CENTRO UNIVERSITÁRIO FEEVALE

LEÔNIDAS KLEIN SALDANHA

FIREWALLS DE BAIXO CUSTO

Novo Hamburgo, dezembro de 2007.

LEÔNIDAS KLEIN SALDANHA

FIREWALLS DE BAIXO CUSTO

Centro Universitário FEEVALE
Instituto de Ciências Exatas e Tecnológicas
Curso de Ciência da Computação
Trabalho de Conclusão de Curso

Professor Orientador: Msc. Vandersílvio da Silva

Novo Hamburgo, dezembro de 2007.

AGRADECIMENTOS

Gostaria de agradecer a todos os que, de alguma maneira, contribuíram para a realização desse trabalho de conclusão, em especial: a minha esposa Luciele e minha família que muito me motivaram e apoiaram.

Aos amigos e às pessoas que convivem comigo diariamente, minha gratidão, pelo apoio emocional - nos períodos mais difíceis do trabalho. Em especial aqueles que já passaram por este período e sabem o quão complicado é.

RESUMO

O trabalho apresenta o estudo sobre dois *firewalls* sob licença *General Public License* (GPL) voltados para o segmento *Small Office/Home Office* (SOHO) de empresas. Também são apresentadas ferramentas para testes de vulnerabilidades aplicadas sobre os *firewalls* estudados. Como a insegurança na *Internet* cresce cada vez mais, esse segmento tende a ser mais vulnerável a ameaças. Seus níveis de segurança são relativamente baixos, atraindo a atenção dos *crackers*, os quais têm a tendência de voltar-se para este segmento pela fragilidade de seus sistemas de proteção, quando existentes. Sendo assim, a preocupação com a falta de segurança destas redes não pode ser esquecida. Os cuidados que elas devem ter com seus dados é crucial para que diminua as chances de algum indivíduo mal intencionado causar alguma forma de prejuízo. As pequenas empresas muitas vezes não possuem alguém especializado para definir uma política de segurança consistente ou para administrar sua rede. Da mesma maneira não dispõem de capital para investir em *firewalls* de alto desempenho e segurança, que possuem custo elevado. Por esse motivo, o estudo é embasado em duas ferramentas que não possuem custo inicial, possibilitando a aquisição das mesmas de uma maneira simplificada. São estudadas características comuns e apresentada uma forma de teste, e posteriormente explicadas para o bom entendimento do funcionamento desta tecnologia indispensável nos dias de hoje. Após estes estudos é elaborada a forma de comparação para averiguar qual das ferramentas melhor se enquadra para este segmento empresarial.

Palavras-chave: *Firewall*. Segurança de Pequenas Empresas. Segurança com custo baixo.

ABSTRACT

The monography shows the study of two firewalls under General Public License (GPL) aimed to the small office/home office (SOHO) enterprise segment. It also shows tools for tests of vulnerabilities to be applied to the studied firewalls. Because the Internet insecurity increases more and more each day, this segment tends to be more vulnerable to threats. They security levels, when they do exist, tend to be relatively low, which may attract crackers attention. This way, the preoccupation with the lack of security cannot be forgotten. The attention that they need to have with their data is crucial in order to decrease the chances of any badly-intentioned person causing any form of damage. The SOHO many times does not have any specialized person to create a consistent security policy or to administrate its network; the same way, they do not have enough capital to invest in high performance and safety firewalls with high costs. Because of this, this study is based on two tools that do not have any initial cost, making their acquisition an easy way. Firewall common characteristics will be studied and a way to test the firewall will be presented; afterwards they will be explained for a good understanding of how this indispensable technology works nowadays. After this study, a way of comparison is developed to verify which is better and which fits better in this enterprise segment.

Key words: Firewall. Small office security. Low cost security.

LISTA DE FIGURAS

Figura 1.1 - Modelo OSI. Camada em que trabalha o filtro de pacotes. 20
Figura 1.2 - Exemplo do funcionamento da NAT. 23
Figura 1.3 - Modelo OSI. Camada em que trabalha o *proxy*. 26
Figura 1.4 - Modelo OSI. Camada em que trabalha o *proxy* a nível de circuitos. 28
Figura 1.5 - Modelo OSI. Camadas em que trabalha *proxy stateful*. 28
Figura 1.6 - Túnel entre duas filiais, de um *firewall* para outro. 30
Figura 1.7 - Arquitetura de rede sem DMZ. 33
Figura 1.8 - Arquitetura de rede com DMZ. 34
Figura 1.9 - DMZ *Dual Firewall.* 35
Figura 2.1 - Interface Gráfica do *ipcop*. 39
Figura 2.2 - Endian Firewall, Página inicial. 42
Figura 3.1 - *Plugins* do *Nessus.* 47
Figura 3.2 - Opções para busca por vulnerabilidade do *Nessus*. 50
Figura 3.3 - Tela inicial do *HoneyBOT*. 54
Figura 3.4 - Tela inicial do *CesarFTP*. 56
Figura 3.5 - Tela inicial do *Wireshark.* 57
Figura 3.6 - *VMware Server* 58
Figura 3.7 - *Linux Kurumin* 7 sobre o *WMware Server*. 59
Figura 3.8 - Ambiente utilizado para os testes. 60
Figura 3.9 - Tela do *HoneyBOT* após a verificação do *Nessus*. 63
Figura 3.10 - Detalhe de um pacote recebido pelo *HoneyBOT*. 63
Figura 3.11 - Gerenciado de tarefas da estação de trabalho durante ataque *syn-flood*. 65
Figura 3.12 - Pacotes Capturados pelo *Wireshark.* 67
Figura 3.13 - *Ipcop*, opção para desabilitar *ping*. 70
Figura 3.14 - Tela de boas vindas do ipcop. 81

Figura 3.15 - Selecão de idioma do ipcop. 82
Figura 3.16 - Seleção de endereço IP e máscara. 82
Figura 3.17 - Configuração do nome de domínio. 83
Figura 3.18 - Configuração de ISDN. 83
Figura 3.19 - Configuração dos tipos de rede. 84
Figura 3.20 - Configuração de senhas, *root, admin, backup*. 84
Figura 3.21 - Tela de boas vindas do EFW. 85
Figura 3.22 - Seleção de idioma. 86
Figura 3.23 - Configuração do endereço IP e máscara. 86
Figura 3.24 - Configuração de senhas. 87
Figura 3.25 - Configuração da interface *RED*. 87
Figura 3.26 - Configuração das interfaces *ORANGE* e *BLUE*. 87
Figura 3.27 - Configuração da interface *GREEN*, *hostname* e *Domain name*. 88
Figura 3.28 - Configuração de endereços IPs de servidores DNS. 88

LISTA DE QUADROS

Quadro 1.1 - Exemplo da tabela de roteamento utilizada pela NAT. 24
Quadro 3.1 - Resumo dos resultados obtidos nos testes. 73
Quadro 3.2 – Comparação das funcionalidades dos *firewalls*. 74

LISTA DE ABREVIATURAS E SIGLAS

ADSL	*Asymmetric Digital Subscriber Line*
CD	*Compact Disc*
DHCP	*Dynamic Host Configuration Protocol*
DMZ	*Demilitarized Zones*
DNAT	*Destination Network Address Translation*
DNS	*Domain Name System*
DoS	*Deny of Service*
DVD	*Digital Video Disc*
EFW	*Endian Firewall*
FTP	*File Transfer Protocol*
GB	*Giga byte*
Ghz	*Gigahertz*
GPL	*GNU Public License*
HTTP	*Hypertext Transfer Protocol*
ICMP	*Internet Control Message Protocol*
ID	*Identificador*
IDS	*Intrusion Detection System*
IGMP	*Internet Group Management Protocol*
IP	*Internet Protocol*
IPv4	IP versão 4
IPv6	IP versão 6
MB	*Mega Bytes*
Mbps	*Mega bytes per second*
Mhz	*Megahertz*
NASL	*Nessus Attack Scripting Language*

NAT	*Network Address Translation*
NTP	*Networ Time Protocol*
OSI	*Open Systems Interconnection*
PC	*Personal Computer*
PCI	*Peripheral Component Interconnect*
PPP	*Point-to-Point Protocol*
POP	*Post Office Protocol*
QoS	*Quality of Service*
RAM	*Random Access Memory*
RPC	*Remote Procedure Call*
SATAN	*Security Analysis Tool for Auditing Networks*
SIP	*Session Initiation Protocol*
SMTP	*Simple Mail Transfer Protocol*
SNAT	*Source Network Address Translation*
SNMP	*Simple Network Management Protocol*
SOCKS	*Sockets*
SOHO	*Small Office/Home Office*
SSH	*Security Shell*
SSL	*Security Sockes Layer*
TCP	*Transmission Control Protocol*
TCP/IP	*Transmission Control Protocol Internet Protocol*
TLS	*Transmission Layer Security*
TOS	*Type of Service*
UDP	*User Datagram Protocol*
USB	*Universal Serial Bus*
VPN	*Virtual Private Network*
WWW	*World Wide Web*

SUMÁRIO

INTRODUÇÃO 13

1 FIREWALLS 16
1.1 Definição de *firewall* 16
1.2 *Firewalls* de *software* 17
1.3 *Firewalls* de *hardware* 18
1.4 *Firewalls* integrados 19
1.5 Tecnologias de *firewall* 19
1.5.1 Filtro de pacotes 19
1.5.1.1 Filtro de pacotes *stateless* 21
1.5.1.2 Filtro de pacotes *stateful* 21
1.5.2 Autenticação de acesso 22
1.5.3 NAT 22
1.5.3.1 NAT Estática 24
1.5.3.2 NAT Dinâmica 25
1.5.4 *Proxy* 25
1.5.4.1 *Proxy* a Nível de Circuito 27
1.5.5 *Stateful Firewall* 28
1.5.6 *Firewalls* Transparentes 29
1.5.7 VPN 29
1.5.8 Relatórios e *Logs* 31
1.6 Localização do *Firewall* 31
1.6.1 DMZ (*Demilitarized Zones*) 32
1.6.2 Arquitetura de *Firewall* Único 32
1.6.3 Arquitetura *Dual-Firewall* 34
1.7 O Que *Firewalls* Não Podem Fazer 35

2 *FIREWALLS* TESTADOS 38
2.1 *Ipcop* 38
2.2 *Endian* 40

3 AVALIAÇÃO DOS *FIREWALLS* 44
3.1 Ferramentas Avaliadas 44
3.1.1 *Nmap* 44
3.1.2 SATAN 45
3.1.3 *Nessus* 45
3.1.3.1 Cliente e Servidor 46
3.1.3.2 *Plugins* 46

3.1.3.3 Base de Conhecimento 47
3.1.3.4 NASL 48
3.1.3.5 Opções Para Busca por Vulnerabilidades 48
3.1.3.6 *Nmap* no *Nessus* 50
3.1.4 *HoneyBOT* 53
3.1.5 *Hping* 54
3.1.6 *CesarFTP* 56
3.2 *Wireshark* 56
3.3 *VMware Server* 57
3.4 Descrição do Ambiente Para Testes dos *Firewalls* 59
3.5 Execução e Resultados dos Testes 61
3.5.1 *Ipcop* 66
3.5.2 *Endian* 70
3.6 *Endian* x *ipcop* – Análise de funcionalidades 73
CONCLUSÃO **75**
REFERÊNCIAS BIBLIOGRÁFICAS **78**
ANEXO I **81**
ANEXO II **85**
ANEXO III **89**

INTRODUÇÃO

Dadas as atuais circunstâncias nas quais as pequenas empresas são submetidas por estarem conectadas na *Internet*, o segmento S*mall Office/Home Office*[1] (SOHO) deve se manter protegido da melhor forma possível. Uma ferramenta indispensável nos dias de hoje é o *firewall*. Ele mantém (na medida do possível) a rede interna em segurança contra a rede externa (*Internet*). É instalado na fronteira com a *Internet*, fazendo o roteamento de pacotes com base na política de segurança aplicada. Os benefícios trazidos por esta ferramenta são muitos, dentre eles podem ser destacados: restrição de acesso à rede interna, controle sobre o tráfego da *Internet*, relatórios de segurança por meio de *logs* e filtro de conteúdo indevido.

Todo o tráfego de entrada ou saída da rede interna deve necessariamente passar pelo *firewall*. Desta forma é possível analisar os pacotes e permitir ou não a passagem dos mesmos nos dois sentidos (entrada e saída). “Achamos que um *firewall* é qualquer dispositivo, *software*, arranjo ou equipamento que limita o acesso à rede. Ele pode ser uma caixa que você compra ou constrói ou uma camada de *software* em alguma outra coisa” (CHESWICK, BELLOVIN e RUBIN, 2005, p.177).

Quando atua na fronteira com a *Internet* poupam-se recursos dos computadores, pois sem ele cada estação de trabalho teria de se proteger sozinha. Outro ponto positivo é a centralização dos serviços de proteção, contra perigos externos, em uma máquina preparada para trabalhar nesta tarefa. Em contrapartida o *firewall* cria um canal estreito por onde passa toda a informação que sai ou entra na rede e que pode comprometer o desempenho da mesma, mas que não justifica a não instalação do *firewall* (STREBE e PERKINS, 2002).

Os *firewalls* podem possuir serviços dentre os quais se destacam: filtro de pacotes,

[1] Este segmento é representado no Brasil pelas micro e pequenas empresas.

Network Address Translation (NAT) e *proxy*. A maioria deles também tem a capacidade de realizar outros dois serviços: autenticação criptografada e rede virtual privada (VPN, *Virtual Private Network*) (STREBE e PERKINS, 2002).

A utilização dos recursos apresentados anteriormente depende das necessidades da empresa, como o ramo de atividade e o tamanho da mesma. Empresas dentro de um mesmo ramo de atividade também podem possuir necessidades diferentes, por isto é importante que seus gestores verifiquem a importância de seus dados para tomar as providências.

As SOHO utilizam cada vez mais as conexões de alta velocidade com a *Internet*, devido ao seu custo ter reduzido ao longo dos anos. Assim, as mesmas ficam expostas na *Internet* por longos períodos de tempo. Devido a isto e pelo fato dos investimentos em segurança da informação serem limitados, o risco de invasão aumenta.

Não utilizar um *firewall* em uma pequena empresa diminui a sua segurança. Em ambos os casos (utilizar ou não um *firewall*) ainda existem algumas medidas que devem ser tomadas como manter as atualizações dos sistemas operacionais em dia e colocar um antivírus em cada estação, atualizando-o regularmente. Se o sistema operacional usado for o *Windows* XP, o mesmo contém um simples *firewall* que somente filtra pacotes e bloqueia portas, mas é preciso ser ativado. No caso de conexão por *Asymmetric Digital Subscriber Line* (ADSL), outra medida de segurança seria adotar *modems* roteadores que possuam filtro de pacotes, desta maneira os mesmos podem ser configurados para aumentar a segurança da rede prevenindo alguns protocolos ou conexões que possam ocasionar risco para a empresa.

As SOHO serão o principal alvo para os *crackers* em um futuro próximo. Isto devido aos poucos recursos e investimentos em segurança que possuem. As grandes empresas comumente se preocupam com a segurança da informação e possuem capital para altos investimentos: novos equipamentos, novos programas e contratam uma pessoa ou uma equipe responsável por esse assunto. Isso as torna menos atrativas para o *cracker*, muitas vezes por incapacidade do mesmo de invadir empresas com altos níveis de segurança (ZMOGINSKI, 2006).

Este motivo demonstra que as SOHO não podem deixar a segurança de lado. Toda a empresa deveria conhecer quais são seus dados sensíveis, desta forma poderiam protegê-los.

Como se sabe existem diferentes tipos de segmentos empresariais e alguns destes possuem dados que, se comprometidos, podem afetar a terceiros e não somente a empresa. Dentre estes segmentos podem ser citados como exemplo: clínicas médicas, escritórios de advocacia e contabilidade. O tipo de dados que a empresa utiliza deve ser levado em conta no planejamento de configuração do *firewall*.

"Como regra geral, quanto mais visível é uma organização, maior é a probabilidade de ela atrair um *hacker* que a coloca em sua agenda" (STREBE e PERKINS, 2002, p. 186).

Os gestores das pequenas empresas podem não possuir o conhecimento devido sobre a necessidade de proteção da informação. Também pelo motivo do capital ser baixo, não é possível um investimento elevado em segurança. Devido a isto, o presente trabalho tem como objetivo testar dois *firewalls* de baixo custo e definir qual deles possui melhor segurança. Assim, pode-se apontar qual é o mais seguro para ser instalado no segmento SOHO de empresas. Com este resultado, as pequenas empresas poderão utilizar um *firewall* com baixo custo em suas redes de forma a aumentar a segurança das mesmas.

A seguir, é apresentado embasamento teórico sobre as definições de *firewall*, os tipos existentes e as tecnologias que este utiliza para oferecer proteção à rede interna. Da mesma forma são descritos os *layouts* básicos para o posicionamento do *firewall* de forma a obter maior segurança na rede.

Após a apresentação teórica sobre as tecnologias, no capítulo 2 são descritos os dois *firewalls* a serem testados. No capítulo 3 são descritas as ferramentas utilizadas para os testes e a maneira de como foram efetuados. Ao fim encontram-se a conclusão do trabalho e as referências bibliográficas.

1 FIREWALLS

Para um melhor entendimento das tecnologias existentes em um *firewall*, bem como as diferentes categorias às quais pertence, e também para uma melhor situação dentro do trabalho, abaixo encontram-se definições, tecnologias e categorias aos quais os *firewalls* pertencem ou oferecem

1.1 Definição de *firewall*

Firewall pode ser um dispositivo de *hardware* ou *software* utilizado para tentar dificultar o acesso indevido a informações sensíveis dentro da rede.

"Os *firewalls* são usados para criar pontos de controle de segurança nas fronteiras das redes privada. Ao fornecer a função de roteamento entre as redes privadas e a *Internet*, os *firewalls* inspecionam toda a comunicação passando entre as redes"[...] (STREBE e PERKINS, 2002. p 3).

Este não deve ser considerado o único meio de proteger a rede, outros mecanismos existentes devem ser utilizados em conjunto com o *firewall* (BORSCHEID, 2005). Em geral, o *firewall* é um meio de reforçar as políticas de segurança existentes na empresa, geralmente pertence a uma política maior, onde é uma das partes.

O *firewall* deve ser instalado na fronteira das redes para limitar o acesso entre elas. É normalmente instalado com o objetivo de proteger a rede interna da *Internet*, pelo fato da mesma ser uma rede pública não confiável. Também pode ser utilizado para separar redes internas que devem ter seu acesso limitado como, por exemplo, a rede do departamento financeiro de uma empresa.

Os recursos de um *firewall* não precisam estar unidos em um equipamento somente, estes podem ser distribuídos dentre vários computadores, embora esta prática não seja muito recomendada pelo fato do acréscimo da manutenção e por algumas de suas características dependerem das outras para se proteger. A recomendação é que seja instalado em um único equipamento, desta maneira os serviços serão integrados e o mesmo será menos vulnerável a ataques (STREBE e PERKINS, 2002).

Em geral, os *firewalls* compartilham algumas características: gerenciar e controlar o tráfego de rede, autenticar o acesso, agir como um intermediário, proteger recursos e gravar e relatar eventos (NOONAN e DUBRAWSKY, 2006).

Segundo Noonan e Dubrawsky (2006) existem três categorias onde os produtos de *firewalls* podem ser classificados: *firewalls* de *software*, *firewalls* de *hardware* e *firewalls* integrados.

1.2 *Firewalls* de *software*

São programas de computador desenvolvidos para serem instalados sobre um sistema operacional comum, como: *Windows*, *Solaris*, *Linux*.

Pelo motivo de seu funcionamento ser sobre um sistema operacional, existe a necessidade de atualizações contra falhas de segurança ou problemas de funcionamento que possam facilitar um ataque ou provocar o mau funcionamento do *firewall*.

A vantagem da utilização desta categoria de *firewalls* é a possibilidade de utilizá-lo não somente para uma aplicação, mas de transformá-lo em um servidor multitarefa. Nele pode ser instalado, por exemplo: *Domain Name System* (DNS), filtro *anti-spam*, servidor de *e-mails*, outra vantagem seria a facilidade de conserto do computador em caso de falha, pois o mesmo provavelmente utiliza componentes de *hardware* que podem ser adquiridos facilmente, até mesmo a troca do *hardware* inteiro é possível.

A desvantagem que *firewalls* de *software* sofrem é a contínua atualização para falhas descobertas no sistema operacional ou aplicação de *firewall*. Desta maneira, o administrador deverá estar atento às atualizações que surgem. Quando uma falha é descoberta o fornecedor do *firewall* cria uma atualização para correção da mesma. Antes de disponibilizá-la, o mesmo irá efetuar testes sobre a atualização. Juntamente com a atualização publicará comentários

alertando para possíveis complicações decorrentes da instalação, quando existentes. Mas, isso não significa que o pacote não está sujeito a problemas, este pode causar o mau funcionamento do sistema operacional ou do *software* de *firewall*. Em alguns casos, quando o administrador da rede for procurar solução para um problema causado por algo que deveria solucionar outro problema, ele poderá ficar sem suporte quando o fabricante do *firewall* culpar o fabricante do sistema operacional e vice-versa. Isso não acontece quando se utilizam programas de um mesmo fabricante.

Também existe a questão do desempenho, como o *hardware* e possivelmente o sistema operacional não foram projetados para ser um *firewall*, é provável que seu desempenho seja inferior a *firewalls* de *hardware* que são produzidos especificamente para este fim.

1.3 *Firewalls* de *hardware*

Estes dispositivos integram o *software* e o *hardware* de forma que ambos sejam altamente compatíveis. Desta forma o desempenho é superior ao de *firewalls* de *software*. O sistema operacional base, onde o programa apóia-se para o devido funcionamento, não é um sistema normal como o utilizado na categoria anterior, este é desenvolvido ou modificado para permitir um alto desempenho e interagir com o programa do *firewall* de uma forma mais eficaz.

Uma das maiores vantagens que se pode obter nesta categoria de *firewalls* é provavelmente o suporte técnico. Como mencionado anteriormente, existe a possibilidade de mais de um fabricante estar envolvido no produto final, como: o fabricante do *hardware*, o fabricante do sistema operacional e o do *firewall*. Isso não acontece na categoria presente, onde o fabricante é um só e somente ele deverá ser contatado na ocorrência de problemas.

As desvantagens são: quando um produto for descontinuado, onde o suporte para o mesmo poderá não ser mais oferecido ou se o fabricante deixar de existir. Se um problema for encontrado no *firewall*, o fabricante definirá quando uma atualização poderá ser lançada. Em *firewalls* de *hardware* a adição de novas funcionalidades pode não ser possível, ou se for, pode ser complicada, pelo motivo da integração com o *hardware* ser alta. Para a adição de novas funcionalidades, a compra de outros equipamentos pode ser necessária, o que irá aumentar a manutenção, o custo e complicar a topologia da rede.

1.4 *Firewalls* integrados

São dispositivos para múltiplos propósitos. Eles combinam as funções de *firewall* com outras funcionalidades como: acesso remoto de VPN, interconexão de redes com VPN, filtros de *spam*, antivírus e detecção ou prevenção de intrusão.

A vantagem desta categoria é a unificação de vários dispositivos em um só, reduzindo o custo e manutenção, o custo de compra de vários dispositivos e simplificando a topologia da rede onde se encontra.

A desvantagem é em respeito à falha do computador, onde vários serviços serão afetados. Outra é a maior exposição de um único equipamento a diferentes técnicas de ataques, pois cada serviço possui as suas. Problemas de conectividade podem ser aumentados, pois existem vários serviços que utilizam o mesmo sistema operacional base.

1.5 Tecnologias de *firewall*

Segundo Strebe e Perkins (2002) um produto *firewall* pode conter uma ou mais características dentre as essenciais abaixo:

1.5.1 Filtro de pacotes

Para gerenciar e controlar o tráfego de rede, existe o filtro de pacotes que é a característica mais básica que um *firewall* pode ter, estes podem ser encontrados em *modems* roteadores ADSL, por exemplo. Esse componente analisa o cabeçalho de cada pacote que entra ou sai da rede. Para tomar as decisões de aceitar ou rejeitar o pacote corrente, o mesmo analisa a porta de origem, o endereço *Internet Protocol* (IP) de origem, a porta de destino, o endereço IP de destino e algumas informações mais como: *ckecsums*, *flags* de dados e outras que o cabeçalho de um pacote *Transmission Control Protocol/Internet Protocol* (TCP/IP) pode conter. Este trabalha nas camadas de rede e transporte do modelo *Open Systems Interconnection* (OSI).

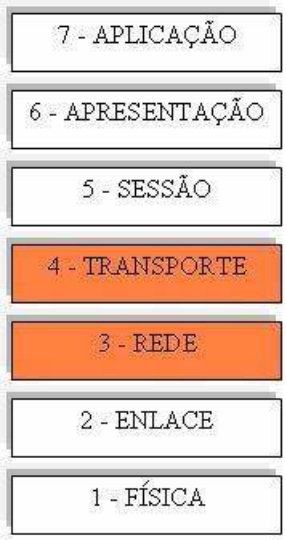

Figura 1.1 - Modelo OSI. Camada em que trabalha o filtro de pacotes.

O filtro pode ser configurado de duas formas: a primeira e mais recomendada é *default deny*, ou seja, todo o tráfego que não for expressamente permitido será bloqueado. A segunda é a de *default allow*, consiste em permitir todo o tráfego menos o que for proibido. Esta é mais falha, sendo que é mais complicado encontrar todo o tráfego que pode apresentar risco e proibi-lo do que simplesmente permitir os necessários.

Existem duas variações do filtro de pacotes, são elas: o filtro que não considera o estado da conexão chamado de filtro sem estados ou filtro *stateless* e o filtro que analisa a conexão como um todo, mantendo o estado das conexões chamado de filtro com estados ou filtro *stateful*.

Segundo Noonan e Dubrawsky (2006) uma conexão é a maneira que dois *hosts* utilizam para conseguirem se comunicar e serve para dois princípios que são: a identificação dos computadores um para o outro, desta maneira garantindo que os pacotes cheguem ao destino correto e não sejam entregues a quem não está participando da conexão. Também para definir como será feita a comunicação, seja por TCP que é orientado à conexão ou por *User Datagram Protocol* (UDP) e *Internet Control Message Protocol* (ICMP) que não são orientados à conexão. Quando um protocolo é orientado à conexão o mesmo possui mecanismos para assegurar que o pacote chegue ao seu destino, diferentemente dos não orientados à conexão.

Os *firewalls* utilizam as conexões entre os *hosts* para poder aplicar suas regras, baseando-se nelas, o mesmo distingue quais são permitidas e quais não são. Desta maneira

pode bloquear as conexões não permitidas e garantir passagem para as permitidas (NOONAN e DUBRAWSKY, 2006).

1.5.1.1 Filtro de pacotes *stateless*

Este tipo de filtro de pacotes não mantém os estados da conexão, por isso analisa pacote a pacote separadamente. Desta maneira o tráfego de retorno para a rede interna deve ser configurado no filtro, caso contrário será bloqueado.

Um filtro de pacotes sem estados pode ser usado para filtrar protocolos, ou seja, permitir ou negar a passagem de protocolos existentes como: UDP, *Transmission Control Protocol* (TCP), ICMP, *Internet Group Management Protocol* (IGMP). Também existe a possibilidade de bloquear ou permitir endereços IP's. Pode-se criar uma lista dos permitidos e restringir o resto ou vice-versa, o que é mais falho, pois mesmo bloqueando um IP de um *cracker*, o mesmo pode modifica-lo e tentar um novo ataque.

Outra falha é a não verificação do conteúdo do pacote, por exemplo: quando um usuário navega na *Internet* traz pacotes permitidos para dentro da rede, mas os mesmos podem conter cavalos-de-tróia que podem abrir brechas na segurança mesmo atrás de um *firewall*.

O filtro de pacotes *stateless* possui uma configuração simplificada e é eficaz no roteameto dos pacotes, o que não prejudica a eficiência da rede em caso de grande tráfego de dados. Quando possui-se um servidor *web* com grande volume de informação passando pela rede, pode-se configurar o filtro para permitir somente a porta 80 e bloquear todas as outras. Assim tem-se reduzido o número de portas de 65535 para 1 (HENMI, 2006).

Por essas razões, o filtro de pacotes sozinho não é considerado uma segurança eficaz tratando-se de redes empresariais, mas leva-se em conta que é melhor tê-lo do que nada, principalmente quando o orçamento (pequenas empresas) não é suficiente para colocar um *firewall* em uma máquina dedicada.

1.5.1.2 Filtro de pacotes *stateful*

O filtro de pacotes *stateful* guarda o estado da conexão, diferente do filtro *stateless*, assim consegue tomar decisões baseando-se na conexão e não por pacote individualmente.

Entende-se então que durante a comunicação entre *hosts* a conexão possui um estado, como por exemplo, quando o *host* A requisita iniciar uma comunicação com *host* B o estado da comunicação é “aguardando resposta de B”. Desta forma, os filtros *stateful*s mantêm este histórico e quando B responder A o filtro saberá que este pacote pertence a uma comunicação ativa e o deixará passar.

O filtro de pacotes *stateful* elimina alguns dos problemas que ocorrem no filtro de pacotes *stateless*, mas ainda assim não consegue analisar o conteúdo do pacote. Para tanto é necessário um filtro em uma camada de aplicação como o *proxy*.

1.5.2 Autenticação de acesso

Os *firewalls* podem efetuar a autenticação de acesso antes de efetuar a conexão, pois nos dias atuais é justo que empresas não queiram que qualquer um se conecte a serviços dentro de suas redes privadas. Desta maneira a autenticação possibilita restringir o acesso aos serviços.

Para autenticar o acesso, um *firewall* pode munir-se de alguns métodos como: a caixa de *login* onde devem ser informados o usuário e senha; certificados de segurança os quais são mais cômodos, pois não requerem intervenção por parte do usuário e o uso de chaves pré-compartilhadas, as mesmas possuem uma configuração mais fácil contra os certificados e também não requerem intervenção do usuário (NOONAN e DUBRAWSKY, 2006).

1.5.3 NAT

Primeiramente o *Network Address Translation* (NAT, tradução de endereço de rede) foi desenvolvido com o propósito de aumentar os endereços IPs destinados à rede interna e evitar o término de endereços para *hosts* de *Internet*.

A NAT converte os endereços IPs dos computadores internos de rede para um endereço externo. Isso possibilita a rede interna possuir muitos computadores e poucos ou só um endereço IP externo, também chamado de endereço IP válido.

Descobriu-se então o benefício de proteção obtido com esta tecnologia, pois ao traduzir os endereços, os computadores internos ficam inacessíveis externamente e seus endereços na rede interna são ocultados, aumentando a segurança da mesma.

O processo de tradução de endereços pode ser descrito desta forma: quando um *host* interno requisita uma conexão para um externo, o servidor NAT recebe o pacote, grava na memória o endereço IP do requisitante e destinatário, troca o endereço de origem do pacote pelo seu endereço de *Internet* válido e envia o pacote ao destino. Quando o pacote retornar será efetuado o processo inverso.

Para um melhor entendimento do funcionamento da NAT segue um exemplo, Figura 1.2: ambos os *hosts* internos 192.168.1.2 e 192.168.1.1 requisitam uma conexão com o *host* externo 10.100.100.44 na porta 80 na qual opera o serviço de *Hypertext Transfer Protocol* (HTTP). Assim a NAT recebe os pacotes e troca o endereço de origem pelo seu endereço externo 172.28.230.55. Desta forma o *host* que receberá a conexão acreditará que esta partiu do endereço 172.28.230.55 e não de algum computador em uma rede interna.

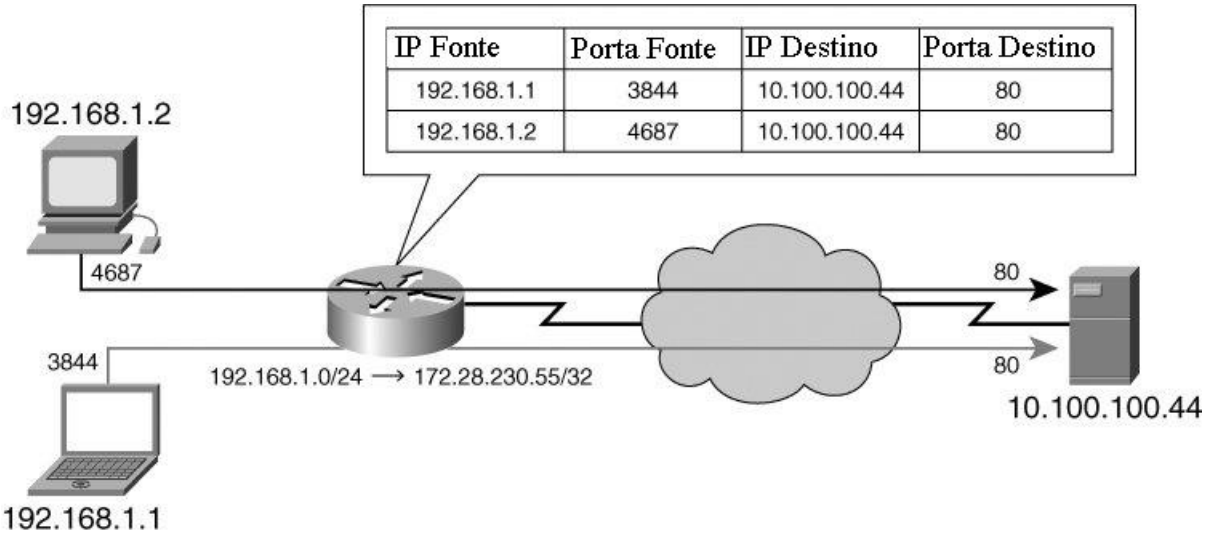

Figura 1.2 - Exemplo do funcionamento da NAT.
Fonte: NOONAN e DUBRAWSKY, 2006 (Adaptação nossa)

Quando for requisitada uma conexão para fora da rede em uma porta que já está em uso por outra conexão o servidor NAT irá modificar a porta na saída além do endereço IP. O Quadro 1.1 mostra este processo.

Note que a porta 4687 já está em uso pelo *host* 192.168.1.2, então quando o *host* 192.168.1.1 requisitar uma conexão na mesma porta o servidor NAT irá modificá-la, como pode ser notado na última linha do quadro, onde a porta foi trocada para 63440.

Devido ao crescimento do número de *hosts* utilizados em redes locais, foram disponibilizadas três classes de IP que podem ser utilizadas sem restrições atrás de um

firewall, os mesmos não são disponíveis na *Internet*. Se tentar efetuar o comando *ping* para um destes endereços o retorno será destino inalcançável (MAIWALD, 2001)

10.0.0.0 – 10.255.255.255 (máscara de 8 *bits*)

172.16.0.0 – 172.31.255.255 (máscara de 12 *bits*)

169.254.0.0 – 169.254.255.255 (máscara de 16 *bits*)

192.168.0.0 – 192.168.255.255 (máscara de 16 *bits*)

IP Fonte	Porta Fonte	IP NAT	Porta NAT	IP Destino	Porta Destino
192.168.1.1	3844	172.28.230.55	3844	10.100.100.44	80
192.168.1.2	4687	172.28.230.55	4687	10.100.100.44	80
192.168.1.1	4687	172.28.230.55	63440	10.100.100.44	80

Quadro 1.1 - Exemplo da tabela de roteamento utilizada pela NAT.
Fonte: NOONAN, 2006 (Adaptação nossa).

Alguns problemas podem surgir com o uso da NAT em determinados protocolos, os mesmos podem não funcionar corretamente, pois necessitam de um canal de comunicação separado para o retorno. Este tipo de problema pode ocorrer em serviços de *File Transfer Protocol* (FTP) que requerem uma conexão separada de retorno dos dados. O funcionamento destes em NATs não preparadas não será possível, porque a mesma não possui em sua tabela, informações referentes ao retorno por uma conexão diferente, ao receber os dados simplesmente irá descartá-los.

A NAT possui a capacidade de fazer dois tipos de conversão de endereços de rede: NAT Estática e NAT Dinâmica.

1.5.3.1 NAT Estática

As conversões de endereços são fixas e não mudam. A tabela de roteamento utilizada para o conhecimento das conexões existentes deve ser configurada manualmente. Esse tipo de conversão é utilizado em *Demilitarized Zones* (DMZs) para disponibilizar algum serviço externamente como *web*, por exemplo.

A NAT estática pode ser usada para o balanceamento de carga[2] em serviços oferecidos na *Internet*. Para tanto a mesma deve ter conhecimento da carga utilizada por cada servidor utilizando algum método proprietário que pode aumentar o custo do serviço. O método alternativo é considerar que cada cliente que se conecta ao serviço consome a mesma carga do servidor, então para cada um que se conecte, a NAT encaminha a conexão ao próximo servidor.

1.5.3.2 NAT Dinâmica

A tabela de roteamento é montada conforme os *hosts* internos requisitam conexões externas. A conversão dinâmica de endereços é mais utilizada em redes onde o endereço IP é fornecido por meio de um servidor utilizando o protocolo *Dynamic Host Configuration Protocol* (DHCP), ou seja, os endereços dos computadores variam. Toda a vez que o computador for iniciado o servidor DHCP fornecerá um IP disponível dentro da classe em que foi configurado. Quando este tipo de conversão é utilizado, a possibilidade de oferecer algum serviço na *Internet* (como *web* ou FTP por exemplo) não existe.

A NAT dinâmica pode efetuar cerca de 64.000 conexões simultâneas, mas deve ser levado em conta que cada computador pode requisitar mais de 32 conexões quando está acessando a *Internet* (MAIWALD, 2001).

Este tipo de NAT pode ser utilizado para o balanceamento de carga[3] em *links* de *Internet*, ou para disponibilidade da mesma. O servidor NAT deve possuir o conhecimento da carga que o *link* está utilizando e desta maneira conseguir efetuar o roteamento dos pacotes para o que possui carga menor. Em caso de falha de um deles, a NAT entenderá que o mesmo está com carga total e não enviará mais conexões para o *link*. Este tipo de balanceamento é transparente para o usuário a não ser que ele esteja utilizando naquele momento um serviço que requer uma sessão como, por exemplo, o FTP onde a conexão será perdida.

1.5.4 *Proxy*

[2] A NAT redireciona cada conexão nova para o servidor com a menor carga ou que possua a menor quantidade de conexões. Este processo permite que a empresa possua mais de um servidor para o mesmo serviço ampliando a quantidade de conexões simultâneas para o serviço oferecido.

[3] Este tipo de balanceamento de carga é similar ao anterior, mas neste caso ao invés de possuir mais de um servidor para um serviço a empresa possui mais de um *link* de *Internet*. O balanceamento, na medida do possível, faz com que nenhum dos *links* fique com excesso de tráfego de dados.

Os *proxies* de aplicação, *gateways* de aplicação, serviços de *proxy* ou filtros de aplicação como é chamado é a arquitetura mais inteligente. Trabalham na camada de aplicação do modelo OSI. Podem averiguar os dados detalhadamente antes de tomarem uma decisão. Conseguem não somente filtrar os pacotes, mas averiguar o conteúdo do mesmo. Podem distinguir dentro de uma comunicação HTTP, por exemplo, os pacotes de tráfego normal e os que contêm alguns tipos de vírus. Isso cria uma grande flexibilidade para os administradores de rede que podem criar regras diversas para melhorar a segurança da empresa permitindo ou não determinados conteúdos (NOONAN e DUBRAWSKY, 2006).

Figura 1.3 - Modelo OSI. Camada em que trabalha o *proxy*.

Utilizam uma filtragem de pacotes *stateful* na camada de aplicação, por esse motivo são tão eficazes. Em contrapartida deve haver um serviço para cada protocolo, ou seja, HTTP, FTP, *Simple Mail Transfer Protocol* (SMTP), etc.

Bons *firewalls* possuem *proxies* que permitem filtrar os protocolos mais utilizados como: SMTP, *Post Office Protocol* (POP), FTP, HTTP, DNS e *Remote Procedure Call* (RPC).

Os *proxies* atuam como intermediários nas comunicações entre *hosts* internos e externos, conexões diretas entre os mesmos não são efetuadas, ou seja, os *proxies* não efetuam roteamento de mensagens como os filtros de pacotes e a NAT. Para obter o comportamento de intermediário o mesmo recebe a requisição de conexão interna, a coloca em sua memória para lembrar a quem responder, e cria uma conexão com o *host* remoto em seu próprio nome. Este é totalmente transparente, ou seja, externamente o tráfego parece ser gerado por um computador somente e nenhum outro saberá que a conexão partiu de dentro de uma rede.

Muitos dos quais suportam o protocolo HTTP ainda podem armazenar em *cache* dados das páginas de *Internet*, fazendo com que a banda utilizada para fora da rede diminua.

Como o *proxy* efetua a filtragem dos pacotes em uma camada mais alta (aplicação) não pode se proteger de ataques originados em camadas inferiores como a de transporte. Para isso é necessário um filtro de pacotes que opere em camadas inferiores.

Por sua capacidade de análise do conteúdo do pacote seu desempenho é inferior a outros tipos de filtros como filtros *stateful* e sua configuração se torna mais complexa. Também o custo de manutenção pode aumentar, pois a análise da camada de aplicação requer mais configurações e manutenção.

Pode-se definir que *proxies* são utilizados para casos mais específicos enquanto o filtro de pacotes é utilizado para condições gerais.

1.5.4.1 *Proxy* a Nível de Circuito

É um *proxy* com limitações, não consegue efetuar filtro baseando-se no conteúdo do pacote.

Opera na camada de seção do modelo OSI (Figura 1.4). Este verifica as negociações dos pacotes para analisar qual é legítimo. O tráfego enviado para o *host* remoto é modificado para que o Nível de Circuitos pareça ser a origem da conexão da mesma forma que a NAT. Isso gera a mesma vantagem que é ocultar os computadores internos da rede para que não sejam acessíveis fora dela.

O *Sockets* (SOCKS) é um *proxy* a nível de circuitos, é muito utilizado quando não existe *proxy* a nível de aplicação para algum tipo de protocolo.

Este *proxy* é mais ágil, seu trabalho exige menor processamento do servidor, por não precisar abrir os pacotes um a um.

Sua desvantagem é não efetuar filtro de conteúdo, isso pode permitir a entrada de dados não autorizados contidos em pacotes autorizados. Realmente a frase anterior parece contraditória, mas isso acontece, um pacote é permitido, mas não se sabe o que contém.

Figura 1.4 - Modelo OSI. Camada em que trabalha o *proxy* a nível de circuitos.

1.5.5 *Stateful Firewall*

O *stateful firewall* consegue fazer a tradução que a NAT faz, a verificação de sessão que o nível de circuitos possui e o filtro na camada de aplicação do *proxy*. Esta, na verdade, é uma tecnologia que agrupa funcionalidades descritas anteriormente. Sua vantagem é conseguir fazer a verificação de sessão que a NAT e o *proxy* não fazem. A desvantagem é que se torna mais complexo pelo fato de agrupar funcionalidades.

Nos dias de hoje boa parte dos produtos de *firewalls* são *stateful*. Na figura abaixo são descritas as camadas em que opera, são as mesmas das funcionalidades agrupadas.

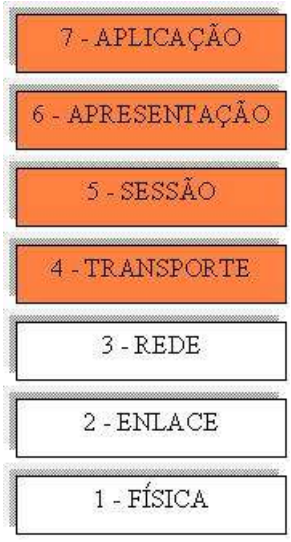

Figura 1.5 - Modelo OSI. Camadas em que trabalha *proxy stateful*.

1.5.6 *Firewalls* Transparentes

Esta tecnologia de *firewall* opera na camada dois, monitora a camada três e mais o tráfego de rede. Consegue efetuar a filtragem de pacotes *stateful* de forma transparente para o usuário final.

Sua performance é melhor se comparado com outros tipos de filtros, pelo fato de tender a ser simples, seu uso de processamento é baixo, possibilitando a filtragem de um alto volume de dados. A configuração é facilitada pelo motivo de operar na camada dois, onde não é necessário endereço IP. Desta forma o *firewall* também não pode ser atacado, como não possui endereço, fica invisível ao atacante.

A sua desvantagem é não trabalhar na camada de aplicação, desta forma fica impossibilitado de efetuar filtro por conteúdo ou identificar possíveis ameaças contidas nos pacotes.

1.5.7 VPN

A VPN tem como principal característica conectar redes remotas através da *Internet* com segurança. O custo de uma VPN é menor do que utilizar circuitos dedicados para o mesmo intento, porém a sua segurança é ligeiramente inferior (STREBE e PERKINS, 2002).

“As VPNs usam a *Internet* para encaminhar o tráfego de rede local oriundo de uma rede privada para outra, encapsulando o tráfego de rede local em pacotes IP.” (STREBE e PERKINS, 2002. p. 167).

A Figura 1.6 representa o procedimento utilizado por uma VPN para transmitir os pacotes através da rede pública (*Internet*). O *host* com endereço IP 135.207.9.12 envia um pacote para o endereço IP 135.207.8.15. O *firewall* verifica que este endereço não está nesta rede, por sua vez, criptografa o mesmo e o encapsula dentro de outro pacote IP. O novo pacote é enviado para o destino 120.6.13.5, este é o endereço do *firewall* da rede destino do pacote encapsulado. O *firewall* desencapsula o pacote, descriptografa o mesmo, verifica se sua procedência é confiável e encaminha o mesmo para o destino final, o IP 135.207.8.15.

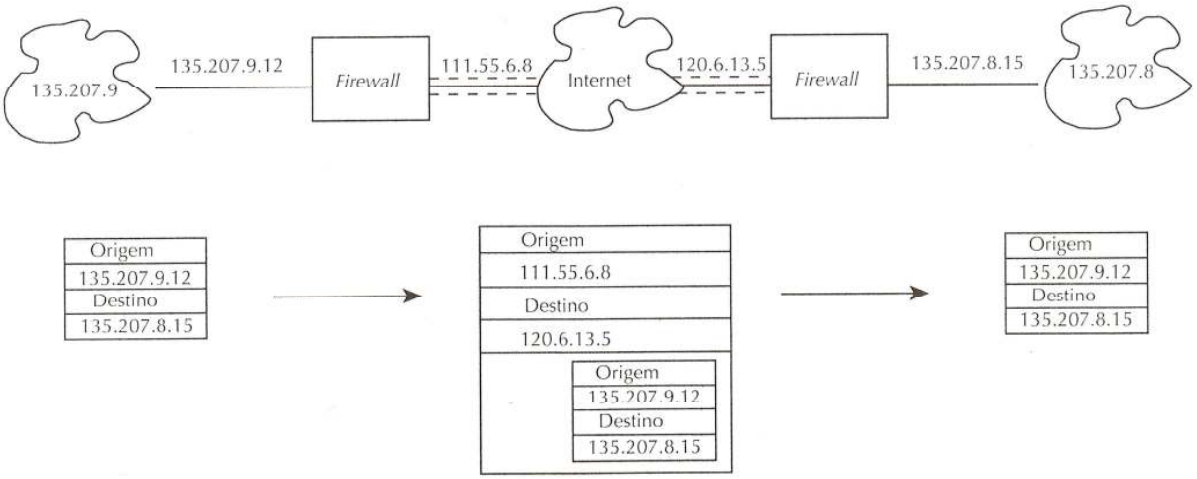

Figura 1.6 - Túnel entre duas filiais, de um *firewall* para outro.
Fonte: CHESWICK, BELLOVIN e RUBIN, 2005. p. 233.

A VPN permite que os dados sejam criptografados. Desta maneira nenhum *cracker* poderá saber o que existe no pacote, somente o *host* destino saberá como descriptografar o mesmo. Uma boa prática para o aumento da credibilidade do túnel criptográfico é criptografar também a autenticação, de forma diferente e independente da criptografia de dados. O primeiro é a criptografia dos dados (geralmente posteriores à autenticação) que circulam entre as redes. Já o segundo é utilizado para a criptografia durante a autenticação (geralmente usuário e senha), os dados posteriores não são criptografados.

Existem três tipos de VPN: as que conectam escritórios e compartilham as redes, as que conectam os clientes, como funcionários que trabalham em casa ou funcionários que viajam e as que conectam duas ou mais redes, mas não compartilham todos os recursos. Este último é muito utilizado quando fornecedores e clientes querem trocar informações de uma forma segura, mas não querem expor inteiramente suas redes uns para os outros.

Para criar uma rede privada virtual é necessário possuir um equipamento (*Hardware* ou *Software*) que possua esta função. Existem três formas de criar a VPN: por servidor onde em cada ponta existe um computador dedicado para este serviço, por *firewall* onde estes criarão a VPN (muitos deles já possuem esta capacidade) ou por *hardware* (geralmente um roteador). É indicado que se crie uma VPN com o mesmo tipo de equipamento em cada ponta, se for *software* o mesmo fornecedor deve ser utilizado. Isto devido a existência de produtos que não são compatíveis uns com os outros. O problema de incompatibilidade pode ocasionar o mau funcionamento da VPN e brechas na segurança podem surgir.

1.5.8 Relatórios e *Logs*

Outra característica que um *firewall* deve possuir é a habilidade de criar *logs* de eventos do sistema. Desta maneira o administrador de rede pode ficar ciente do que está acontecendo com o *firewall*. Como se sabe o mesmo não é infalível, por esse motivo deve possuir maneiras de relatar: falhas de segurança, violações de acesso, entre outros.

Para relatar os acontecimentos extraordinários os *firewalls* podem se munir de diversos métodos como: um aviso no console, notificação por *Simple Network Management Protocol* (SNMP), notificação por *pager*, notificação por *e-mail* entre outras. O administrador da rede deve averiguar qual delas condiz melhor com suas necessidades e fazer uso da mesma.

Ao utilizar relatórios no *firewall* o administrador perceberá cedo ou tarde de que esta ferramenta é muito importante. A utilização da mesma ajuda a resolver problemas de filtros, por exemplo. Ao analisar os *logs* o administrador pode verificar detalhes do problema. Com esta ferramenta, também é possível averiguar a consistência do *firewall*, espaço em disco, processamento, etc. Além disto, os relatórios podem avisar o administrador sobre incidentes ou invasões, assim o mesmo pode tomar as devidas medidas.

Para que as vantagens obtidas pelos *logs* sejam efetivas, é necessária monitoração. O administrador de rede deve periodicamente analisar as informações contidas nos relatórios do *firewall* (NOONAN e DUBRAWSKY, 2006).

O *firewall* pode criar relatórios de diferentes maneiras: pode utilizar recursos do próprio *firewall* ou pode utilizar uma ferramenta externa[4] paga ou gratuita. Isso dependerá dos recursos existentes do *firewall* ou do conhecimento que o administrador possui sobre uma ferramenta.

1.6 Localização do *Firewall*

Como se sabe o *firewall* deve se localizar na fronteira entre as redes, mas existem maneiras diferentes de posicioná-lo na topologia da mesma. Dependendo das necessidades em

[4] Que não acompanha o produto *firewall* original.

questão na empresa, a localização do *firewall* deverá ser estudada, pois o posicionamento incorreto poderá diminuir a segurança que oferece.

Segundo Noonan e Dubrawsky (2006) existem duas arquiteturas predominantes para a disposição do *firewall* na rede. A primeira e mais simples é a arquitetura de *firewall* único e a segunda é a arquitetura *dual-firewall*.

1.6.1 DMZ (*Demilitarized Zones*)

As zonas desmilitarizadas (DMZ) são segmentos da rede interna que não são totalmente confiáveis, por este motivo ficam isoladas da rede interna. Em geral todos os serviços que serão acessados pela *Internet* devem estar em uma DMZ e os que não serão devem estar na rede interna confiável.

A política de segurança de uma DMZ é simples: requisições de conexões oriundas da *Internet* só podem se conectar a serviços na DMZ, a mesma não pode originar conexões para a rede interna (havendo exceções) e a rede interna pode efetuar conexões para a DMZ e para a *Internet*.

Existem algumas peculiaridades para o uso da DMZ, como quando se disponibiliza acesso a um servidor *web*, o correto é não permitir o acesso deste diretamente ao banco de dados (que está na rede interna), mas sim por meio de um outro servidor de aplicação o qual receberá a requisição do servidor *web*, fará a requisição ao servidor de banco de dados e transmitirá a resposta para o servidor *web*. Esta arquitetura parece complicada, mas desta maneira protege-se o servidor do banco de dados que possui informações críticas. Como o servidor *web* é acessível externamente, está mais suscetível a ataques, se este possui comunicação direta com o banco de dados, o atacante não terá maiores problemas para alcançá-lo também (MAIWALD, 2001).

As DMZs são construídas por roteadores ou por *firewalls*, possuindo algumas arquiteturas distintas como serão melhor explicadas posteriormente.

1.6.2 Arquitetura de *Firewall* Único

Na arquitetura que utiliza somente um *firewall* existem basicamente três formas de posicioná-lo:

Sem DMZ: esta arquitetura é utilizada quando não se deseja disponibilizar nenhum serviço na *Internet*. Caso contrário, a regra fundamental do *firewall* (proteger a rede interna de conexões oriundas da *Internet*) será quebrada. Quando se oferece um serviço com esta topologia, permite-se a entrada de conexões vindas da *Internet* e desta maneira coloca-se toda a rede em perigo (NOONAN e DUBRAWSKY, 2006). A Figura 1.7 exemplifica a arquitetura de rede sem DMZ.

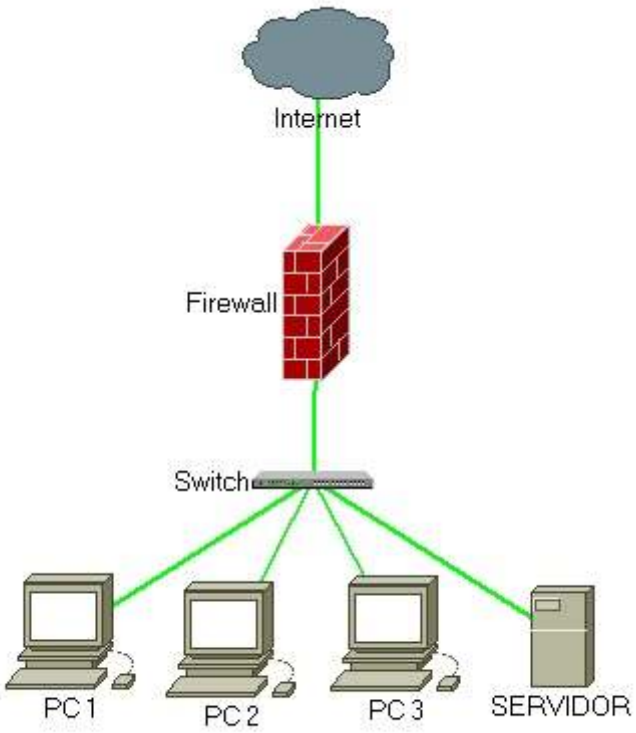

Figura 1.7 - Arquitetura de rede sem DMZ.

Com uma DMZ: desta forma o *firewall* possui três interfaces de rede, uma conectada a *Internet*, outra a rede interna protegida e a terceira com a DMZ.

As regras padrão são: a rede interna pode se conectar com *hosts* na *Internet* ou na DMZ; a DMZ pode se conectar com a *Internet* e com a rede interna (mas só se a conexão partir da rede interna); a *Intenet* pode se conectar com a DMZ e jamais pode gerar uma conexão para a rede protegida. Esta configuração é a mais utilizada, pois não é complexa e ao oferecer um serviço externamente, a rede interna estará isolada. . A Figura 1.8 exemplifica a arquitetura de *firewall* único com uma DMZ.

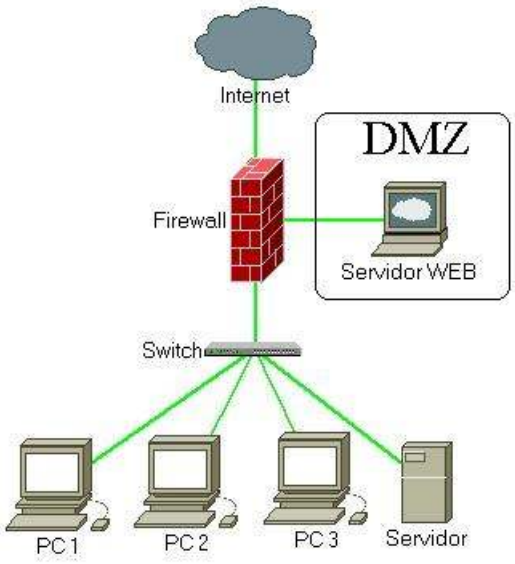

Figura 1.8 - Arquitetura de rede com DMZ.

Com duas ou mais DMZs: É utilizada para separar serviços críticos uns dos outros em mais de uma DMZ. Ela tem seu funcionamento parecido com a anterior só separa seus serviços, criando um ambiente com mais segurança. Sua vantagem é que se um *host* de uma DMZ for comprometido os separados continuarão seguros, o que não acontece em arquiteturas de uma DMZ, pois se um *cracker* comprometer um computador da DMZ é provável que ele conseguirá comprometer os outros também.

1.6.3 Arquitetura *Dual-Firewall*

Dual-Firewall significa que são utilizados dois *firewalls* para efetuar a proteção da rede. O primeiro é configurado com uma das interfaces para a *Internet* e outra para a DMZ. O segundo é configurado com uma interface para a DMZ e outra para a rede interna como pode ser observado na Figura 1.9. Desta forma a segurança da rede fica elevada, ainda mais se forem utilizados *firewalls* de fabricantes diferentes. Deste modo um *cracker*, que deseje invadir a rede, deverá possuir conhecimento sobre duas ferramentas de proteção distintas que provavelmente não estão sujeitas ao mesmo tipo de ataque.

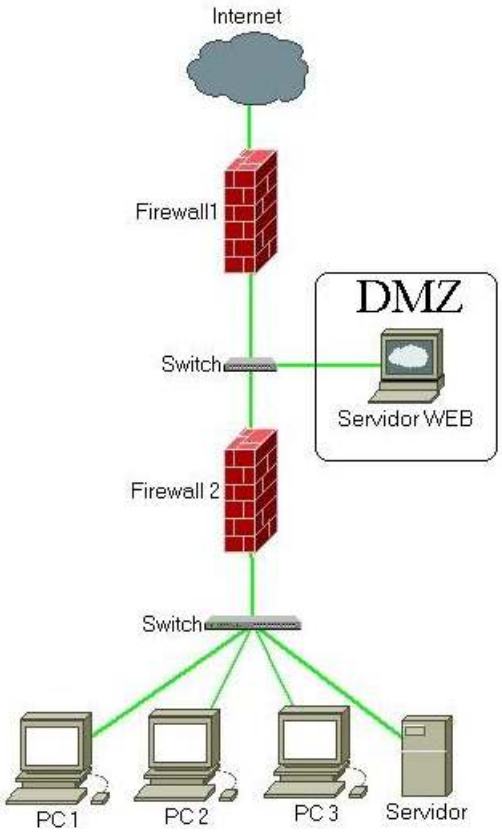

Figura 1.9 - DMZ *Dual Firewall.*

Sua desvantagem é quanto ao custo, é necessária a aquisição de dois equipamentos. Para administrar é preciso que se tenha um administrador com experiência nos dois produtos ou que se tenha um para cada. A configuração dos mesmos será diferente, aumentando a manutenção. Devido a isto este *design* não é comumente utilizado por pequenas empresas.

1.7 O Que *Firewalls* Não Podem Fazer

O *firewall* é a primeira linha de defesa que a empresa deve possuir. Este não é capaz de efetuar a proteção da rede na íntegra. O mesmo possui algumas fraquezas, as quais quando não tratadas enfraquecerão a segurança da rede.

Todo o tráfego que sai ou entra na rede deve passar pelo *firewall*, pois o mesmo não pode analisar dados aos quais não possui acesso. Portanto, o administrador da rede deve se precaver com usuários que transportam dados em qualquer meio digital (*Compact Disc* (CD), *Digital Vídeo Disc* (DVD), *Pen Drive*, disquete, etc.). O *firewall* também não pode proteger contra dados que são permitidos, muitas vezes *cracker* utilizam falhas em serviços permitidos para invadir uma rede.

Por exemplo: um funcionário recebe um *e-mail* forjado, o qual necessita da instalação de um programa no computador. Este programa é um Cavalo-de-Tróia e ao ser ativado abre uma porta local requisitando conexão com o computador de um *cracker*. Deste modo, como a comunicação é iniciada do na parte interna da rede, fica mais complicado de um *firewall* detectar esse problema dependendo de sua configuração (STREBE e PERKINS, 2002).

Quando se utiliza uma VPN, é interessante que o próprio *firewall* seja uma das terminações. Se for utilizado outro computador que esteja dentro da rede interna (atrás do *firewall*) para efetuar a VPN, o tráfego entrante somente será descriptografado pelo servidor de VPN, ou seja, passará pelo *firewall* ainda criptografado o que impedirá o mesmo de efetuar análise dos dados.

Somente o *firewall* não é suficiente, além dele devem ser adotadas outras medidas de segurança como: antivírus nas estações de trabalho, educação dos usuários contra a engenharia social, restrição de acesso a serviços não necessários e aos servidores (MOREIRA e CORDEIRO, 2002).

Em grandes empresas deve existir uma política de segurança bem desenvolvida para minimizar ao máximo os riscos existentes na *Internet*. Também deve-se frisar que as senhas nunca devem ser entregues a pessoas estranhas devendo ser trocadas regularmente e as estações de trabalho precisam estar com as atualizações de segurança em dia.

Segundo Kizza (2005) pelo motivo das fraquezas que o *firewall* possui são desenvolvidas outras ferramentas para a proteção de redes. Uma delas é o *IPSec* (conjunto de padrões utilizados para efetuar uma comunicação segura entre dois computadores. Maiores informações sobre *IPSec* podem ser encontradas em Loshin (1999)) que segundo o autor, fará com que a tecnologia de *firewall* se torne obsoleta.

Para reforçar a segurança da rede interna é necessário que se observem os aspectos descritos acima. Desta forma a rede se tornará mais segura, uma vez que o *firewall* não é capaz de protegê-la sozinho.

Com as informações contidas neste capítulo, pode-se concluir que o *firewall* cria benefícios à rede privada. O maior é a segurança que oferece utilizando as características

comentadas previamente. Nos dias presentes não é aconselhável que empresas utilizem recursos disponíveis na *Internet* sem a proteção devida de um *firewall*.

Para que empresas do segmento SOHO possam ser beneficiadas com recursos e proteção oferecidos por *firewalls*, o capítulo posterior descreve dois, sob licença GPL. Ambos possuem algumas das características descritas acima.

2 *FIREWALLS* TESTADOS

Devido ao baixo poder aquisitivo de empresas do segmento SOHO, as mesmas não podem investir altos valores para a aquisição de *firewalls* com recursos superiores. Muitas também não possuem serviço para a manutenção de seus equipamentos de informática, da mesma forma não possuem um responsável pela manutenção do *firewall* quando existente.

Por este motivo os programas testados no trabalho estão sob a licença *GNU Public License* (GPL). Sendo assim, não existe custo de aquisição (FREE, 2007). Ambos são baseados em *software* e possuem *Linux* como sistema operacional.

O licenciamento GPL tem como vantagem não somente o fato de ser gratuito, mas todos os programas sob esta licença devem possuir o código fonte. Desta maneira o usuário pode modificá-lo para que atenda as suas necessidades e se for de seu intuito, redistribuir a versão com as modificações executadas. A exigência é que a versão modificada também deve estar sob a licença GPL e possuir o código fonte para que outros possam usufruir da mesma (FREE, 2007).

Para o teste são avaliadas duas ferramentas: o *ipcop* e o *Endian firewall*, ambos são baseados no sistema operacional *Linux*. O *Endian firewall* é baseado no *ipcop* e mantido por uma empresa italiana chamada *Endian.* Embora mantido por uma empresa privada, possui uma versão gratuita voltada para o mercado de pequenas empresas.. O *ipcop*, segundo Cota (2005) e Dempster e Eaton-Lee (2006) é uma ferramenta especialmente desenvolvida para o segmento SOHO de empresas.

2.1 *Ipcop*

O *ipcop* foi desenvolvido com o intuito de operar em pequenas empresas como escritórios em casa, empresa com poucos computadores ou que possuem poucas filiais. Partiu

da necessidade das mesmas de utilizarem funcionalidades de *firewall* que somente grandes empresas utilizam, mas sem investir muito dinheiro.

O *ipcop* é um sistema especialmente desenvolvido com o propósito de ser *firewall*. Diferentemente dos *firewalls* que podem ser instalados sob um sistema operacional com múltiplos propósitos, o *ipcop* possui em seu sistema somente os componentes necessários para seu funcionamento, anulando a possibilidade de algum conflito com outro programa ou módulo (DEMPSTER e EATON-LEE, 2006).

Para a sua instalação o computador não necessita possuir um sistema operacional, pois o *ipcop* o instalará, assim o administrador não precisa ter conhecimentos sobre o sistema operacional *Linux* para fazer a instalação e configurações. O *ipcop* utiliza o *Apache* (programa para disponibilizar serviços na *web*) para disponibilizar a página de administração na *web* como a Figura 2.1 demonstra.

Figura 2.1 - Interface Gráfica do *ipcop*.
Fonte: DEMPSTER e EATON-LEE, 2006, p. 60.

O *ipcop* tem a capacidade de gerenciar quatro interfaces de rede, levando em conta que as pequenas empresas não possuem muitas redes normalmente, este número se torna adequado. O mesmo utiliza cores para distinguir as interfaces, a cor verde é sempre a rede confiável, ou seja, a rede interna. Dela são permitidas conexões para outras redes. A cor vermelha é utilizada normalmente para a rede não confiável (*Internet*, ou outra rede em uma

topologia maior). Neste caso, por padrão todas as tentativas de conexões oriundas desta interface serão negadas (*default deny*), para exceções o administrador deverá configurar o *ipcop*. A cor laranja é utilizada para DMZ e a cor azul para redes *wireless* (rede sem fio).

Como este *firewall* é desenvolvido para pequenas empresas, existe a possibilidade de utilizar *modem* ADSL *Peripheral Component Interconnect* (PCI) ou *Universal Serial Bus* (USB), de algumas marcas[5] diretamente no *firewall*. Para utilizar *modem* roteador com interface *ethernet* (rede), é necessário que o mesmo permita desabilitar a opção de NAT. Assim o endereço IP válido da *Internet* será fornecido para a interface de rede do computador que possui o *ipcop*. Caso contrário haverá problemas, pois a maioria destes *modems* faz NAT, ou seja o endereço que o *ipcop* receberá não será o IP real, mas sim outro. Sendo que o *ipcop* também efetua NAT, o processo será refeito, e a rede poderá não funcionar conforme o esperado.

O *ipcop* não é somente um *firewall* com filtro de pacotes. O mesmo permite a configuração ou instalação de outros serviços para a segurança da rede como: *Intrusion Detection System* (IDS , sistema de detecção de intrusão), VPN com uso de *IPSec*, *Caching* de DNS, *Web Proxy*, Servidor DHCP, Servidor de hora, *Traffic Shaping* e NAT (GROOM, 2007).

A desvantagem do *ipcop* é possuir algumas limitações no crescimento da topologia de rede. No momento em que a empresa se expandir para uma topologia complexa, ou seja, possuir muitos segmentos interconectados, ou possuir grande variedade de serviços, o mesmo poderá não se adequar às necessidades da empresa.

2.2 *Endian*

O desenvolvimento do *Endian firewall* é baseado no *firewall* descrito anteriormente, *ipcop*. Este produto é desenvolvido pela *Endian*, empresa italiana de gerenciamento e segurança de rede (LIMA, 2007).

[5] Maiores informações sobre marcas de *modems* ADSL compatíveis com o *ipcop* podem ser encontradas em: http://ipcop.org/modules.php?op=modload&name=phpWiki &file=index&pagename=IPCopHCLv01

Da mesma forma que o *ipcop*, o *Endian firewall* tem o propósito de ser somente *firewall*. Assim, não é possível instalar outros programas que não sejam destinados à proteção de rede como FTP ou servidor de *e-mails* em um mesmo computador.

A *Endian* mantém versões gratuitas e comerciais do *firewall*. A versão comercial (conhecida como *enterprise*) possui uma equipe de desenvolvedores, enquanto a versão gratuita (conhecida como EFW *community*) é desenvolvida pela comunidade que a *Endian* mantém. Para o comércio, a empresa disponibiliza tanto a versão *firewall* de *software* quanto a versão *firewall* de *hardware*, com opções de contrato para suporte técnico (ENDIAN, 2007).

No site oficial da empresa existe um comparativo[6] entre a versão comercial e a versão mantida pela comunidade, deixando claras as diferenças existentes entre os produtos. Do mesmo podem ser destacados itens como o foco: a versão *enterpise* é direcionada às empresas com necessidade de maior estabilidade e disponibilidade como governo e grandes empresas, enquanto a versão EFW *community* é direcionada a desenvolvedores, usuários domésticos ou ambientes não críticos como pequenas empresas. Outro item destacável é a garantia, que possui um período de dois anos para a versão comercial, enquanto a gratuita não a possui (ENDIAN, 2007).

Segundo a comunidade mantida pela *Endian* [...] "o *software* foi projetado com a usabilidade em mente, assim este é de simples instalação, uso e gerenciamento sem perder sua flexibilidade"[7] (tradução nossa). Por esse motivo o mesmo possui interface gráfica de gerenciamento como pode ser observado na Figura 2.2.

Pelo motivo de ter se baseado no *ipcop*, o *Endian firewall* possui algumas características semelhantes. Uma delas é a utilização de cores para diferenciar as interfaces de rede. Este utiliza as mesmas cores e objetivos para cada: cor vermelha é a interface conectada à *Internet*, verde é a rede confiável (rede interna), azul *wireless* e laranja DMZ.

[6] O comparativo com informações mais detalhadas pode ser encontrado no endereço eletrônico http://www.endian.com/en/community/about/compare/
[7] Frase retirada do site da *Endian*. http://www.endian.com/en/community/about/

O trabalho possui caráter universitário e tem o intuito de ajudar o segmento SOHO de empresas a aumentar a segurança em suas redes. Por este motivo a versão utilizada do *Endian firewall* será EFW *community*, ou seja, a gratuita.

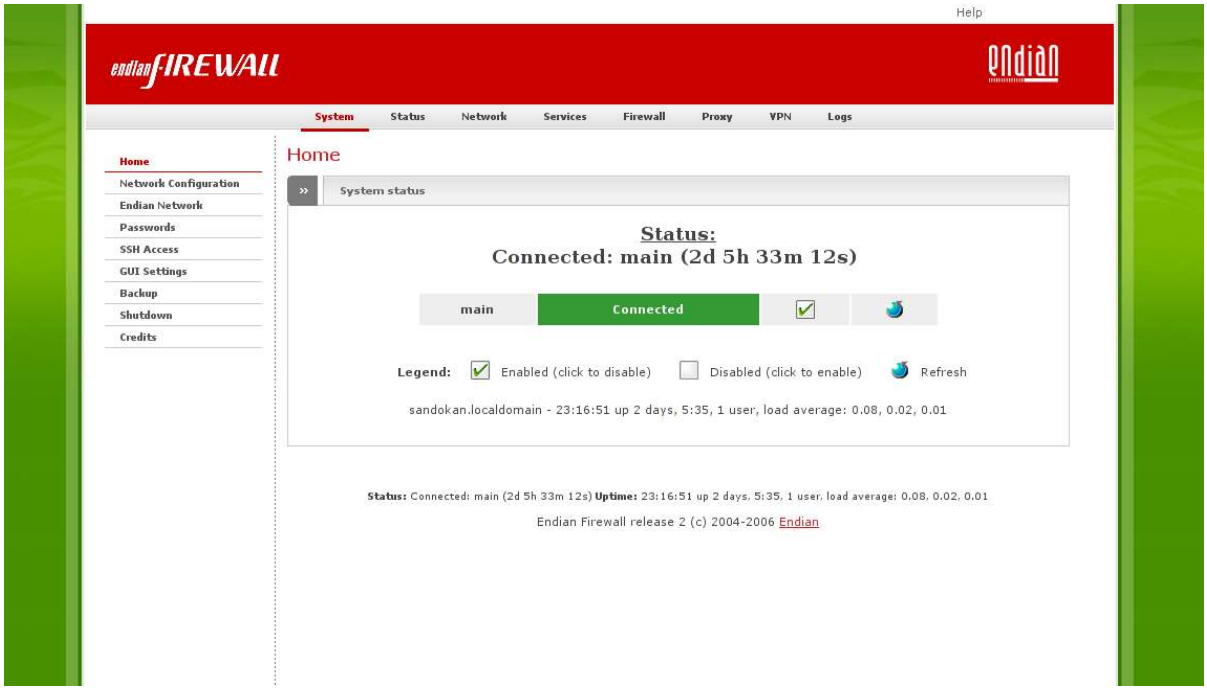

Figura 2.2 - Endian Firewall, Página inicial.
Fonte: GAGLIARDO, 2006.

A desvantagem do *Endian firewall* versão EFW *community* é possuir algumas limitações principalmente referentes a suporte e garantia, como comentado anteriormente. O exemplo que pode ser tomado, é referente ao gerenciamento. Na versão comercial existe a assistência, monitoração e atualização por parte da *Endian*[8], enquanto na versão gratuita estes devem ser efetuados pela própria empresa que o utiliza.

Entre os dois *firewalls* apresentados podem-se fazer várias comparações, dentre elas duas destacam-se, a interação com o usuário durante a instalação e a utilização da língua portuguesa do Brasil.

Interação: Este é modo como os dois *firewalls* interagem com o usuário ao aplicar as configurações iniciais (endereço IP das interfaces e senha para administração remota e para o console, por exemplo). O *ipcop* efetua a interação com o usuário durante a instalação do mesmo, ou seja, as primeiras configurações devem ser efetuadas no próprio console. Este

[8] Durante o período de vigência para atualizações sem novo contrato. O tempo é de cinco anos segundo o site oficial.

firewall possui uma interface gráfica durante a instalação, mas não permite o uso do *mouse*, assim dificultando a configuração. O *Endian firewall* somente requisita o endereço IP da interface verde durante a instalação. Depois de completa, as primeiras configurações são efetuadas pelo navegador de *Internet* do usuário.

Língua portuguesa do Brasil: A primeira opção apresentada durante a instalação do *ipcop* é a linguagem do mesmo. Dentre elas existe a "*Brazilian*", ou seja, o idioma português do Brasil. A instalação, bem como configuração, do *Endian* são efetuadas no idioma inglês. Depois de configurado é possível trocar o idioma utilizado para português do Brasil.

Os dois *firewalls* descritos pertencem ao grupo de *firewalls* de *software*. Os mesmos possuem as tecnologias[9]: filtro de pacotes *stateful*, NAT, *proxy*, VPN e *logs*. Desta forma pode-se entender que atentem as características desejáveis para a proteção da fronteira das redes (STREBE e PERKINS, 2002).

Pelo motivo dos *firewalls* não serem mantidos pela mesma comunidade, acredita-se que podem não oferecer um nível igual de proteção à rede. Assim, o teste procura averiguar a segurança oferecida. A forma de como é efetuado, ferramentas utilizadas e resultados obtidos são descritos nos capítulos posteriores.

[9] Características extraídas das bibliografias: Dempster e Eaton-Lee (2006) e Gagliardo (2006).

3 AVALIAÇÃO DOS *FIREWALLS*

Para avaliar uma rede ou mesmo um *firewall* são necessários programas de computador que executem tarefas como: buscar por portas abertas, verificar o serviço que elas oferecem, etc. Outros conseguem apresentar vulnerabilidades que o computador alvo possui, dentre elas: vulnerabilidades que são exploradas por vírus ou cavalos-de-tróia, ataques que possíveis sobre as mesmas, etc. Para este tipo de serviço existem tanto ferramentas proprietárias quanto ferramentas gratuitas e de código aberto.

3.1 Ferramentas Avaliadas

Pelo motivo deste trabalho possuir caráter universitário e estar voltado para o segmento de pequenas empresas, descreve-se o funcionamento de quatro ferramentas gratuitas: *Nmap*, *Security Analysis Tool for Auditing Networks* (SATAN), *Nessus* e *hping*.

3.1.1 *Nmap*

Network Mapper (*Nmap*) é uma ferramenta de código aberto, possui a capacidade de efetuar buscas em todas as portas de cada *host* da rede. Nos resultados, dependendo da opção utilizada, apresenta o número da porta, se está aberta (*open*), fechada (*closed*), possui algum tipo de bloqueio (*filtered)* ou se não o possui (*unfiltered*). Também é possível averiguar qual sistema operacional é utilizado pelo computador alvo e quais são os serviços que oferece (INSECURE, 2007).

O *Nmap* foi desenvolvido com o intuito de trabalhar em redes com muitos computadores, por esse motivo funciona bem em redes com poucos *hosts* (INSECURE.ORG, 2007).

A sua utilização é efetuada por linha de comando, desta forma é necessário digitar os tipos de busca que irá efetuar e os *hosts* ou a rede que irá averiguar. O *Nmap* possui versões para suporte aos sistemas operacionais *Window*s e *Linux*.

O *Nmap* pode ser utilizado por administradores de rede para efetuar inventário da rede como versão de sistema operacional, por exemplo.

3.1.2 SATAN

Security Analysis Tool for Auditing Networks (SATAN, ferramenta para análise de segurança para auditorias de rede) é uma aplicação *Linux* utilizada para efetuar auditorias em redes de computadores da mesma plataforma, busca por serviços operando em *hosts* da rede em questão e as possíveis falhas destes serviços (SATAN, 2007)

Esta ferramenta possui três modos de operação para efetuar a auditoria da rede: leve (*light*) onde é efetuada somente uma consulta ao sistema, os *logs* gerados não carregam informações sobre vulnerabilidades; média (*medium*) é mais complexo, o SATAN irá efetuar a busca por determinados serviços dentro da topologia da rede como: *World Wide Web* (WWW) na porta 80 e FTP na porta 21; o modo profundo (*heavy*) irá averiguar todos os serviços conhecidos para cada computador da rede. Este último modo de operação colocará informações completas dentro dos arquivos de *logs*, desta forma podem ser analisados para corrigir brechas existentes na segurança.

Da mesma forma que um administrador de rede pode utilizar o SATAN para buscar por falhas na sua rede ou até mesmo para auditorá-la, um *cracker* pode submeter as consultas do SATAN contra uma rede desejada, se a mesma estiver desprotegida e com serviços em aberto o *cracker* irá saber por onde pode começar as tentativas de invasão.

3.1.3 *Nessus*

Em 1998 Renault Derasion, em resposta ao constante aumento dos preços de produtos para auditoria em redes fundou o projeto que se chamaria *Nessus*. Nesta época a última ferramenta gratuita de código aberto SATAN foi ultrapassada pelos concorrentes pagos e ficou estagnada (DERASION, 2004).

O *Nessus* possui capacidade de detectar vulnerabilidades conhecidas para tipos diferentes de serviços. A versão 3.X desta ferramenta de auditoria em segurança de rede pode ser utilizada em sistemas operacionais *Linux Fedora Core* versões 5 e 6, *FreeBSD* versões 5 e 6, *Solaris* versões 9 e 10 e *Mac* OS X versão 10.4 e *Windows* versões 2000, XP e 2003 32 *bits*.

A arquitetura do *Nessus* é constituída por quatro componentes básicos: o cliente, o servidor, os *plugins* e a base de conhecimento.

3.1.3.1 Cliente e Servidor

O *Nessus* possui uma arquitetura de cliente e servidor, desta forma quando o administrador ou auditor executa alguma tarefa de busca por vulnerabilidades, seu computador fica disponível, pois o servidor é quem executará a tarefa.

Outro benefício que este tipo de arquitetura traz é a manutenção por conexão remota do servidor, o operador não necessariamente precisa estar perto do computador que possui a função de servidor.

A conexão com o servidor pode ser protegida por *Transmission Layer Security* (TLS) ou por *Security Socket Layer* (SSL).

3.1.3.2 *Plugins*

Pode se dizer que os *plugins* são uma das partes mais importantes no *Nessus*, pois são contêm as características das vulnerabilidades de cada serviço.

Para que a busca por vulnerabilidades na rede não se resuma aos *plugins* existentes, o usuário pode criar novos, utilizando a linguagem *Nessus Attack Scripting Language* (NASL). Esta opção cria uma grande gama de flexibilidades para que novos tipos de vulnerabilidade ou casos raros possam ser acrescentados.

O *Nessus* organiza os *plugins* de forma que sejam agrupados por categoria como apresentado na figura 3.1. Na parte esquerda da tela são apresentadas as categorias de *plugins* chamadas de *Families*, no caso da Figura 3.1 a categoria selecionada é a *Denial of Service*. A parte direita lista os *plugins* pertencentes à categoria selecionada na esquerda, neste caso os que pertencem a *Familie Denial of Service*.

Cada *plugin* possui um identificador (ID) numérico que não se repete, desta forma é possível identificá-los apenas pelo número.

A base do *Nessus* foi atualizada no dia 14 de agosto de 2007, um dia antes do início dos testes. Neste momento o *Nessus* possuia em sua base 15.314 *plugins*.

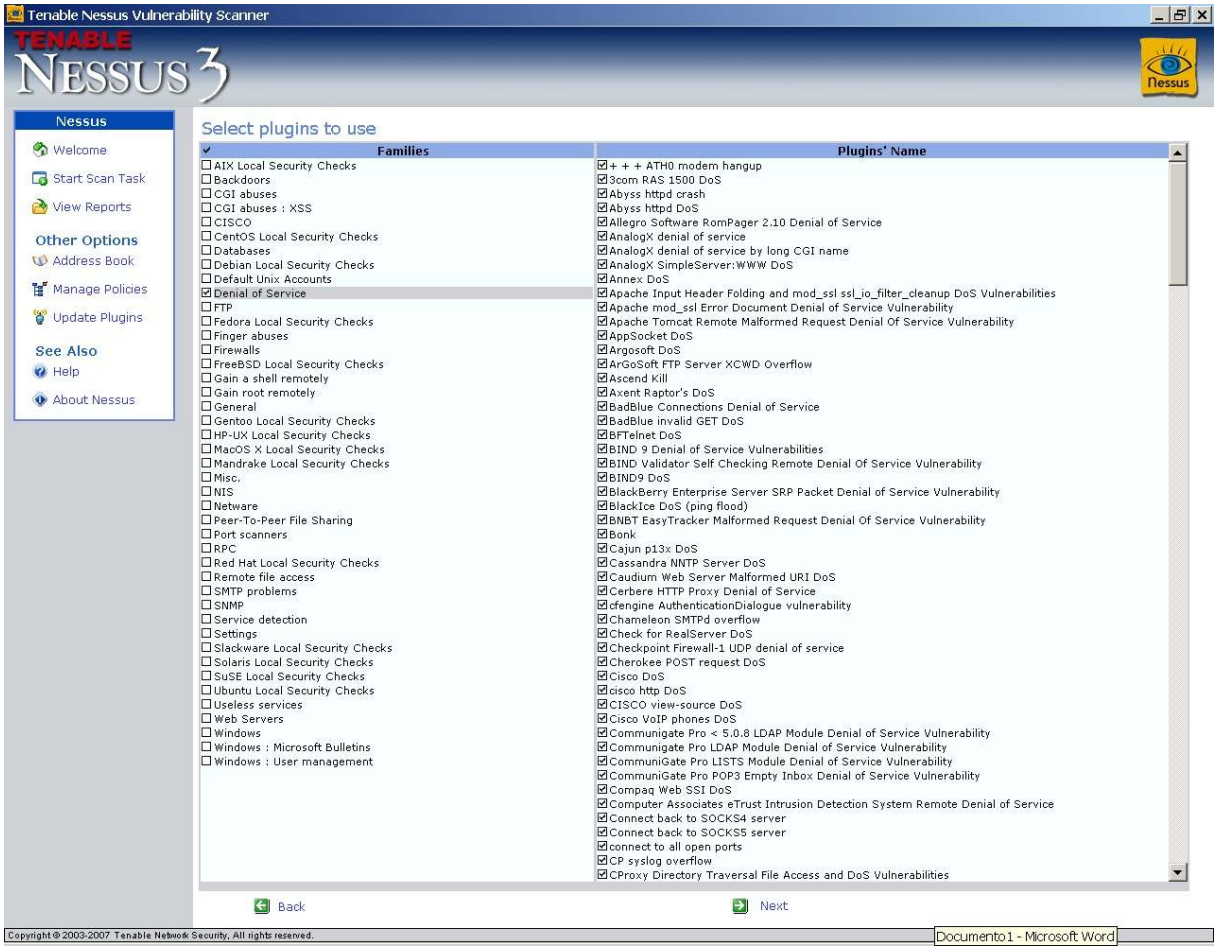

Figura 3.1 - *Plugins* do *Nessus*.

3.1.3.3 Base de Conhecimento

A base de conhecimento é utilizada para pesquisa pelos *plugins*. Os mesmos podem necessitar da resposta de outro *plugin* que já foi executado, este então deverá pesquisar na base para buscar a resposta que necessita. Desta forma o *Nessus* diminui significativamente a quantidade de vezes que cada *host* da rede deve ser acessado, a quantidade de dados que irá trafegar na rede e o tempo para o término da procura por vulnerabilidades.

Derasion (2004) sugere que os *plugins* utilizem a base de conhecimento o máximo possível, assim haverá um crescimento para futuros *plugins*, de maneira que poderão localizar

informações referentes aos *hosts* que serão analisados sem haver a necessidade de cruzar a rede em busca da mesma.

3.1.3.4 NASL

Nessus Attack Scripting Language (NASL) foi desenvolvida para que os usuários do *Nessus* possam criar ou modificar os *plugins* existentes. Ela simplifica a manutenção dos mesmos e também dos erros que possam ser encontrados.

Deraison (2004) desenvolveu esta linguagem de *script* devido ao tempo e a dificuldade que existia ao se desenvolver ou manter as vulnerabilidades acrescentadas na linguagem C. Em 1998 desenvolveu esta linguagem para o *Nessus* e em 2001 Michel Arboi contribuiu para que a linguagem fosse melhorada e para consertar alguns problemas existentes.

A NASL é dividida em duas seções, sendo a primeira chamada *script description*, esta parte contém o nome para o *plugin*, as dependências dele e outros atributos que podem ser utilizados para controlar o funcionamento do mesmo como: quando será executado e como será executado. A segunda seção é o corpo do *plugin* chamado *script body*, o mesmo contém todas as informações necessárias para a verificação da vulnerabilidade.

3.1.3.5 Opções Para Busca por Vulnerabilidades

Além da escolha dos *plugins* que o *Nessus* irá utilizar na busca por vulnerabilidades, podem ser escolhidas outras opções que tornarão a busca mais completa, ou não. Esta escolha tem um impacto direto na quantidade de tráfego que este procedimento irá gerar na rede e o tempo que decorrerá até o término da tarefa.

Dentre as opções que o *Nessus* possui, existem:

- Teste seguro (*safe checks*): esta opção fará com que o *Nessus* não execute procedimentos que podem acarretar a perda de estabilidade dos serviços a serem analisados. A mesma torna a busca menos completa, mas ao mesmo tempo torna-a mais confiável, sendo que os serviços não serão afetados.

- Performance: é possível escolher o número de *hosts* que serão verificados ao mesmo (*max number of hosts*) tempo e o número de testes (*max number of security checks*) aplicados ao mesmo tempo. Deve se levar em conta que configuração de *hardware* está sendo

utilizada para o servidor, pois se for selecionada, por exemplo, a execução em 10 computadores e 30 tipos de testes significa que o servidor irá executar 300 processos simultâneos. A performance também pode ser otimizada marcando a opção *optimize test* (otimizar o teste), deste modo o *Nessus* não irá executar todos os testes para um mesmo tipo de serviço dependendo da resposta dos primeiros testes. Por exemplo: se o *Nessus* averiguar o serviço FTP em um servidor, mas o serviço não estiver ativo, todos os testes que pertencem ao serviço FTP não serão executados. Esta opção irá decrescer o tempo de trabalho do *Nessus* significativamente. O porém desta opção é: se a busca por vulnerabilidades em um serviço retornar um valor não verdadeiro, as suas sucessoras poderão não ser executadas, decrescendo a confiabilidade no resultado da busca.

- *Report verbosity* (verbosidade do relatório) indica a quantidade de dados que o *Nessus* irá gerar para o relatório.

- *Report paranoia* (paranóia do relatório) modifica a sensibilidade de alguns *plugins*, para colocar no relatório possíveis vulnerabilidades.

- *Port range to scan* (a range de portas): esta opção determina quais portas serão verificadas dentro das 65535 existentes.

- *Consider unscanned ports as closed* (portas que não serão verificadas): esta opção fará com que a busca por vulnerabilidades não verifique as portas marcadas como fechadas. Esta também pode fazer com que algumas brechas existentes não sejam encontradas pelo *Nessus*.

- *Auto enable dependencies* (habilitar dependências automáticas). Alguns *plugins* necessitam do retorno de outros, esta opção fará com que o *Nessus* execute os *plugins* necessários mesmo que não sejam escolhidos para o teste.

- *Silent dependencies* (Dependência silenciosa) fará com que não conste nos relatórios o retorno que os *plugins* necessários executados pela opção anterior obtiveram.

- *Thorough scan* (teste completo) fará com que o *Nessus* execute alguns testes a mais para determinados *plugins*, esta opção pode acarretar no desempenho e tempo para o término do teste.

A Figura 3.2 demonstra a tela de configuração das opções de busca por vulnerabilidades.

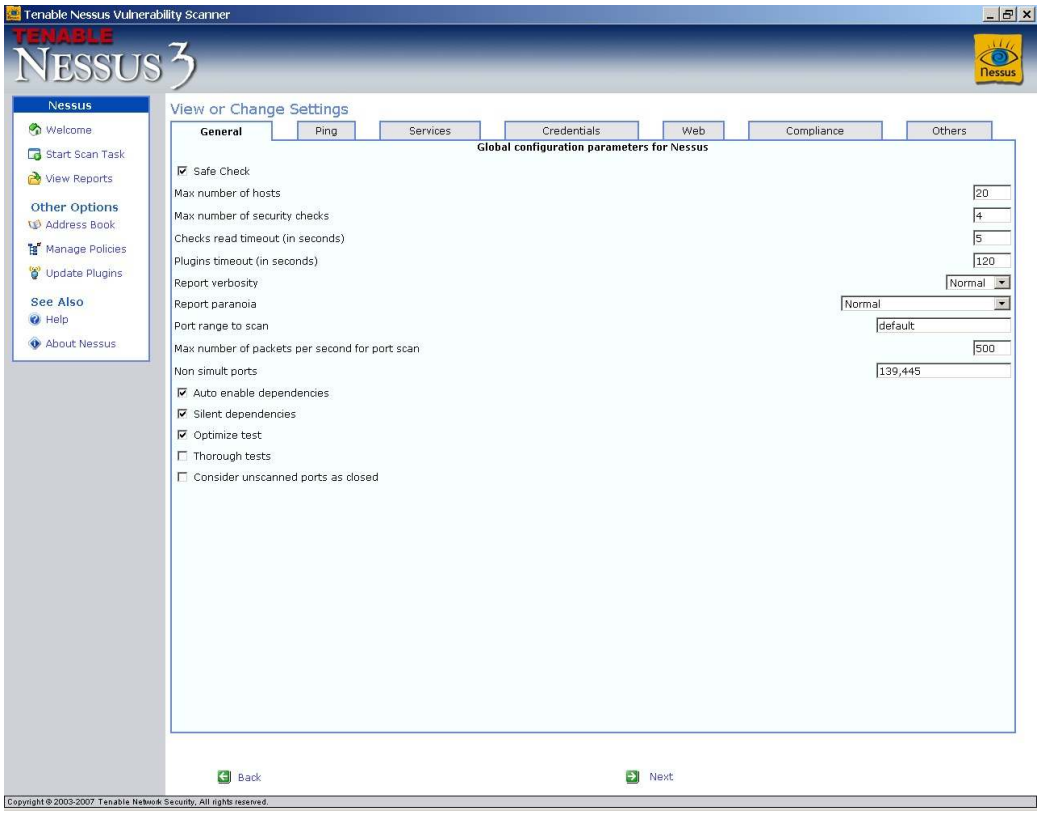

Figura 3.2 - Opções para busca por vulnerabilidade do *Nessus*.

3.1.3.6 *Nmap* no *Nessus*

O *Nessus* incorporou as funcionalidades do *Nmap*. Primeiramente era utilizado como *scanner* padrão por buscas a vulnerabilidades. Depois o *scanner* padrão passou a ser o *synscan.nasl*.

A partir da versão 3.X do *Nessus* o *scanner* do *Nmap* foi removido, isto devido aos usuários com pouco conhecimento sobre as ferramentas, efetuarem um mau uso do mesmo. Desta forma interpretavam erroneamente os resultados obtidos e efetuavam julgamento precipitado sobre o *Nessus* e o *Nmap*.

Outro motivo que levou a retirada do *Nmap* foi devido a esta ferramenta ser externa. O *Nessus* trabalha com *plugins*, assim possibilita um baixo consumo de memória para efetuar seus testes. O mesmo não ocorre quando é necessária a utilização de uma ferramenta externa. Desta forma a velocidade da busca por vulnerabilidades era prejudicada.

Se o usuário fizer questão de utilizar o *Nmap*, o mesmo pode ser adquirido no endereço eletrônico www.nessus.org e incorporado à versão 3.x do *Nessus*.

Para obter-se uma melhor compreensão sobre as vulnerabilidades averiguadas pelo *Nessus* é possível destacar alguns dos *plugins* que serão convenientes nos testes a serem realizados:

Categoria *Denial of Service:*

- O *plugin Proxy accepts CONNECT requests to itself* com identificador número 17154 analisa o *proxy* testado para averiguar se este permite ao usuário interno fazer requisições repetidas ao próprio *proxy*. Este tipo de vulnerabilidade permitiria que qualquer computador na rede interna provocasse a saturação de memória e processamento do servidor.

- O *plugin Ping of Death* com identificador (ID) 11903 averigua se o servidor pára de responder quando é submetido ao recebimento de pacotes malformados. Desta forma um *cracker* poderia efetuar um ataque de negação de serviço deixando o servidor indisponível para o uso.

Categoria *Firewalls*:

- O *plugin Usable Remote Proxy on Any Port* com ID 10193 verifica se o *proxy* aceita conexões em qualquer número de porta. Isto permite que *crackers* utilizem portas de serviços conhecidos como a porta 25 utilizada para enviar mensagens eletrônicas via SMTP. Desta forma o mesmo poderia fazer com que o servidor que possui essa vulnerabilidade fosse utilizado para atacar outras redes.

- O *plugin Weak Initial Sequence Number* com ID 11057 identifica se o computador gera a seqüência inicial de números dos pacotes TCP de uma maneira fraca. Esta vulnerabilidade pode ser explorada por um atacante, o mesmo poderia identificar a seqüência de números e efetuar um ataque do tipo homem-no-meio (*men-in-the-middle*). Desta forma poderia capturar os pacotes que trafegam entre o remetente e o destinatário.

- O *plugin Proxy Accepts POST Requests* com ID 10194 identifica se o *proxy* aceita requisições *post*, em portas utilizadas para serviços conhecidos. Esta vulnerabilidade permite que um atacante possa conseguir acesso por *telnet* ou que usuários internos possam se

conectar a portas as quais não deveriam. Também é possível que o servidor com esta vulnerabilidade seja utilizado para atacar outras redes.

- O *plugin Open Web Proxy Server* com ID 10195 analisa se o servidor *proxy* aceita conexões HTTP originadas do lado de fora da rede. Isso pode ser utilizado para que pessoas mal intencionadas pratiquem crimes no anonimato, utilizando o servidor vulnerável para este propósito. Neste caso a empresa que possui o *proxy* vulnerável poderá ser indiciada por crimes que parecem ter sido cometidos de dentro da mesma, mas na realidade um terceiro se aproveitou da falha para cometê-lo.

Categoria *Gain Root Remotely*:

- O *plugin Header Overflow Against* HTTP *proxy* com ID 11715 verifica se o servidor *proxy* HTTP suporta pacotes inválidos de requisição com cabeçalhos muito extensos. Este tipo de vulnerabilidade pode ser explorada de forma a executar códigos remotos no servidor ou fazer com que o mesmo pare de operar continuamente.

- O *plugin Too Long Authorization* com ID 10515 analisa se o servidor *web* pára de operar ou aceita execução de código remoto quando é submetido a um texto de autorização muito longo. O atacante pode aproveitar esta vulnerabilidade para executar código malicioso no servidor afetado.

Categoria *Service Detection*:

- O *plugin Service Detection* (2*nd pass*) com ID 14772 analisa serviços comuns que podem ter sido esquecidos durante a configuração do servidor.

- O *plugin LinuxConf Detection* com ID 10135 verifica se o serviço *LinuxConf* está ativo, o mesmo é utilizado para administração remota do *Linux*. A sugestão segundo o *site* do *Nessus*[10] é desabilitar este serviço caso não utilize ou fazer um filtro na porta do mesmo para prevenir acesso indevido.

[10] Informação extraída do site: http://www.nessus.org/plugins/index.php?view=single&id=10135.

Categoria SNMP:

- O *plugin Obtain Network Interfaces List via* SNMP com ID 10551 identifica se o servidor disponibiliza uma lista de interfaces de rede operantes no mesmo quando é submetido a uma requisição SNMP com o identificador 1.3.6.1.2.1.2.1.0. O atacante poderia utilizar esta vulnerabilidade para aprimorar seu conhecimento a respeito do alvo.

- O *plugin Obtain Installed Software via* SNMP com ID 19763 averigua se o servidor disponibiliza uma lista com os programas instalados no mesmo ao ser submetido por uma requisição SNMP com identificador 1.3.6.1.2.1.25.6.3.1.2. Esta vulnerabilidade é utilizada com o mesmo propósito da anterior.

Como pode ser notado, o *Nessus* é uma ferramenta para teste de vulnerabilidades em rede bastante abrangente. Esta ferramenta foi utilizada para efetuar os testes sobre os dois *firewalls* de baixo custo apresentados anteriormente. A utilização da mesma refere-se a sua grande base de dados de *plugins*, pela possibilidade de efetuar uma busca por portas abertas, da mesma forma que o *Nmap*, e pelo fato do SATAN ter sido ultrapassado e estar estagnado.

3.1.4 *HoneyBOT*

HoneyBOT é programa utilizado em *honeypots*[11], opera sobre o sistema operacional *Windows*. Foi utilizada nos testes a versão 0.1.3 deste programa. Segundo o site do fabricante, a configuração mínima desejada para a sua utilização é um computador que possua sistema operacional *Windows* 2000 e 129 MB de memória RAM (ATOMIC, 2007).

Depois de instalado e configurado o *HoneyBOT* abre cerca de 1000 portas no computador. Estas possuem o intuito de imitar serviços vulneráveis, fazendo com que o atacante tenha a impressão de estar em um servidor real. O *HoneyBOT* gera *logs* de todas as comunicações que recebeu para uma análise futura (ATOMIC, 2007).

Este programa foi utilizado, devido a sua capacidade de abrir portas imitando serviços diversos. Também nota-se a sua capacidade de guardar todas as comunicações

[11] *Honeypot*: “Recurso computacional de segurança dedicado a ser sondado, atacado ou comprometido”. Definição retirada do *site* Cert.Br. Maiores informações podem ser encontradas no endereço eletrônico http://www.cert.br/docs/whitepapers/honeypots-honeynets/.

efetuadas e detalhamento do conteúdo de pacotes recebidos. Chegou-se a este *software* por intermédio do trabalho de conclusão de curso de Caletti (2007).

A Figura 3.3 demonstra a tela inicial apresentada pelo programa ao ser aberto. Para que o este entre em execução é necessário clicar no botão *start*. O *HoneyBOT* também permite que sejam adicionados ou retirados serviços, para isso basta clicar no botão *services*.

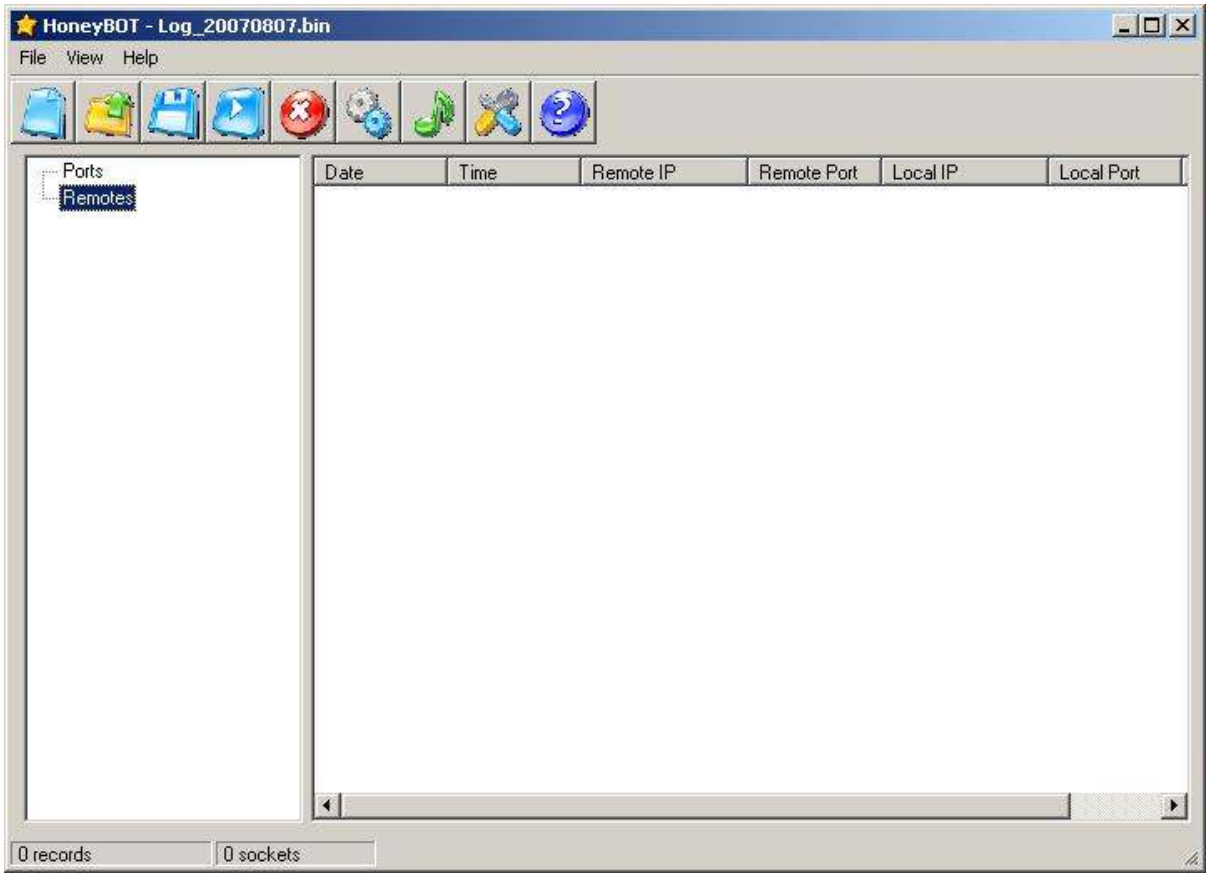

Figura 3.3 - Tela inicial do *HoneyBOT*.

3.1.5 *Hping*

Hping é um programa com utilização por linha de comando. O mesmo foi inspirado no comando *ping* do sistema operacional *Unix*. Mas, este não é capaz somente de efetuar requisições *echo*, também suporta protocolos TCP, UDP, ICMP e faz *traceroute* (HPING, 2007).

Este programa foi muito utilizado como ferramenta para segurança em redes. Dentre as funcionalidades que possui destacam-se:

- Teste de *firewall*;
- *Scanner* de portas avançado;
- Utilizado por estudantes para aprender sobre o protocolo TCP/IP.

A versão do *hping* utilizada nos testes é a 3 para sistema operacional *Linux*. Esta foi utilizada devido a versão para *Windows* possuir alguns problemas de execução no *Windows* XP *service pack* 2 (HPING, 2007).

Nos testes executados foi utilizado o método de *scan* com *flags* SYN para portas TCP. Isto devido aos outros *flags* possuírem falsos positivos ou não demonstrarem exatamente se a porta está aberta ou não. Como por exemplo, podem-se ser explicados os métodos de *scan* com *flags* ACK e FIN.

Com um *scan* utilizando-se o *flag* ACK, o *firewall* irá responder um pacote com *flag* RST para portas que estejam abertas ou fechadas e não responderá no caso das portas serem filtradas. Desta forma não se consegue uma resposta convincente, pois não se sabe exatamente a situação da porta (LYON, 2007).

Por sua vez, um *scan* efetuado com o *flag* FIN recebe muitos falsos-positivos. Isto devido a este ser o *flag* para finalizar uma comunicação TCP. Se o *firewall* responder com um pacote contendo um RST significa que a porta está fechada. Se não obtiver resposta significa que está aberta. Isto não é muito confiável, uma vez que, *firewalls* podem não responder a estes pacotes por terem sido configurados para isso, não por estarem com as portas abertas (AUDIT, 2007).

O *hping* foi utilizado nos testes aplicados, devido à variedade de opções e por possuir muitas iguais ao *nmap*. Sua vantagem quanto ao *nmap* é possuir a opção de ataque *syn-flood*.

Para que um melhor entendimento do funcionamento do *hping*, o Anexo III contém o manual do mesmo. Desta forma os comandos utilizados no trabalho não possuem explicação devido à existência do manual.

3.1.6 *CesarFTP*

CesarFTP é um programa servidor FTP, com este é possível compartilhar arquivos com serviço FTP sem a necessidade de reorganizar pastas. O *CesarFTP* foi desenvolvido com o intuito de simplificar a configuração de um servidor FTP (CESARI, 2007).

Este programa possui várias funcionalidades, dentre elas podem ser destacadas: tela de *logs* colorida; possibilidade de banir ou desconectar usuários; suporte para limitação da velocidade de transferência de arquivos; possibilidade de limitar número de conexões de um mesmo usuário, entre outras. A Figura 3.4 mostra a tela inicial do *CesarFTP*.

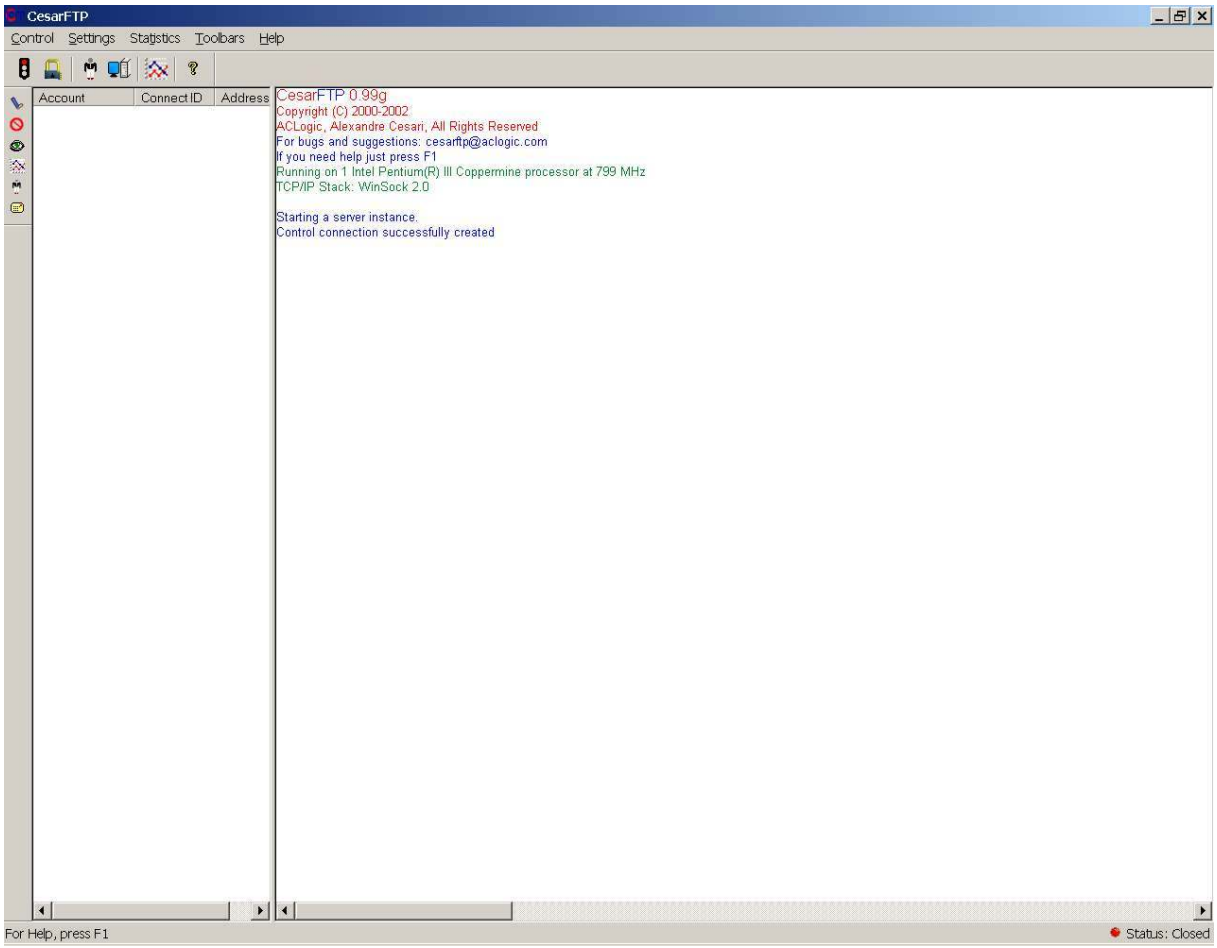

Figura 3.4 - Tela inicial do *CesarFTP*.

3.2 *Wireshark*

O *Wireshark* foi criado em maio de 2006 quando Gerald Combs, autor do *Ethereal*, foi trabalhar na CACE *Technologies*. Desta maneira, para a continuação do desenvolvimento de um analisador gratuito pacotes, a comunidade teve de trocar o nome (COMBS, 2007).

Segundo Combs (2007) o *Wireshark* suporta a maioria dos sistemas operacionais como: *Windows*, *Linux* e *Mac* OS X; pode analisar mais de cem protocolos diferentes e suporta análise de pacotes *offline*. Assim, depois de capturar alguns pacotes, o usuário pode parar a captura e analisar os recebidos. A Figura 3.5 representa a tela inicial apresentada pelo *Wireshark*.

Este programa é baseado na licença GPL, portanto pode ser utilizado sem custo. O código fonte do mesmo está disponível em seu *web site* para *download*.

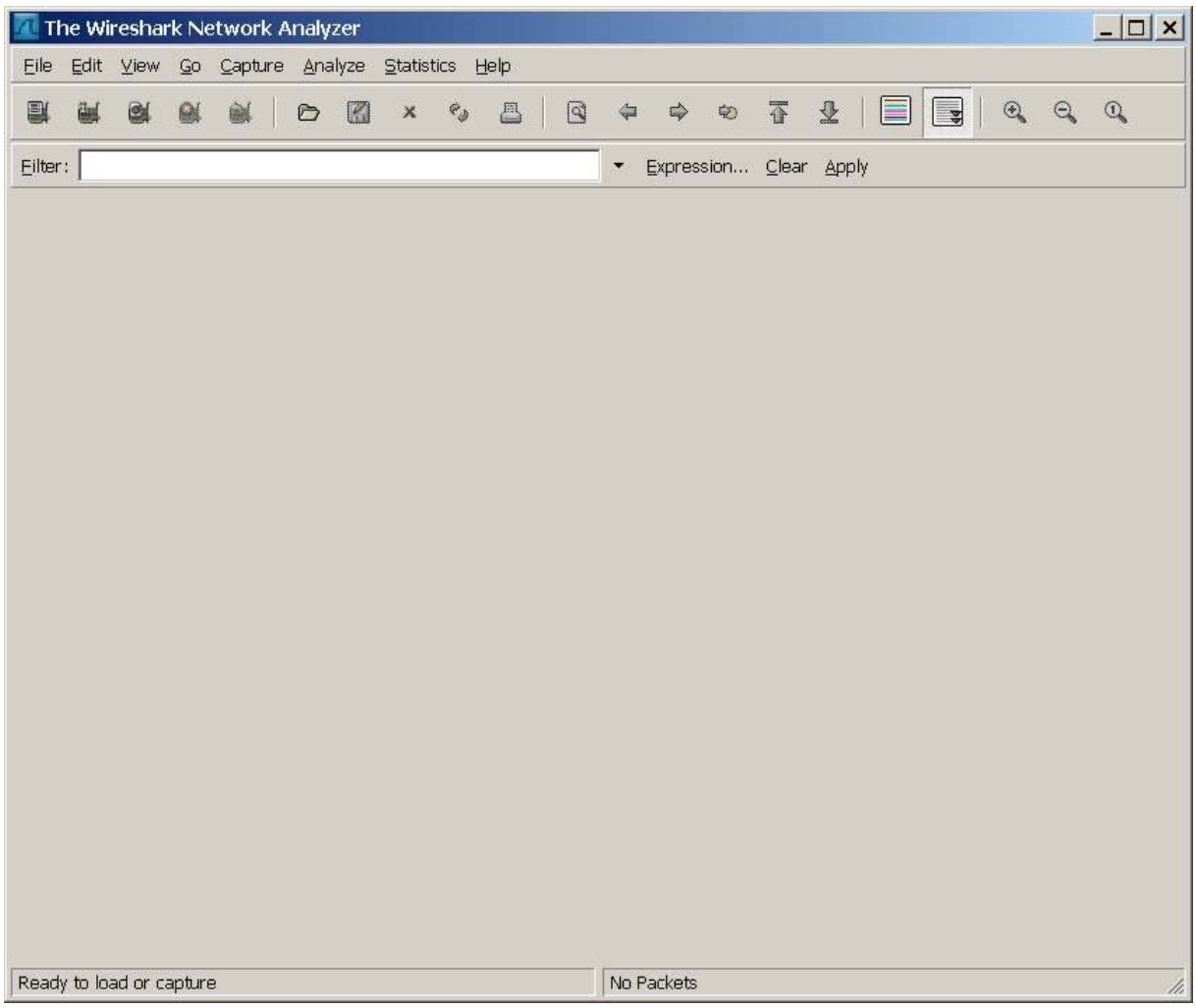

Figura 3.5 - Tela inicial do *Wireshark*.

3.3 *VMware Server*

VMware Server é um *software* utilizado para virtualização do *hardware*, ou seja, permite que dois ou mais sistemas operacionais sejam utilizados ao mesmo tempo em um só computador. Desta forma, profissionais da área de tecnologia da informação podem utilizar

um computador para oferecer diferentes serviços (VMWARE, 2007). A Figura 3.6 representa o *VMware Server*.

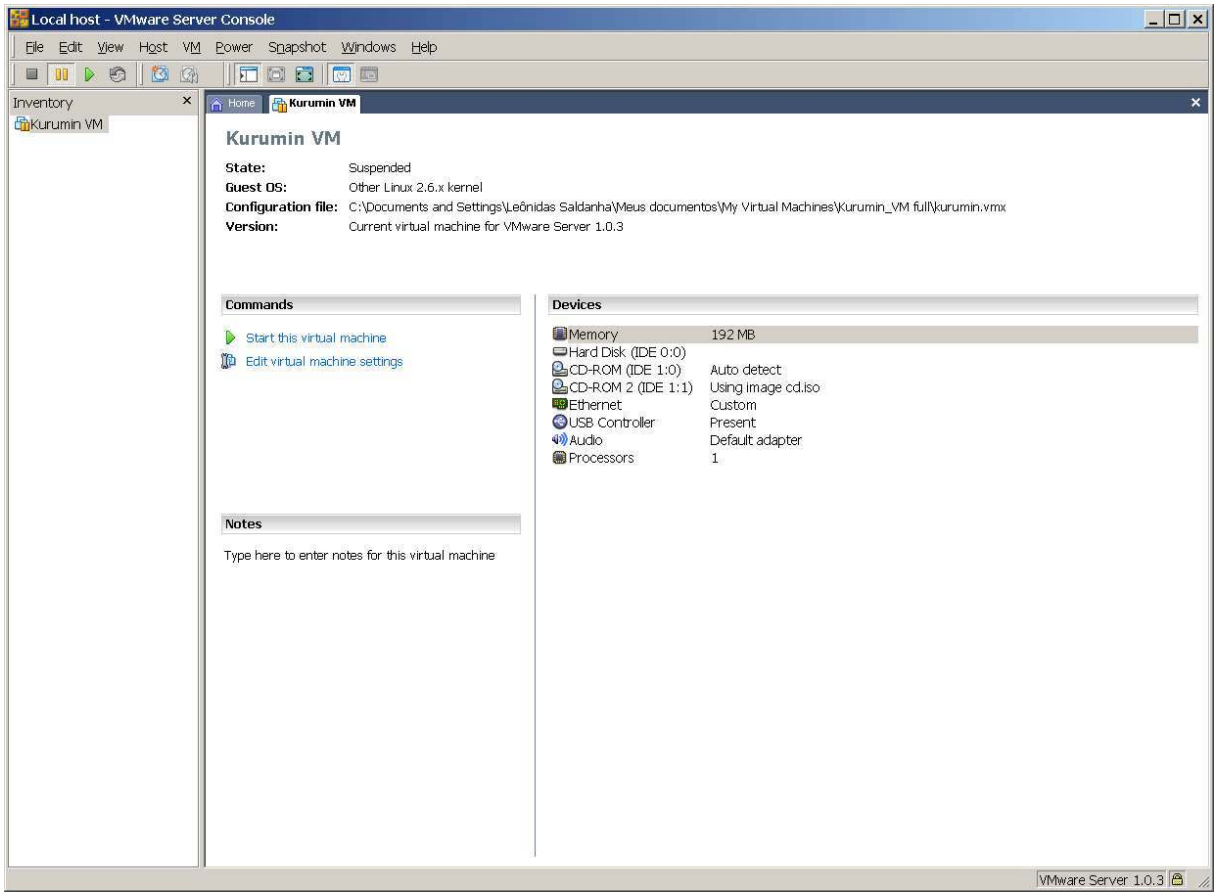

Figura 3.6 - *VMware Server*

A versão *Server* do *VMware* é gratuita e pode ser adquirida no site do fabricante. Empresas utilizam este programa para diminuir gastos com novos servidores, ou até mesmo quando uma questão de espaço é observada.

Neste trabalho o programa foi utilizado para virtualizar o *Linux Kurumin 7 Light*, sobre o sistema operacional *Windows XP*. A versão do *VMware Server* utilizada nos testes foi 1.0.3 *build*-44356. A Figura 3.7 demonstra o *VMware Server* virtualizando o *Linux* sobre o *Windows*.

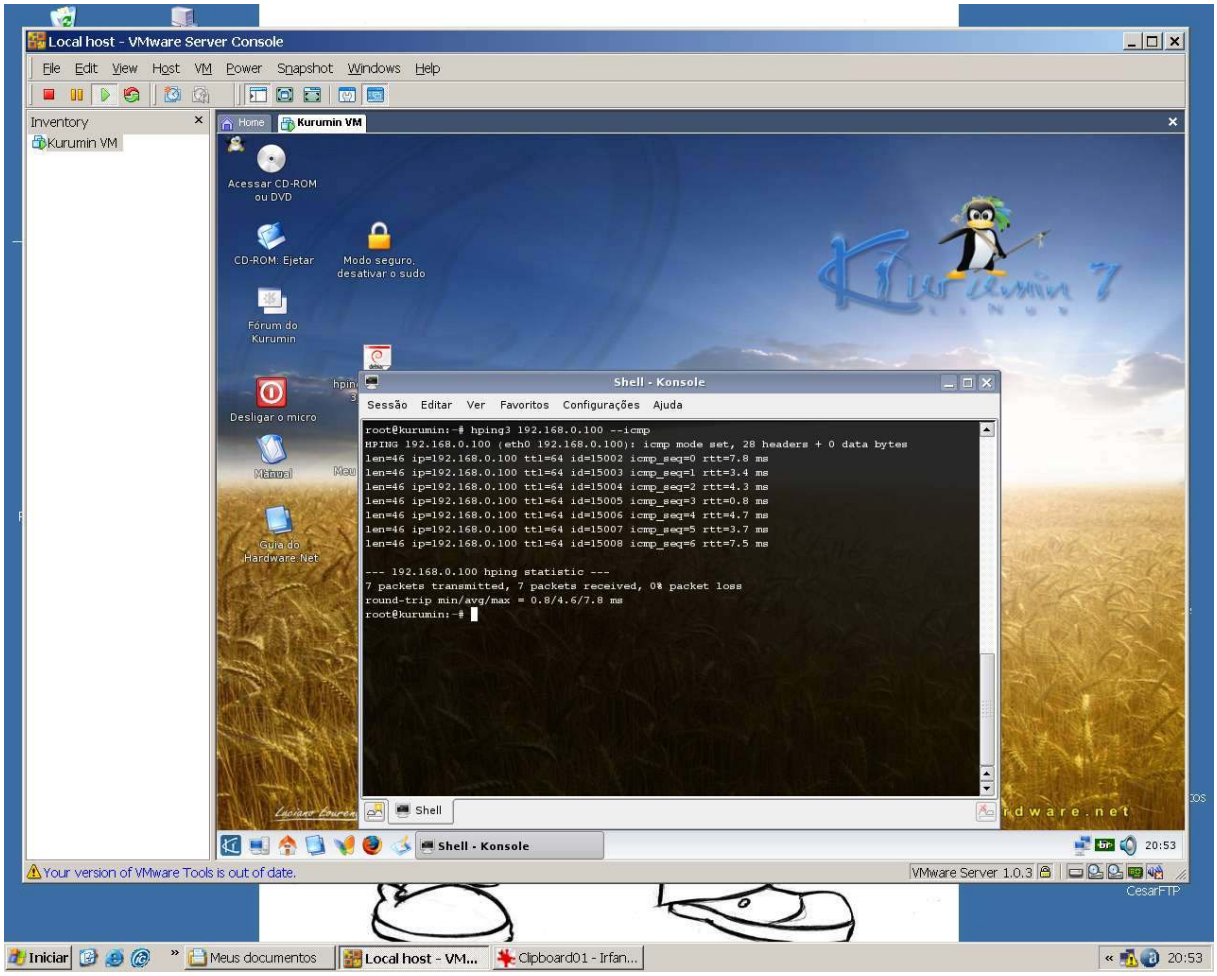

Figura 3.7 - *Linux Kurumin* 7 sobre o *WMware Server*.

Estas são as ferramentas utilizadas para efetuar os testes sobre os dois *firewalls* descritos anteriormente. As mesmas foram escolhidas, baseando-se em pesquisas efetuadas na *Internet*, trabalhos de conclusão de curso e artigos ou livros. Nos sub-capítulos posteriores são descritos o ambiente onde foram realizados os testes e os resultados obtidos.

3.4 Descrição do Ambiente Para Testes dos *Firewalls*

Os testes foram efetuados em um ambiente controlado, de forma a fornecer resultados mais confiáveis. Para o mesmo foram utilizados três computadores: uma máquina de ataque (Configuração: processador *Pentium* III 800 Mhz, 256 Mb de memória RAM, HD de 80 Gb, duas placas de rede de 10Mbps e 100Mbps, sistema operacional Windows XP *Professional Service Pack* 2 e *VMware server* com *Linux Kurumin* 7 *light*). Um *firewall* (Configuração: processador AMD *Duron* 950 Mhz, 512 Mb de memória RAM, HD de 40 Gb, duas placas de rede de 100 Mbps) e uma estação de trabalho (configuração: processador

AMD *Athlon* 1.2 Ghz, 256 Mb de memória RAM, HD de 40 Gb, placa de rede de 100 Mbps e sistema operacional *Windows* 2000 *Professional Service Pack* 4).

Na conexão entre a máquina de ataque e o *firewall* foram utilizados um *switch Encore* de oito portas com velocidade de 100Mbps e três cabos de rede par trançado. Dos cabos conectados, um partia do *firewall*, enquanto os outros partiam da máquina de ataque. A conexão entre o *firewall* e a estação de trabalho foi estabelecida por meio de um cabo par trançado com a configuração pinagem *cross over*[12].

O endereçamento de IP utilizado na rede externa, entre máquina de ataque e *firewall*, foi 192.168.0.X com máscara 255.255.255.0. Para a rede interna, entre *firewall* e a estação de trabalho, utilizou-se 10.0.0.X com máscara 255.0.0.0. A Figura 3.3 detalha o ambiente utilizado nos testes.

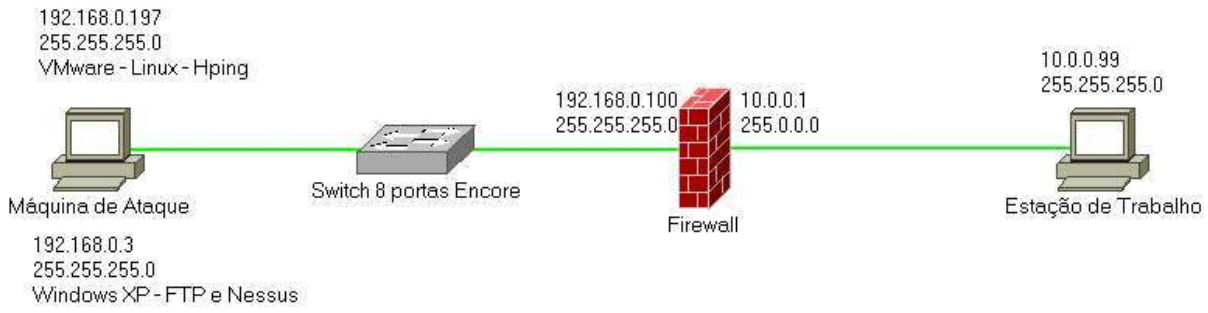

Figura 3.8 - Ambiente utilizado para os testes.

Para a realização dos testes foi necessário preparar os computadores. Em cada um foram instalados os programas necessários para que este fornecesse o serviço desejado:

Máquina de Ataque: Neste computador foi instalado o *Nessus*, para analisar as vulnerabilidades; o *CesarFTP*, para que a estação de trabalho se conecte nele, o *Wireshark* para analisar pacotes de rede e o *VMware Server* com *Linux Kurumin 7 Light*. Sobre o *Linux* foi instalada a ferramenta *hping* para efetuar uma busca por portas e um possível ataque *syn-flood* caso alguma esteja aberta.

[12] Configuração de pinagem utilizada para conectarem-se dois computadores sem a necessidade de um *switch* ou *hub*. Maiores informações podem ser encontradas no endereço eletrônico: http://br.geocities.com/conexaopcpc/par_trancado.htm.

Como este computador possui duas placas de rede, conseguiu-se separar o tráfego de cada ferramenta. O *VMware*, o qual foi utilizado para o funcionamento do *hping*, utilizou a placa de 10Mbps enquanto o *Windows* XP utilizou a de 100Mbps.

O objetivo deste computador é tentar obter informações, bem como falhas sobre o *firewall* e a rede interna. Estas poderiam ser exploradas para efetuar um ataque real ao mesmo.

Estação de Trabalho: foi instalado o *HoneyBOT*, para simular serviços vulneráveis. Também foi desinstalado o antivírus que o mesmo possuía, assim assume-se que o computador será mais vulnerável.

A estação de trabalho tem o objetivo de simular vulnerabilidades diversas dentro da rede interna, as quais podem estar presentes em empresas do segmento SOHO. Além da simulação, este computador está apto a capturar possíveis informações que o *firewall* não foi capaz de bloquear. Por isso a utilização do *honeyBOT*, o qual irá monitorar as entradas do computador em busca de tentativas de conexão.

Firewall: Este computador foi destinado a servir como hospedeiro para os *firewalls* testados. Portanto no mesmo foi instalado o *ipcop* versão 1.4.15 e posteriormente o *Endian* versão EFW *community*. Como comentado anteriormente no capítulo *Firewalls* Testados, ambos são destinados a servirem somente ao propósito de proteção da rede. Estes não podem ser utilizados como servidores multitarefa, ou seja, não podem possuir serviços além do *firewall* como: servidor de *e-mail*, servidor FTP, etc. Desta forma, durante a instalação, ambos exigem que o computador seja formatado para que nenhuma configuração prévia interfira no seu funcionamento.

3.5 Execução e Resultados dos Testes

Após o preparo do ambiente de testes, antes de iniciar o processo com os *firewalls*, a estação de trabalho foi submetida a alguns testes diretos, sem a proteção do *firewall*. Os resultados obtidos nestes, foram comparados com os dos *firewalls*.

Primeiramente o endereço IP da estação de trabalho foi trocado para 192.168.0.10 com mascara 255.255.255.0. Desta forma ficou na mesma subrede da máquina de ataque. Após, a estação de trabalho foi submetida a duas verificações do *Nessus*. Estas foram

efetuadas antes e após a ativação do *HoneyBOT*. Para que o *Nessus* obtivesse o maior número de vulnerabilidades em seu relatório, foi utilizada a opção *Thorough tests*, desabilitada a opção *Safe check*, desabilitada a opção *Optimize test* e utilizada toda a base de *plugins*.

Os resultados obtidos na primeira verificação apontaram que o computador possuía 9 portas abertas. Também apresentou 29 avisos como: a possibilidade de obter o nome do computador pela rede, obter o nome dos usuários cadastrados no computador, entre outros. O mesmo relatório ainda apresentou 3 alertas: a possibilidade de acesso aos arquivos compartilhados, verificar usuários que nunca efetuaram *logon* ou nunca tiveram a senha expirada, entre outros. Por fim o relatório identificou uma falha a qual descreve a possibilidade do *host* sofrer um ataque de DoS.

Na segunda verificação, com o *HoneyBOT* ativado, obteve-se um relatório com as seguintes informações: foram detectadas 755 portas abertas, 767 avisos, 3 alertas e 6 falhas. Das falhas pode-se destacar: *serialgateway trojan* operando na porta 1234, *terabase* operando na porta 4000. Segundo o relatório do *Nessus*, é possível fazer com que o computador pare de operar ao receber pacotes mal formados na porta 4000 do *terabase*.

O *HoneyBOT* por sua vez, ao ser ativado e sofrer uma verificação de vulnerabilidades do *Nessus*, apresentou em seu *log* 4698 linhas. Cada uma é a referência a alguma comunicação recebida. O *HoneyBOT* grava em seu relatório a data, hora, endereço do *host* remoto, porta do *host* remoto, endereço IP local e a porta local, como pode ser observado na Figura 3.9.

Além de armazenar todas as comunicações recebidas nas portas especificadas, o *HoneyBOT* também salva o conteúdo dos pacotes recebidos. Assim, podem ser analisados posteriormente. Para abrir o conteúdo de uma conexão basta clicar duas vezes sobre uma das linhas do *log*. Uma nova janela contendo as informações se abrirá, como demonstra a Figura 3.10. Assim, é possível analisar detalhes da conexão e o histórico de pacotes da mesma, além do conteúdo de cada pacote. Após a conclusão dos testes sobre o *HoneyBOT*, os *logs* do mesmo foram apagados para não serem eventualmente confundidos com os gerados nos testes dos *firewalls*.

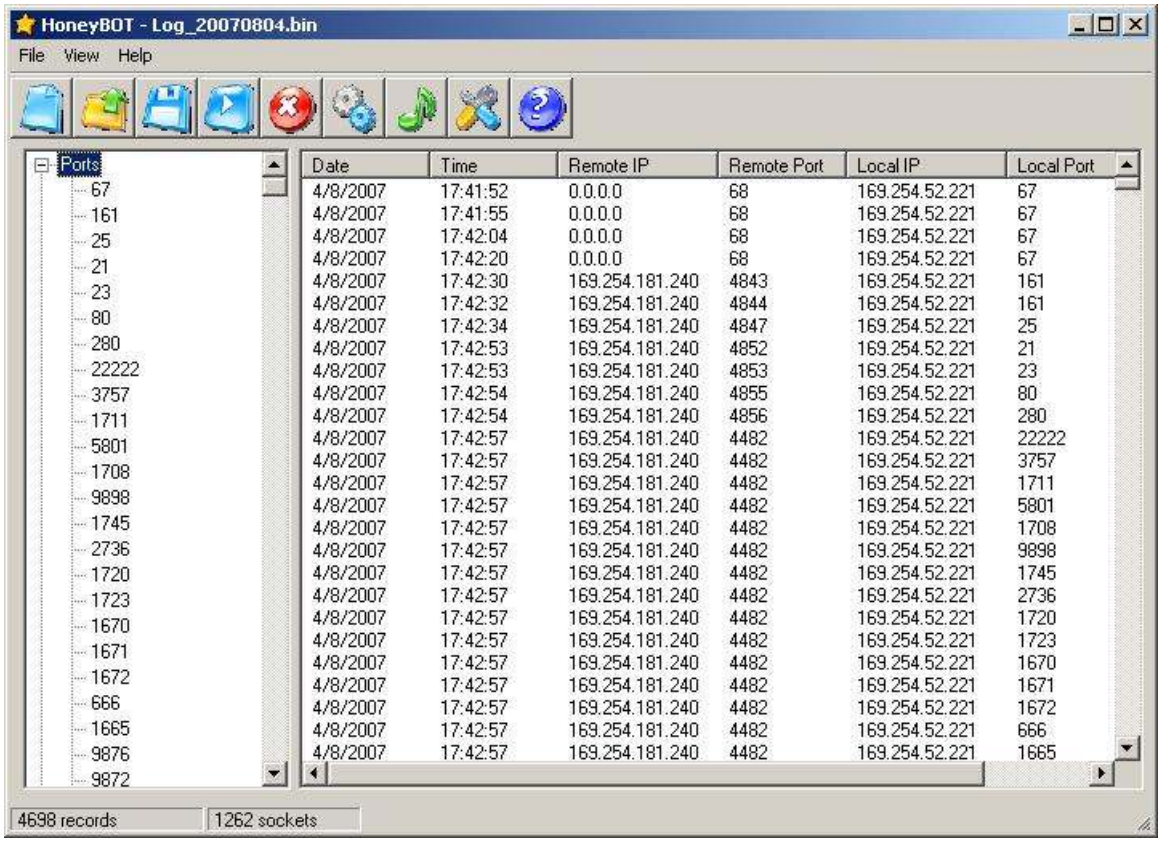

Figura 3.9 - Tela do *HoneyBOT* após a verificação do *Nessus*.

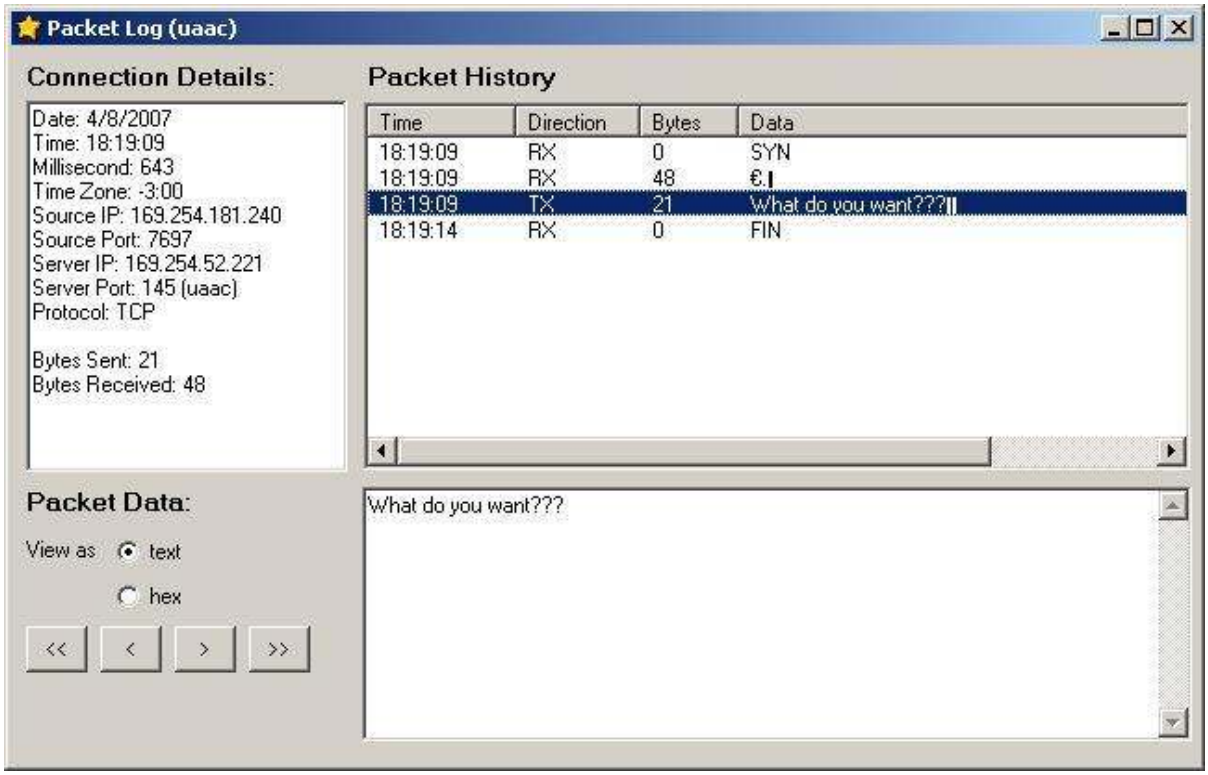

Figura 3.10 - Detalhe de um pacote recebido pelo *HoneyBOT*.

Completados os testes com o *Nessus*, a estação de trabalho recebeu mais três testes com o *hping*. O primeiro e o segundo consistiram em averiguar portar abertas, sendo que o *HoneyBOT* foi ativado somente no segundo. O terceiro teste consistiu em um ataque *syn-flood* para uma das portas abertas e com o *HoneyBOT* desativado.

O primeiro teste resultou em 4 portas abertas, o segundo, com o *HoneyBOT* ativado, obteve 880 portas abertas. Abaixo se encontra o resultado do resultado do primeiro teste, bem como o comando utilizado no *hping*. O segundo não está presente devido a sua extensão, 880 portas abertas geraram um arquivo com 16 páginas.

```
root@kurumin:~# hping3 -S --scan 0-65535 192.168.0.10

Scanning 192.168.0.10 (192.168.0.10), port 0-65535
65536 ports to scan, use -V to see all the replies
+----+-----------+---------+---+-----+-----+-----+
|port| serv name |  flags  |ttl| id  | win | len |
+----+-----------+---------+---+-----+-----+-----+
  135 loc-srv        : .S..A... 128 50432 16616    46
  139 netbios-ssn    : .S..A... 128 51456 16616    46
  445 microsoft-d    : .S..A... 128 64257 16616    46
 1025                : .S..A... 128 16132 16616    46
All replies received. Done.
Not responding ports:
```

O terceiro teste consistiu em um ataque *syn-flood* sobre a estação de trabalho. Assim, verificou-se a possibilidade da mesma ser vulnerável a este tipo de ataque. O comando utilizado foi *hping3 -S 192.168.0.10 --p 135 --flood*. Este comando não recebe resposta, somente envia pacotes com o *flag syn* para a porta programada. Para obter certeza de que o ataque foi bem sucedido, averiguou-se o gerenciador de tarefas da estação de trabalho. O mesmo ficou com a utilização do processador em 35% durante o teste. A Figura 3.11 mostra o gerenciador de tarefas da estação de trabalho, assim prova-se que este computador é vulnerável a ataques *syn-flood*. O mesmo somente não ocupou 100% de uso do processador, pois o ataque foi efetuado somente por um computador.

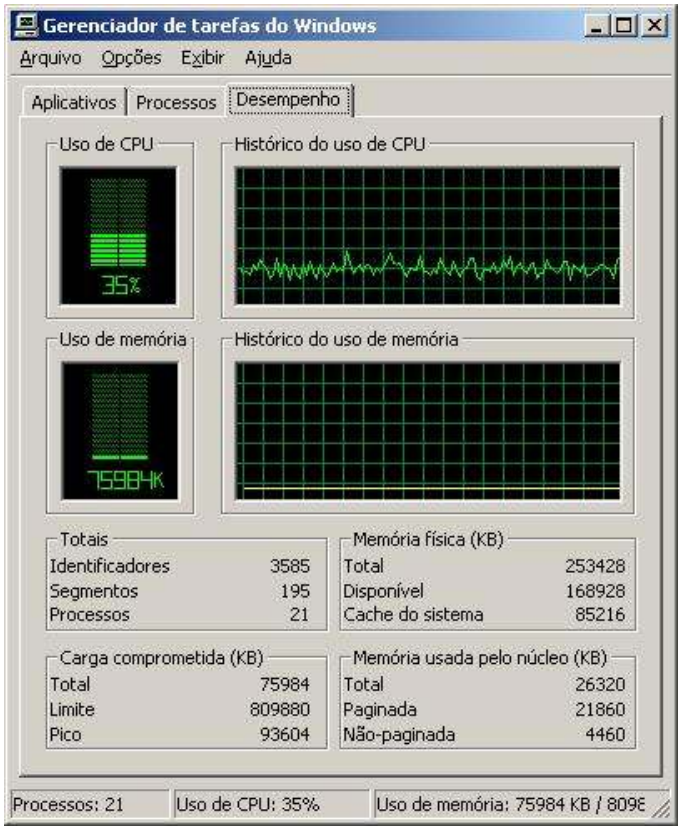

Figura 3.11 - Gerenciado de tarefas da estação de trabalho durante ataque *syn-flood*.

Verifica-se assim que o desempenho do *HoneyBOT*, *hping* e *Nessus* podem ser considerados bons. Descartando-se a hipótese das ferramentas estarem interferindo nos resultados.

Os resultados aqui obtidos também foram utilizados como meio de comparação. Estes foram comparados com os resultados dos testes sobre os *firewalls*. Desta forma, verificaram-se quais vulnerabilidades cada um conseguiu ou não proteger.

Depois da confirmação do funcionamento por parte das ferramentas acima, iniciaram-se os testes com os *firewalls*. Como comentado anteriormente o primeiro a ser testado foi o *ipcop*. Os testes realizados consistem de verificações do *Nessus*, verificações do *hping* e possíveis ataques *syn-flood* caso uma porta esteja aberta. Os resultados obtidos estão separados por dois subtítulos referentes a cada *firewall* testado.

Estes testes têm como objetivo verificar se o *firewall* em questão consegue manipular pacotes de conexões simultâneas sem deixar de averiguar suas regras. De forma que, mesmo com tráfego constante de 100Mbps da comunicação FTP o *firewall* conseguirá ou não

prevenir que suas portas sejam encontradas como abertas ou que tráfego não permitido seja direcionado para a rede interna.

3.5.1 *Ipcop*

Nos dois primeiros testes executados, o *Nessus* foi utilizado para averiguar a existência de vulnerabilidades no *firewall*. No primeiro a estação de trabalho estava desativada, assim assume-se que o resultado obtido é referente somente ao *firewall*. No segundo estação de trabalho foi ativada e o *honeyBOT* também. Como resultado, em ambos os testes, o *Nessus* não apresentou nenhuma porta aberta, falha ou vulnerabilidade existente no *ipcop*.

O terceiro teste consistiu em efetuar uma comunicação FTP durante o *scan* do *Nessus*. A estação de trabalho utilizou o *Internet Explorer* 6.0 para a cópia de arquivos e o *HoneyBOT* estava ativado. Na máquina de ataque, o *CesarFTP* foi usado como servidor FTP e o *Wireshark* foi executado para descobrir qual a porta usada pelo *firewall* durante a comunicação.

O *Wireshark* capturou os primeiros pacotes da conexão FTP. Com esta informação conseguiu-se determinar que o *firewall* utilizou a porta 1077. Então, este programa foi desativado para que o *Nessus* operasse com mais recursos do computador. A Figura 3.12 detalha a captura de pacotes do *Wireshark*. Com este teste, pôde-se verificar se o *firewall* continuava efetuando a filtragem de pacotes mesmo com tráfego de entrada e um *scan* de vulnerabilidades simultâneo vindos do mesmo computador.

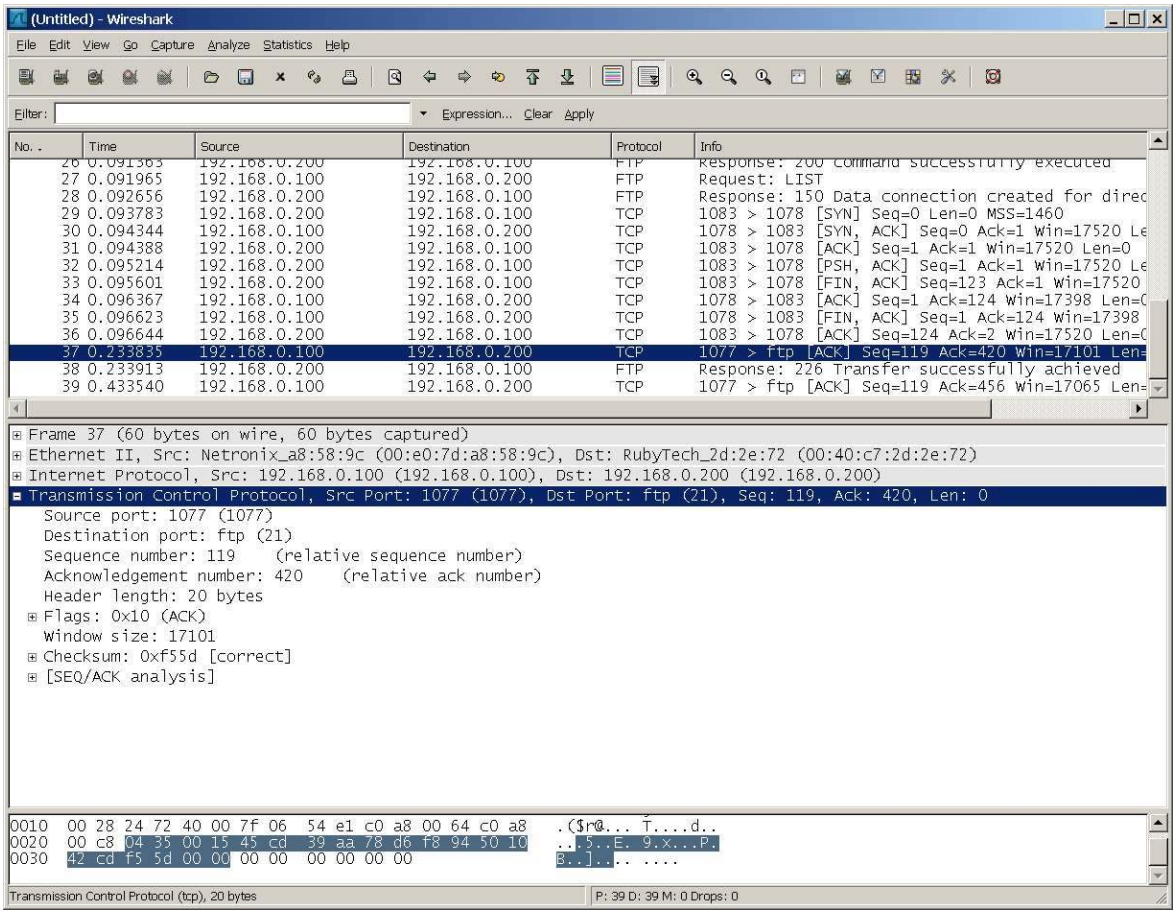

Figura 3.12 - Pacotes Capturados pelo *Wireshark*.

O relatório obtido pela busca de vulnerabilidades apresentou o mesmo resultado anterior, ou seja, nenhuma porta ou vulnerabilidade foi encontrada. Assim pode-se definir que o *ipcop* não possui nenhuma das vulnerabilidades analisadas pelo *Nessus*, ou, o *firewall* consegue ocultá-las mesmo com uma conexão direta com a máquina de ataque.

Após a conclusão dos testes com o *Nessus*, o *Linux Kurumin* foi iniciado no *VMware Server* para que o *hping* pudesse ser utilizado. Os testes efetuados com esta ferramenta consistem em: verificações de portas TCP, UDP, requisição *echo* e um possível ataque *syn-flood* em portas abertas.

O primeiro teste consistiu em uma busca por portas abertas. Esta foi efetuada sobre todas as portas existentes de 0 à 65535. O comando utilizado foi *hping3 -S --scan 0-65535 192.168.0.100*. O resultado obtido com este teste foi que nenhuma porta estava aberta, o *firewall* foi programado para nem mesmo responder quando as portas estão fechadas. Devido

a isto o *hping* apresentou como resultado que todas as portas verificadas não responderam a requisição *syn*[13].

O segundo teste efetuado também verificou por portas abertas, mas foi efetuado durante uma comunicação FTP. Como mencionado anteriormente, o sistema operacional *Windows* XP utilizou a placa com 100Mbps para o FTP enquanto o *Linux Kurumin* utilizou a de 10Mbps para efetuar o *scan*. Neste teste, o *Wireshark* capturou alguns pacotes da comunicação FTP e pôde-se averiguar que o *firewall* utilizou a porta 1113.

O comando executado no *hping*, foi o mesmo anterior, mas neste teste o *firewall* possuía tráfego e outra placa de rede (sem tráfego) na máquina de ataque efetuou o *scan*. Como resultado o *hping* apresentou que nenhuma porta respondeu. Desta forma verifica-se que o *ipcop* mantém suas regras ativas mesmo com tráfego de dados de dois *hosts* diferentes e consegue discernir conexões entre eles.

O terceiro teste consistiu em enviar pacotes *syn* para a mesma porta utilizada durante a comunicação FTP. Para isso o *Wireshark* verificou que o *ipcop* usou a porta 1103 nesta comunicação. O comando *hping* foi *hping3 -S 192.168.0.100 -p 1103*. Este foi executado durante um minuto, gerando um total de 60 pacotes com o *flag syn* para a porta usada pelo *firewall*. Do mesmo modo que os testes anteriores, o *hping* não apresentou resposta, ao fim do teste o mesmo mostrou que 100% dos pacotes foram perdidos. Com esta verificação nota-se que o *ipcop* consegue distinguir os *hosts* aos quais possui uma comunicação em aberto.

Após os testes com portas de comunicação TCP, partiu-se para as UDP. Estas mesmo não sendo orientadas a conexão também são utilizadas para comunicações que não requerem muitos pacotes, como consultas DNS e consultas em servidores de hora.

O primeiro teste efetuado foi a verificação novamente de todas as portas, mas com pacotes UDP. No *scan* de portas UDP, quando uma porta está aberta o *hping* não recebe resposta, caso contrário um pacote com *flag* RST é retornado. Os *firewalls* podem ser programados para não responder a portas fechadas, portanto somente será assumida que uma porta está aberta se o *firewall* em questão encaminhar a mesma para a rede interna. Caso contrário assume-se que a mesma está fechada.

[13] O *bit* SYN é utilizado no *three-way-handshake* para requisitar uma conexão TCP a um computador. Maiores informações podem ser obtidas em Tanenbaum 2003.

O comando utilizado para este foi *hping3 192.168.0.100 -p ++0 -c 65536 --udp --faster*. Este não retornou pacotes para o *hping*, mas também não encaminhou nenhum para a rede interna. Assim assume-se que todas as portas UDP do *ipcop* estão fechadas.

O segundo teste UDP consistiu em efetuar uma comunicação FTP com a estação de trabalho e a máquina de ataque no *Windows* XP, durante a cópia de arquivos, novamente foi feito um *scan* de portas UDP. O comando utilizado foi idêntico ao anterior e mais uma vez o *hping* não obteve resposta, tampouco a estação de trabalho recebeu pacotes UDP. Sendo assim acredita-se que o *ipcop*, mesmo com tráfego TCP, consegue prevenir os pacotes UDP de cheguem na rede interna.

Por último, o teste efetuado foi uma requisição *echo*. O comando utilizado pelo programa *hping* foi *hping3 192.168.0.100 --icmp*. Este comando efetua a mesma requisição que o comando *ping* utilizado no *Windows*. Neste caso procurou-se descobrir se o *firewall* respondia a esta requisição. O teste foi executado durante um minuto, gerando 60 pacotes ICMP.

```
root@kurumin:~# hping3 192.168.0.100 --icmp

HPING 192.168.0.100 (eth0 192.168.0.100): icmp mode set, 28 headers + 0 data bytes
len=46 ip=192.168.0.100 ttl=64 id=15002 icmp_seq=0 rtt=7.8 ms
len=46 ip=192.168.0.100 ttl=64 id=15003 icmp_seq=1 rtt=3.4 ms
len=46 ip=192.168.0.100 ttl=64 id=15004 icmp_seq=2 rtt=4.3 ms
len=46 ip=192.168.0.100 ttl=64 id=15005 icmp_seq=3 rtt=0.8 ms
len=46 ip=192.168.0.100 ttl=64 id=15006 icmp_seq=4 rtt=4.7 ms
len=46 ip=192.168.0.100 ttl=64 id=15007 icmp_seq=5 rtt=3.7 ms
len=46 ip=192.168.0.100 ttl=64 id=15008 icmp_seq=6 rtt=7.5 ms

--- 192.168.0.100 hping statistic ---
7 packets transmitted, 7 packets received, 0% packet loss
round-trip min/avg/max = 0.8/4.6/7.8 ms
```

Como resultado acima, o *firewall ipcop* responde a requisições *echo*. Portanto, está sujeito a ataques do tipo DoS. Mas, a interface de gerência via página *web* permite ao administrador configurar o *firewall* para que não responda a este tipo de requisição. A Figura 3.13 mostra a opção onde pode ser desabilitada a resposta para requisições *echo*.

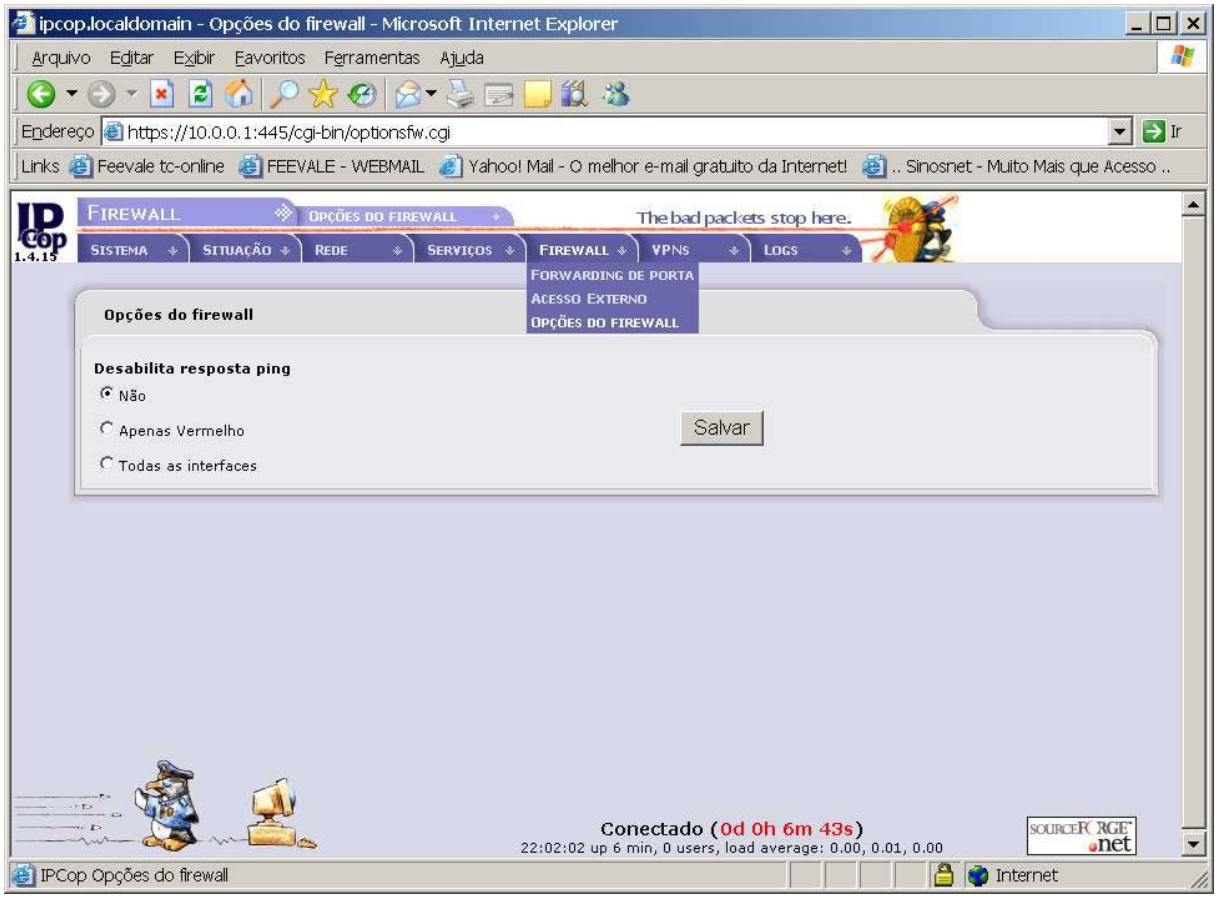

Figura 3.13 - *Ipcop*, opção para desabilitar *ping*.

3.5.2 *Endian*

Terminados os testes com o *firewall ipcop* o computador destinado a estes, foi formatado pela instalação do *EFW community*. O *Endian* foi configurado da mesma maneira que o *ipcop*, os mesmos endereços IPs foram utilizados.

Os testes se procederam de forma idêntica, na mesma ordem que foram executados sobre o *ipcop*. Sendo assim, para que os comandos não se repitam massivamente neste capítulo, aqui apresenta-se somente os resultados obtidos.

Os dois primeiros testes executados pelo *Nessus* apresentaram o *Endian* com nenhuma vulnerabilidade ou portas abertas. O terceiro, o qual possuía a comunicação FTP, o *Endian* utilizou a porta 1121 para se conectar na máquina de ataque. O resultado obtido foi o mesmo, nenhuma vulnerabilidade encontrada. Da mesma forma que o *ipcop*, pode-se definir que o *Endian* não possui nenhuma das vulnerabilidades verificadas pelo *Nessus* ou as consegue ocultar mesmo com uma conexão direta com a máquina de ataque.

Após os testes efetuados com o *Nessus* começaram com o *hping*. No primeiro teste, o qual averiguou se o *Endian* possuía alguma porta aberta, resultou sem resposta. Ou seja, mesmo as portas estando fechadas o *firewall* foi configurado para não responder.

Com a conexão FTP ativa o segundo teste foi lançado. Verificou-se novamente se o *Endian* conseguiu manter as portas fechadas, mesmo com uma comunicação à 100Mbps ativa. Como comprovou o *hping*, o EFW conseguiu. Nenhuma porta respondeu aos pacotes com *flag syn*.

Como terceiro teste, o *hping* enviou pacotes *syn* para a porta que estava sendo utilizada pela comunicação FTP. O *Wireshark* capturou alguns pacotes durante a comunicação. Com a análise dos mesmos, verificou-se que a porta utilizada foi 1048. Como resultado, obteve-se nenhuma resposta. Todos os pacotes *syn* enviados para a porta 1048 foram perdidos. Sendo assim, o *Endian* consegue distinguir os *hosts* aos quais possui comunicação em aberto.

O protocolo UDP que não é orientado a conexão foi testado após o TCP. Primeiramente, verificou se o *Endian* não possuía nenhuma porta aberta. Como resultado, todos os pacotes foram perdidos, como nenhum chegou à estação de trabalho, assume-se que o *firewall* foi programado para descartar os pacotes UDP nas portas fechadas.

O segundo teste UDP consistiu na verificação das portas durante a comunicação FTP. Este obteve o mesmo resultado, nenhuma resposta. Da mesma forma a estação de trabalho não recebeu nenhum pacote. Com estes dois presume-se que o EFW consegue distinguir conexões com protocolos diferentes, mesmo durante uma comunicação FTP à 100Mbps.

Por último, um teste de requisição *echo* foi efetuado. Neste, o *endian* respondeu à requisição como pode ser visto abaixo no resultado obtido. Sendo assim este *firewall* por padrão está vulnerável a ataques de negação de serviço (DoS). Ao contrário do *ipcop*, o *endian* não possui em suas opções nada referente ao bloqueio das requisições *echo*. Sendo assim, para desabilitar as respostas ao comando *ping* é necessário que o administrador possua conhecimentos em *linux* e desabilite a mesma diretamente no console.

```
root@kurumin:~# hping3 192.168.0.100 –icmp

HPING 192.168.0.100 (eth0 192.168.0.100): icmp mode set, 28 headers + 0 data bytes
len=46 ip=192.168.0.100 ttl=64 id=15002 icmp_seq=0 rtt=7.8 ms
len=46 ip=192.168.0.100 ttl=64 id=15003 icmp_seq=1 rtt=3.4 ms
len=46 ip=192.168.0.100 ttl=64 id=15004 icmp_seq=2 rtt=4.3 ms
len=46 ip=192.168.0.100 ttl=64 id=15005 icmp_seq=3 rtt=0.8 ms
len=46 ip=192.168.0.100 ttl=64 id=15006 icmp_seq=4 rtt=4.7 ms
len=46 ip=192.168.0.100 ttl=64 id=15007 icmp_seq=5 rtt=3.7 ms
len=46 ip=192.168.0.100 ttl=64 id=15008 icmp_seq=6 rtt=7.5 ms

--- 192.168.0.100 hping statistic ---
7 packets transmitted, 7 packets received, 0% packet loss
round-trip min/avg/max = 0.8/4.6/7.8 ms
```

Para uma melhor visualização dos testes e resultados efetuados neste trabalho, abaixo apresenta-se uma lista enumerada com o resumo dos testes executados. No quadro XX estão os resumos dos resultados obtidos juntamente com a numeração dos testes executados.

1. *Scan* com o *Nessus* no *firewall*;
2. *Scan* com o *Nessus* no *firewall* com o *HoneyBOT*;
3. *Scan* com o *Nessus* no *firewall* com conexão FTP;
4. *Scan* com *hping* nas portas TCP;
5. *Scan* com *hping* nas portas TCP com conexão FTP;
6. *Scan* com *hping* na porta utilizada pela conexão FTP;
7. *Scan* com *hping* nas portas UDP;
8. *Scan* com *hping* nas portas UDP com conexão FTP;
9. Testes com pacotes ICMP (*echo request*)

Teste	Ipcop	Endian
1	Nenhuma falha ou vulnerabilidade encontrada;	Nenhuma falha ou vulnerabilidade encontrada;
2	Nenhuma falha ou vulnerabilidade encontrada e nenhum registro no *HoneyBOT*;	Nenhuma falha ou vulnerabilidade encontrada e nenhum registro no *HoneyBOT;*
3	Nenhuma falha ou vulnerabilidade encontrada;	Nenhuma falha ou vulnerabilidade encontrada;
4	Nenhuma porta aberta foi encontrada e o *firewall* foi programado para não responder em portas fechadas;	Nenhuma porta aberta foi encontrada e o *firewall* foi programado para não responder em portas fechadas;
5	Nenhuma porta aberta foi encontrada mesmo com conexão FTP;	Nenhuma porta aberta foi encontrada mesmo com conexão FTP;
6	Porta utilizada na comunicação FTP não respondeu ao *hping*;	Porta utilizada na comunicação FTP não respondeu ao *hping;*
7	Nenhuma porta UDP aberta foi encontrada;	Nenhuma porta UDP aberta foi encontrada;
8	Nenhuma porta UDP aberta foi encontrada mesmo com a conexão FTP;	Nenhuma porta UDP aberta foi encontrada mesmo com a conexão FTP;
9	Respondeu as requisições *echo* (*ping*);	Respondeu as requisições *echo* (*ping*);

Quadro 3.1 - Resumo dos resultados obtidos nos testes.

3.6 *Endian* x *ipcop* – Análise de funcionalidades

Para a definição de qual *firewall* melhor se enquadra para pequenas empresas, além dos testes de segurança efetuados, também foi feito um comparativo das funcionalidades. O Quadro 3.1 demonstra as funcionalidades encontradas em cada um.

Pode-se notar que o EFW possui algumas a mais, destacando-se. Este quadro foi produzido visualizando-se as interfaces de administração de cada *firewall*. O *ipcop* possui uma funcionalidade que pode ser salientada, suporte a atualizações. Esta não está presente no *Endian EFW Community*, isto prejudica o acesso a correções de possíveis problemas que possam ser encontrados no mesmo. Para que o administrador da rede possa atualizar o *EFW*

ele teria de fazer um *backup* das configurações efetuadas no mesmo. Após isto, instalar uma versão mais atualizada do *firewall* e recuperar as informações do *backup*.

FUNCIONALIDADE	IPCOP	ENDIAN
Servidor *cron*[14]	SIM	SIM
Servidor DHCP	SIM	SIM
NAT	SIM	SIM
Proxy HTTP	SIM	SIM
Proxy Transparente	SIM	SIM
Proxy filtro de conteúdo	NÃO	SIM
Proxy antivírus	NÃO	SIM
Proxy POP3	NÃO	SIM
Proxy SMTP	NÃO	SIM
Proxy FTP	NÃO	SIM
Proxy SIP[15]	NÃO	SIM
Proxy DNS	NÃO	SIM
Servidor de DNS	SIM	SIM
Servidor de Horário	SIM	SIM
Controle de tráfego	SIM	SIM
Detecção de Intrusão (IDS)	SIM	SIM
Possibilidade de criação de DMZ	SIM	SIM
Suporte a VPN	SIM	SIM
Logs	SIM	SIM
Cópia de segurança	SIM	SIM
Suporte a atualizações	SIM	NÃO
Filtro de SPAM	NÃO	SIM

Quadro 3.2 – Comparação das funcionalidades dos *firewalls*.

[14] *cron* é um agendador de tarefas para sistemas operacionais *Linux* maiores informações podem ser obtidas em: http://www.guiadohardware.net/termos/cron.

[15] SIP é um protocolo usado para sinalização de telefonia sobre IP. Maiores informações podem ser encontradas em: http://www.ietf.org/rfc/rfc3261.txt.

CONCLUSÃO

Neste trabalho, foram apresentadas as definições para o *firewall*, que segundo Cheswick, Bellovin e Rubin (2005, p. 177) é "...qualquer dispositivo, *software*, arranjo ou equipamento que limita o acesso à rede". Da mesma forma os *firewalls* podem ser classificados como: *firewalls* de *software*, *firewalls* de *hardware* e *firewalls* integrados (NOONAN e DUBRAWSKY, 2006).

Apresentaram-se também as tecnologias de *firewall* existentes como: filtro de pacotes, autenticação de acesso, NAT, *proxy*, *proxy* a nível de circuitos, *stateful firewall*, *firewall* transparente, VPN, relatórios e *logs*.

Este trabalho também abordou o posicionamento do *firewall* no *layout* da rede utilizando as teorias da DMZ. Dentre os *layouts* comuns utilizados podem ser destacados os que não necessitam oferecer serviços na *Internet*, sendo assim, não necessitam de DMZ. Aquelas empresas que precisam oferecer algum tipo de serviço na rede externa devem averiguar qual é a capacidade financeira e a necessidade de segurança para seus dados, desta forma podem optar por *layouts* com um *firewall* ou com dois *firewalls*.

Também foram destacadas as proteções que os *firewalls* não são capazes de efetuar como verificar qualquer dado ao qual não tenha acesso, como arquivos em CD ou *pen drives*, por exemplo.

Foram estudadas as teorias sobre dois *firewall* sob licença GPL para uma posterior comparação. O motivo de se utilizar *firewall* sob licença GPL é pelo fato das empresas SOHO possuírem baixo poder aquisitivo.

O nome dos *firewalls* pesquisados neste trabalho são *ipcop* e *Endian EFW community*. O *Endian firewall* é baseado no *ipcop* e mantido por uma empresa italiana

chamada *Endian*. Embora mantido por uma empresa privada, o *Endian firewall* possui uma versão gratuita voltada para o mercado de pequenas empresas. O *ipcop*, segundo Cota (2005) e Dempster e Eaton-Lee (2006) é uma ferramenta especialmente desenvolvida para o segmento SOHO de empresas. Ambos os *firewalls* são gratuitos e podem ser obtidos nos *web sites* de seus respectivos fabricantes.

Ambos possuem como sistema operacional base GNU/*Linux*. Todos os dois foram projetados para servir somente ao propósito de proteção da rede, ou seja, não podem ser utilizados como servidores multitarefa. Os mesmos não suportam a instalação de componentes ou programas como servidor FTP, servidor de *e-mail* ou ser um servidor hospedeiro para páginas da *web*.

Para avaliar se os *firewalls* escolhidos não possuem vulnerabilidades foram estudadas algumas ferramentas, descritas a seguir*:* o *Nessus* que é um verificador de vulnerabilidades utilizado para buscar as mesmas em *hosts* de redes privadas. Este foi atualizado em 14 de agosto e possuía 15.314 *plugins* que correspondem às vulnerabilidades que o mesmo pode testar; *hping* o qual pode ser usado para encontrar portas abertas ou até mesmo efetuar ataques DoS; o *Wireshark* que é um analisador de pacotes, foi utilizado para verificar pacotes em uma comunicação FTP; *CesarFTP* utilizado como servidor FTP nos testes; *VMware* que possui como função virtualizar mais de um sistema operacional dentro de um mesmo *hardware*, foi utilizado para o funcionamento de uma versão do *linux*; *Linux Kurumin 7 light* no qual foi instalada a ferramenta *hping* que foi utilizado sobre o *VMware* e o *HoneyBOT*, um programa para o sistema operacional *Windows*, este simulou serviços vulneráveis atrás do *firewall*.

Como processo, cada *firewall* foi testado em um ambiente controlado. Desta forma preveniu-se que houvesse alguma influência externa. Os testes executados procuraram por falhas e portas abertas em ambas as soluções propostas. Sabe-se que estes testes não possuem como meta verificar se os *firewall* são imunes à falhas, mas sim, se os mesmos não possuem falhas que sejam mais visíveis. Não se testou os mesmos com todas as possibilidades devido a numerosidade das mesmas.

Como resultado dos testes aplicados, nenhum dos dois *firewalls* apresentou falhas que comprometessem seriamente a segurança da rede interna. O *Endian* não possui uma opção para desabilitar a resposta a requisições *echo*, mas a mesma pode ser desabilitada em

seu terminal, via linha de comando. O *ipcop* por sua vez, respondeu as requisições *echo*, mas possui uma opção para que a mesma seja desativada.

Sendo assim, para que o trabalho pudesse oferecer dentre as ferramentas a melhor, partiu-se para a comparação de funcionalidades. Notou-se que ambos possuem funcionalidades necessárias para o segmento de empresas SOHO. Mas, o *firewall Endian* destacou-se com algumas a mais.

Por comparação das funcionalidades, conclui-se que o *firewall Endian* pode ser uma melhor opção para o uso em pequenas e médias empresas. Das funcionalidades que este possui a mais, as mesmas podem ajudar a empresa a se proteger contra vírus em páginas *web* ou mesmo em mensagens eletrônicas. Também podem ser utilizadas para obter um maior controle sobre a utilização da *Internet* dentro da empresa.

Para trabalhos futuros, dão-se as seguintes sugestões:

- Estudo aprofundado das regras aplicadas a cada *firewall*;
- Comparação com outros produtos da mesma categoria que venham a ser desenvolvidos;
- Teste das funcionalidades de cada *firewall* para comparação;

REFERÊNCIAS BIBLIOGRÁFICAS

AUDIT My PC. **Port Scanning**. Disponível em: <http://www.auditmypc.com/freescan/readingroom/port_scanning.asp>. Acesso em 28 de setembro de 2007.

CESARI, Alexandre. **ACLogic Web Site**. Disponível em: <http://www.aclogic.com/>. Acesso em: 26 de agosto de 2007.

ATOMIC Software Solutions. **HoneyBOT.** Disponível em <http://www.atomicsoftware solutions.com/honeybot.php>. Acesso em 4 de agosto de 2007.

BORSCHEID, Régis Maciel. **Protótipo de Aplicação Web Para Gerenciamento de Firewall Linux**. Blumenau: 2005. p 18-26. Monografia (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, 2005.

CALETTI, Marcos. **IPS (Intrusion Prevention Systems) um estudo teórico experimental.** Rio Grande do Sul, 2007. Trabalho de Conclusão de Curso (Graduação do Curso de Ciências da Computação) – Instituto de Ciências Exatas e Tecnológicas (ICET), Centro Universitário FEEVALE, Rio Grande do Sul, 2007.

CHESWICK, Wiliam R; BELLOVIN, Steven M; RUBIN, Aviel D. **Firewalls e Segurança na Internet**: Repelindo o Hacker Ardiloso. 2º edição. Porto Alegre: Artmed, 2005. 400 p.

COMBS, Gerald. **Wireshark**. Disponível em <http://www.wireshark.org>. Acesso em 28 de agosto de 2007.

COTA, Alan. **IPCOP Firewall:** Uma ótima opção de prevenção para sua rede ADSL. 2005. Disponível em <http://www.vivaolinux.com.br/artigos/verArtigo.php?codigo=2683&pagina=1>. Acesso em 05 de junho de 2007.

DEMPSTER, Barrie; EATON-LEE, James. **Configuring IPCop Firewalls**: Closing Borders with Open Source. Reino Unido: Packt Publishing, 2006. 241 p.

DERAISON, Renaud at al. **Nessus Network Auditing**. Estados Unidos da América: Syngress Publishing, 2004. 408 p.

ENDIAN. Site oficial da empresa. Disponível em <http://www.endian.it>. Acesso em 27 de julho de 2007.

FREE Software Foundation. **GNU General Public License**. Disponível em <http://www.gnu.org/copyleft/gpl.html>. Acesso em: 10 de maio de 2007.

GAGLIARDO, Diego et al. **Administrative Guide**. 2006. Disponível em <http://www.endian.it/fileadmin/documentation/efw-admin-guide/en/efw-admin-guide.html>. Acesso em 27 de julho de 2007.

GROOM, Ryan. **IPCOP Firewall Review**. Business Security. Disponível em <http://bizsecurity.about.com/od/securityproductreviews/fr/ipcopreview.htm>. Acesso em: 12 de maio de 2007.

HENMI, Anne et al. **Firewalls Polices and VPN Configurations**. Canada: Syngress, 2006. 504 p.

HPING. Disponível em <http://www.hping.org/>. Acesso em 07 de agosto de 2007.

INSECURE.ORG. Nmap. Disponível em <http://insecure.org/nmap/index.html>. Acesso em: 31 de maio de 2007.

KIZZA, Joseph Migga. **Computer Network Security**. Estados Unidos da América: University of Tennessee-Chattanooga, 2005. p 209 a 423.

LIMA, Fábio Marcelo de. ***Endian Firewall*** – Uma solução completa para um servidor de *Internet*. Disponível em <http://www.vivaolinux.com.br/artigos/verArtigo.php?codigo=6000&pagina=1>. Acesso em 27 de julho de 2007.

LYON, Gordon. **Insecure.Org**. *Nmap security scanner*. Disponível em <http://insecure.org/nmap/man/>. Acesso em 30 de setembro de 2007.

LOSHIN, Peter. **Big Book of IPSec RFCS**. Estados Unidos da América: Morgan Kaufmann Publishers Inc, 1999. 560 p.

MAIWALD, Eric. **Network Security**: A Beginners Guide. Estados Unidos da América: McGraw-Hill/Osborne, 2001. 441 p.

MOREIRA, Marcelo Eduardo da Silva; CORDEIRO, Rômulo Facuri Miranda. **Desenvolvimento de Procedimentos de Segurança e Implantação de Firewall no Laboratório de Bioinformática da Rede Genoma Centro-Oeste**. Brasília, 2002. p. 1 a 66. Monografia (Graduação do Curso de Ciências da Computação) – Instituto de Ciências Exatas, Universidade de Brasília. Brasília, 2002.

NOONAN, Wes; DUBRAWSKY, Ido. **Firewall Fundamentals**. Indianópilos, USA: Cisco Press, 2006. 408 p.

SATAN Documentation. Disponível em <http://www.porcupine.org/satan/demo/satan_documentation.html>. Acesso em: 24 de maio de 2007.

STREBE, Matthew; PERKINS Charles. **Firewalls**. São Paulo: Makron Books, 2002. 411 p.

TANENBAUM, Andrew S. **Redes de Computadores**. 4ª edição. Rio de Janeiro: Elsevier Editora LTDA, 2003. 945 p.

VMWARE. VMware Server. Disponível em < http://www.vmware.com/>. Acesso em 3 de novembro de 2007.

ZMOGINSKI Felipe. **Crackers preferem PCs de pequenas empresas**. Brasil: 2006. Disponível em <http://info.abril.com.br/aberto/infonews/112006/21112006-3.shl>. Acesso em 01 de maio de 2007.

ANEXO I

Este anexo contém algumas imagens do processo de instalação do *firewall ipcop*. Foram selecionadas as que possuem as informações mais importantes.

Figura 3.14 - Tela de boas vindas do ipcop.

Figura 3.15 - Selecão de idioma do ipcop.

Figura 3.16 - Seleção de endereço IP e máscara.

Figura 3.17 - Configuração do nome de domínio.

Figura 3.18 - Configuração de ISDN.

Figura 3.19 - Configuração dos tipos de rede.

A figura 3.19 apresenta a configuração de senhas para o *ipcop*. A primeira é o *root*, usado no acesso local ao *ipcop*. A segunda para o *admin*, utilizado na administração via *http*. Por fim a senha do *backup*, utilizada para a exportação segura da chave de *backup*.

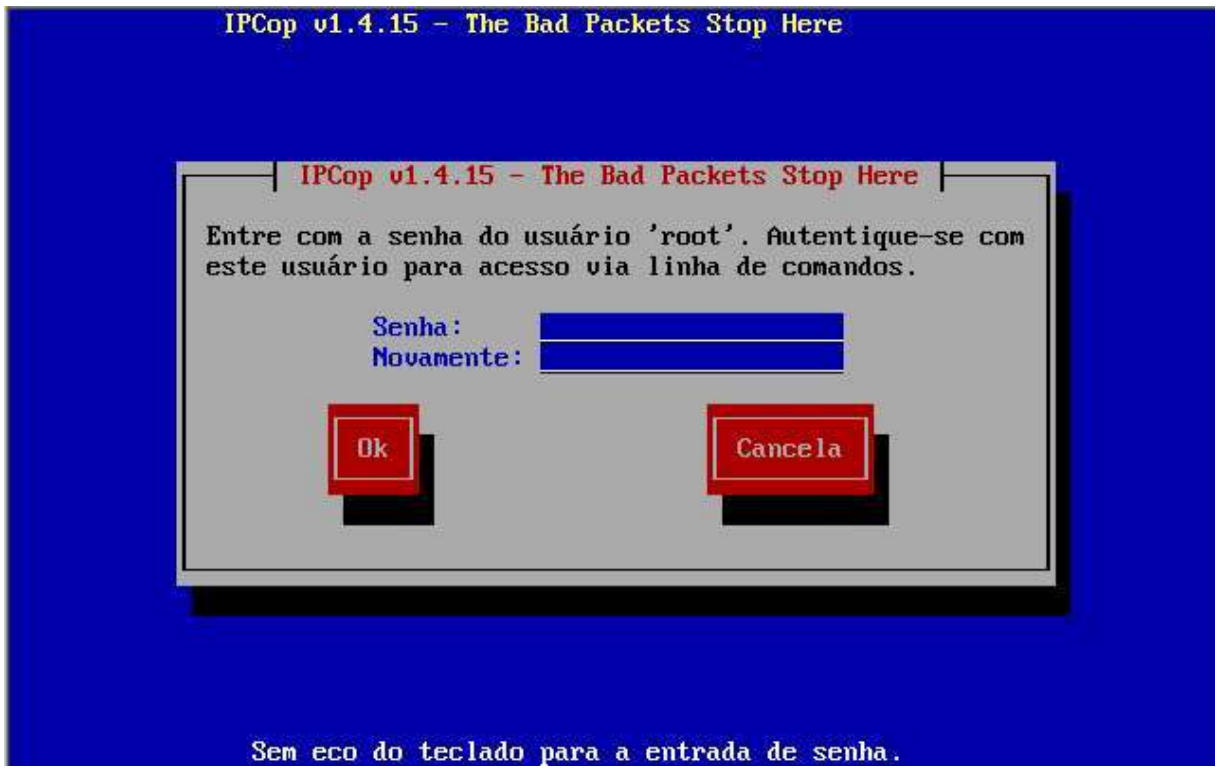

Figura 3.20 - Configuração de senhas, *root*, *admin*, *backup*.

ANEXO II

Neste anexo estão presentes algumas imagens feitas a partir da instalação do *firewall* EFW. Estas foram selecionadas, pois possuem informações importantes durante a instalação.

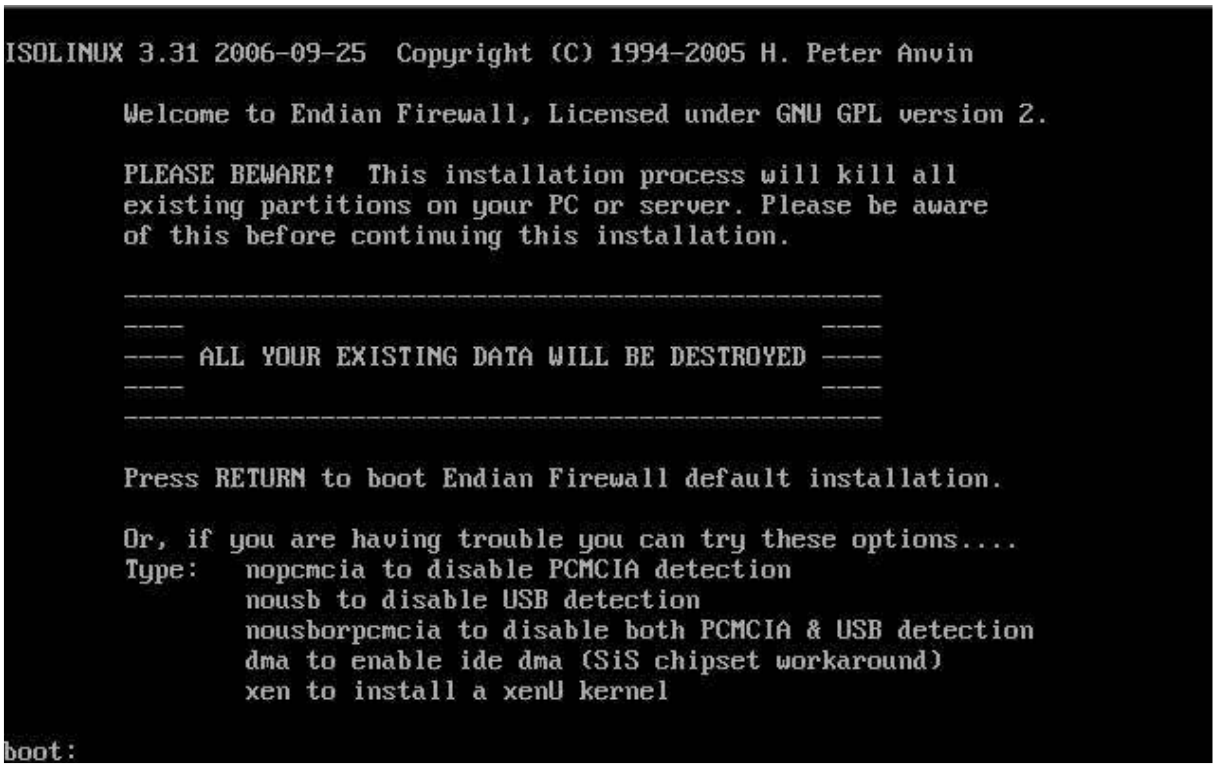

Figura 3.21 - Tela de boas vindas do EFW.

Figura 3.22 - Seleção de idioma.

Figura 3.23 - Configuração do endereço IP e máscara.

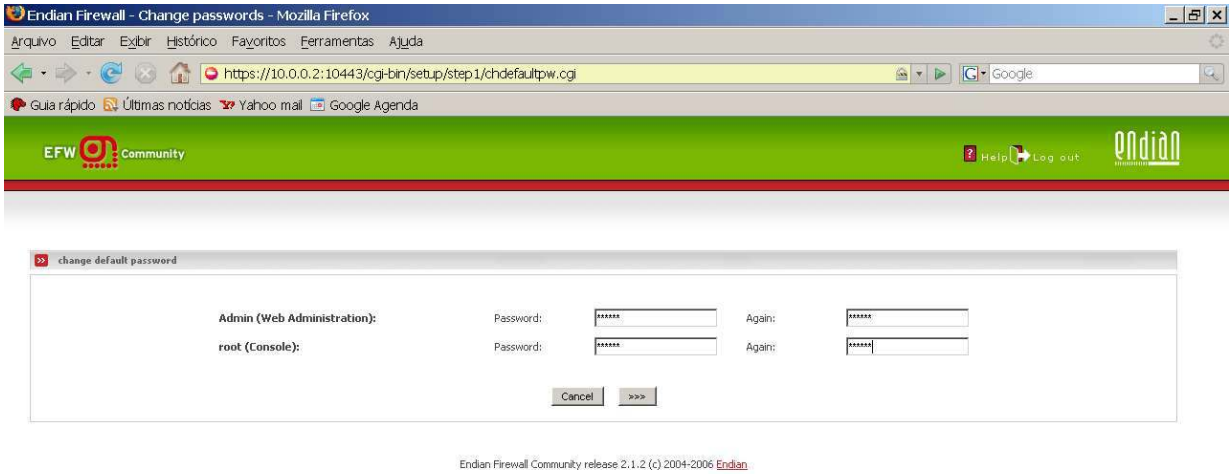

Figura 3.24 - Configuração de senhas.

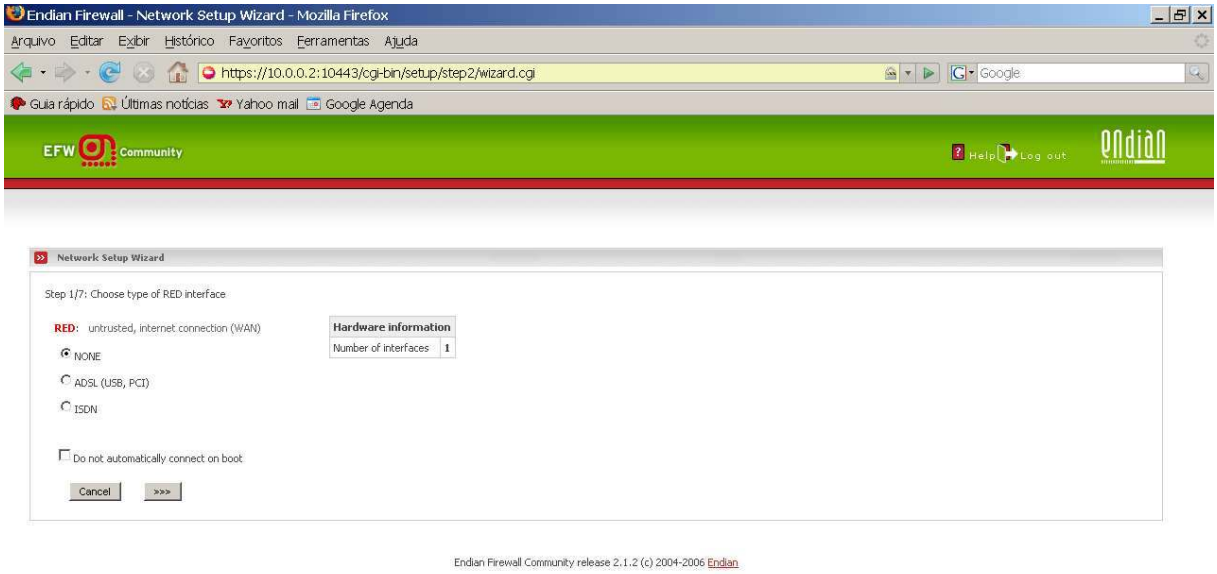

Figura 3.25 - Configuração da interface *RED*.

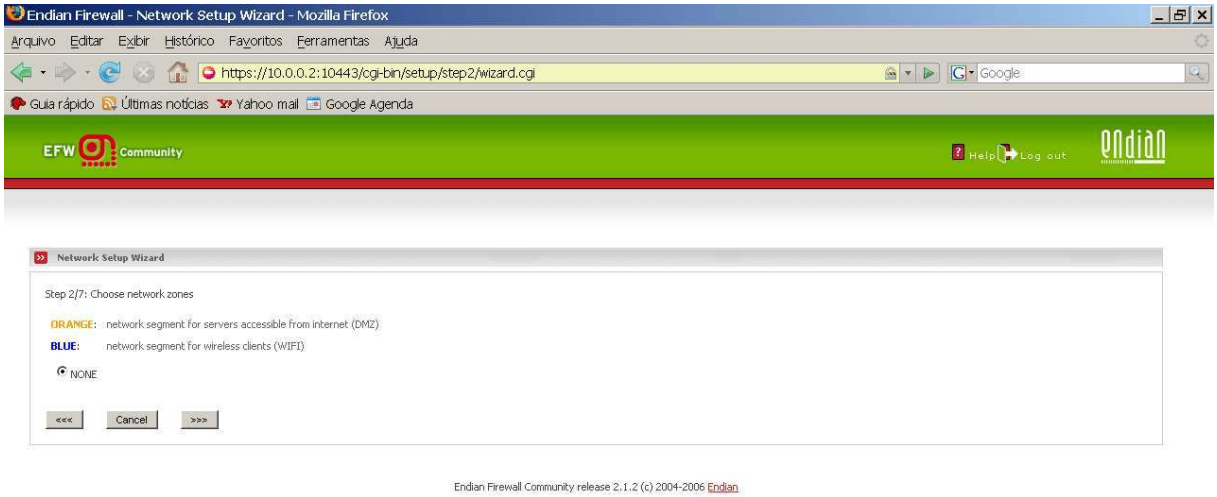

Figura 3.26 - Configuração das interfaces *ORANGE* e *BLUE*.

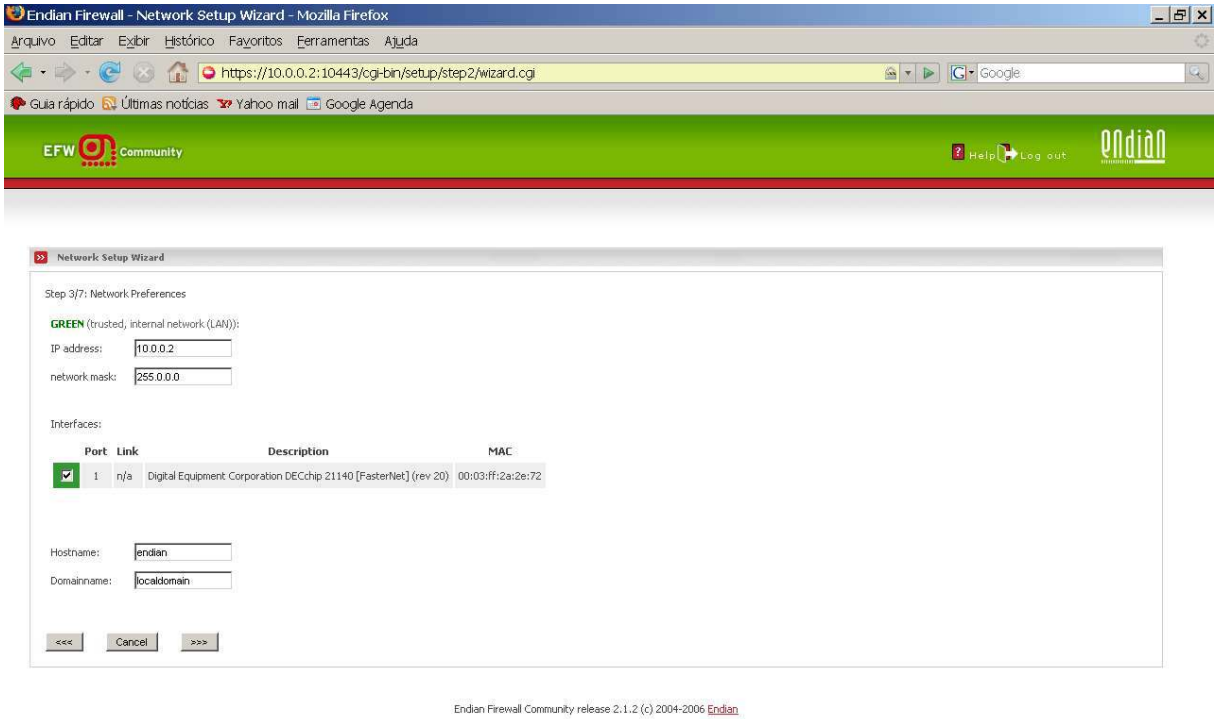

Figura 3.27 - Configuração da interface *GREEN*, *hostname* e *Domain name*.

Figura 3.28 - Configuração de endereços IPs de servidores DNS.

ANEXO III

Para acessar a ajuda do *hping*, depois de instalado, deve-se digitar *hping2 --help*. O sistema operacional *Linux* diferencia as letras maiúsculas das minúsculas, ou seja, as opções listadas abaixo devem ser utilizadas exatamente como estão.

```
usage: hping host [options]
  -h  --help       show this help
  -v  --version    show version
  -c  --count      packet count
  -i  --interval   wait (uX for X microseconds, for example -i u1000)
      --fast       alias for -i u10000 (10 packets for second)
  -n  --numeric    numeric output
  -q  --quiet      quiet
  -I  --interface  interface name (otherwise default routing interface)
  -V  --verbose    verbose mode
  -D  --debug      debugging info
  -z  --bind       bind ctrl+z to ttl           (default to dst port)
  -Z  --unbind     unbind ctrl+z
Mode
  default mode TCP
  -0  --rawip      RAW IP mode
  -1  --icmp       ICMP mode
  -2  --udp        UDP mode
  -8  --scan       SCAN mode.
                   Example: hping --scan 1-30,70-90 -S www.target.host
  -9  --listen     listen mode
IP
  -a  --spoof      spoof source address
  --rand-dest      random destionation address mode. see the man.
  --rand-source    random source address mode. see the man.
  -t  --ttl  ttl (default 64)
  -N  --id   id (default random)
  -W  --winid      use win* id byte ordering
  -r  --rel  relativize id field       (to estimate host traffic)
  -f  --frag       split packets in more frag.  (may pass weak acl)
  -x  --morefrag       set more fragments flag
  -y  --dontfrag   set dont fragment flag
  -g  --fragoff    set the fragment offset
```

```
  -m  --mtu      set virtual mtu, implies --frag if packet size > mtu
  -o  --tos      type of service (default 0x00), try --tos help
  -G  --rroute   includes RECORD_ROUTE option and display the route buffer
  --lsrr    loose source routing and record route
  --ssrr    strict source routing and record route
  -H  --ipproto set the IP protocol field, only in RAW IP mode
ICMP
  -C  --icmptype      icmp type (default echo request)
  -K  --icmpcode      icmp code (default 0)
      --force-icmp    send all icmp types (default send only supported types)
      --icmp-gw set gateway address for ICMP redirect (default 0.0.0.0)
      --icmp-ts   Alias for --icmp --icmptype 13 (ICMP timestamp)
      --icmp-addr     Alias for --icmp --icmptype 17 (ICMP address subnet mask)
      --icmp-help     display help for others icmp options
UDP/TCP
  -s  --baseport base source port          (default random)
  -p  --destport [+][+]<port> destination port(default 0) ctrl+z inc/dec
  -k  --keep     keep still source port
  -w  --win      winsize (default 64)
  -O  --tcpoff   set fake tcp data offset     (instead of tcphdrlen / 4)
  -Q  --seqnum shows only tcp sequence number
  -b  --badcksum      (try to) send packets with a bad IP checksum
                 many systems will fix the IP checksum sending the packet
                 so you'll get bad UDP/TCP checksum instead.
  -M  --setseq   set TCP sequence number
  -L  --setack   set TCP ack
  -F  --fin      set FIN flag
  -S  --syn      set SYN flag
  -R  --rst      set RST flag
  -P  --push     set PUSH flag
  -A  --ack      set ACK flag
  -U  --urg      set URG flag
  -X  --xmas     set X unused flag (0x40)
  -Y  --ymas     set Y unused flag (0x80)
  --tcpexitcode use last tcp->th_flags as exit code
  --tcp-timestamp      enable the TCP timestamp option to guess the HZ/uptime
Common
  -d  --data     data size                 (default is 0)
  -E  --file     data from file
  -e  --sign     add 'signature'
  -j  --dump     dump packets in hex
  -J  --print    dump printable characters
  -B  --safe     enable 'safe' protocol
  -u  --end      tell you when --file reached EOF and prevent rewind
  -T  --traceroute      traceroute mode           (implies --bind and --ttl 1)
  --tr-stop      Exit when receive the first not ICMP in traceroute mode
  --tr-keep-ttl  Keep the source TTL fixed, useful to monitor just one hop
  --tr-no-rtt    Don't calculate/show RTT information in traceroute mode
ARS packet description (new, unstable)
  --apd-send     Send the packet described with APD (see docs/APD.txt)
```