ser usado para informar para onde irão os pacotes.

Rules						slackware
F12:help INS:new DEL:del RET:edit n:move f:filter F10:back						
Nr.	Action	Service	Source	Destination	Options	
----- [Informatica] -----						
[x] 2	Drop	ssh	Bruno.clientes.interno	FileServer.servers.interno	log,loglimit="30"	
[x] 3	Accept	any	Informatica.clientes.interno	any	log,loglimit="30"	
----- [Administrativo] -----						
[x] 5	Drop	http	Administrativo.clientes.inter	Uol.internet.externo	log,loglimit="30"	
[x] 6	Accept	http	Administrativo.clientes.inter	internet.externo	log,loglimit="30"	
[x] 7	Accept	https	Administrativo.clientes.inter	internet.externo	log,loglimit="30"	
[x] 8	Accept	pop3	Administrativo.clientes.inter	MailServer.servers.interno	log,loglimit="30"	
[x] 9	Accept	smtp	Administrativo.clientes.inter	MailServer.servers.interno	log,loglimit="30"	
[x] 10	Accept	samba	Administrativo.clientes.inter	FileServer.servers.interno	log,loglimit="30"	
[x] 11	Accept	msn	Administrativo.clientes.inter	internet.externo	log,loglimit="30"	
[x] 12	Accept	SAPo	Administrativo.clientes.inter	FileServer.servers.interno	log,loglimit="30"	
----- [Contabilidade] -----						
[x] 14	Accept	msn	Contabilidade.clientes.intern	internet.externo	log,loglimit="30"	
[x] 15	Accept	pop3	Contabilidade.clientes.intern	MailServer.servers.interno	log,loglimit="30"	
[x] 16	Accept	smtp	Contabilidade.clientes.intern	MailServer.servers.interno	log,loglimit="30"	
[x] 17	Accept	ping	Contabilidade.clientes.intern	internet.externo	log,loglimit="30"	
----- [Financeiro] -----						
[x] 19	Accept	SAPo	Financeiro.clientes.interno	FileServer.servers.interno	log,loglimit="30"	
[x] 20	Accept	ftp	Financeiro.clientes.interno	FileServer.servers.interno	log,loglimit="30"	
[x] 21	Accept	http	Financeiro.clientes.interno	MoneyWorld.internet.externo	log,loglimit="30"	
----- [Compras] -----						
[x] 23	Accept	pop3	Teodoro.clientes.interno	MailServer.servers.interno	log,loglimit="30"	
[x] 24	Accept	smtp	Teodoro.clientes.interno	MailServer.servers.interno	log,loglimit="30"	
[x] 25	Drop	any	Compras.clientes.interno	any	log,loglimit="30"	
----- [Impressora] -----						
[x] 27	Accept	samba	hp9000dn.clientes.interno	FileServer.servers.interno	log,loglimit="30"	
----- [Servers] -----						
[x] 31	Portfw	pop3	internet.externo	MailServer.servers.interno	log,loglimit="30"	
[x] 32	Portfw	smtp	internet.externo	MailServer.servers.interno	log,loglimit="30"	

Status
Ready.

Figura 87: Rules: portforward simples

Regra 12: Somente a IBM poderá nos pingar pela Internet

Rules						slackware
F12:help INS:new DEL:del RET:edit n:move f:filter F10:back						
Nr.	Action	Service	Source	Destination	Options	
----- [Informatica] -----						
[x] 2	Drop	ssh	Bruno.clientes.interno	FileServer.servers.interno	log,loglimit="30"	
[x] 3	Accept	any	Informatica.clientes.interno	any	log,loglimit="30"	
----- [Administrativo] -----						
[x] 5	Drop	http	Administrativo.clientes.inter	Uol.internet.externo	log,loglimit="30"	
[x] 6	Accept	http	Administrativo.clientes.inter	internet.externo	log,loglimit="30"	
[x] 7	Accept	https	Administrativo.clientes.inter	internet.externo	log,loglimit="30"	
[x] 8	Accept	pop3	Administrativo.clientes.inter	MailServer.servers.interno	log,loglimit="30"	
[x] 9	Accept	smtp	Administrativo.clientes.inter	MailServer.servers.interno	log,loglimit="30"	
[x] 10	Accept	samba	Administrativo.clientes.inter	FileServer.servers.interno	log,loglimit="30"	
[x] 11	Accept	msn	Administrativo.clientes.inter	internet.externo	log,loglimit="30"	
[x] 12	Accept	SAPo	Administrativo.clientes.inter	FileServer.servers.interno	log,loglimit="30"	
----- [Contabilidade] -----						
[x] 14	Accept	msn	Contabilidade.clientes.intern	internet.externo	log,loglimit="30"	
[x] 15	Accept	pop3	Contabilidade.clientes.intern	MailServer.servers.interno	log,loglimit="30"	
[x] 16	Accept	smtp	Contabilidade.clientes.intern	MailServer.servers.interno	log,loglimit="30"	
[x] 17	Accept	ping	Contabilidade.clientes.intern	internet.externo	log,loglimit="30"	
----- [Financeiro] -----						
[x] 19	Accept	SAPo	Financeiro.clientes.interno	FileServer.servers.interno	log,loglimit="30"	
[x] 20	Accept	ftp	Financeiro.clientes.interno	FileServer.servers.interno	log,loglimit="30"	
[x] 21	Accept	http	Financeiro.clientes.interno	MoneyWorld.internet.externo	log,loglimit="30"	
----- [Compras] -----						
[x] 23	Accept	pop3	Teodoro.clientes.interno	MailServer.servers.interno	log,loglimit="30"	
[x] 24	Accept	smtp	Teodoro.clientes.interno	MailServer.servers.interno	log,loglimit="30"	
[x] 25	Drop	any	Compras.clientes.interno	any	log,loglimit="30"	
----- [Impressora] -----						
[x] 27	Accept	samba	hp9000dn.clientes.interno	FileServer.servers.interno	log,loglimit="30"	
----- [Servers] -----						
----- [portfw] -----						
[x] 31	Portfw	pop3	internet.externo	MailServer.servers.interno	log,loglimit="30"	
[x] 32	Portfw	smtp	internet.externo	MailServer.servers.interno	log,loglimit="30"	
[x] 34	Accept	ping	Ibm.internet.externo	firewall	log,loglimit="30"	

Status
Ready.

Figura 88: Rules: IBM ping

Regra 13: Bloquear todo o tráfego do IP 200.1.5.10

Esta regra já foi desenvolvida facilmente na seção anterior, na página 89.

Regra 14: pela Internet, somente Linus poderá conectar-se via SSH em nossa máquina firewall.

Nr.	Action	Service	Source	Destination	Options
1					
[x] 2	Drop	ssh	Bruno.clientes.interno	FileServer.servers.interno	log,loglimit="30"
[x] 3	Accept	any	Informatica.clientes.interno	any	log,loglimit="30"
4					
[x] 5	Drop	http	Administrativo.clientes.inter	Uo1.internet.externo	log,loglimit="30"
[x] 6	Accept	http	Administrativo.clientes.inter	internet.externo	log,loglimit="30"
[x] 7	Accept	https	Administrativo.clientes.inter	internet.externo	log,loglimit="30"
[x] 8	Accept	pop3	Administrativo.clientes.inter	MailServer.servers.interno	log,loglimit="30"
[x] 9	Accept	smtp	Administrativo.clientes.inter	MailServer.servers.interno	log,loglimit="30"
[x] 10	Accept	samba	Administrativo.clientes.inter	FileServer.servers.interno	log,loglimit="30"
[x] 11	Accept	msn	Administrativo.clientes.inter	internet.externo	log,loglimit="30"
[x] 12	Accept	SAPo	Administrativo.clientes.inter	FileServer.servers.interno	log,loglimit="30"
13					
[x] 14	Accept	msn	Contabilidade.clientes.intern	internet.externo	log,loglimit="30"
[x] 15	Accept	pop3	Contabilidade.clientes.intern	MailServer.servers.interno	log,loglimit="30"
[x] 16	Accept	smtp	Contabilidade.clientes.intern	MailServer.servers.interno	log,loglimit="30"
[x] 17	Accept	ping	Contabilidade.clientes.intern	internet.externo	log,loglimit="30"
18					
[x] 19	Accept	SAPo	Financeiro.clientes.interno	FileServer.servers.interno	log,loglimit="30"
[x] 20	Accept	ftp	Financeiro.clientes.interno	FileServer.servers.interno	log,loglimit="30"
[x] 21	Accept	http	Financeiro.clientes.interno	MoneyWorld.internet.externo	log,loglimit="30"
22					
[x] 23	Accept	pop3	Teodoro.clientes.interno	MailServer.servers.interno	log,loglimit="30"
[x] 24	Accept	smtp	Teodoro.clientes.interno	MailServer.servers.interno	log,loglimit="30"
[x] 25	Drop	any	Compras.clientes.interno	any	log,loglimit="30"
26					
[x] 27	Accept	samba	hp9000dn.clientes.interno	FileServer.servers.interno	log,loglimit="30"
28					
29					
30					
[x] 31	Portfw	pop3	internet.externo	MailServer.servers.interno	log,loglimit="30"
[x] 32	Portfw	smtp	internet.externo	MailServer.servers.interno	log,loglimit="30"
33					
[x] 34	Accept	ping	Ibm.internet.externo	firewall	log,loglimit="30"
[x] 35	Accept	ssh	Linus.internet.externo	firewall	log,loglimit="30"

Figura 89: Rules: SSH

Com isso, finalizamos o *set* proposto de regras na elaboração deste cenário-exemplo. Entretanto, existem ainda algumas regras não citadas diretamente nesta seção, que são necessárias para o funcionamento do firewall. Frisando *novamente* a questão do Vuurmuur: para que ele funcione corretamente, devemos liberar absolutamente **tudo** que se for necessário utilizar. Sendo assim, vamos tentar levantar algumas regras que faltaram:

- Alterar o endereço na saída para a Internet das máquinas internas para o endereço IP válido, de modo que possa estabelecer a comunicação com a Internet.
- Liberar a comunicação no firewall de possíveis serviços que deverão estar disponíveis caso estejamos fazendo uso diretamente do firewall.
- Fazer a liberação de possíveis serviços adicionais que se deseje.

Vamos à primeira questão:

Todos sabemos hoje que o sistema de endereçamento IPv4 já se encontra demasiadamente sobrecarregado, e que em breve o IPv6 irá resolver este problema. Para solucionar o problema da falta de endereços válidos para todos, utiliza-se o conceito de mascaramento de IP. Geralmente, temos nossa rede interna formada por segmentos de IP inválidos na Internet, que utilizam faixas específicas para tal fim, como a rede 192.168.0.0/16 utilizada em nossos exemplos. O problema é que, se o pacote de nossa rede interna sair para a Internet com este endereço, a outra ponta não irá saber para quem responder, visto que podem existir milhares ou milhões de máquinas no mundo utilizando este endereço. Para resolver esse problema, ordenamos ao nosso roteador de Internet que substitua o endereço interno pelo seu próprio endereço, que com pouquíssimas exceções, é o endereço válido. Desta forma, a outra ponta saberá de quem exatamente é este pacote, podendo assim estabelecer a comunicação.

Existem duas formas de conseguirmos fazer o mascaramento de IPs através do Iptables: utilizando o *target MASQ* ou o *target SNAT*. O SNAT consome menos CPU para este processo, uma vez que já especificamos diretamente qual o endereço a ser trocado, eliminando o trabalho do servidor de ter que reconhecer o endereço a ser utilizado.

Regra 15: estabelecer o mascaramento de IPs.

A tela para o SNAT é bem simples:

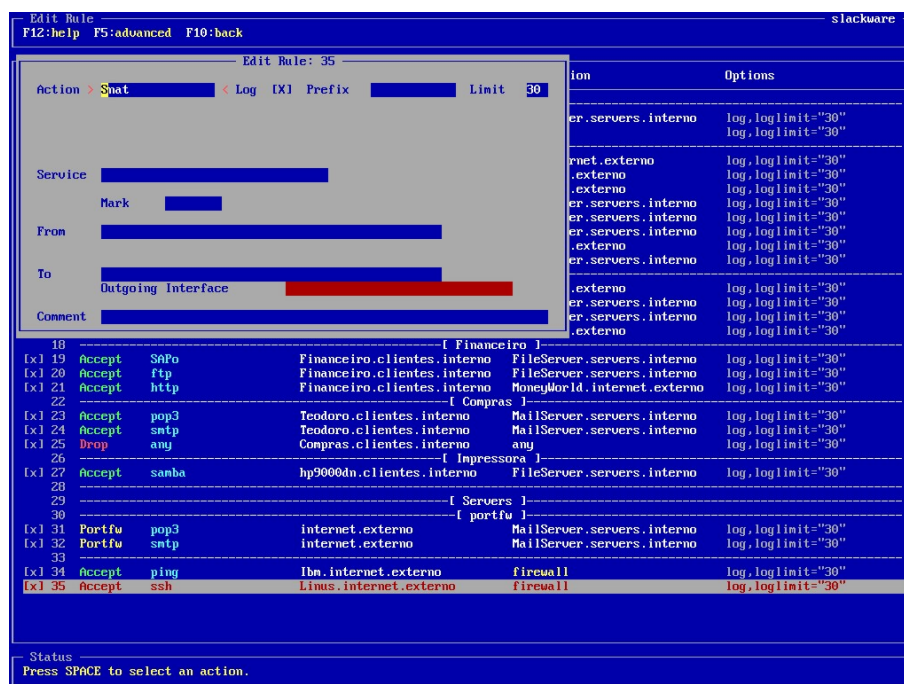


Figura 90: SNAT

Service: Informamos o serviço, que em nosso caso é **Any** (qualquer). Assim, não precisamos colocar o SNAT para cada serviço autorizado para cada máquina cliente. É importante que coloquemos esta regra no início, para que o processamento do pacote não pare antes de chegar ao mascaramento.

From e To: Esses já são campos tradicionais do Vuurmuur. Em nosso caso, o campo de destino (**To**) será a Internet, desta forma o firewall irá saber para qual interface deverá mascarar o endereço. No Vuurmuur, não precisamos informar o endereço que queremos que seja alterado no pacote, pois ele mesmo já captura o endereço da interface, e monta diretamente na regra.

Adicionalmente, temos um campo, **Outgoing Interface**, que serve para especificarmos uma interface para a qual o endereço deve ser alterado. Usamos isto caso existam duas interfaces em uma mesma rede. Observe no topo da figura 92 nossa regra criada:

Nr.	Action	Service	Source	Destination	Options	
1				[Snat]		
[x]	2	Snat	any	clientes.interno	internet.externo	out_int="ip", log, loglimit
	3			[Informatica]		
[x]	4	Drop	ssh	Bruno.clientes.interno	FileServer.servers.interno	log, loglimit="30"
[x]	5	Accept	any	Informatica.clientes.interno	any	log, loglimit="30"
	6			[Administrativo]		
[x]	7	Drop	http	Administrativo.clientes.inter	Uol.internet.externo	log, loglimit="30"
[x]	8	Accept	http	Administrativo.clientes.inter	internet.externo	log, loglimit="30"
[x]	9	Accept	https	Administrativo.clientes.inter	internet.externo	log, loglimit="30"
[x]	10	Accept	pop3	Administrativo.clientes.inter	MailServer.servers.interno	log, loglimit="30"
[x]	11	Accept	smtp	Administrativo.clientes.inter	MailServer.servers.interno	log, loglimit="30"
[x]	12	Accept	samba	Administrativo.clientes.inter	FileServer.servers.interno	log, loglimit="30"
[x]	13	Accept	msn	Administrativo.clientes.inter	internet.externo	log, loglimit="30"
[x]	14	Accept	SAPo	Administrativo.clientes.inter	FileServer.servers.interno	log, loglimit="30"
	15			[Contabilidade]		
[x]	16	Accept	msn	Contabilidade.clientes.intern	internet.externo	log, loglimit="30"
[x]	17	Accept	pop3	Contabilidade.clientes.intern	MailServer.servers.interno	log, loglimit="30"
[x]	18	Accept	smtp	Contabilidade.clientes.intern	MailServer.servers.interno	log, loglimit="30"
[x]	19	Accept	ping	Contabilidade.clientes.intern	internet.externo	log, loglimit="30"
	20			[Financeiro]		
[x]	21	Accept	SAPo	Financeiro.clientes.interno	FileServer.servers.interno	log, loglimit="30"
[x]	22	Accept	ftp	Financeiro.clientes.interno	FileServer.servers.interno	log, loglimit="30"
[x]	23	Accept	http	Financeiro.clientes.interno	MoneyWorld.internet.externo	log, loglimit="30"
	24			[Compras]		
[x]	25	Accept	pop3	Teodoro.clientes.interno	MailServer.servers.interno	log, loglimit="30"
[x]	26	Accept	smtp	Teodoro.clientes.interno	MailServer.servers.interno	log, loglimit="30"
[x]	27	Drop	any	Compras.clientes.interno	any	log, loglimit="30"
	28			[Impressora]		
[x]	29	Accept	samba	hp9000dn.clientes.interno	FileServer.servers.interno	log, loglimit="30"
	30			[Servers]		
	31			[portf]		
[x]	33	Portfu	pop3	internet.externo	MailServer.servers.interno	log, loglimit="30"
[x]	34	Portfu	smtp	internet.externo	MailServer.servers.interno	log, loglimit="30"
	35					
[x]	36	Accept	ping	Ibn.internet.externo	firewall	log, loglimit="30"
[x]	37	Accept	ssh	Linus.internet.externo	firewall	log, loglimit="30"

Status Ready.

Figura 91: Liberação de regras para o firewall.

Regra 16: Liberar os serviços para o Firewall.

Em nosso caso, iremos autorizar todos os serviços no sentido Firewall → Rede Interna, e alguns serviços no sentido Firewall → Internet. Com isso, finalizamos a criação cenário-exemplo, e estamos prontos para aplicar nossas regras.

Nr.	Action	Service	Source	Destination	Options
[x] 8	Accept	http	Administrativo.clientes.inter	internet.externo	log,loglimit="30"
[x] 9	Accept	https	Administrativo.clientes.inter	internet.externo	log,loglimit="30"
[x] 10	Accept	pop3	Administrativo.clientes.inter	MailServer.servers.interno	log,loglimit="30"
[x] 11	Accept	smtp	Administrativo.clientes.inter	MailServer.servers.interno	log,loglimit="30"
[x] 12	Accept	samba	Administrativo.clientes.inter	FileServer.servers.interno	log,loglimit="30"
[x] 13	Accept	msn	Administrativo.clientes.inter	internet.externo	log,loglimit="30"
[x] 14	Accept	SAPo	Administrativo.clientes.inter	FileServer.servers.interno	log,loglimit="30"
15			[Contabilidade]		
[x] 16	Accept	msn	Contabilidade.clientes.intern	internet.externo	log,loglimit="30"
[x] 17	Accept	pop3	Contabilidade.clientes.intern	MailServer.servers.interno	log,loglimit="30"
[x] 18	Accept	smtp	Contabilidade.clientes.intern	MailServer.servers.interno	log,loglimit="30"
[x] 19	Accept	ping	Contabilidade.clientes.intern	internet.externo	log,loglimit="30"
20			[Financeiro]		
[x] 21	Accept	SAPo	Financeiro.clientes.interno	FileServer.servers.interno	log,loglimit="30"
[x] 22	Accept	ftp	Financeiro.clientes.interno	FileServer.servers.interno	log,loglimit="30"
[x] 23	Accept	http	Financeiro.clientes.interno	MoneyWorld.internet.externo	log,loglimit="30"
24			[Compras]		
[x] 25	Accept	pop3	Teodoro.clientes.interno	MailServer.servers.interno	log,loglimit="30"
[x] 26	Accept	smtp	Teodoro.clientes.interno	MailServer.servers.interno	log,loglimit="30"
[x] 27	Drop	any	Compras.clientes.interno	any	log,loglimit="30"
28			[Impressora]		
[x] 29	Accept	samba	hp9000dn.clientes.interno	FileServer.servers.interno	log,loglimit="30"
30					
31			[Servers]		
32			[portfu]		
[x] 33	Portfu	pop3	internet.externo	MailServer.servers.interno	log,loglimit="30"
[x] 34	Portfu	smtp	internet.externo	MailServer.servers.interno	log,loglimit="30"
35					
[x] 36	Accept	ping	lbn.internet.externo	firewall	log,loglimit="30"
[x] 37	Accept	ssh	Linus.internet.externo	firewall	log,loglimit="30"
38			[Firewall]		
[x] 39	Accept	any	firewall	clientes.interno	log,loglimit="30"
[x] 40	Accept	dns	firewall	internet.externo	log,loglimit="30"
[x] 41	Accept	ftp	firewall	internet.externo	log,loglimit="30"
[x] 42	Accept	http	firewall	internet.externo	log,loglimit="30"
[x] 43	Accept	ping	firewall	internet.externo	log,loglimit="30"
[x] 44	Accept	traceroute	firewall	internet.externo	log,loglimit="30"
[x] 45	Accept	whois	firewall	internet.externo	log,loglimit="30"

Status
Ready.

Figura 92: Snat no topo das regras

Quando tivermos todas nossas regras prontas, podemos voltar ao Menu principal e aplicá-las. Caso contrário, elas só ficarão na configuração do Vuurmuur, e não serão repassadas ao Iptables. Podemos fazer isso pressionando <F11>. Com isso, temos nosso firewall rodando e nossa rede segura, podendo se alterar qualquer parâmetro, a qualquer momento.

3.6 Limitações e Contribuição

Obviamente, o Vuurmuur não é um programa perfeito, como qualquer outro existente. Entretanto, por encontrar-se em estado *beta*, já agrega inúmeras das funcionalidades esperadas.

O Vuurmuur é mantido pelo seu criador, Victor Julien, até hoje, mas a comunidade open-source sugere diariamente, através da lista de discussão oficial, ou conversa com o próprio Victor. Muitos casos específicos de coisas que não eram possíveis de se fazer com o Vuurmuur já foram implementadas, como o suporte a servidor e cliente DHCP, que está em testes na última versão *alpha*, e deverá ser integrada já no próximo release *beta* do Vuurmuur.

Alguns dos recursos à serem implementados no Vuurmuur, de acordo com o criador, e o site oficial:

- Suporte à IPv6
- Suporte à ulog
- Suporte à criação de grupos de serviços
- Aprimoração de serviços fornecidos pelo Iptables, para serem fornecidos também pelo Vuurmuur
- Suporte ao Layer-7, que adiciona inúmeras funcionalidades ao Iptables.

Com o tempo e a ajuda da comunidade, todos esses recursos serão com certeza implementados, tornando este produto ainda mais completo. Abaixo, está a última tela do Vuurmuur que ainda não foi mostrada, o **About**, que disponibiliza algumas informações sobre a Licença, os links oficiais, e agradecimentos às pessoas envolvidas no projeto.

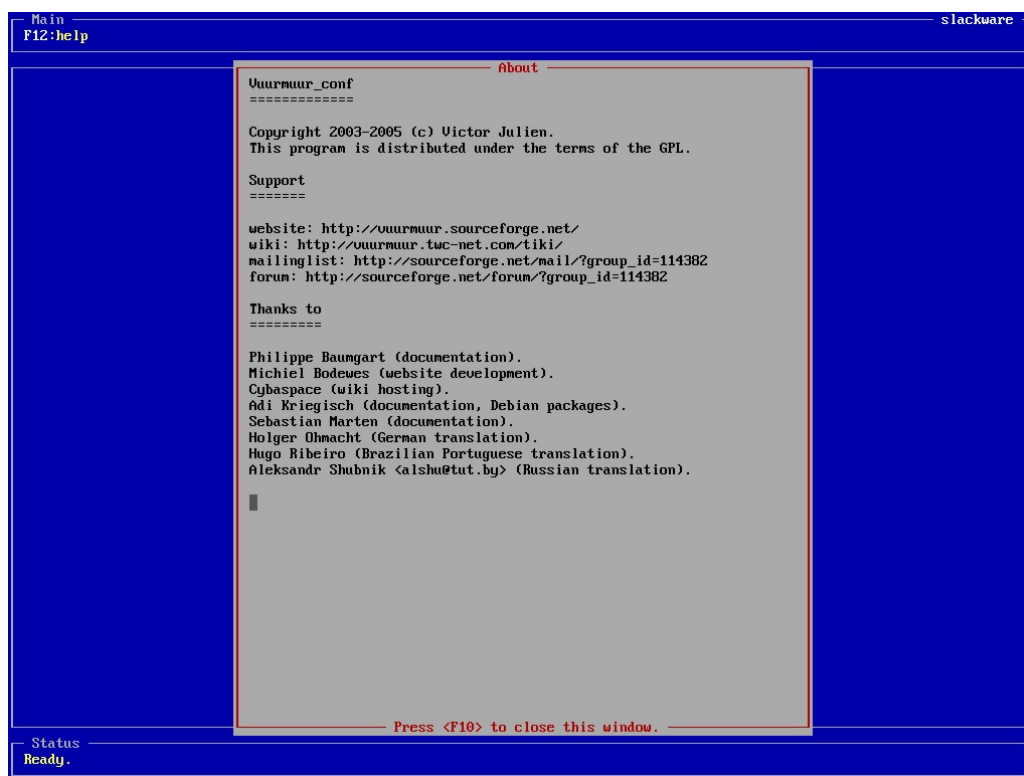


Figura 93: About

4 *Conclusão*

Neste trabalho, pudemos acompanhar e conhecer completamente o Vuurmuur, desde sua filosofia de trabalho, até um completo guia sobre sua implementação, com exemplos detalhados e explicações sobre cada opção disponível.

É indiscutível a necessidade da segurança em uma rede de computadores hoje em dia, por menor que seja, se estendendo até a ausência da rede, ou seja, o computador pessoal. Várias práticas de segurança juntas aumentam o nível de segurança, e ao menos tentam eliminar ao máximo os vários tipos de ameaça existentes. A situação dos ataques se torna cada vez mais evidente, com milhares de pessoas mal intencionadas à procura de qualquer coisa que sirva de brecha para o ganho de informações, ou simplesmente para a destruição delas. Neste contexto, o firewall é ponto principal e vital para a implementação de uma boa segurança, e o Vuurmuur se mostra um grandioso aliado a essa luta.

Pudemos conhecer neste trabalho uma alternativa livre de *front-end* para Iptables, capaz de agregar funcionalidades em relação a ele, atuando tanto como facilitador para as pessoas que não tem um conhecimento avançado da sintaxe do Iptables, quanto para os administradores já experientes que desejam ter um firewall excelente no sentido da simplicidade e mobilidade com as regras. A partir deste trabalho, temos a capacidade de partir do zero, de sabermos como ele identifica as coisas, como as estrutura, e como fazemos isso virar realidade.

Pudemos ver que se trata de um produto novo, porém relativamente maduro, que para seu tempo de vida já consegue reunir muitas funcionalidades aliadas a uma forma simples de se operacionalizar tudo isso, combinação difícil de se encontrar em produtos disponíveis, acentuada ainda mais por ser um produto completamente livre, de código-aberto, disponível para toda a comunidade.

Adicionalmente, as 5 línguas disponíveis refletem o seu sucesso, que aponta o anseio de diversas comunidades ao redor do mundo em aliar a já existente facilidade do produto com a acessibilidade de estar fazendo algo em sua língua.

A intenção deste trabalho foi desenvolver algo que fosse de grande contribuição para a comunidade open-source, em especial as pessoas interessadas em segurança de informação e *firewalls*, agindo de certa forma como incentivo às boas práticas na elaboração de planos de proteção para redes, além de obviamente promover o uso de software livre.

Finalmente, fica clara e desmistificada aqui a impressão de que elaborar planos de firewall é algo complicado, ao mesmo tempo que deixa evidente a necessidade de uma elaboração bem pensada, de algo que atenda às necessidades, mas que não prejudique a praticidade por isso.

Como sugestão para trabalhos futuros, podemos destacar a de se fazer uma análise mais a fundo nos mecanismos internos do Vuurmuur, afim de entender realmente o que acontece dentro dele. Além dessa, a especificação a fundo dos parâmetros e regras, talvez em um trabalho que integrasse o Vuurmuur e os recursos do Iptables.

Referências

- 1 RUSSEL, Rusty. *Linux 2.4 Packet filtering HowTO*. 01 2001. Disponível em: <http://www.netfilter.org/documentation/HOWTO/packet-filtering-HOWTO.html>. Acesso em: 13 jun. 2005.
- 2 VIANNA, William da Silva. *Implementação de Segurança para Redes Locais com acesso à Internet*. Dissertação (Mestrado) — Universidade Federal de Lavras, 2004.
- 3 JULIEN, Victor. *Vuurmuur: an iptables front-end*. Disponível em: <http://vuurmuur.sourceforge.net>. Acesso em: 28 mai. 2005.
- 4 MULLER, Mary Stela. *Normas e Padrões para teses, dissertações e monografias*. Londrina: Eduel, 2003.
- 5 RUSSEL, Rusty. *Linux 2.4 NAT HowTO*. 01 2002. Disponível em: <http://www.netfilter.org/documentation/HOWTO/NAT-HOWTO.txt>. Acesso em: 12 jun. 2005.
- 6 RUSSEL, Rusty. *Linux 2.4 Networking concepts HowTO*. 07 2001. Disponível em: <http://www.netfilter.org/documentation/HOWTO/networking-concepts-HOWTO.html>. Acesso em: 12 jun. 2005.
- 7 AUTOPACKAGE. *Autopackage – a distributionless package program*. Disponível em: <http://www.autopackage.org>. Acesso em: 03 jun. 2005.
- 8 IP Traffic Volume. Disponível em: <http://iptrafficvolume.sourceforge.net>. Acesso em: 01 jun. 2005.
- 9 BLACK, Uyless. *TCP/IP and Related Protocols*. 2. ed. Nova Iorque: [s.n.], 1994.
- 10 BARTZ, Manfred. *Netfilter log format*. 2001. Disponível em: <http://logi.cc/linux/netfilterlog-format.php3>. Acesso em: 12 jun. 2005.
- 11 ELIZABETH, David. *Building Internet Firewalls*. 2. ed. Denver: [s.n.], 2001.