CENTRO UNIVERSITÁRIO FEEVALE

LEÔNIDAS KLEIN SALDANHA

FIREWALLS DE BAIXO CUSTO

Novo Hamburgo, junho 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, junho 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 General Public License* (GPL) voltados para o segmento *Small Office/Home Office* (SOHO, Pequenos escritórios/escritórios em casa) de empresas. Também é apresentada uma ferramenta de teste de vulnerabilidades para ser aplicada 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 custo. Por esse motivo o estudo será embasado em duas ferramentas que não possuem custo inicial, possibilitando a aquisição do mesmo de uma maneira simplificada. Serão 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 será 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 General Public License (GPL) firewalls aimed to the small office/home office (SOHO) enterprise segment. It also shows a tool 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 cost firewalls. 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 will be 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

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

SOHO	*Small Office/Home Office*
VPN	*Virtual Private Network*
NAT	*Network Address Translation*
ADSL	*Asymmetric Digital Subscriber Line*
DNS	*Domain Name System*
IP	*Internet Protocol*
TCP/IP	*Transmission Control Protocol over Internet Protocol*
OSI	*Open Systems Interconnection*
UDP	*User Datagram Protocol*
ICMP	*Internet Control Message Protocol*
TCP	*Transmission Control Protocol*
IGMP	*Internet Group Management Protocol*
HTTP	*Hypertext Transfer Protocol*
FTP	*File Transfer Protocol*
DMZ	*Demilitarized Zones*
DHCP	*Dynamic Host Configuration Protocol*
SMTP	*Simple Mail Transfer Protocol*
POP	*Post Office Protocol*
RPC	*Remote Procedure Call*
SOCKS	*Sockets*
SNMP	*Simple Network Management Protocol*
CD	*Compact Disc*
DVD	*Digital Video Disc*
GPL	*GNU Public License*
SNAT	*Source Network Address Translation*

DNAT	*Destination Network Address Translation*
QoS	*Quality of Service*
TOS	*Type of Service*
DoS	*Deny of Service*
RAM	*Random Access Memory*
IPv4	IP versão 4
IPv6	IP versão 6
PCI	*Peripheral Component Interconnect*
USB	*Universal Serial Bus*
PPP	*Point-to-Point Protocol*
IDS	*Intrusion Detection System*
SATAN	*Security Analysis Tool for Auditing Networks*
WWW	*World Wide Web*
SSL	*Security Sockes Layer*
TLS	*Transmission Layer Security*
NASL	*Nessus Attack Scripting Language*
ID	*Identificador*
PC	*Personal Computer*

SUMÁRIO

INTRODUÇÃO

Dadas as atuais circunstâncias nas quais as pequenas empresas são submetidas por estarem conectadas na *Internet*, os pequenos escritórios/escritórios em casa (SOHO, *Small Office/Home Office*) devem se manter protegidos 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: restrição de acesso à rede interna, controle sobre o tráfego da *Internet*, relatórios de segurança por meio de *logs*, filtro de conteúdo indevido...

Todo o tráfego entre a rede privada e a *Internet* deve necessariamente passar pelo *firewall*, assim ele pode 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, mas é um pequeno preço que deve ser pago pela segurança. (STREBE e PERKINS, 2002).

Os *firewalls* podem possuir serviços dentre os quais se destacam: filtro de pacotes, *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 necessidade de utilizar ou não os recursos apresentados anteriormente varia, existem casos em que a empresa não precisará de todos. Isto depende do ramo de atividade, do tamanho da empresa e de outras variáveis.

As pequenas empresas possuem características distintas das atividades, o que gera necessidades de proteção diferenciadas. Seus gestores devem averiguar a importância de seus dados e tomar as providências.

Observando-se que o custo das conexões de alta velocidade com a *Internet* vem diminuindo ao longo dos anos, cada vez mais as SOHO as utilizam, o que agrava as vulnerabilidades: pelo fato da exposição à *Internet* durar longos períodos de tempo, pela falta de política de segurança e pelo investimento em segurança da informação ser limitado.

Não utilizar um *firewall* em uma pequena empresa é um agravante para a baixa 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.

Mesmo que as SOHO não sejam alvos comuns, por muitas vezes não possuírem dados que interessem a outros, a segurança não pode ser deixada de lado. Ao contrário destas, existem empresas pequenas que lidam com dados sigilosos como: clínicas médicas, escritórios de advocacia, contabilidade, etc. Isso deve ser levado em conta na hora de configurar o nível de segurança e ou bloqueio que o *firewall* irá oferecer.

Ainda assim uma pequena empresa pode atrair um *cracker*, a tendência é que ele seja impulsionado a invadi-la pelo motivo da baixa segurança. 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).

"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.

A seguir são apresentados embasamentos teóricos 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 são descritos os dois *firewalls* a serem testados e também as ferramentas utilizadas para o teste. Na seqüência é apresentada a maneira de como serão efetuados os testes e a conclusão do trabalho.

1 *FIREWALLS*

1.1 Definição de *firewall*

"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).

Firewall é um dispositivo *hardware* ou *software* utilizado para tentar impedir a invasão, limitar os danos que podem ser causados por ela e isolar uma porção de dados sensíveis do resto do mundo.

Ele 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* baseia-se nas regras à que foi submetido para efetuar o controle de acesso à rede.

Ele 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 que não é 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.

Ele não precisa ser um equipamento só, seus recursos 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 ele 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, gravar e relatar eventos.

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*, etc.

Outra seria a facilidade de conserto do equipamento em caso de falha, pois o mesmo provavelmente utiliza peças que podem ser adquiridas facilmente, até mesmo a troca do *hardware* inteiro é possível.

A desvantagem que *firewalls* de *software* sofrem é a continua 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 será criado um pacote para correção, antes de disponibilizar para instalação os fornecedores fazem testes e publicam notas sobre a instalação do mesmo. Mas, isso não significa que o pacote não está sujeito a problemas, ele 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*

São dispositivos integrados de *software* e *hardware*, os mesmo são desenvolvidos para serem 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, ele é desenvolvido ou modificado para permitir um alto desempenho e interagir com o programa do *firewall* da melhor maneira.

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 funções é provável que seja necessário a compra de outros equipamentos, 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, o que reduz o custo de manutenção, o custo de compra de vários dispositivos e simplifica a topologia da rede onde ele se encontra.

A desvantagem é que em caso de falha, 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*

Um produto *firewall* pode conter uma ou mais características dentre as seguintes:

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, ele 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 over Internet Protocol* (TCP/IP) pode conter. Ele 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. A segunda forma 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 *stateles* 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 BRAWSKY (2006) uma conexão é a maneira que dois *hosts* utilizam para conseguirem se comunicar, ela 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 ele

pode bloquear as conexões não permitidas e garantir passagem para as permitidas. (NOONAN e BRAWSKY, 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 o *cracker* pode modificar o seu próprio endereço e tentar um novo ataque.

Outra falha é a não verificação do conteúdo do pacote, por exemplo: quando um usuário navega na *Internet* ele 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*.

Por outro lado ele é fácil de configurar e eficaz no roteamento dos pacotes, o que não prejudica a eficiência da rede em caso de grande tráfego de dados. Quando se tem 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 ele 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 ele 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.

1.5.3 NAT

Primeiramente o *Network Address Translation* (NAT, tradução de endereço de rede) foi desenvolvido com o propósito de aumentar os endereços IP’s 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 que ela traz, pois ao traduzir os endereços, os computadores internos ficam inacessíveis externamente e seus endereços na rede interna são ocultados, aumentando a segurança da rede.

Para conseguir traduzir os endereços a NAT efetua os processos basicamente desta forma: quando um *host* interno requisita uma conexão para um externo ela recebe o pacote anota 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:

Ambos os *hosts* internos 192.168.1.2 e 192.168.1.1 querem 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 ela partiu do endereço 172.28.230.55 e não de algum computador em uma rede interna.

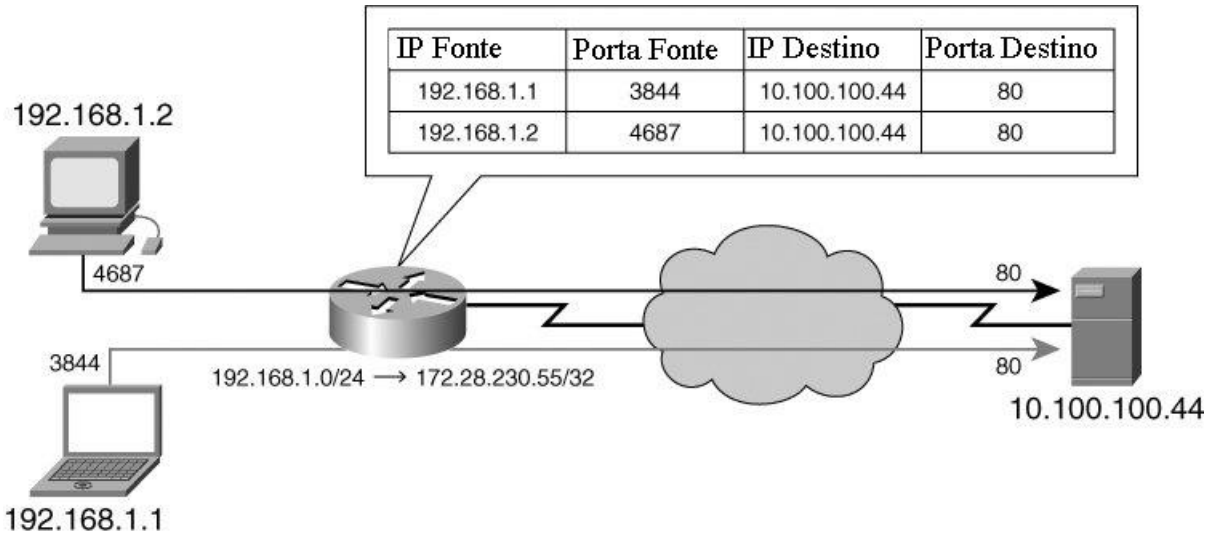

Figura 1.2 - Exemplo do funcionamento da NAT.
Fonte: NOONAN e BRAWSKY, 2006 (Adaptação nossa)

Quando for requisitada uma conexão para fora da rede em uma porta que já está em uso por outra conexão a 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 a 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 – 172.16.255.255 (máscara de 12 *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 não funcionam corretamente com ela, 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 em serviços oferecidos na *Internet*. Para tanto ela 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 *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 ele foi configurado. Quando este tipo de conversão é utilizado a possibilidade de oferecer algum serviço na *Internet* 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 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*

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 BRAWSKY, 2006).

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

Utilizam uma filtragem de pacotes *statefull* 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 na verdade como intermediários nas comunicações, eles não permitem que o *host* interno se comunique diretamente com o *host* externo, ou seja, ele não faz roteamento das mensagens como ocorre nos filtros de pacotes e na NAT. Para tanto ele 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. Ele é totalmente transparente, ou seja, externamente o tráfego parece ser gerado por um computador extremamente estressado e nenhum computador 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) ele 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, ele 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 que é 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 esconder 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 ele contém.

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

1.5.5 *Stateful Firewall*

Ele 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 que consegue fazer a verificação de sessão que a NAT e o *proxy* não fazem. A desvantagem é que ele 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 ele opera, elas são as mesmas das funcionalidades agrupadas.

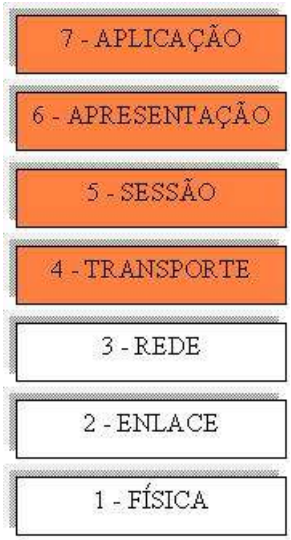

Figura 1.5 - Modelo OSI. Camadas em que trabalha *proxy stateful*.

1.5.6 *Firewalls* Transparentes

Esta tecnologia de *firewall* opera na camada dois, monitora a camada três e mais o tráfego de rede. Ela 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 dele tender a ser simples, seu uso de processamento é baixo, possibilitando a filtragem de um alto volume de dados. A sua configuração é facilitada pelo motivo dele operar na camada dois, onde não é necessário endereço IP. Desta forma o *firewall* também não pode ser atacado, como não possui endereço, ele fica invisível ao atacante.

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 linhas dedicadas para o mesmo intento, porém a sua segurança é um pouco inferior, o que não justifica a sua crucificação. Em uma VPN bem configurada as chances de invasão são pequenas.

Ela utiliza encapsulamento para que os pacotes sejam transportados até a rede destino. Por encapsulamento entende-se que é a maneira de colocar um protocolo dentro do outro, possibilitando assim a transmissão dos mesmos em redes distintas. (CHESWICK e BELLOVIN, 2005).

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. Para uma segurança maior a senha deve ser trocada no mínimo a cada 30 dias. Outra prática para o aumento da credibilidade do túnel criptográfico é a autenticação criptografada, ela não é igual a criptografia de dados. O primeiro é utilizado para a criptografia durante a autenticação (geralmente usuário e senha), os dados posteriores não são criptografados. Já o segundo é a criptografia dos dados (geralmente posteriores à autenticação) que circulam entre as redes.

A rede virtual privada funciona basicamente desta forma: o *host* I deseja se comunicar com o *host* II que está em uma rede diferente (outra cidade, por exemplo). Então o *host* I verifica que o *host* II não está na mesma rede e encaminha o pacote para o *gateway* da rede. Desta forma o *gateway* criptografa o pacote, encapsula o mesmo e envia-o pela interface de rede que está ligada na *Internet*. O *gateway* da rede onde o *host* II se encontra recebe o pacote, analisa a veracidade do mesmo e, se ele provém de uma fonte confiável, abre e descriptografa o pacote encapsulado e o entrega para o *host* II. Este exemplo é uma das formas mais simples para entender o funcionamento da VPN.

Existem três tipos de VPN: as que conectam escritórios e compartilham as redes, as que conectam na rede, 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 eles é que 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 ele for por *software* o mesmo fornecedor deve ser utilizado. Isto não é uma regra, sim uma sugestão, porque existem produtos que não são compatíveis uns com os outros. O problema de compatibilidade 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 ele deve possuir maneiras de relatar: falhas de segurança, violações de acesso, entre outros.

Para relatar os acontecimentos extraordinários os *firewalls* podem se munir de diversas maneiras para avisar o administrador sobre o ocorrido 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. Ela ajuda a resolver problemas de filtros, como: quando algo que deveria ser permitido não está sendo, o administrador usa os *logs* para verificar o problema. Com ela, também é possível averiguar como está a saúde do *firewall* espaço em disco, processamento, etc. Além disto, os relatórios podem avisar o administrador sobre incidentes ou invasões, desta maneira o mesmo pode tomar as devidas medidas.

Os *logs* exigem uma monitoração, o administrador deverá fazê-la periodicamente.

O *firewall* pode criar relatórios de diferentes maneiras: ele pode utilizar recursos do próprio *firewall* ou pode utilizar uma ferramenta extra: 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 posicionar ele na topologia da mesma. Dependendo das necessidades em questão na empresa, a localização do *firewall* deverá ser estudada, pois o posicionamento incorreto poderá diminuir a segurança que ele oferece.

Em princípio existem duas arquiteturas que podem ser utilizadas para a disposição do *firewall*: arquitetura de um único *firewall* e arquitetura de dois *firewalls*, também conhecida como *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 DMZ 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 que ele se conecte 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 se está protegendo o servidor do banco de dados que possui informações críticas. Como o servidor *web* é acessível externamente, está mais suscetível a ataques, se ele 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 de locais não confiáveis (*Internet*) será quebrada). Quando se oferece um serviço com esta topologia, está se permitindo a entrada de conexões vindas da *Internet* e desta maneira colocando toda a rede em perigo. (NOONAN e BRAWSKY, 2006).

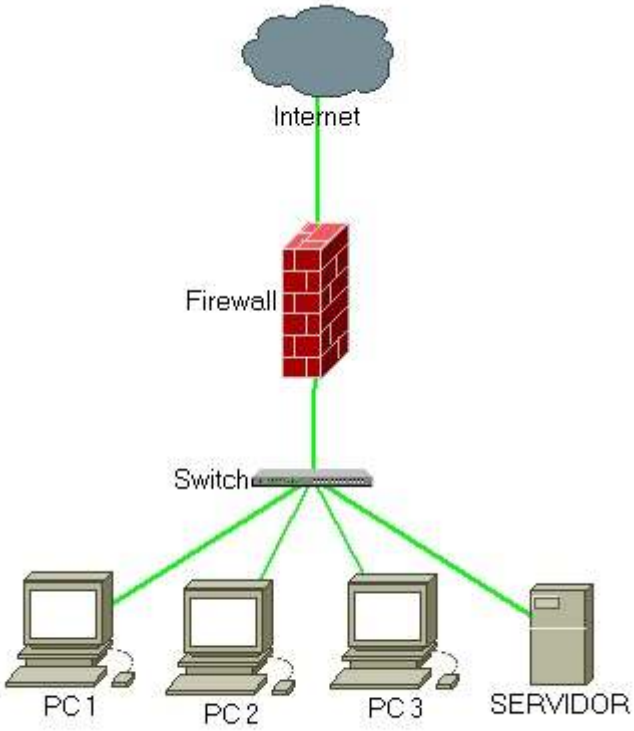

Figura 1.6 - Arquitetura de rede sem DMZ.

Com uma DMZ: desta forma o *firewall* possui três interfaces de rede: uma é 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, pois só serão acessados *hosts* da DMZ.

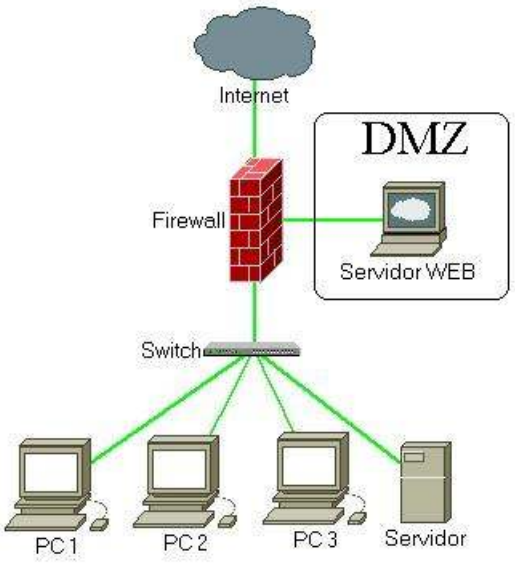

Figura 1.7 - Arquitetura de rede com DMZ.

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. Desta forma a segurança da rede fica elevada, ainda mais se forem utilizados *firewalls* de fabricantes diferentes, pois para um *cracker* conseguir acesso à rede interna ele terá de ter conhecimento sobre duas ferramentas de proteção diferentes e que provavelmente não são sujeitas ao mesmo tipo de ataque.

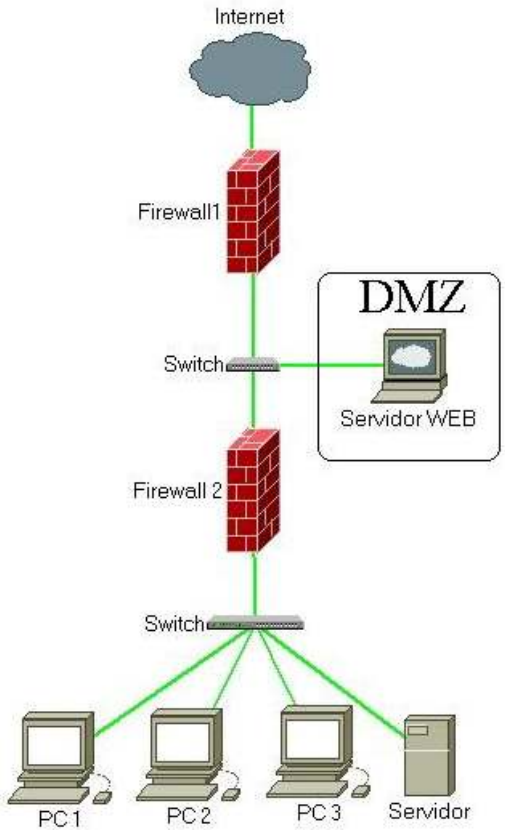

Figura 1.8 - DMZ *Dual Firewall.*

Sua desvantagem é quanto ao custo, é necessária a aquisição de dois equipamentos, para administrar, também é necessário que se tenha um administrador com experiência nos dois produtos ou que se tenha um administrador para cada. A configuração dos mesmos será diferente aumentando a manutenção.

Pelo seu custo, esse *design* é utilizado somente em redes altamente críticas como bancos e o governo.

1.7 O Que *Firewalls* Não Podem Fazer

O *firewall* é a primeira linha de defesa que a empresa deve possuir, ele 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 necessariamente passar pelo *firewall.* Todo e qualquer tipo de dado que não passar por ele não poderá ser analisado. 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.). Ele

também não pode proteger contra dados que são permitidos, muitas vezes *cracker* utilizam falhas em serviços permitidos para invadir uma rede.

Um exemplo: Imagine que um funcionário recebe um *e-mail* forjado e que pede para instalar um programa no computador, mas na verdade este programa é um Cavalo-de-Tróia que ao ser ativado abre uma porta local e pede a conexão com o computador de um *cracker*. Deste modo, como a comunicação é feita de dentro para fora, fica mais complicado de um *firewall* detectar esse problema dependendo de sua configuração. (STREBE e PERKINS, 2002).

Ao utilizar uma VPN o ideal é que o próprio *firewall* faça este serviço, pois se for utilizado outro computador que está dentro da rede, o tráfego que entrará vai estar criptografado e não haverá possibilidade de ser analisado.

Em grandes redes o *firewall* não conseguirá garantir a integridade dos dados, pois se torna difícil de fazer com que ele analise todos os pacotes em ambos os sentidos: entrada e saída.

Todos estes pontos devem ser bem analisados para que o *firewall* ganhe reforços e possa proteger da melhor forma a rede.

Somente um *firewall* não é o 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 que seja bem desenvolvida para minimizar ao máximo os riscos existentes na *Internet*. Como nunca é demais quando se fala em segurança, deve-se frisar que as senhas nunca devem ser entregues à pessoas estranhas e devem ser trocadas regularmente, as estações de trabalho devem 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, é só uma questão de tempo.

2 *FIREWALLS* A SEREM TESTADOS

Devido ao baixo poder aquisitivo de empresas do segmento SOHO, as mesmas não podem investir altos valores para a aquisição de *firewalls* com alto custo. Muitas também não contam com um serviço para a manutenção de seus equipamentos de informática, da mesma forma não possuem alguém 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).

O sistema operacional é *Linux* e a maioria das suas versões de distribuição também são gratuitas.

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 o mesmo 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 *iptables* que é nativo do *kernel* do *Linux* e é bastante utilizado, devido a sua existência em quase todas as distribuições do *Linux*. Também o *ipcop*, que é um *firewall* gráfico, de fácil utilização. Segundo COTA (2005) e DEMPSTER e EATON-LEE (2006) é uma ferramenta especialmente desenvolvida para o segmento SOHO de empresas.

2.1 *Iptables*

O *iptables* é parte integrante do *kernel* (núcleo de um sistema operacional) do *Linux*, ele é utilizado para manipular o módulo *netfilter*. Apesar do conhecimento errôneo por muitas pessoas, o *iptables* não é o *firewall* do *Linux*, ele é a última versão do manipulador do *netfilter*. Antes dele havia o *ipchains* e antes o *ipfadm*. O criador do *iptables* foi Rusty com ajuda de Michael Neuling, Rusty também foi o responsável pela criação do *netfilter*.

O *iptables* foi incorporado na versão 2.4 do *kernel* do *Linux* em meados de julho de 1999. (SILVA, 2006).

Segundo SILVA (2006), algumas das vantagens obtidas com o *firewall* do *Linux* são:

- Filtro de pacotes *stateful*;

- Suporte a pacotes TCP, UDP, ICMP;

- Serviço de *proxy;*

- Mecanismos para rejeitar automaticamente pacotes malformados ou duvidosos;

- Seus *logs* podem receber dados personalizados;

- Suporte a *Source Network Address Translation* (SNAT) (NAT de origem);

- Suporte a *Destination Network Address Translation* (DNAT) (NAT de destino).

Em OLIVERIRA (2007) são destacadas outras funcionalidades obtidas com o uso do *firewall* integrante ao *kernel* 2.4:

- *Quality of service* (QoS - qualidade de serviço);

- *Type of Service* (TOS - Implementação de tipo de serviço);

- Bloqueio de alguns tipos de invasão como:*spoofing* (troca de um IP original por outro), *Syn-Flood* (sobrecarga por conexão), negação de serviço (DoS, *Deny of Service*), *Scanners* Ocultos, *Ping* da Morte, etc.

O autor também destaca que existe a possibilidade de adicionar novas funcionalidades através de módulos externos.

Os requisitos mínimos para o funcionamento do *netfilter* dependem da quantidade de tráfego que ele terá de suportar, mas em casos onde o tráfego não é grande, pode-se dizer que ele necessita de um processador 386SX com 4 *Mega Bytes* de memória *Random Access Memory* (RAM). Estes requisitos são para a função de *firewall* do *kernel* 2.4 do *Linux*. Dependendo da versão *Linux* utilizada os requisitos mencionados não serão válidos.

O *iptables* não possui uma configuração fácil, uma vez que ele não possui regras padrões e sua administração nativa é efetuada por linha de comando, para tanto o administrador deve ter experiência sobre o funcionamento do *Linux* e do *iptables*. No intuito de facilitar a mesma, existem programas de interface gráfica para configuração do *firewall*, desta forma ele gera o código e o transmite para o *iptables*. Sendo assim, a necessidade de conhecimento sobre o *Linux* se restringe à configuração inicial do *iptables* e da ferramenta gráfica.

O *iptables* possui basicamente quatro componentes, cada um possui suas funcionalidades. Podem ser destacados:

- *iptables*: esta é a aplicação principal, ela é dedicada ao trabalho com pacotes da versão IP versão 4 (IPv4) do protocolo IP;

- *ip6tables*: possui as mesmas funções do *iptables*, mas é destinado ao trabalho com a versão IP Versão 6 (IPv6) do protocolo IP;

- *iptables-save*: o *Linux* permite que sejam adicionadas regras diretamente no *shell*, sendo que ao reiniciar o sistema elas são perdidas. Este componente grava as regras para que as mesmas possam ser restauradas em seções futuras;

- *iptables-restore*: este possui a função de restaurar as regras salvas pelo componente anterior.

Para o entendimento inicial do funcionamento do *iptables*, apresenta-se o comando: *iptables [-t tabela] [opção] [chain] [dados] -j [ação]*

As tabelas armazenam as *chains*, em uma regra *iptables* referencia-se uma tabela a qual se deseja usar no determinado momento. Caso esta opção do comando não seja preenchida, o *Linux* utilizará a tabela padrão. As tabelas existentes são: *[-t tabela]*

- *filter*: tabela padrão, armazena *chains* do tráfego de dados para ocorrências de NAT;

- *nat*: usada para tradução de endereços NAT;

- *mangle*: esta é utilizada somente quando existe a necessidade de alguma alteração especial em pacotes.

As opções que podem ser utilizadas no comando são escritas com letras maiúsculas: *[opção]*

- -P : Utilizada para a definição de uma regra padrão;

- -A: Utilizada para acrescentar uma nova regra às existentes;

- -D: Utilizada para a exclusão de uma determinada regra existente;

- -L: Faz a listagem de todas as regras existentes;

- -F: Esta opção excluirá todas as regras existentes;

- -I: Inserir uma regra nova;

- -H: Ajuda;

- -R: Utilizado para a substituição de regras;

- -C: Verifica as regras existentes;

- -Z: Zerar uma regra específica;

- -N: Serve para a criação de regras com nomes;

- -X: excluir uma regra baseando-se no seu nome;

As *chains* são utilizadas para especificar como será o tratamento dos pacotes, para cada tabela existem opções diferentes como: *[chains]*

Filter:

Input – para dados de entrada, rede externa para a rede interna;

Output – dados de saída, rede interna para a rede externa;

Forward – dados que tem seu destino em outra máquina ou interface de rede;

Esta tabela permite ao usuário criar suas próprias *chain*, as quais são comumente chamadas de *user-defined chains*.

NAT:

Prerouting – Para dados que chegam no computador (*firewall)*;

Postrouting – Para modificar os dados depois do roteamento;

Output – Para modificar os dados antes que eles sejam roteados;

Mangle:

Esta tabela utiliza os mesmos comandos das tabelas anteriores. (OLIVEIRA, 2006)

Para os *[dados]* as opções disponíveis de inserção de regras no *iptalbles* são:

-s e -d especificam os dados de origem e destino respectivamente. Eles aceitam quatro formas para a especificação dos endereços sendo: pelo IP completo (10.1.1.1), pelo nome do computador (*linux*), pelo endereço na *Internet* (www.linux.com) e pelo par de rede e máscara, sendo possível também utilizar com máscara completa ou resumida (200.200.200.0/255.255.255.0 ou 200.200.200.0/24).

As opções -i e -o especificam qual a interface de rede por onde o pacote entrará e por qual sairá especificamente. Para a especificação das interfaces o cuidado a ser tomado é com as *chains output* e *input*, as quais são saída e entrada. As *chains* de entrada não podem conter regras para controlar dados de saída e vice-versa.

Para especificar o protocolo ao qual a regra será aplicada, a opção utilizada é -p: -p TCP, -p UDP. Juntamente com a especificação do protocolo é possível determinar a porta ou uma faixa de portas para entrada e saída, sendo especificamente: -*sport* e -*dport*.

As *[ações]* são o tratamento que os dados receberão, cada regra possui uma ação e as mesmas podem ser as seguintes: *accept* (aceita o pacote), *reject* (rejeita o pacote e envia um

aviso ao remetente), *drop* (descarta o pacote sem emitir nenhum alerta para o remetente), *log* (cria uma linha no *log*, para pacotes aceitos ou rejeitados).

Pelo motivo do *iptables* ser um *firewall* de configuração manual, e possuir várias funcionalidades e flexibilidades para tratar os dados, administradores podem se deparar com o decréscimo de desempenho do mesmo devido ao aumento da quantidade de regras em ambientes que possuem diferentes necessidades para cada tipo de dado, o tamanho de uma *chain* que deve ser percorrida e o número de regras ao qual o pacote é submetido até encontrar uma que combine. Desta forma em SUEHRING e ZIEGLER (2005) descreve-se algumas regras para otimizar o desempenho do *firewall*:

- O *firewall* do *Linux* possui o método de leitura de regras *top-donw* (de cima para baixo), sendo assim, para otimizar o funcionamento do *firewall* recomenda-se que se organizem as regras de forma onde as que tratam os pacotes mais gerais fiquem anteriormente as que tratam de pacotes mais específicos. Um exemplo que pode ser considerado é de organizar as regras que tratam de tráfego TCP previamente as que tratam de UDP, isso devido ao maior uso do protocolo TCP e porque a comunicação UDP geralmente é procedida de poucos pacotes.

- Manter a proximidade das regras que tratam pacotes de serviços muito utilizados (ex.: HTTP).

- As regras que bloqueiam tráfego em portas de numeração alta devem ficar no início das *chains*, menos as regras para FTP, pois comumente esta comunicação possui um volume de dados maior.

2.2 *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. Ele partiu da necessidade das mesmas de utilizarem funcionalidades de *firewall* que somente grandes empresas utilizam, mas sem investir muito dinheiro. Começou a ser desenvolvido utilizando o *kernel* 2.4 do *Linux*, desta maneira ele é tão estável quanto o *iptables*.

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, o que anula a possibilidade de algum conflito com outro programa ou módulo. (DEMPSTER e EATON-LEE, 2006).

Para sua instalação o computador não necessita um sistema operacional, pois o *ipcop* já o instala, 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*, ao contrário do *iptalbes*, ele já possui sua administração via interface gráfica.

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, sendo que este número é adequando, levando em conta que as pequenas empresas não possuem muitas redes normalmente. Ele 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 rede vermelha é a não confiável (*Internet*, ou outra rede em uma topologia maior), neste caso, por padrão todas as tentativas de conexões oriundas da interface vermelha 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 ele é desenvolvido para pequenas empresas, existe a possibilidade de utilizar *modem* ADSL *Peripheral Component Interconnect* (PCI) ou *Universal Serial Bus* (USB), de algumas marcas[1] diretamente no *firewall*. Para utilizar *modem* roteador com interface *ethernet* (rede), é necessário que o mesmo possua a possibilidade de operar no modo *Point-to-Point Protocol* (PPP) *half bridge*. Assim o endereço IP válido da *Internet* será fornecido para a interface de rede o 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. Como ele também efetua NAT, o processo será feito duas vezes, e a rede poderá não funcionar conforme o esperado.

O *ipcop* não é somente um *firewall* com filtro de pacotes, com ele é possível se munir 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 que ele possui algumas limitações no crescimento da topologia de rede, no momento em que a empresa se expandir para uma topologia complexa com mais de três redes diferentes o mesmo poderá não se adequar às necessidades da empresa.

[1] Maiores informações sobre as 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

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 procedem sobre as mesmas, etc. Para este tipo de serviço existem tanto ferramentas proprietárias quanto ferramentas gratuitas e de código aberto.

3.1 Ferramentas Avaliadas

Devido a este trabalho possuir caráter universitário e estar voltado para o segmento de pequenas empresas, descreve-se o funcionamento de três ferramentas gratuitas: *Nmap*, *Security Analysis Tool for Auditing Networks* (SATAN) e *Nessus*.

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, ele apresenta o número da porta, se ela 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 ele oferece. (INSECURE, 2007).

O *Nmap* foi desenvolvido com o intuito de trabalhar em redes com muitos computadores, por esse motivo ele funciona bem em redes com poucos *hosts*.

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, ela 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 eles podem ser analisados para corrigir as brechas na segurança existentes.

Da mesma forma que um administrador de rede pode utilizar o SATAN para buscar por falhas na sua rede ou até mesmo para auditora-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 Darinson, 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 e 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, ele pode ficar com seu computador disponível, pois o servidor é quem fará a tarefa.

Outro benefício que este tipo de arquitetura traz é que o servidor pode ser mantido com conexão remota, 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 eles que 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 eles 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. No presente momento o *Nessus* possui em sua base 14.910 *plugins*.

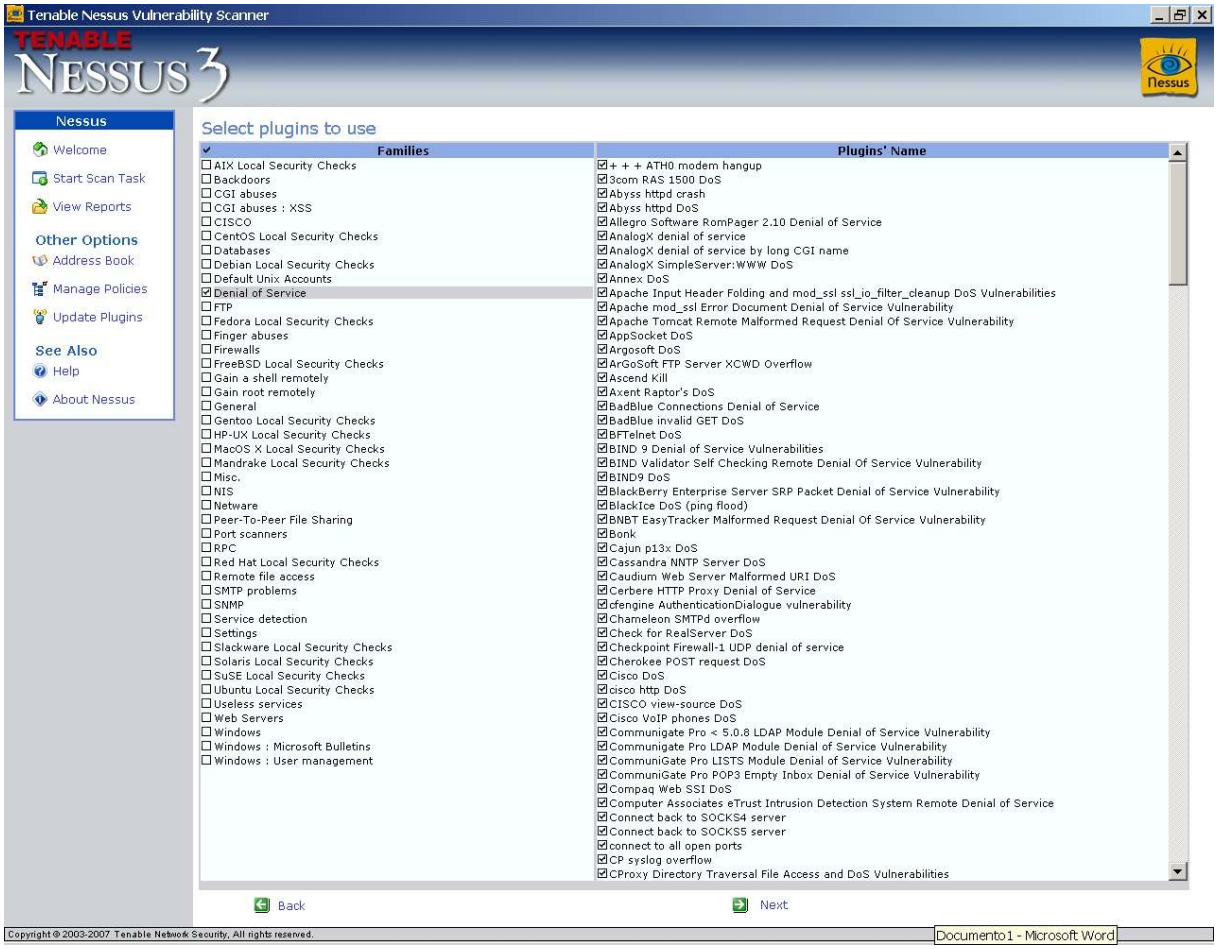

Figura 3.1 - *Plugins* do *Nessus*.

3.1.3.3 Base de Conhecimento

A base de conhecimento é utilizada para pesquisa por *plugins*, no momento em que um necessitar da resposta de outro mais primitivo, deverá efetuar a busca na base de conhecimento. 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 eles poderão procurar por 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 Leanguage (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 ele desenvolveu esta linguagem para o *Nessus* e em 2001 Michel Arboi contribuiu para que a linguagem fosse melhorada e para consertar alguns problemas existentes na mesma.

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 ele será executado e como ele 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 ao mesmo tempo 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.

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

- Portas que não serão verificadas (*consider unscanned ports as closed*): esta opção fará com que a busca por vulnerabilidades não verifique as portas marcadas como fechadas. Esta opção 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 eles 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.

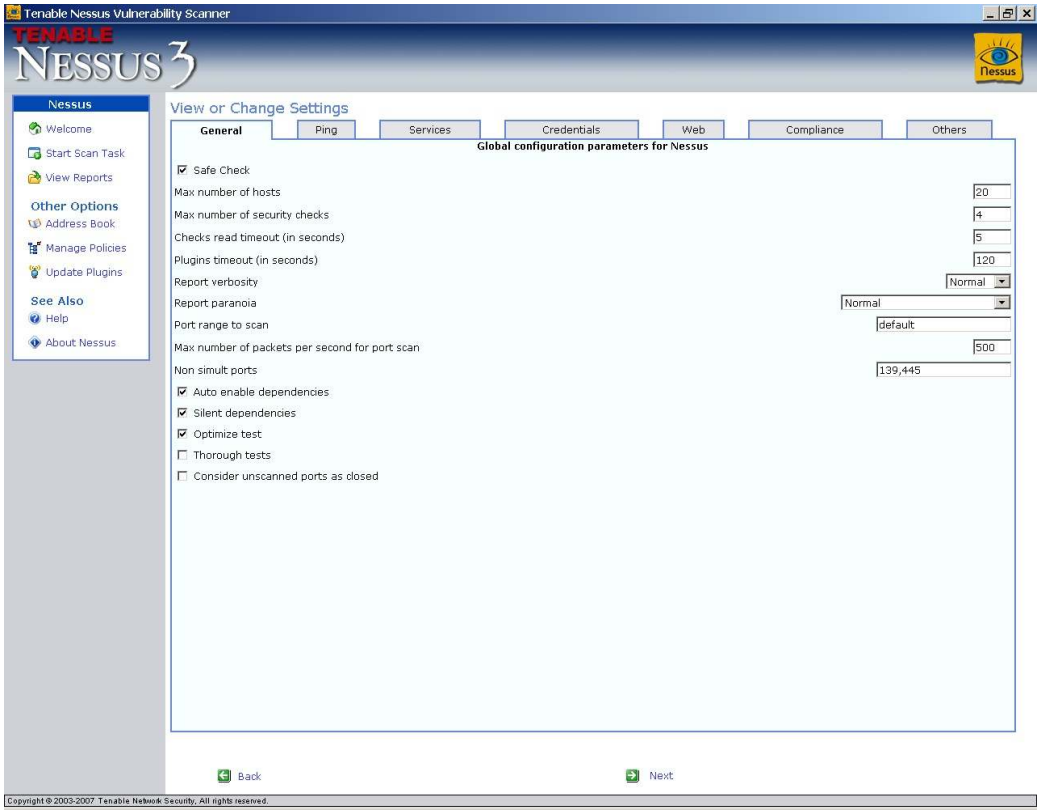

Figura 3.2 - Opções para busca por vulnerabilidade do *Nessus*.

3.1.3.6 *Nmap* no *Nessus*

O *Nessus* incorporou as funcionalidades do *Nmap*, primeiramente ele 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 ele possibilita um baixo consumo de memória para efetuar seus testes. O mesmo não ocorre quando ele necessita utilizar 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 www.nessus.org e incorporado a versão 3.x do *Nessus*.

3.2 Procedimento Para Testes dos *Firewalls*

Como pode ser notado o *Nessus* é uma ferramenta para teste de vulnerabilidades em redes bastante abrangente. Esta ferramenta será 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.

Para realizar os testes utilizar-se-ão dois computadores. Em um deles serão instalados os *firewalls*, um de cada vez, o outro irá possuir o *Nessus*. Ambos estarão conectados por rede *ethernet*.

Os testes serão efetuados em ambos os lados de cada *firewall*. Desta maneira obtêm-se as mesmas visões que um usuário mal intencionado localizado na rede interna ou de um *cracker* localizado na rede externa. Ambos serão configurados com a política de negação padrão, desta forma o teste averiguará, se mesmo com a proibição de todas as conexões externas alguma vulnerabilidade será encontrada.

Ao efetuar a busca por vulnerabilidades o *Nessus* será configurado para utilizar todos os *plugins* que ele possui, mesmo que muitos deles não sejam específicos para *firewalls*. Desta maneira evita-se o engano de não selecionar algum *plugin* que possua a capacidade de averiguar vulnerabilidades críticas existentes. Também será utilizada a opção *Thorough* para que a análise seja completa. Podendo assim obter-se resultados mais abrangentes.

Para obter-se uma melhor compreensão sobre as vulnerabilidades que o *Nessus* é capaz de averiguar 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 ele permite que um usuário interno faça 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 ele 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*[2] é desabilitar este serviço caso não utilize ou fazer um filtro na porta do mesmo para prevenir acesso indevido.

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.

Também serão efetuados testes com o *Nessus* ao mesmo tempo em que o *firewall* é submetido a um gerador de tráfego que irá simular a utilização do *firewall*. Para isso será utilizada a ferramenta *TrafficEmulator* 1.4 da *NsaSoft*[3], o mesmo é capaz de gerar tráfego ICMP, TCP e UDP. Ele é utilizado com o intuito de estressar o computador alvo, de maneira que se possa analisar o comportamento do mesmo em ambientes com alto volume de tráfego.

[2] Informação extraída do site: http://www.nessus.org/plugins/index.php?view=single&id=10135.
[3] Mais informações sobre o *TrafficEmulator* podem ser obtidas no endereço eletrônico http://www.nsauditor.com/network_tools/network_traffic_generator.html

Devido à disponibilidade de somente dois computadores, poderá haver a necessidade de virtualizar um terceiro, para isso será utilizado o Virtual *Personal Computer* (PC) 2007 da *Microsoft*.

Cada *firewall* aqui testado possui uma arquitetura diferenciada, o *ipcop* possui uma ferramenta de gerência via interface *web*, isto significa que por padrão ele possui um servidor *web* habilitado. Desta forma o mesmo será submetido por *plugins* especializados na análise em servidores deste serviço.

O *iptables* não possui nenhuma política padrão configurada, desta forma, baseando-se nos manuais e livros, as regras serão criadas para poder validar o teste. Este *firewall* utiliza um sistema operacional base, sendo que o mesmo também poderá apresentar vulnerabilidades, acredita-se que as mesmas devem ser contabilizadas e descritas, devido ao usuário poder escolher a versão *Linux* que utilizará caso utilize o *iptables*.

Além dos testes para averiguar a segurança dos *firewalls* também serão analisados aspectos como: tempo destinado à instalação e ativação, facilidade na manutenção, facilidade na instalação, possibilidade e capacidade de integração a novos serviços e a necessidade de conhecimento sobre a plataforma *Linux* para a operação do *firewall*.

Para apresentação dos resultados, serão anexados os relatórios gerados pelo *Nessus* para cada um dos *firewalls* testados. Os quesitos que não são relacionados à segurança das ferramentas serão apresentados em forma de quadro, onde haverá duas colunas nomeadas com as ferramentas de segurança e linhas para cada característica averiguada. Cada linha possuirá comentários referentes à característica em questão e ao *firewall* em questão.

Observa-se que todos os procedimentos aqui mencionados poderão sofrer modificações durantes os testes, devido ao ambiente não estar disponível para maior detalhamento. Também poderão ser inseridas novas ferramentas de testes encontradas no decorrer do trabalho, de forma que se obtenha a maior quantidade de informação possível sobre os *firewalls*, assim a análise dos resultados possuirá uma consistência reforçada.

CONCLUSÃO

Neste trabalho de conclusão I, 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 que não passe por ele em arquivos em CD ou *pen drives*, por exemplo.

Foram estudadas as teorias sobre dois *firewall* GPL para uma posterior comparação. Os escolhidos foram o *ipcop* que é específico para exercer esta função, pois o mesmo opera sobre o sistema operacional *Linux* somente com os recursos necessários, sem outros programas que não sejam úteis para um *firewall* e o *iptables* que é um manipulador do módulo *netfilter* que contém o *firewal*. O *iptables* por sua vez opera também sobre o sistema

operacional *Linux*, desta forma o computador utilizado pode também exercer outras funcionalidades como servidor de *e-mail*, servidor de arquivos, etc.

Para avaliar se os *firewalls* escolhidos não possuem vulnerabilidades foram estudadas três ferramentas, dentre elas foi escolhido o *Nessus*. O mesmo é um verificador de vulnerabilidades utilizado para buscar as mesmas em *hosts* de redes privadas. Ele possui 14.910 *plugins* que correspondem às vulnerabilidades que o mesmo pode testar.

O processo de testes será efetuado com dois computadores, os *firewalls* serão instalados em um deles, um de cada vez, e submetidos aos testes do *Nessus*. Para a comparação também serão analisadas características como: facilidade de instalação, necessidade de conhecimento em *Linux* para efetuar a configuração, etc...

A apresentação dos resultados será efetuada com o anexo dos resultados obtidos pelos testes do *Nessus* e um quadro comparativo com as características mencionadas.

REFERÊNCIAS BIBLIOGRÁFICAS

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.

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.

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.

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

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.

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; BRAWSKY, Ido. **Firewall Fundamentals**. Indianópilos, USA: Cisco Press, 2006. 408 p.

OLIVEIRA, Marcelo Fonseca de. **Entendendo a Teoria do Iptables**, 2006. Disponível em <http://www.vivaolinux.com.br/artigos/verArtigo.php?codigo=5354&pagina=1>. Acesso em: 15 de maio de 2007.

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

SILVA, Gleydson Mazioli da. **Guia Foca Linux**. 2006. Disponível em <http://focalinux.cipsga.org.br/guia/avancado/>. Acesso em: 15 de maio de 2007.

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

SUEHRING, Steve; ZIEGLER, Robert. **Linux Firewalls**. 3ª edição. Estados Unidos da América: Sams Publishing, 2005. 552 p.

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.