Richer Tiago Araújo Barros

Segurança de Perímetro com FreeBSD

Monografia de Pós-Graduação "*Lato Sensu*" apresentada ao Departamento de Ciência da Computação para obtenção do título de Especialista em "Administração em Redes Linux"

Orientador
Prof. Joaquim Quinteiro Uchôa

Lavras
Minas Gerais - Brasil
2011

Richer Tiago Araújo Barros

Segurança de Perímetro com FreeBSD

Monografia de Pós-Graduação "*Lato Sensu*" apresentada ao Departamento de Ciência da Computação para obtenção do título de Especialista em "Administração em Redes Linux"

Aprovada em *01 de Maio de 2011*

Prof. Eric Fernandes de Mello Araújo

Prof. Sanderson Lincohn Gonzaga de Oliveira

Prof. Joaquim Quinteiro Uchôa
(Orientador)

Lavras
Minas Gerais - Brasil
2011

Dedico esta monografia à Ana Thalita, minha filha. Amor verdadeiro, sentimento puro, algo inexplicável que sinto por ela e que me motiva a buscar novas conquistas nessa vida.

Agradecimentos

Agradeço especialmente a Deus, pois só nele acredito que tudo é possível. Aos meus pais, pelo amor, educação, credibilidade e carinho. À minha esposa, pela atenção, carinho e paciência durante essa jornada. Ao meu orientador, o professor, mestre e doutor Joaquim Quinteiro Uchôa pelos ensinamentos, sugestões, pela paciência e pelo profissionalismo no qual conduz a coordenação desse curso de pós-graduação. Aos demais familiares e amigos que de uma forma ou de outra me ajudaram a obter mais essa conquista em minha vida.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

A construção de um perímetro seguro sob a plataforma FreeBSD visou aumentar a segurança e a integridade dos dados na rede da Prefeitura Municipal de Itabuna. Sendo assim, qualquer usuário da intranet passou a utilizar com segurança diversos recursos que uma rede pode proporcionar como: acesso à Internet, troca de arquivos, serviços de impressão entre outros. Com o advento da configuração do servidor de perímetro, políticas de seguranças foram estabelecidas referente ao uso da Internet, intranet, dos sistemas internos da empresa, acessos remotos e outros tipos de recursos. Este trabalho objetiva apresentar e avaliar o processo de implantação deste perímetro de segurança. É explanado no decorrer dos capítulos, a instalação e configuração dos aplicativos que permitiram testar a segurança do servidor e também adquirir um desempenho considerável relativo à navegação de Internet. Desempenho este que foi alcançado através da implementação de um controle de banda e de *software*, tais como o Squid, que possui um mecanismo para realizar *cache* de objetos acessados na *web*. Nesse processo de implantação do perímetro de segurança, é importante ressaltar que apenas *software* livre foi utilizado. Isso permitiu uma economia financeira significativa para a prefeitura.

Palavras-Chave: Segurança, *Firewall*, FreeBSD, *proxy*, internet.

Capítulo 1

Introdução

A Internet e o sistema de telefonia são os principais meios de comunicação entre os diversos departamentos que existem na prefeitura de Itabuna. A partir do ano de 2007, a Internet passou a ser ainda mais utilizada neste orgão público. A Internet é um meio de comunicação que vem auxiliando no desenvolvimento de tecnologias em diversas áreas do conhecimento. Hoje a Internet é uma realidade cada vez mais presente em nosso cotidiano. Seja nas universidades, empresas privadas ou públicas, em nossas casas, o fato é que esse meio de comunicação tornou-se imprescindível no mundo globalizado do qual fazemos parte. A busca rápida por informações e o rompimento de limites geográficos para o compartilhamento do conhecimento, são apenas alguns pontos proporcionados por ela. De acordo com (MONTEIRO, 2008), com o desenvolvimento da Internet e do seu acesso de forma "gratuita", além do crescimento do comércio eletrônico, cada vez mais temos usuários conectados à grande rede, bem como empresas querendo fazer seu *marketing* e negociar através deste novo mercado. Junto com este crescimento, também surgiu a espionagem dos *hackers* e *crackers*, além dos mais diversos problemas relacionados com segurança computacional.

A Prefeitura de Itabuna precisa disponibilizar acesso *on-line* aos seus funcionários, por conta de diversos sistemas que existem na Internet que auxiliam no trabalho das suas funções. Entre esses sistemas, podemos destacar: sistema de contra-cheque, consignação bancária, emissão de notas fiscais eletrônicas, entre outros. Na prefeitura existe uma política de uso da Internet estabelecida pelo diretor do Departamento de Tecnologia da Informação. Os usuários não podem, por exemplo, acessar sites de relacionamento, *sites* de bate-papo e usar o MSN sem a devida justificativa.

Para que a rede interna (intranet) da prefeitura não sofra ataques externos, nasceu a necessidade de informar aos usuários dessa rede sobre um melhor uso da Internet e a implementação de um perímetro de segurança envolvendo controles de acessos.

A segurança de perímetro é um conjunto de procedimentos envolvendo tecnologias de rede para proteger uma determinada rede de computadores contra possíveis intrusos. A inexistência de uma segurança de perímetro na Prefeitura de Itabuna, pode ocasionar uma falha na segurança da rede interna por onde *hackers* podem explorar e causar danos a instituição.

1.1 Motivação

A navegação na Internet é cada vez mais necessária, principalmente em um ambiente corporativo de um orgão público. Mas é preciso também preocupar-se com a qualidade dessa navegação e com sua segurança. A ausência de uma política de segurança para o uso da Internet, pode comprometer o funcionamento dos computadores da rede interna e também ocasionar a perda ou roubo de informações importantes da prefeitura.

Percebeu-se que alguns usuários que utilizam a Internet da rede da prefeitura fazem *download* em excesso, outros assistem videos *on-line*, comprometendo a qualidade da navegação. Sendo assim, nasceu a necessidade da construção de um perímetro seguro para obter êxito na resolução desses problemas. Nesse perímetro de segurança, foi escolhido o sistema operacional Unix FreeBSD, devido a sua grande estabilidade, confiabilidade, centralização de gerência de seus arquivos e grande capacidade de *hardening*[1]

No FreeBSD serão implementados serviços importantes, para garantir a integridade e segurança de dados que são trafegados através da Internet na rede da Prefeitura Municipal de Itabuna. Ferramenta como o *software* livre Squid merece destaque na implementação do perímetro seguro. No momento o atual, o Squid é o que mostra ser mais eficiente. Além de possuir um esquema baseado em ACL (*Access Control List*) para controle de acesso, ele também permite criar um *cache* de páginas *web* visitadas, auxiliando assim "economia" de banda. Essa economia de banda de Internet, pretende ser adquirida, com o uso Squid e o do módulo Dummynet implementado no *kernel* do FreeBSD.

[1] *Hardening* - são procedimentos que tem a finalidade de eliminar configurações desnecessarias no servidor.

Na prefeitura, existem também sistemas de outras empresas. Essas empresas implementam o sistema e oferecem o suporte. Algumas dessas empresas realizam seu suporte remotamente, por estarem localizadas outras cidades ou estados, por isso deve-se fazer o monitoramento de todas as ações efetuadas. Também existe a necessidade de bloquear a conexão de rede para usuários desconhecidos que entram em alguns setores e conectam seus *notebooks* seja por cabeamento ou via *wireless*, para fazer uso de serviços da rede.

1.2 Objetivos

O objetivo geral deste trabalho é apresentar o processo de implementação de uma Segurança de Perímetro em um servidor utilizando o sistema operacional Unix FreeBSD. Esse servidor foi configurado como o *gateway* da rede, tendo a função de proteger e gerenciar os dados que são trafegados através da Internet na rede da Prefeitura Municipal de Itabuna. Entre os objetivos específicos, destacam-se:

- Implementar um *firewall* utilizando IPFW
- Implementação e configuração de um *proxy* transparente para controle de acesso de usuários à Internet.
- Gerar relatórios de usuário quanto ao acesso à Internet
- Implementar políticas de uso de programas de chat. Ex MSN.
- Implementar controle de banda utilizando Dummynet
- Controle de acesso à programas indevidos ex. UltraSurf
- Prover redirecionamento de conexões externas de NAT e PAT

Assim, o presente trabalho aborda a utilização do *software* Squid, a implementação do módulo Dummynet e utilização do *firewall* IPFW, visando a construção de um perímetro seguro sob a plataforma Unix FreeBSD. Com esse perímetro implementado, o administrador de rede da Prefeitura Municipal de Itabuna obteve maior controle de acesso à Internet por parte dos usuários, um incremento no nível de segurança e a garantia de uma melhor performance relativa ao consumo de banda de Internet.

1.3 Metodologia

Este trabalho de apresentação e avaliação foi feito em duas etapas. A primeira etapa, caracterizou-se pelo levantamento das necessidades do departamento de Tecnologia da Informação da prefeitura de Itabuna. Na segunda etapa, à medida que as ferramentas eram instaladas e configuradas, os testes eram feitos e o resultados colhidos. Foram utilizados na implementação desse perímetro de segurança apenas *software* livre. Todos os aplicativos foram testados e implementados, em paralelo com o desenvolvimento dos capítulos. As referências do trabalho e as *man pages*, auxiliaram na avaliação das ferramentas que foram escolhidas para o processo de implantação do perímetro de segurança.

1.4 Organização do Trabalho

No Capítulo 2, descreve-se a fundamentação teórica sobre a plataforma de sistema operacional no qual foi construído o perímetro de segurança do presente trabalho e também é abordado o processo de instalação do mesmo e seus ajustes.

No Capítulo 3, são apresentadas as configurações relativas ao perímetro de rede. Essa fase é caracterizada por etapas como: recompilação de *kernel*, configuração do *firewall*, implementação do controle de banda e do controle de acesso na camada 2 do modelo OSI.

No Capítulo 4, inicialmente é apresentado um conceito sobre o *software* Squid utilizado para fazer o *proxy*. É abordado também o processo de instalação e configuração do Squid. A instalação de uma ferramenta para geração de relatórios de acesso à internet também é vista nesse capítulo.

No Capítulo 5, é abordada de que maneira é feita o redirecionamento de conexões externas para intranet, utilizando configurações de NAT e PAT no servidor.

No capítulo 6, trata-se da etapa das configurações de *hardening* e *tuning* do servidor.

No capítulo 7, trata-se da apresentação dos resultados obtidos.

Por fim, no capítulo 8, temos a conclusão do trabalho e as perspectivas de trabalhos futuros.

Capítulo 2

Introdução ao FreeBSD

FreeBSD[1] é um avançado sistema operacional para x86 compatível (incluindo Pentium ® e Athlon ™), amd64 compatível (incluindo Opteron ™, Athlon ™ 64, e EM64T), UltraSPARC , IA-64, PC-98 e ARM. É derivado do BSD, a versão do UNIX desenvolvido na Universidade da Califórnia, Berkeley. É desenvolvido e mantido por uma grande equipe de pessoas. Outras plataformas estão em vários estágios de desenvolvimento. De acordo com (TFBSD, 2010), o FreeBSD é amplamente utilizado por empresas, provedores de serviço de Internet, pesquisadores, profissionais de informática, estudantes e usuários domésticos no mundo todo, para trabalho, educação e recreação.
O FreeBSD é considerado um sistema operacional bastante confiável para aplicações que exigem grande poder de processamento. Devido a sua grande estabilidade, o nível de sua manutenção corretiva é baixa. Para se ter uma idéia, na Copa do Mundo de futebol realizada em 2006, de acordo com (FUG-BR, 2006), oito dos dez patrocinadores do evento, utilizavam FreeBSD em seus servidores *web*. Isso pode ser observado na Tabela 2.1

[1] Vide http://www.freebsd.org

Tabela 2.1: Sistemas Operacionais usados nos servidores de grandes empresas patrocinadoras na copa do mundo em 2006 - Fonte:(FUG-BR, 2006)

Patrocinador Copa do Mundo	Sistemas Operacionais
FIFA *World Cup*	FreeBSD
Yahoo!	FreeBSD
Phillips	FreeBSD
Toshiba	Solaris 9/ FreeBSD
McDonalds	Solaris 8/ FreeBSD
Cola-Cola Company	Linux / IBM Appliances
Adidas	Linux / Windows 2003
Budweiser	Linux/ Windows 2000
Avaya	Solaris 8 / Windows 2000 / FreeBSD
Continental	Linux / IBM
Deutsche Telekom	Solaris 8 / FreeBSD
Fly Emirates	Windows 2003
Fuji Film	Solaris 8
Gillette	Windows 2000
Matercard	Solaris 8 / FreeBSD / Windows 2000

FreeBSD x Linux

Exitem algumas diferenças entre o FreeBSD e o Linux que vale ressaltar. Isto é necessário, pois ambos podem ser utilizados para atuar como um servidor de perímetro seguro em uma rede de computadores.

Licenças: Um aspecto importante a ser levado em consideração, é o tipo de licença que os os sistemas Linux e FreeBSD utilizam. O Linux está licenciado sob GPL (*General Public License*), enquanto o FreeBSD segue a licença BSD (*Berkeley Software Distribution*). A licença BSD é considerada um tipo de licença "mais permissiva", ou seja, possui poucas restrições quando comparada a licença GPL utilizada pelo linux. Qualquer software que esteja sob a licença BSD pode ser modificado por um usuário ou uma empresa proprietária sem que ela seja forçada a disponibilizar o código fonte para o público. Um exemplo disso, é o *kernel* Darwin utilizado no sistema operacional Mac OS da empresa Apple.

Suporte de Hardware: O sistema FreeBSD apresenta suporte menor de *drivers* de fornecedores de *hardware* quando comparado ao Linux. O linux nesse quesito leva vantagem, pois muitos usuários utilizam esse sistema operacional em seus computadores pessoais. Não é impossível utilizar o FreeBSD

em computadores pessoais, mas ele é um sistema operacional cujo objetivo é atuar como servidor. O lema do FreeBSD é "*The Power To Server*".

Desenvolvimento dos Sistemas: O desenvolvimento do *kernel* do Linux é gerenciado basicamente por Linus Torvalds (criador do Linux). No FreeBSD, o *kernel* é gerenciado por uma equipe central que participa ativamente das decisões e mantem um histórico de desenvolvimento do projeto. Enquanto o projeto do Linux visa principalmente o *kernel*, o projeto do FreeBSD mantém o sistema operacional completo.

Semelhança com UNIX: O linux foi criado baseado em um sistema operacional chamado Minix[2], que por sua vez, era um sistema derivado do UNIX. Já o FreeBSD, foi originado diretamente do tradicional UNIX, apresentando assim maior semelhança com o mesmo.

Base do Sistema: A afirmação que o "Linux é o *kernel*" é bastante utilizada. Porém, o *kernel* torna-se inútil sem qualquer aplicativo que seja útil. Os programas que não fazem parte do *kernel*, são componentes de uma distribuição. Existem diversas distribuições Linux. Sendo assim, o Linux é um conjunto de sistemas menores que se acoplam para formar um todo. No FreeBSD, por outro lado, existe um sistema base que contém diversas ferramentas. A biblioteca "libc", por exemplo, é uma parte da base do sistema. Portanto, como essas "partes" são todas tratadas como um sistema base, todos elas são desenvolvidas e empacotadas juntas.

Instalação de Programas: O FreeBSD instala seus programas através do sistema de *Ports*[3]. Sendo assim, é mais fácil para o usuário instalar um programa a partir do seu código fonte ao invés de pacotes binários pré-compilados. Uma das vantagens deste mecanismo de instalação, é que existe a possibilidade de compilar todo o sistema com otimizações para alguns hardware do computador. Vale lembrar que existe uma distribuição Linux chamada Gentoo, que utiliza um sistema de instalação de programas semelhantes ao FreeBSD, cujo nome é Portage. A desvantagem desse sistema de instalação de programas do FreeBSD, é que em caso de instalação de um programa muito grande, o processo pode durar horas e até dias.

Ainda segundo (SOUSA, 2008), o FreeBSD possui operações de I/O (*Input/Output*) no disco rígido melhor que o do Linux com o uso do UFS2 (Unix

[2]É sistema operacional baseado no Unix e desenvolvido por Andy Tannenbaum.

[3]É uma árvore de banco de dados que contém mais de 1000 programas divididos por categorias

File System com Soft Update). Assim, tem-se um melhor desempenho de aplicações de banco de dados, de vídeo o que façam uso de muita leitura e gravação, o que faz com que muitas empresas mundialmente reconhecidas (SunOS/Solaris, System V Release 4, HP-UX, Tru64 UNIX, etc) utilizem o sistema operacional FreeBSD em seus servidores.

2.1 Instalação do sistema operacional FreeBSD 7.0

A instalação do sistema operacional FreBSD 7.0 é feita através de um único CD de tamanho 640 MegaBytes e em modo texto. Logo no inicio do processo, o programa de instalação do FreeBSD (`sysinstall`) aparece na tela, dando boas vindas ao usuário e solicitando o tipo de instalação a ser feito. Isso é observado na figura 2.1

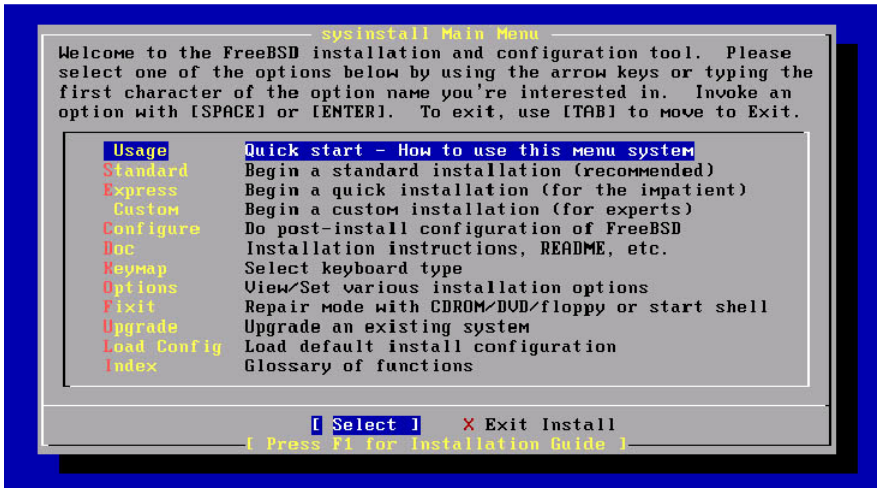

Figura 2.1: Tela do `sysinstall`

O processo de instalação do FreeBSD é um processo muito rápido e simples, na maioria da vezes. Basicamente, esse processo é constituído de quatro etapas diferentes, realizadas em sequência, nas quais podemos citar:

- Criar uma partição para a instalação do sistema operacional
- Organizar e definir o espaço para os sistemas de arquivos que serão criados
- Escolher os itens (programas, etc) da distribuição a serem instalados

- Escolher a mídia utilizada para a instalação (disquetes, CD-ROM, FTP, etc)

A opção de instalação escolhida para o presente trabalho, foi a *standard*. Utilizando a opção *standard*, as etapas do processo de instalação são comentadas e mostradas através de caixas de diálogo. Essa opção, permite maior simplicidade na configuração de diretivas do sistema, tais como criação de usuários, configuração de rede, entre outras.

2.2 Ajustando configurações básicas pós-instalação (rede, inicialização, atualização do PORTS)

Para configurar as diretivas de rede no FreeBSD, deve-se editar o arquivo "`rc.conf`" que está localizado no diretório "`/etc`". Nesse arquivo podemos configurar os endereços IP das placas de rede. Ao contrário do Linux, onde as placas são detectadas como eth0, eth1, eth2, e etc, no FreeBSD elas são detectadas de acordo com o *chipset* das mesmas. Por exemplo, "`rl0`" identifica uma placa de rede da marca Realtek, o "`xl0`" identifica uma placa de rede da marca 3Com e o "`lo0`" indentifica o dispositivo de *loopback*.

Foram utilizados duas interfaces de rede na implementação do presente trabalho. A interface "`rl0`", é placa que recebeu um endereço IP válido de número 200.151.208.229 e a outra interface "`xl0`", recebeu um endereço IP da rede interna de número 173.173.1.3. Para o *gateway*, utilizamos o endereço IP 200.151.208.225. Na configuração do DNS, utilizamos o arquivo "`resolv.conf`", localizado no diretório "`/etc`". O endereço IP utilizado para o DNS, foi 200.223.0.84. Na figura 2.2, observamos a saída do comando `ifconfig`.

Após a instalação do sistema, foi desabilitada a inicialização do Sendmail (servidor de *emails*), uma vez que, em nosso perímetro de segurança, ele não foi utilizado. Então, no arquivo "`/etc/rc.conf`", foi colocada a linha `Sendmail_enable="NONE"`. Esse procedimento, portanto, auxilia na economia de memória RAM pelo servidor, pois será menos um processo em execução.

O *Ports Collections* do FreeBSD é uma árvore de banco de dados que contém mais de 1000 programas. Após a instalação do sistema operacional, é necessário realizar alguns procedimentos para que ela seja atualizada. A árvore de banco de dados de programas do FreeBSD precisa ser atualizada, pois na instalação padrão do sistema operacional, podem existir versões antigas de determinados programas. Existem maneiras diferentes de realizar esse procedimento de atualização. No pre-

```
rl0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=9b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM>
        ether 00:1c:c0:63:37:a5
        inet 200.151.208.229 netmask 0xffffff80 broadcast 200.151.208.255
        media: Ethernet autoselect (100baseTX <full-duplex>)
        status: active
xl0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=8<VLAN_MTU>
        ether 00:08:54:a5:f8:c5
        inet 173.173.1.3 netmask 0xffffff00 broadcast 173.173.255.255
        media: Ethernet autoselect (100baseTX <full-duplex>)
        status: active
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> metric 0 mtu 16384
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x5
        inet6 ::1 prefixlen 128
        inet 127.0.0.1 netmask 0xff000000
```

Figura 2.2: Configurações de rede

sente trabalho, isto foi feito através de uma ferramenta, chamada `Portsnap` e pelo programa `Portupgrade`. O `Portsnap` não atualiza o *Ports Collections*, apenas a árvore de diretórios. O comando executado na figura 2.3, realiza a atualização da árvore de diretórios.

```
# portsnap fetch extract
```

Figura 2.3: Comando inicial para atualizar a árvore de diretórios do *Ports Collections*

Se for necessário realizar uma nova atualização da árvore de diretórios, basta apenas executar o comando `portsnap` com alguns parâmetros. Isso é mostrado na figura 2.4

```
# portsnap fetch update
```

Figura 2.4: Comando para atualizar a árvore de diretórios do *Ports Collections* pela segunda vez

Para fazer a instalação do programa Portupgrade, foi necessário entrar no diretório `/usr/ports/ports-mgmt/portupgrade` e executar o comando mostrado

na figura 2.5

```
# cd /usr/ports/ports-mgmt/portupgrade && make install clean
```

Figura 2.5: Instalação do programa `Portupgrade`

Em seguida, foi executado o comando "`portupgrade -a`" para o inicio do processo de atualização do *Ports Collections*.

Após a instalação do sistema operacional e atualização de suas ferramentas básicas, o FreeBSD ficou apto para receber modificações na configurações de seu *kernel* e a instalação de outros *software* necessários para torna-ló um servidor de segurança da intranet da Prefeitura de Itabuna.

Capítulo 3

Configuração dos Perímetros de Rede

3.1 Recompilação do *kernel* com suporte a *Proxy* Transparente, NAT e Controle de Banda

O processo de recompilação usado no presente trabalho, foi baseado no *handbook* do FreeBSD (OPENHB, 2008). Para prosseguir com a recompilação do *kernel*, as seguintes etapas foram seguidas:

- cópia do arquivo principal, onde são configuradas as opções de compilação do *kernel* na instalação padrão;
- em seguida, edição do novo *kernel* que foi recompilado;

Os comando que caracterizam esses procedimentos são evidenciados na Figura 3.1

```
#cp /usr/src/sys/i386/conf/GENERIC KERNEL-PMI
#ee /usr/src/sys/i386/conf/KERNEL-PMI
```

Figura 3.1: Cópia e edição do arquivo de compilação do kernel

A opção “ident GENERIC”, foi substituida por “ident KERNEL-PMI”. O nome do arquivo do *kernel* do FreeBSD , “KERNEL-PMI” serve para identificar que

o *kernel* que foi compilado é da prefeitura municipal de Itabuna (PMI). Depois foram adicionadas as opções, de acordo com a figura 3.2

```
options    MROUTING
options    IPFIREWALL
options    IPFIREWALL_VERBOSE
options    IPFIREWALL_FORWARD
options    IPFIREWALL_VERBOSE_LIMIT=1000
options    IPFIREWALL_DEFAULT_TO_ACCEPT
options    IPDIVERT
options    IPFILTER
options    IPFILTER_LOG
options    IPSTEALTH
options    TCPDEBUG
options    ACCEPT_FILTER_DATA
options    ACCEPT_FILTER_HTTP
options    DUMMYNET
```

Figura 3.2: Opções usadas na compilação do kernel

Dentre as opções de compilação, observadas na Figura 3.2, podemos destacar as principais:

Item options IPFIREWALL: opção responsável por habilitar o serviço principal de *firewall* do FreeBSD;

Item options IPFIREWALL_VERBOSE: opção responsável por imprimir informações sobre pacotes descartados;

Item options IPFIREWALL_FORWARD: opção responsável por habilitar o suporte ao uso do *proxy* transparente;

Item options IPFIREWALL_VERBOSE_LIMIT=1000: opção responsável pelo limite de ocorrências geradas pelo syslogd;

Item options IPFIREWALL_DEFAULT_TO_ACCEPT: com essa opção, todos os pacotes serão aceitos por padrão;

Item options IPDIVERT: opção responsável por permitir o uso dos recursos de NAT (*Network Address Translate*) na rede;

Item options DUMMYNET: opção que possibilita que o controle da largura de banda de Internet seja efetuado utilizando o IPFW;

Depois de habilitar as opções no *kernel*, o arquivo foi salvo e o comando "`config KERNEL-PMI`" foi utilizado para gerar o código-fonte para a compilação. Após a execução desse comando, a mensagem *"Kernel build directory is ../../compile/KERNEL-PMI Don't forget to do a 'make cleandepend; make depend'"* foi retornada. Em seguida, foram necessários executadas alguns comandos de compilação. Esses comandos são observados na Figura 3.3 Após o termino do

```
# cd ../../compile/KERNEL-PMI
# make depend
# make
# make install
```

Figura 3.3: Comandos para compilação do *kernel*

processo de compilação, bastou reiniciar o sistema operacional e o novo *kernel* já estava atualizado e pronto para utilização.

3.2 *Firewall*

Com a popularização e crescimento da *Internet*, novas ferramentas tecnológicas são cada vez mais frequentes no cotidiano das pessoas. O *e-mail* hoje é algo bem comum. Serviços de comércio eletrônico são cada vez mais utilizados para realizar compras, efetuar pagamentos, etc. Hoje em dia, já existem diversas universidades que ministram cursos através desse rápido canal de comunicação, que é a *Internet*. Por outro lado, esse canal de comunicação deve assegurar confiabilidade para seus usuários.

Apesar de proporcionar todos os benefício citados, existem diversos riscos envolvidos com a utilização da *Internet*. Podemos citar alguns: ataques de *hackers*, *e-mails* com anúncios falsos, *malwares* que capturam senhas, entre outros perigos. Nasce daí a necessidade de implementar mecanismos de segurança. Um dos principais dispositivos de segurança de uma rede de computadores é o *firewall*.

Existem diversos conceitos sobre o que é um *firewall*, podemos citar alguns, para um melhor entendimento sobre esse mecanismo de segurança que será implementado no servidor responsável pela segurança da rede.

> "*Firewalls* são dispositivos de *hardware* e *software* que servem para criar uma barreira de proteção lógica entre sua rede interna e externa com regras para determinar o que pode ou não passar por eles.

> Dito em outras palavras, os *firewalls* compõem o portão de entrada da empresa com um porteiro inspecionando tudo que entra e sai, e tomando as devidas providências sobre o tipo de tráfego".(MONTEIRO, 2008)

> "*Firewall* é uma ferramenta de *software* ou *hardware* situada entre duas redes (uma interna e outra externa), responsável por filtrar pacotes, evitando o acesso externo a determinados serviços."(UCHÔA, 2005)

> "*Firewall* é um programa que como objetivo proteger a máquina contra acessos indesejados, tráfego indesejado, proteger serviços que estejam rodando na máquina e bloquear a passagem de coisas que você não deseja receber (como conexões vindas da Internet para sua segura rede local, evitando acesso aos dados corporativos de uma empresa ou a seus dados pessoais). "(SILVA, 2009)

A topologia básica de um *firewall* pode ser observada na figura 3.4. Nessa figura, existem duas redes externas (secretaria de educação e saúde) que são conectadas ao servidor de perímetro (FreeBSD) através de uma conexão *wireless* de alta velocidade.

Um *firewall* pode ser projetado de acordo com a necessidade do que se deseja proteger. Mas em resumo, um *firewall* sempre obedece as seguintes políticas de acesso aberta e fechada.

Um *firewall* que adota a política "aberta" em sua arquitetura de implementação, permite que todos pacotes trafeguem pelo servidor, bloqueando apenas aqueles que são especificados para serem descartados. Na política "fechada"de implementação, o *firewall* nega por padrão todos os pacotes que chegam e saem do servidor. A liberação é feita de acordo com a necessidade de determinado serviço. A implementação de um *firewall* com a política "fechada", torna-o mais seguro em comparação à um *firewall* com política "aberta". Um *firewall* de política "fechada" requer uma elaboração mais complexa e consequentemente mais demorada. Em muitos casos, faz-se necessário uso de outras ferramentas *(sniffers)* para monitorar conexões e configurar as regras de filtragem.

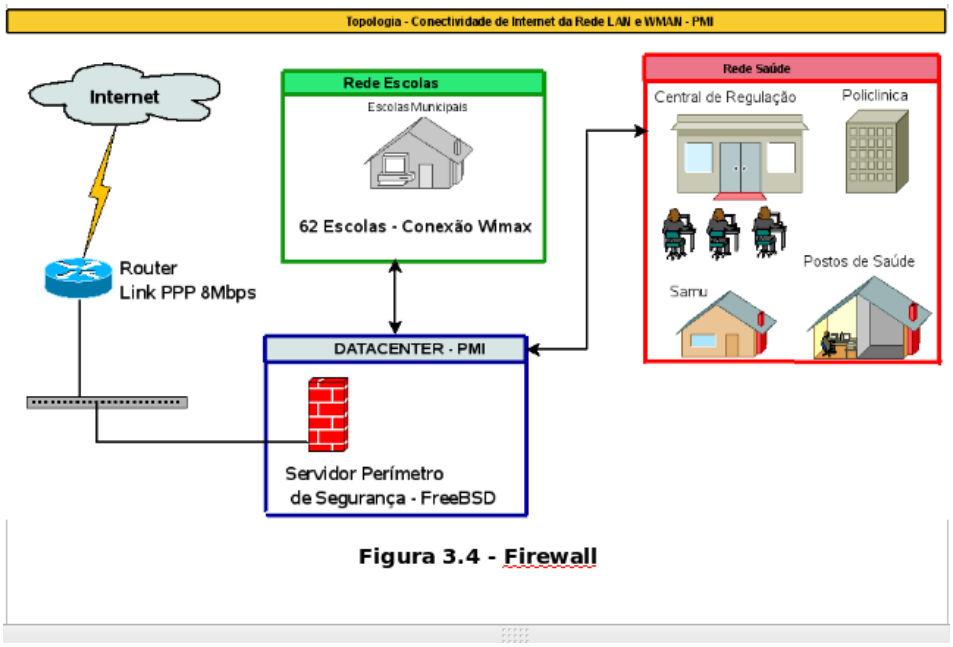

Figura 3.4: Topologia de um *Firewall*

3.2.1 Introdução e configuração do IPFW

O IPFW *(ipfirewall)* é o *firewall* padrão do FreeBSD. Ele é baseado na filtragem de pacotes e implementado à nível de *kernel*. De acordo com (TFBSD2, 2010), o IPFW é um *software* de *firewall* mantido por membros da equipe de voluntários do FreeBSD. IPFW suporta as duas versões do protocolo IP: IPv4[1] e IPv6[2] As regras que podem ser usadas do IPFW, estão disponíveis no arquivos "`/etc/rc.firewall`" e "`/etc/rc.firewall6`". Após a instalação básica do sistema operacional, o IPFW por padrão, inicialmente encontra-se desabilitado. Sendo assim, é necessária a recompilação do *kernel* com algumas opções vistas na secão 3.1, para que o filtro de pacotes possa ser utilizado. A partir da versão 4.x, do FreeBSD, o IPFW passou também a trabalhar gerenciando conexões por estado *(statefull e statless)*.

Utilizando o comando `ipfw`, qualquer regra pode ser adicionada através do console do sistema. O comando mostrado na figura 3.5, de acordo com (TRACANELLI, 2002), lista todas as regras ativas do *firewall* no momento, seguindo a

[1]IPV4 - É a versao 4 do protocolo IP.
[2]IPV6 - É a versao 6 do protocolo IP.

ordem dos números das regras. A ordem das regras também influi na forma como o IPFW se comporta.

```
#ipfw list
```

Figura 3.5: Listagem de regras ativas

Ao executar os comandos ilustrados na figura 3.6, segundo (TRACANELLI, 2002), serão listados o número de vezes que um pacote passou (ou foi bloqueado) por uma regra, e o número de bytes que esse tráfego gerou. Os dois comandos vão apresentar as mesmas informações, dispostas da mesma maneira. A primeira coluna é o número da regra, em ordem como ela é verificada. A segunda coluna informa quantas vezes aquela regra coincidiu com um pacote, seguido do (terceira coluna) número de *bytes* que trafegaram pela regra, e finalmente a regra em sí. Existem algumas sintaxes curtas pra esses comandos. Por exemplo, "`ipfw s`" tem o mesmo efeito que "`ipfw show`"; "`ipfw l`" o mesmo que "`ipfw list`", etc.

```
# ipfw -a list
ou
# ipfw show
```

Figura 3.6: Características das regras ativas

De acordo com (TRACANELLI, 2002), no IPFW, grande parte das vezes, sempre que um pacote combinar com uma regra, o IPFW para de examinar as outras regras para aquele pacote, ou seja, a ordem como as regras são processadas no *firewall* do FreeBSD são do tipo *"first match wins"* onde, a primeira constatação do pacote evita que as outras sejam lidas. Por isso é extremamente importante estar atento a ordem (número) das regras. Sob circunstâncias especiais as outras regras podem ser processadas. Então, de forma geral, a sintaxe mais simples pro IPFW é ilustrado na figura 3.7. As explicações de uso para cada sintaxe representada na figura 3.7, foram baseadas em (TRACANELLI, 2002)

```
<comando> [<número da regra>] <ação> <protocolo> from <origem> to <destino>
```

Figura 3.7: Sintaxe de regra do ipfw

<comando> Os comandos mais importantes são "add" e "delete". Uma simples tradução seria o suficiente pra explicar que "add" adiciona uma regra e "delete" a

exclui.
<número da regra> Varia de 0 até 65535, e indica a ordem que essas regras serão processadas no *firewall*, portanto a última regra sempre define a política padrão do *firewall* no *kernel*. Mesmo se for utilizada uma política de *firewall* "aberta" (*OPEN*) definida no seu "/etc/rc.conf", a última regra vai sempre indicar a política do *kernel*. Isso é ótimo porque, como a ordem de busca das regras para de processar ao encontrarem uma regra que combina com o pacote, e se a penúltima regra é a de número 65000, definida pelo "rc.firewall" para permitir todo o tráfego da rede, então todo tráfego vai ser liberado, mesmo que a última regra (65535) negue todos os pacotes, já que essa regra nunca vai ser processada.

<ação> na sintaxe do comando ipfw pode ser uma das seguintes:

"allow", "pass", "permit", "accept" - Quando uma regra define essa ação, sempre que um pacote combinar com essa regra, será permitido seu roteamento pelo *firewall*, e o processamento das regras para esse pacote termina.

"deny", "drop" - Qualquer pacote que combinar com uma regra cuja ação seja "deny" ou "drop" são silenciosamente bloqueados pelo *firewall*, e o processamento das regras para esse pacote termina. Conforme a figura 3.8, temos um exemplo de regra que nega todos os pacotes vindos de qualquer origem e indo pra qualquer destino.

"reset" - Quando um pacote encontra uma regra com essa ação, o pacote é bloqueado, e o IPFW tenta enviar um sinal (flag) de TCP Reset (RST) para o endereço de origem do pacote. O processamento das regras para esse pacote termina. Como esse tipo de regra apenas se aplica para pacotes TCP, o protocolo especificado na regra deve ser TCP, para que apenas tais pacotes sejam processados por essa regra, e não todos (proto "all") os protocolos de pacotes IP. Vale notar que o uso do "reset" pode ser muito interessante pra enganar ferramentas que escaneiam as redes (*network scanners*), já que normalmente podem detectar um serviço por trás de uma porta filtrada, mesmo que ele esteja por trás de um *firewall* de bloqueio. Por outro lado, se alguém estiver praticando um ataque do tipo *network flood*[3] em uma porta específica a qual o IPFW está configurado para responder com pacotes RST, isso duplicaria o uso da sua banda de rede. Uma solução simples é bloquear,

[3] *Network flood* - É um tipo de ataque à redes de computadores, no qual diversas requisições são enviadas por um ou mais computadores para um único computador alvo, acarretando queda de performance nos seus serviços.

com uma regra prévia o endereço da máquina que está agindo dessa forma, endereço esse obtido de forma dinâmica por monitoramento.

```
add 1100 deny all from any to any
```

Figura 3.8: Sintaxe de regra com negação de pacotes

A regra de exemplo, mostrada na figura 3.9, tem como propósito bloquear todas as conexões TCP vindas de qualquer destino, indo para qualquer origem, e responderia com um pacote RST para cada uma delas.

```
add 700 reset tcp from any to any
```

Figura 3.9: Sintaxe de regra utilizando o "reset"

"count" - Todos os pacotes que combinarem com uma regra cuja ação seja "count", determinará que o IPFW incremente o contagem de pacotes, ou seja, a saída de "ipfw show" indicará mais uma ocorrência de pacotes nessa regra. O processamento das regras do *firewall* continuam a buscar por outras regras que combinem com os pacotes. Observa-se na figura 3.10, que a primeira regra incrementa o número de vezes que um pacote passou pelo *firewall*, vindo de qualquer lugar e indo pra onde quer que seja. Já a segunda regra conta quantos pacotes, dentre todos, estariam sendo enviados da rede 200.230.50.0/24 (origem) pra rede 173.173.0.0/16 (destino).

```
add 800 count all from any to any
add 800 count all from 200.230.50.0/24 to 173.173.0.0/16
```

Figura 3.10: Sintaxe de regra utilizando o "count"

"skipto <número da regra>" - Todos os pacotes que combinem com uma regra cuja ação seja "skipto <número da regra>" vão fazer com que o IPFW continue processando esse pacote e buscando ocorrência nas regras que sejam de ordem maior ou igual ao <número da regra> indicado pela ação. Um exemplo de aplicação do uso do "skipto" é mostrado na figura 3.11. Essa regra faria com que todo o tráfego vindo da rede 192.168.1.0/24 e indo pra qualquer destino seja processado pelas regras a partir da regra de número 1800.

```
add 1400 skipto 1800 all from 192.168.1.0/24 to any
```

Figura 3.11: Sintaxe de regra utilizando o "skipto"

"reject" - Essa ação é pouco utilizada atualmente. Quando um pacote combina com uma regra cuja ação seja "reject", então o IPFW bloqueia esse pacote e responde com uma mensagem ICMP do tipo "*host unreachable*", dando a impressão que a máquina se encontra fora da rede. Essa é uma forma não silenciosa de negar o tráfego pelo *firewall*, contudo, assim como a ação "reset", essa ação também aumenta o uso da banda de rede.

<protocolo> "protocolo" na sintaxe básica de uso do IPFW, é o protocolo de comunicação que você quer que aquela regra combine. Definições de protocolos do tipo "ip" ou "all" são especificações gerais que englobam todos os protocolos. Os protocolos mais comuns são ICMP, UDP e TCP, mas a relação de protocolos com os quais o IPFW trabalha é enorme. Na verdade são todos os protocolos possíveis de uma rede.

<origem> e <destino> O endereço de origem e de destino de um pacote tem o mesmo formato em uma regra de *firewall*. Eles podem ser um nome, definido no "`/etc/hosts`" e resolvido por DNS, pode ser um endereço IP, um endereço de rede com máscara de rede (bitmask/netmask) e, ainda podem definir uma ou várias portas, se o protocolo for TCP ou UDP. Usar nomes ou endereços IP é indiferente, basta atentar ao fato que a resolução de nomes via DNS pode mudar sem conhecimento prévio do administrador.

```
add 1000 allow all from minhamaquina to outramaquina
add 1100 deny all from 10.0.0.5 to any
```

Figura 3.12: Sintaxe de regra origem/destino

De acordo com a figura 3.12, a primeira regra permite todo o tráfego vindo da "minhamaquina" para a "outramaquina", e a segunda regra nega toda conexão da máquina `10.0.0.5` para qualquer outra estação. Sempre que um pacote coincidir com uma regra do *firewall*, o processamento para aquele pacote termina, e o pacote é permitido ou negado, de acordo com a ação definida pela regra. Esse é um exemplo muito simples de um filtro baseado em estações, ou "*host-based filtering*", que filtra de acordo com o destino ou origem do pacote.

Um *firewall* por filtragem de redes funciona de forma similar, distinguindo-se apenas a notação de redes, onde é necessário definir a máscara de subrede (*netmask*) ou ainda o *bitmask*. Isso é demonstrado na figura 3.14.

```
add 2000 allow all from 192.168.0.0/16 to any
add 2100 deny all from any to 10.0.0.0:255.0.0.0
```

Figura 3.13: Sintaxe de regra filtragem por redes

A primeira regra permite todo o tráfego de pacotes vindo da rede cujo conjunto de endereços IP começa em `192.168.0.0` até `192.168.255.255`. A regra usa uma máscara (bitmask) pra indicar a abrangência do endereçamento da rede. A máscara em bits também conhecida como bitmask especifica quantos bits do endereço de rede (192.168.0.0) devem permanecer o mesmo pra cada pacote. Nesse caso, os primeiros 16 bits dos 32 bits do endereço vão continuar os mesmos. Como os primeiros 16 bits são os primeiros dois octetos do endereçamento da rede, então todos os endereços cuja origem seja a indicada nos dois primeiros octetos (ou nos 16 bits iniciais), nesse caso a rede 192.168, serão permitidos por essa regra. A segunda regra tem uma função similar, utilizando as máscaras de rede. A máscara de rede (*netmask*) indica quantos bits do endereço de rede devem ser monitorados pela regra do *firewall*. No exemplo acima, a segunda regra (número 2100) tem a máscara de rede "255.0.0.0". O primeiro octeto dessa regra é definido como "bits altos", ou seja, os primeiros 8 bits são "bits altos", o que indica pro *firewall* que apenas os pacotes cujos primeiros 8 bits do endereço da rede devem ser filtrados. Como os primeiros 8 bits da rede é igual a 10, então todos os pacotes cujo endereço de destino seja 10 no primeiro octeto (ou seja, toda a faixa de 10.0.0.0 até 10.255.255.255) vão combinar com essa regra, e então serão bloqueados, como indicado na ação da regra (*deny*). O *firewall* do FreeBSD oferece ainda a possibilidade de inverter a expressão apresentada na regra com o comando "not". Na figura 3.14, por exemplo, observa-se uma regra para que o IPFW efetue o bloqueio de todos os pacotes que não sejam da estação `192.168.0.3`.

```
add 1000 deny all from not 192.168.0.3
```

Figura 3.14: Sintaxe de regra usando o "not"

As configurações de regras que foram realizadas no *firewall* do perímetro de segurança do presente trabalho, serão abordadas à seguir. No início do *script*, foram definidas as variáveis que contém, o caminho do executável do IPFW, o valor

do alcance da intranet, a variável “nosquid” é utilizada para assegurar que determinadas faixas de endereços IP não passem pelo Squid. Essa técnica foi utilizada, por exemplo, para acesso a sistemas da Caixa Econômica Federal. A variável “portas”, foi utilizada para liberar o acesso para alguns seviços utilizados pela intranet. A variável “nat” representa a configuração da interface de rede externa do servidor. E por fim, a variável “type” representa se é utilizado IPv4 ou IPv6. As definições dessas variáveis são ilustradas na figura 3.15.

```
# DEFININDO VARIAVEIS
fwcmd="/sbin/ipfw"
rede_local="173.173.0.0/16"
nosquid="{ 67.15.2.43/32 or 200.201.0.0/16 or 161.148.185.46/32 or
200.209.70.130/32 or 207.44.142.6/32 or 200.223.0.0/16 or 200.187.6.0/16 }"
portas="20,21,22,25,53,80,443,3306,2631" # Portas de entrada autorizadas
nat="rl0"  # Interface de Internet
type="ip4" # ip4 ou ip6
```

Figura 3.15: Definição de variáveis no script do IPFW

De acordo com a figura 3.16, observa-se que as regras de números 00001 e 00002, tem por objetivos liberar todo o tráfego de pacotes da intranet com destino a porta 3128 e também liberar o tráfego da intranet com destino a porta 80 para determinadas faixas de endereços IP. Na regra de número 00003, toda conexão de origem da intranet para qualquer destino, é redirecionada para a porta 3128. Sendo assim, garante a passagem do tráfego de navegação da intranet pelo *software* Squid. A regra de número 00010, realiza o compartilhamento da conexão da Internet para acesso da intranet.

A figura 3.17, apresenta um outro trecho do código do *script* do *firewal*, onde temos a implementação do controle de acesso para uso do programa de bate-papo (MSN). Foi utilizada uma estrutura de repetição (*for*) que lê um arquivo de texto. Esse arquivo de texto contém todos os endereços IP autorizados para acesso. Dentro dessa estrutura de repetição, foram adicionadas regras 00029 e 00030 para liberar o trafego com a utilização da porta 1863 e da faixa da rede IP `207.46.0.0/16`. Essa porta é utilizada pelos servidores do msn. O mesmo acontece com essa rede `207.46.0.0/16`. Portanto, os computadores da intranet,que não está cadastrado no arquivo “acesso-msn”, tem seu tráfego bloqueado para uso do msn. Esse bloqueio é observado através das regras 00031 e 00032.

```
# Permite que o trafego originado da rede local com destino a porta
3128 (porta do Squid) seja aceito
${fwcmd} add 00001 allow tcp from ${rede_local} to me dst-port 3128

# Regra abaixo para permitir que alguns sistemas não passem pelo
proxy Squid
#${fwcmd} add 00002 allow tcp from ${rede_local} to ${nosquid}
dst-port 80

# Redirecionando o trafego p/ o Squid !
${fwcmd} add 00003 fwd 127.0.0.1,3128 TCP from ${rede_local} to
any  80 via rl0

# Nega os demais pacotes com destino ao servidor na porta 3128
${fwcmd} add 00004 deny log logamount 500 TCP from any to me
dst-port 3128

# Dando inicio ao natiamento com IPDIVERT
${fwcmd} add 00010 divert natd all from any to any via ${nat}
```

Figura 3.16: Regras do Squid e NAT no *script* do IPFW

```
# Politica de acesso/bloqueio MSN
# O codigo de erro que aparece no msn quando esta bloqueado,
geralmente é esse: 80072efd

for IP in $(awk '{ print $1 }' /usr/local/etc/Squid/acesso-msn); do
   PREFIX=$(echo ${IP} | cut -f2 -d/)
   if [ -z "${PREFIX}" ]; then
     IP=${IP}/16
   fi
   ${fwcmd} add 00029 allow tcp from any 1863 to $IP
   ${fwcmd} add 00030 allow tcp from 207.46.0.0/16 to $IP
done
# Bloqueando p acesso p/ o restante da rede!
${fwcmd} add 00031 deny tcp from any 1863 to ${rede_local}
${fwcmd} add 00032 deny tcp from 207.46.0.0/16 to ${rede_local}
```

Figura 3.17: Implementação do controle de acesso do programa MSN

No trecho do código do *script*, observado na figura 3.18, o tráfego é liberado das portas autorizadas para a intranet e também, esse tráfego é liberado da intranet para qualquer destino, desde que usem aquelas portas autorizadas para acesso. Isso

está configurados nas regras 02100 e 02110.
Na regra 65535, negamos o tráfego de pacotes de qualquer destino para origem e vice-versa. Isso é necessário, pois estamos utilizando *firewall* com a diretiva "IPFIREWALL_DEFAULT_TO_ACCEPT". Ou seja, um *firewall* com arquitetura "*open*". Outras regras do IPFW que tratam do controle de acesso da camada 2 do modelo OSI, do controle de banda, de redirecionamento com PAT, entre outras, serão vistas nas seçoes adiante.

```
# Permitir conexoes para os servicos de SSH, SMTP, FTP, HTTPS, DNS, CEF
${fwcmd} add 02100 allow tcp from any ${portas} to ${rede_local}
${fwcmd} add 02110 allow tcp from ${rede_local} to any ${portas}

# Abrindo seu servidor para a Rede!
${fwcmd} add 02200 allow ip from any to any

#bloqueia qualquer trafego - regra padrao do IPFIREWALL
${fwcmd} add 65535 deny ip from any to any

echo "IPFW CARREGADO !"
```

Figura 3.18: Liberação de portas especificas para uso da intranet

3.3 Configurando controle de banda utilizando Dummynet

Segundo (DUMMY, 2002), o Dummynet é uma ferramenta originalmente concebida para testar protocolos de rede e, desde então usado para uma variedade de aplicações, incluindo gerenciamento de banda. De acordo com (GEIB, 2004b), ele funciona interceptando os pacotes e passando-os por um ou mais *pipes* ou *queues*, que podem efetuar a limitação de banda, perdas de pacotes, retardos de propagação etc. Os *pipes* são canais com largura de banda fixa, enquanto as *queues* representam filas de pacotes, associadas a um peso (*weight*). As *queues* compartilham, em proporção ao peso, a largura de banda dos *pipes* aos quais estão associadas.

Os comandos para gerenciar as regras de *pipes* e *queues* são iguais as das regras comuns do IPFW. A estrutura do comando de gerenciamento dessas regras, podem ser observados na figura 3.19.

```
ipfw {pipe | queue} {delete | list | show} número
```

Figura 3.19: Estrutura de comandos para regras de *pipe* e queue

As opções do IPFW específicas para o módulo Dummynet estão descritas a seguir, de acordo com (GEIB, 2004b)

Para definir um *pipe* utiliza-se o seguinte comando / regra:

- ipfw *pipe* número config opções-configuração

Para a definição de uma queue utiliza-se o comando / regra:

- ipfw queue número config opções-configuração

As opções de configuração são as seguintes:

Opções de configuração específicas do *pipe*:

- bw banda - Define a largura de banda do *pipe*. A banda deve ser especificada em Kbit/s, Mbit/s, KByte/s ou MByte/s.
- *delay* tempo - Define o *delay* de propagação do *pipe*. O *delay* deverá ser especificado em milisegundos.

Opções de configuração específicas da queue:

- *pipe* número - Conecta a queue ao *pipe* especificado. Podem ser conectadas várias *queues* a um único *pipe*.
- weight peso - Especifica o peso daquela queue. O valor pode variar de 1 a 100.

Principais opções de configuração, comuns ao *pipe* e à queue:

- *buckets* tamanho-tabela - Especifica o tamanho da tabela usada para guardar as diversas *queues*. O valor pode variar de 16 a 65536, o padrão é 64. mask especificação-máscara - Define diferentes fluxos através da aplicação

da máscara especificada. Cada fluxo é enviado para uma *queue* ou *pipe* separado, criados dinamicamente. Cada *pipe* dinâmico terá a mesma largura de banda do *pipe* original, enquanto que cada *queue* dinâmica irá compartilhar com as demais dinâmicas a largura de banda do *pipe* ao qual está conectada a original. A especificação de máscara deve ser uma ou mais das seguintes: dst-ip máscara, src-ip máscara, dst-port máscara, src-port máscara, proto máscara ou all. O parâmetro all define que todos os bits em todos os campos são significantes.

- noerror - Não reporta o erro quando um pacote for perdido, por exemplo em uma simulação de perda de pacotes ou congestionamento.

- plr taxa-perda - Define a taxa de perda de pacotes. O valor deverá ser entre 0 e 1, com 0 significando nenhuma perda, e 1 significando 100% de perda.

- queue {slots | tamanhoKBytes} - Tamanho da fila, em slots ou KBytes.

Para utilizar o Dummynet, é necessário inserir a diretiva "options DUMMYNET" no arquivo de configuração do novo *kernel* (KERNEL-PMI) que será compilado. Após o sistema ter o módulo Dummynet carregado corretamente, foram configuradas regras no *script* de *firewall* (IPFW) para o controle de banda. Na figura 3.20, observa-se um trecho do controle de banda feito para o presente trabalho numa máquina da diretoria.

Observando a figura 3.20, conclui-se que a máquina da recepção do setor de TI, ficou limitada com uma taxa de *download* de 64 Kbit/s, configurada no *pipe* 1. A taxa de *upload* também limitada com com sua velocidade de *upload* de 64 Kbit/s, configurada pelo *pipe* 2.

3.4 Implementando controle de acesso na Camada 2

O controle de acesso feito na camada 2 do modelo OSI[4] do presente trabalho, foi realizada com os recursos do IPFW. Para tal, faz se necessário a desabilitação da passagem única de pacotes. A opção "`disable one_pass`"foi utilizada no inicio das regras do *script* do *firewall*.

[4]OSI (Open Systems Interconnection) - É um modelo de arquitetura que divide uma rede de computadores em 7 camadas que contribuem para seu funcionamento.

```
# Controle de BANDA by Richer Barros
# Maquina da recepção
# de qualquer lugar para 173.173.1.3
#${fwcmd} add 00023 pipe 1 tcp from any to 173.173.1.3 out
# Valor de download
#${fwcmd} pipe 1 config bw 64Kbit/s queue 8Kbytes

# da maquina 173.173.1.3 para qualquer lugar...
#${fwcmd} add 00024 pipe 2 tcp from 173.173.1.3 to any in
# Valor upload
#${fwcmd} pipe 2 config bw 64Kbit/s queue 8Kbytes
```

Figura 3.20: Trecho do script firewall com implementação do controle de banda

O suporte ao *layer*2 no FreeBSD, também pode ser habilitado através do comando ilustrado na figura 3.21. Esse comando ativa esse recurso diretamente no *kernel*.

```
sysctl net.link.ether.ipfw=1
```

Figura 3.21: Comando de ativação no *kernel* para controle de camada 2

Para utilizar o bloqueio através dos endereços MAC (*Media Access Control*) dos usuários, são necessários algumas modificações no *firewall* principal. Conforme a figura 3.21, observa-se que a regra de número 00001 não permite a navegação de Internet para os endereços MAC dos computadores da intranet que não estão cadastrados. Foi utilizada uma estrutura de repetição(*while*) que lê um arquivo de texto (cadastro_macs). Esse arquivo de texto contém todos os endereços IP e MAC autorizados para acesso. Dentro dessa estrutura de repetição, foi adicionada a regra 00040 para liberar o tráfego para um computador que tenha seu MAC amarrado à um determinado endereço IP. Ainda dentro da estrutura de repetição, a regra 00100, enfim, realiza o compartilhamento de conexão da Internet. Por último, a regra 65099 bloqueia o tráfego dos computadores com endereços MAC não estejam cadastrados no arquivo cadastro_macs. A estrutura do arquivo "cadastro_macs", pode ser vista na figura 3.23.

```
# Desativando passagem única  de pacotes
${fwcmd} disable one_pass

# Liberando a tabela arp - limpando as regras existentes
${fwcmd} -f flush

# Não permitir navegacao a macs não cadastrados
${fwcmd} add 00001 allow layer2 not mac-type ip

# libera o tráfego localhost
 ${fwcmd} add allow layer2 via lo0

# libera tráfego da interface de saída
\${fwcmd} add allow layer2 via rl0
if [ -f /usr/local/etc/Squid/cadastro_macs ]; then
    awk '{ print $1,$2 }' /usr/local/etc/Squid/cadastro_macs | while read MAC IP;
  do
     # Clientes IPxMAC:
    ${fwcmd} add 00040 allow layer2 src-ip $IP MAC any ${MAC} in via ${if_lan}
    #NAT  IPDIVERT
    ${fwcmd} add 00100 divert natd all from any to any via ${nat}
done
fi

#Nega todos MACs que nao estejam cadastrados no arquivo cadastro_macs
${fwcmd} add 65099 deny MAC any any in recv ${if_lan} layer2
```

Figura 3.22: Trecho do *script firewall* com diretivas de controle para camada 2

```
00:e4:21:aa:ff:b3  173.173.4.1 #tributos_01
00:bb:ee:a8:99:aa  173.173.4.3 #tributos_02
00:bb:ee:a8:92:a1  173.173.4.4 #seplan_01
00:aa:ff:aa:99:bb  173.173.4.5 #seplan_02
```

Figura 3.23: Estrutura do arquivo cadastro_macs

Capítulo 4

Implementando o proxy transparente

4.1 Fundamentação teórica do Squid

A implantação de um *proxy* transparente em um servidor de perímetro é uma boa alternativa para trabalhar em conjunto com a filtragem de pacotes (*firewall* IPFW) e com o sistema de tradução de endereços de rede(NAT). O Squid é um *software* livre licenciado nos termos da *General Public License* (GPL) e engloba-se na categoria *web proxy* cache. Ele permite que os usuários da rede possam acessar à Internet através dele. Além de auxiliar na segurança e no controle de acessos à Internet por parte dos usuários da intranet, ele também aumenta a performance de navegação . Isso acontece porque ele possui um sistema de *cache* de objetos e de sites visitados. Também é possível através do Squid, realizar controle e economia de banda de Internet. Segundo (MARCELO, 2006), o Squid pode ser executado nas seguintes plataformas : Linux; FreeBSD; AIX; NetBSD; BSDI; HP-UX; OSF; Digital Unix; IRIX; SunOS/Solaris; NeXTStep; SCO Unix; OS/2 e Windows NT.

4.2 Processo de instalação

No FreeBSD, o Squid pode ser instalado através dos *ports collections*. Isso é ilustrado na figura 4.1

```
#cd /usr/ports/www/squid/ & make install clean
```

Figura 4.1: Processo de instalação do Squid

Executa-se o comando "`rehash`", para que as alterações do caminho da variável do Squid seja atualizada. Após a instalação, os arquivos de configuração do Squid, encontram-se dentro do diretório "`/usr/local/etc/squid`". O diretório de *cache* estará localizado em "`/usr/local/squid/cache`". Os arquivos de log do Squid, estarão localizados no diretório "`/usr/local/squid/logs`". O arquivo executável do Squid estará em "`/usr/local/sbin/`" e também seu *script* de inicialização em "`/usr/loca/etc/rc.d/`"

4.3 Configurações

O arquivo principal de configuração do Squid é o "`squid.conf`". Ele está localizado no diretório "`/usr/local/etc/squid`". Entre tantas diretivas importantes de configuração do Squid, algumas se destacam:

- Na diretiva **http_port**, definimos a porta e como será a funcionalidade do *proxy*. No presente trabalho, foi configurado como transparente. "http_port 3128 transparent"

Na seção de opções que podem afetar o *cache*, definimos as seguintes variáveis:

- A diretiva **cache_mem**, especifica até quanto de memória o Squid pode utilizar para sua atividade na realização do *cache* de objetos no sistema "cache_mem 1200 MB"
- A diretiva **cache_dir**, especifica o tamanho do *cache* de objetos. Para o presente trabalho, o HD do computador possui tamanho de 160 GB, então foi configurado o tamanho do *cache* para 100 GB.
  "cache_dir ufs /usr/local/Squid/cache 100000 16 256"
- A diretiva **maximum_object_size**, define o tamanho dos objetos que podem ser colocados em cache no servidor. Para o presente trabalho, foi escolhido o tamanho de 350 MB. Ou, seja, se um usuário da rede fizer algum *download* de um arquivo maior que 350 MB, ele não ficará guardado no servidor.
  "maximum_object_size 350 MB"

O controle de usuários para acesso à Internet, é feito através de regras de ACL (*Acess Control List*). Essas regras também são configuradas no arquivo "`squid.conf`". A implementação desse controle de acesso é ilustrado na figura 4.2. Foram definidos três níveis de acesso à Internet:

- **Nivel 1** – Para usuários que podem acessar apenas sites com extensão ".ba", ."gov", ".edu", entre outros institucionais.
- **Nível 2** – Para secretários e diretores da instituição. Esse nível, permite navegação normal, com exceção para os sites que estão cadastrados no arquivo "bloqueados". Nesse arquivo "bloqueados", contém, por exemplo, diversos sites de conteúdo erótico.
- **Nível 3** – Esse é o acesso completo. Qualquer usuário desse nível pode acessar qualquer site sem nenhuma restrição.

Após configurar as diretivas e o controle de acesso de usuários no arquivo de configuração (squid.conf), foi necessário executar o comando "`squid -z`". O comando "`squid -z`" é necessário para que seja criado toda a estrutura de árvore de diretórios de *cache* do Squid. Isso é ilustrado na figura 4.3. Antes de criar os diretórios, observa-se que o caminho do diretório de *cache* do Squid, está completamente vazio.

```
# Descrição do Controle de Acesso: Politica DROP

# Liberando acesso a Internet para IPs cadastrados
acl acesso_restrito src "/usr/local/etc/squid/acesso-net"
# Nível 1 - Sites permitidos para navegacao (apenas .ba.gov)
acl sites url_regex -i "/usr/local/etc/squid/sites_permitidos"

acl bloqueados url_regex -i "/usr/local/etc/squid/bloqueados"
acl liberados url_regex -i "/usr/local/etc/squid/sites_liberados"

# Nível 2 - Grupo_Secretario_diretores_dti
acl usuarios_sec_dir_dti src "/usr/local/etc/squid/ips_acesso_normal"

# Nível 3 - Acesso-Full
acl acesso_full src "/usr/local/etc/squid/acesso-total"

http_access allow localhost

http_access allow acesso_full
http_access deny bloqueados
http_access allow usuarios_sec_dir_dti

http_access allow acesso_restrito sites
# Bloqueando todos acessos externos a esse PROXY!
http_access deny all
```

Figura 4.2: Trecho do arquivo de configuração "Squid.conf" com regras de acesso.

```
gw-PMI# ls /usr/local/squid/cache
00/ 01/ 02/        03/ 04/ 05/ 06/ 07/
08/ 09/ 0A/ 0B/ 0C/ 0D/ 0E/ 0F/
squid.core  swap.state   swap.state.last-clean
```

Figura 4.3: Estrutura de diretórios criados pela execução do comando "squid -z"

Após a etapa em que foi gerado os diretórios de *cache* do squid, fez-se necessário configurar o arquivo de inicialização do serviço. Foi feita a edição do arquivo "`/usr/local/etc/rc.d/squid`", colocando a diretiva "`SQUID_ENABLE`" para "YES".

Apesar de o Squid ser executado com sucesso e desempenhar suas funções, faz-se necessário a manutenção do seu principal arquivo de log. Trata-se do arquivo "`access.log`". Conforme os acessos aos sites da Internet são feitos, esse

arquivo vai aumentando de tamanho. Não é aconselhável deixar ele atinjir o tamanho superior à 1 GB. Caso contrário, à Internet pode ficar com problemas de lentidão e travamento. Tendo em vista essa necessidade de limpeza, um *shell script* foi desenvolvido para automatizar à manutenção do *proxy* Squid. O script "`limpa_logs_squid.sh`" tem o objetivo de uma vez na semana, fazer o *backup* do arquivo "`access.log`" compactando o mesmo.

```
# Script desenvolvido por Richer Barros
# Manutenção do arquivo access.log

DATA=`date +%Y_%m_%d`
NOME="backup-access_log-\$DATA.tar.gz"
ARQUIVO="access.log.0"
echo "Iniciando rotate..."
/usr/local/sbin/squid -k rotate
sleep 2
cd /usr/local/squid/logs/
tar zcf $NOME $ARQUIVO
sleep 4
cp $NOME /tmp/backups_squid
rm access.log.0
rm cache.log.0
sleep 5
rm $NOME
echo "Backup Feito!"
#EOF
```

Figura 4.4: Código fonte do ShellScript de backup do Squid

4.3.1 Testes e resultados

Ao executar o comando "`tail -f /usr/local/squid/logs/`" no terminal , são verificados os registros de acesso das máquinas da intranet que utilizam à Internet. Observa-se esse teste na figura 4.5. Quando um usuário tenta acessar um site que o seu nível de acesso não permite, ele recebe um aviso no navegador. Isso é ilustrado na figura 4.6

```
1267307704.124    316 173.173.3.5 TCP_MISS/200 477 \
POST http://yahoo.com.com/wild/rt.ashx - DIRECT/91.199.100.2 image/gif

1267307706.204    250 173.173.3.21 TCP_IMS_HIT/304 355 \
GET http://www.uol.com.br/saude_e_negocios-550x100.swf - \
NONE/- application/x-shockwave-flash

1267307735.205    235 173.173.3.81 TCP_IMS_HIT/310 355 \
GET http://www.ig.com.br/esporte/n130013417.html
```

Figura 4.5: Registros de acesso aos sites da Internet

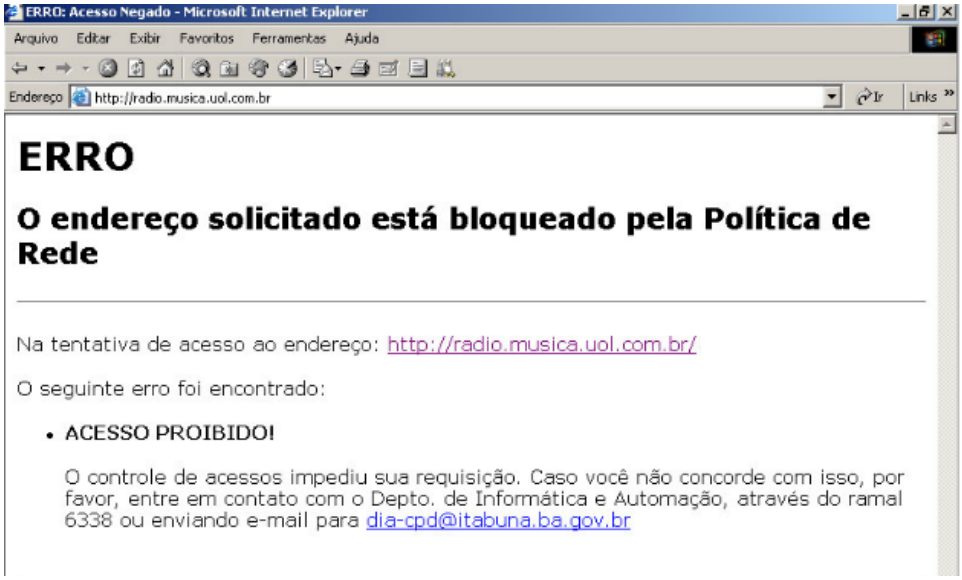

Figura 4.6: Acesso bloqueado à um determinado site

4.4 Implementando ferramenta para gerar relatórios (SARG)

O SARG (*Squid Analysis Report Generator*) é uma ferramenta desenvolvida pelo brasileiro Pedro Orso (ORSO, 2004). Essa ferramenta atua lendo o arquivo de log do Squid (access.log) e gerando as páginas com os registros de acesso à Internet no formato HTML.

Com a utilização do SARG, é possível saber, de que maneira os usuários da rede estão utilizando o acesso à Internet. É possível saber os dias, os horários, e os

sites mais acessados por determinados usuários. Enfim, o SARG é uma ferramenta que facilita o gerenciamento de um *proxy web*.
A instalação do SARG, foi realizada através dos *ports collections*, de acordo com a figura 4.7 Após a instalação do SARG, é necessário fazer a edição do seu arquivo de configuração para gerar os relatórios. Vale ressaltar que a geração do relatório, efetuada pelo SARG, possui certa dependência de como o *proxy* Squid está configurado. Se o *proxy* Squid, estiver funcionando como *proxy* transparente da rede, não vai aparecer, por exemplo, os nomes de usuários da rede. Mas outros tipos de informações, tais quais, endereços IP ou nome das máquinas. Se o Squid estiver configurado com suporte à autenticação, os nomes dos usuários irão aparecer no relatório.

```
#cd /usr/ports/www/sarg
#make install clean
```

Figura 4.7: Instalação do SARG

O arquivo de configuração principal do SARG, está localizado dentro do diretório "`/usr/local/etc/sarg/sarg.conf`". No apêndice, seção A.2, podem ser verificadas todas as opções de configurações do SARG. Pra que usuários comuns da rede não tenham acesso aos relatórios de acesso à Internet, foi configurado no servidor *web* Apache, o arquivo "`.htaccess`", com o conteúdo mostrado na Figura 4.8.

```
AuthName "Acesso Restrito a Usuarios"
AuthType Basic
AuthUserFile /usr/local/www/data/admin/acesso
require valid-user
```

Figura 4.8: Configurações do arquivo "htaccess"

4.4.1 Realizando teste da geração de relatórios

Quando o comando "`sarg`" é executado no sistema, o processo de interpretação do log do Squid é iniciado. Após o termino desse processo, basta ir através de um navegador qualquer e digitar o caminho no qual foi configurado no arquivo "`sarg.conf`". A figura 4.9, ilustra os sites acessados por um determinado usuário.

Relatorios Usuarios PMI
Periodo: 07Oct2009-22Oct2009
Usuario: 173.173.3.246
Ordem: BYTES, reverse
Usuario Relatorio

LOCAL ACESSADO	CONEXÃO	BYTES	%BYTES	IN-CACHE-OUT		TEMPO GASTO	MILISEG	%TEMPO	
storage.mais.uol.com.br	62	458.25M	23.50%	12.67%	87.33%	00:05:08	308.680	0.34%	
ak.worldofwarcraft.com.edgesuite.net	1.09K	287.33M	14.74%	0.00%	100.00%	00:25:57	1.557.864	1.71%	
ne.edgecastcdn.net	91	96.77M	4.96%	0.64%	99.36%	00:02:42	162.490	0.18%	
up18.uploading.com	1	89.32M	4.58%	0.00%	100.00%	00:27:14	1.634.037	1.79%	
72.233.85.146	6	58.01M	2.98%	0.01%	99.99%	00:23:08	1.388.865	1.52%	
virgula.uol.com.br	195	51.54M	2.64%	0.79%	99.21%	00:03:55	235.437	0.26%	
www.meiobit.com	1	1.82K	0.00%	100.00%	0.00%	00:00:00	6	0.00%	NEGADO
www.todojuegos.com	1	1.82K	0.00%	100.00%	0.00%	00:00:00	82	0.00%	NEGADO
digg.analytics.live.com	3	1.82K	0.00%	0.00%	100.00%	00:00:01	1.864	0.00%	
blocoanuncios.clickafiliados.com.br	2	1.81K	0.00%	0.00%	100.00%	00:00:00	904	0.00%	
update4ex.jdownloader.org	1	1.81K	0.00%	100.00%	0.00%	00:00:00	4	0.00%	NEGADO
www.catalogopcgames.com	1	1.81K	0.00%	100.00%	0.00%	00:00:00	38	0.00%	NEGADO
www.jacotei.com.br	3	1.80K	0.00%	0.00%	100.00%	00:00:00	507	0.00%	
www.gamesbloggo.com	1	1.80K	0.00%	100.00%	0.00%	00:00:00	82	0.00%	NEGADO
www.baixatudogames.com	1	1.77K	0.00%	100.00%	0.00%	00:00:00	1	0.00%	NEGADO
www.silkroadonline.net	1	1.77K	0.00%	0.00%	100.00%	00:00:01	1.627	0.00%	
g.msn.com	4	1.77K	0.00%	0.00%	100.00%	00:00:01	1.626	0.00%	
www.premiummegaupload.com.br	1	1.77K	0.00%	100.00%	0.00%	00:00:00	151	0.00%	NEGADO
www.aregiao.com.br	2	1.77K	0.00%	0.00%	100.00%	00:00:02	2.091	0.00%	
i583.photobucket.com	1	1.77K	0.00%	100.00%	0.00%	00:00:00	133	0.00%	NEGADO
251e2c4c1efc3da6.users.storage.live.com	1	1.77K	0.00%	0.00%	100.00%	00:00:01	1.363	0.00%	
www.windowslivetranslator.com	1	1.77K	0.00%	100.00%	0.00%	00:00:00	2	0.00%	NEGADO
dc162.4shared.com	1	1.77K	0.00%	100.00%	0.00%	00:00:00	1	0.00%	NEGADO
dc147.4shared.com	1	1.77K	0.00%	100.00%	0.00%	00:00:00	1	0.00%	NEGADO
dc141.4shared.com	1	1.77K	0.00%	100.00%	0.00%	00:00:00	1	0.00%	NEGADO
dc140.4shared.com	1	1.77K	0.00%	100.00%	0.00%	00:00:00	1	0.00%	NEGADO
dc136.4shared.com	1	1.77K	0.00%	100.00%	0.00%	00:00:00	1	0.00%	NEGADO
www.facebook.com	1	1.76K	0.00%	100.00%	0.00%	00:00:00	217	0.00%	NEGADO
dc99.4shared.com	1	1.76K	0.00%	100.00%	0.00%	00:00:00	1	0.00%	NEGADO
domo.aeriagames.com	1	1.76K	0.00%	100.00%	0.00%	00:00:00	344	0.00%	NEGADO
pavao.videolog.tv	1	1.76K	0.00%	100.00%	0.00%	00:00:00	115	0.00%	NEGADO
www.radios.com.br	1	1.75K	0.00%	100.00%	0.00%	00:00:00	161	0.00%	NEGADO

Figura 4.9: Sites acessados por um usuário da rede

Capítulo 5

Configurações de NAT e PAT

5.1 Introdução ao NAT

O NAT (*Network Address Translation*) é uma técnica utilizada para realizar o mapeamento de toda uma rede interna (intranet) para um único endereço IP. Essa técnica é necessária quando o número de endereços IP atribuídos pelo provedor de Internet é menor do que o número de máquinas que devem ter acesso a uma conexão de Internet.O NAT também conserva o número de endereços IP global que uma empresa precisa e deixa a empresa usar um único endereço IP na sua comunicação com o "mundo exterior." O NAT tem um papel fundamental na segurança de uma rede interna, pois cada solicitação de entrada ou de saída, deve passar por um processo de tradução, que também oferece a oportunidade de qualificar ou autenticar o pedido, ou combiná-la a um pedido anterior. Essa técnica de NAT funciona geralmente em um dispositivo de *firewall* ou *gateway* (roteador). Esses dispositivos recebem diferentes solicitações de conexões de redes, fazendo a conversão de um endereço IP inválido em IP válido e vice-versa. Sendo assim, a rede interna fica mais segura, pois os computadores externos que utilizam a Internet geralmente não tem condições de verificar o verdadeiro endereço IP (que está protegido pelo *firewall*), dessa maneira, não é necessário usar um endereço IP válido fixo para um computador que nem sempre pode estar utilizando a Internet.

No início de sua utilização, o NAT era uma implementação idealizada somente para resolver os problemas de escassez de endereços IP. Com o passar do tempo, ficou provado que ele pode ser útil em vários outras áreas de aplicação, como por exemplo a segurança (criação de *firewall*). O NAT tem como principal função

traduzir vários endereços de IP (inválidos) da rede local para vários endereços de IP globais (válidos), sendo a sua relação então N para N. O NAT pode funcionar de duas maneiras: Estática e Dinâmica.

5.1.1 Tipos de NAT

As diferenças entre os tipos de NAT, são explanadas de acordo com (CUNHA, 2006).

- **NAT Estático:** Define-se NAT estático (*static* NAT) aquele no qual as traduções são sempre as mesmas. Um dado IP global é sempre o resultado da tradução de um determinado IP local todo o tempo. Nenhum outro IP local é traduzido naquele IP global. Neste caso se houver 30 endereços IP válidos serão atribuídos 30 acessos simultâneos sempre para as mesmas estações. Através do NAT estático é possível que uma estação da Internet possa acessar uma estação ou servidor da rede local, desde que haja permissão para isso.
- **NAT Dinâmico:** (*dynamic* NAT), a tradução de um IP local em um IP global depende de uma série de fatores definidos em tempos de execução (permissão, número de endereços IP, MAC, etc). No NAT dinâmico os endereços globais (válidos) são atribuidos aleatoriamente para as estações da nova rede local, se houver 30 endereços globais eles serão alocados para 30 estações dinamicamente e a 31ª estação ficará sem acesso. A cada nova tradução é possível que um IP local obtenha um novo endereço global.

5.2 Entendendo o PAT

Para o melhor entendimento no presente trabalho, a explicação do PAT foi baseada em (ROSS, 2008). De acordo com a figura 5.1, os usuários da rede local 10.10.7.xx pretendem acessar o servidor do site de busca Cadê, que possui um endereço roteável (válido) 200.9.149.100. O administrador encontra um problema: como sua rede 10.10.7.xx vai acessar o servidor do Cadê? A resposta é óbvia: fazendo um NAT, assim todos terão um endereço válido para isso. Mas e como fazer para não usar muitos endereços válidos e não esgotar minha faixa de endereços IP válidos que o meu provedor de Internet me deu ? A Resposta é o *Port Address Translation*. PAT é o tipo de NAT que mais economiza endereços válidos (roteáveis) pois a tradução é feita no modelo N para 1, ou seja, todos os endereços da rede local

são traduzidos para um único endereço válido, diferenciando uma máquina da outra apenas pela porta usada. Este modelo apresenta uma limitação para o número máximo de conexões simultâneas. Como é sabido, há uma limitação do número de portas de comunicação. Por existir um total de 65535 portas disponíveis para comunicação, só é possível, em teoria, se traduzir um número menor que 65000 endereços simultâneos.

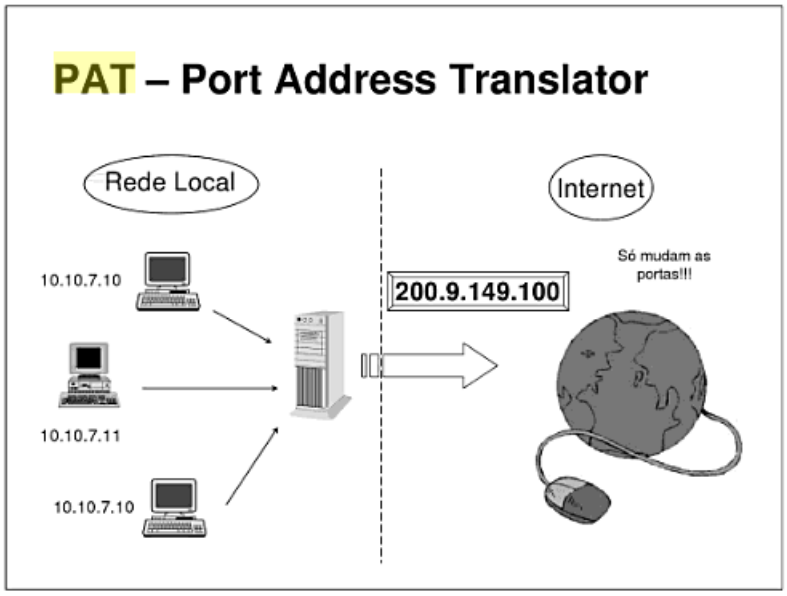

Figura 5.1: Conexoes de PAT - Fonte:(ROSS, 2008)

As conexões de NAT e PAT no FreeBSD para o presente trabalho, são feitas através da configuração dos arquivos "`rc.conf`" e "`natd.conf`".

5.3 Configurando o arquivo "rc.conf"

De acordo com o arquivo "`/etc/defaults/rc.conf`", a opção "`natd_enable`" define se o NAT será ou não habilitado. Já a opção "`natd_flags`", indica o arquivo que deve ser lido para realizar o processo de NAT.

```
natd_enable="YES"
natd_flags="-f /etc/natd.conf"
```

Figura 5.2: Diretivas de configuração no arquivo "rc.conf"

5.4 Configurando o arquivo "natd.conf"

Para que as conexões de PAT possam funcionar corretamente, visando atender os objetivos do presente trabalho, é necessário criar e configurar o arquivo "`natd.conf`". Esse arquivo está localizado no diretório "`/etc`". Seus parâmetros de configuração são mostrados na figura 5.3

```
interface rl1
dynamic yes
same_ports yes
use_sockets yes
# Realizando conexão PAT para aplicativo VNC
redirect_port tcp 173.173.1.3:5900 200.217.76.28:5900
```

Figura 5.3: Arquivo natd.conf

Dentre as opções vistas na figura 5.3, do arquivo "`natd.conf`", vale destacar as duas mais importantes:

- A opção "interface rl1", refere-se a interface de rede na qual será feita o NAT. No caso será utilizada a interface rl1
- A opção "redirect_port tcp 173.173.1.3:5900 200.217.76.28:5900" , indica para qual máquina da rede interna, a solicitação externa deve ser encaminhada. Nesse exemplo, uma conexão externa é solicitada ao endereço IP 200.217.76.28 utilizando a porta 5900. O servidor, então redireciona essa conexão para a máquina da intranet de endereço IP 173.173.1.3.

Vale lembrar que esse arquivo deve conter uma linha em branco no seu final, para evitar erros.

Capítulo 6

Configurações de Hardening e Tuning

6.1 Introdução

Depois de proceder com a instalação e atualização do sistema operacional, é necessário realizar procedimentos a nível de *kernel*, protocolos, aplicativos, com a finalidade de deixar mais seguro e "enxuto" o servidor. Os procedimentos de *hardening* e *tuning* , além de melhorar a segurança , visam aumentar a performance e a estabilidade do servidor. Esses procedimentos, baseados em (GEIB, 2004a), são mostrados nas próximas seções desse capítulo.

6.2 Ajustes na pilha do protocolo TCP/IP

Ao executar um sistema operacional do tipo UNIX-*like*[1], como o FreeBSD, é possível torná-lo mais seguro, modificando o comportamento de execução do protocolo TCP/IP. Para a maior parte, estes são comandos que iria entrar em boot, como um script "`/etc/rc.sysinit`" ou "`/etc/rc.local`"

[1]O termo Unix-like é amplamente utilizado para descrever sistemas operacionais que compartilham muitas das características do UNIX original. Disponivel em: http://www.linfo.org/unix-like.html.

- **No protocolo ARP**

  Diminuir o intervalo de limpeza do *cache* ARP. No FreeBSD, isso é possível através do comando abaixo:
  "`sysctl-w net.link.ether.inet.max_age = 1200`"

- **No protocolo ICMP**

  Desabilitando atividade broadcast ICMP echo - Do contrário, o sistema poderia ser utilizado como parte de um ataque Smurf:
  "`sysctl-w net.inet.icmp.bmcastecho = 0`"

  Desativando encaminhamento redirecionamento ICMP - Se isso não for desabilitado, o redirecionamento da rota do protocolo ICMP pode ser alterada por um atacante.
  "`sysctl -w net.inet.ip.redirect=0`"
  "`sysctl-w net.inet.ip.redirect = 0`"

- **Desabilitando ICMP broadcast sondas -** Se isso não for desabilitado, um invasor pode fazer engenharia reversa alguns detalhes da sua infra-estrutura de rede.
  "`sysctl-w net.inet.icmp.masqrepl = 0`"

- **No protocolo IP**

  Desabilitando origem do roteamento IP - Com essa opção, é possível enganar um atacante, pois o encaminhamento de pacotes de IP roteados são desabilitados.
  "`sysctl -w net.inet.ip.sourceroute = 0`"
  "sysctl -w net.inet.ip.accept_sourceroute = 0"

- **No protocolo TCP**

  Aumentando o TCP enviar e TCP receber no tamanho de janela para pelo menos 32 Kbytes.

  "`sysctl -w net.inet.tcp.sendspace = 32768`"
  "`sysctl -w net.inet.tcp.recvspace = 32768`"

6.3 Alteração de outras diretivas de segurança do sistema

De acordo com (GEIB, 2004a) os ajustes abaixo podem ser feitos, deixando o perímetro de segurança ainda mais eficiente.

- **Usuário toor**
  O usuário toor vem por padrão na instalação do FreeBSD como uma conta alternativa ao root mas com os mesmos poderes, pois possui UID 0. Seu principal objetivo é ser usado com um interpretador de comandos fora do padrão, que seja instalado em um *filesystem* diferente do raiz - caso do `bash`, que fica em "`/usr/local/bin/bash`". Então pode-se usar o `toor` com este *shell* e manter o root com um *shell* padrão, como por exemplo `/bin/sh`. No caso de algum problema, em que não puder ser montado o filesystem "`/usr`", ficaríamos sem *shell*, e conseqüentemente sem acesso ao sistema. Contudo, ao entrar em modo *single*, o sistema nos pede o caminho completo de algum *shell*. Se houver interesse em manter essa conta `toor` como um "*backup*" ao root, deve-se alterar a sua senha através do comando "`passwd`". Caso não houver interesse, ou houver alguma chance de essa conta cair no esquecimento, é recomendado que seja removida, visto o perigo que representa caso for utilizada indevidamente.

- **Senha no console**
  Ao inicializar o FreeBSD, podemos pressionar qualquer tecla (diferente de enter) durante a contagem regressiva para que esta seja parada. Se entrarmos o parâmetro "`boot -s`" e dermos `enter`, o sistema será inicializado em modo *single* (monousuário), da mesma forma que quando digitamos no *shell* o comando "*shutdown now*", mas com a diferença que no primeiro modo não será montado automaticamente nenhum *filesystem* além do raiz. Por padrão, quando entramos no modo single o sistema não nos pede uma senha, e fornece acesso a nível de root, ou seja, a tudo. Isto pode representar um grande risco caso alguém mal intencionado consiga acesso físico ao servidor, então poderá entrar no modo *single* e até mesmo alterar a senha do root. Para evitar isto, podemos instruir o sistema para que peça senha (do root) mesmo no modo *single*. No arquivo "`/etc/ttys`", é feita a alteração da linha:

  console none unknown off secure
  para
  console none unknown off insecure

6.4 Ajuste de Aplicativos

Aumentando a segurança no SSH

O SSH (*Secure Shell*) é o principal programa de acesso remoto em ambientes UNIX. É possível configurar alguns parâmetros para que ele possa ficar ainda mais seguro. O arquivo principal de configuração do SSH é o "`sshd_config`" e ele está localizado no diretório "`/etc/ssh/`"

- **Alteração de porta:** Esse é um procedimento que minimiza bastantes ataques do tipo *brute force*. É comum, pessoas mal intencionadas tentarem programas que realiza inúmeras tentativas de acesso através da porta 22. Isso é feito alterando a opção "Port" do arquivo "`/etc/ssh/sshd_config`"

- **Limitando o acesso para somente um usuário:** Isso é feito utilizando a diretiva "AllowUsers". Faz-se necessário inserir no arquivo "`/etc/ssh/sshd_config`" uma linha com "AllowUsers login_do_usuario_autorizado".

- **Negando acesso do usuário "root":** Essa é uma das melhores práticas de segurança. Afinal, em caso de uma possível invasão, o atacante estará com todos os poderes no sistema operacional. Ao configurar a diretiva "`PermitRootLogin no`" no arquivo "`/etc/ssh/sshd_config`", assegura-se que em caso de uma suposta invasão, o atacante terá que descobrir a senha de root.

Capítulo 7

Avaliação dos resultados

Neste capítulo, são apresentados os resultados obtidos. A avaliação desses resultados foram realizadas com base em testes feitos antes e depois da implementação do servidor de perímetro seguro FreeBSD.

7.1 Redução na Ocorrência de Vírus nos Computadores da Prefeitura

A ocorrência de virús nos computadores da prefeitura antes da implementação do servidor de segurança, de acordo com o setor de Divisão de Manutenção do Departamento de Tecnologia da Informação, foi de aproximadamente 250 notificações no mês de janeiro de 2009. Isso pode ser observado na Tabela 7.1. Após o servidor de perímetro entrar em produção, no mês de janeiro de 2010, foi significativa a diminuição das notificações de vírus. A maioria desses vírus que infectaram os computadores chegaram na intranet da prefeitura através do uso de email e do MSN. Após o *firewall* ser implementado, o problema foi quase que totalmente solucionado. As ocorrências de vírus relativas ao ano de 2010, foram ocasionadas pelo mau uso de *pendrive* na instituiçao.

Tabela 7.1: Ocorrência de vírus antes e depois nos computadores da Prefeitura de Itabuna

MES/JAN	Conficker	Downadup	Virus por Email	Total de Notificacoes
2009	20	50	180	250
2010	3	10	2	15

7.2 Melhoria na Performance da Navegação de Internet

Com a instalação do Squid, a velocidade de navegação da intranet aumentou e a economia da banda de *link* foi significativa. Essa economia é observada na comparação da Figura 7.1 e da Figura 7.2 no comparativo de acordo com o uso *software* MRTG[1].

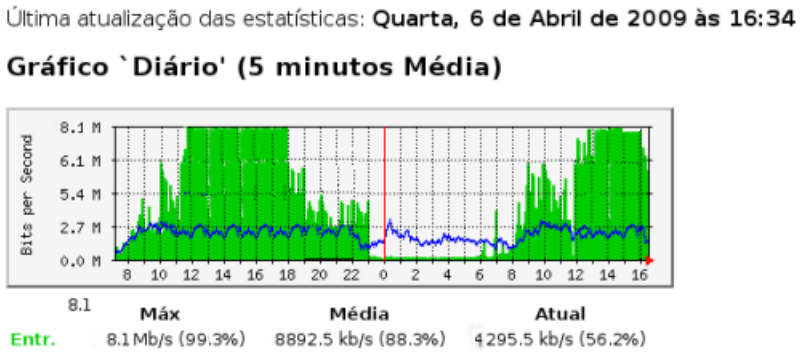

Figura 7.1: Consumo de banda de link de Internet antes do servidor de perimetro entrar em produção

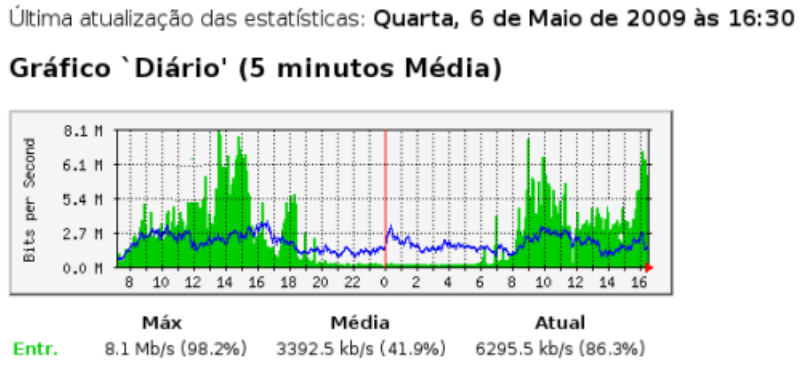

Figura 7.2: Consumo de banda de link de Internet depois do servidor de perimetro entrar em produção

[1] *Multi Router Traffic Grapher* (MRTG) - É um *software* livre utilizado para medir o consumo de banda de *link* de Internet

7.3 Impacto e Avanços

A entrada do servidor de perímetro em produção trouxe alguns impactos e avanços para os diversos segmentos de usuários da prefeitura. A maioria dos usuários acharam que o acesso à Internet ficou mais rápido. Entretanto, inicialmente houve uma grande resistência de muitos usuários por tais mudanças. Isto aconteceu por causa das políticas de rede que foram estabelecidas. Usuários que antes acessavam o programa de *chat* MSN livremente por exemplo, passaram a fazer uma solicitação por ofício ao Departamento de Tecnologia da Informação justificando o seu uso. O mesmo aconteceu também com o setor de imprensa, que para acessar *sites* de vídeos como *Youtube*, tiveram que elaborar um ofício justificando o uso.

O diretor do Departamento de Tecnologia da Informação ficou satisfeito com o trabalho e implantação desse servidor, pois agora, ele pode acessar os relatórios de acesso à Internet de qualquer computador da prefeitura, utilizando o próprio computador dele. Com esse relatórios "em mãos", ele pode provar que um determinado usuário desacatou as políticas de rede e tomar providências cabíveis contra ele. Desta maneira, os usuários da rede da prefeitura foram tomando consciência sobre como se deve utilizar a Internet. O analista e pós-graduando deste presente trabalho, inicialmente não adquiriu ganhos financeiros extras com o desenvolvimento do servidor de perímetro seguro. Porém, após três meses de produção do servidor, o mesmo, que é servidor público municipal efetivo, foi promovido para o cargo de chefe do setor de Divisão de Controle e Segurança de Redes do Departamento de Tecnologia da Informação.

Capítulo 8

Conclusão

Este presente trabalho permitiu aprimorar ainda mais o conhecimento sobre segurança digital em um ambiente corporativo. Nesse orgão público que trata-se de uma prefeitura, pessoas desempenham as mais variadas funções e diversos serviços são prestados à comunidade. Com a implementação de um perímetro seguro, a integridade e confidencialidade dos dados, foram devidamente asseguradas. Os usuários da rede passaram a ser monitorados, porém esses mesmos usuários, passaram à ter maior confiança em seus acessos à Internet.

O desenvolvimento desta monografia possibilitou à prefeitura aumentar o nível de segurança de seus computadores interligados em rede e, ao pós-graduando, adquirir experiência e conhecimento na área Segurança de Redes utilizando uma plataforma Unix BSD, que será de grande importância na sua vida profissional e acadêmica.

Com o desenvolvimento, houve um melhor gerenciamento do *link* de Internet na prefeitura de Itabuna. Isso foi possível graças as ferramentas eficientes que foram instaladas. Ferramentas por exemplo, como o SARG. Foi notada a diminuição de computadores no setor de manutenção do departamento de tecnologia da informação. Antes da implementação do perímetro de segurança, a reincidência de computadores por vírus era grande.

Por ser o FreeBSD um sistema totalmente estável e que não exige tantos recursos a nível de *hardware*, os custos financeiros para a implementação do servidor foram relativamente baixos. Não foi necessária a aquisição de uma "super máquina" com arquitetura servidor, foi utilizado um computador com processador

Core 2 Duo, com 2 GB de memória RAM e um HD de 160GB com a tecnologia SATA.

Como perspectivas de trabalhos futuros, pretende-se efetuar a implantação de um servidor de VPN sob a plataforma BSD, entre os órgãos anexos da prefeitura municipal de Itabuna utilizando o *software* OpenVPN. Será planejada a implementação do FreeBSD Jail em alguns servidores da prefeitura. O FreeBSD Jail é um recurso nativo do FreeBSD que permite criar diversos ambientes virtualizados. Com esse recurso é possível gerenciar e centralizar diversos serviços em apenas um servidor. Um outro projeto também deve ser executado futuramente, trata-se da realização de um estudo visando transformar um computador com o sistema operacional FreeBSD em um ponto de acesso de uma rede *wireless*.

Referências Bibliográficas

CUNHA, A. R. da. *NAT - Network Address Translation*. 2006. Disponível em: <http://www.gta.ufrj.br/grad/98_2/adriano/nat_index.html>.

DUMMY. *Dummynet - Description*. Dummynet home page, 2002. Disponível em: <http://info.iet.unipi.it/~luigi/dummynet/>.

FUG-BR. *FreeBSD na Copa do Mundo 2006 e seus Patrocinadores*. Redação FUG-BR, 2006. Disponível em: <http://www.fug.com.br/content/view/64%-/54/>.

GEIB, H. T. *FreeBD: Configurações Básicas de Segurança*. 2004. Disponível em: <http://www2.unijui.edu.br/~heini/freebsd/confseg.html>.

GEIB, H. T. *FreeBSD, Controle de Tráfego IPFW2 + Dummynet*. 2004. Disponível em: <http://www2.unijui.tche.br/~heini/freebsd/dummynet.html>.

MARCELO, A. *SQUID-CONFIGURANDO O PROXY PARA LINUX*. 5. ed. Rio de Janeiro: Brasport, 2006.

MONTEIRO, E. S. *Segurança em Ambientes Corporativos*. 1. ed. Florianópolis: Visual Books, 2008.

OPENHB. *FreeBSD Handbook - Configurando o kernel do FreeBSD*. OPENIT, 2008. Disponível em: <http://www.openit.com.br/freebsd-hb/kernelconfig-building.html>.

ORSO, P. *Squid Analysis Report Generator*. [s.n.], 2004. Disponível em: <http://sarg.sourceforge.net>.

ROSS, J. *Redes de Computadores*. 1. ed. Rio de Janeiro: Antenna Edições Técnicas, 2008.

SILVA, G. M. da. *Firewall Iptables*. 2009. Disponível em: <http://mktecnologia-.net.br/index.php?option=com_content&view=article&id=18:firewall-iptables&catid=1:linux&Itemid=5>.

SOUSA, D. Linux ou freebsd? *4Linux – Free Software Solutions*, I, n. 1, p. 1, dez. 2008. Disponível em: <http://www.4linux.com.br/artigos/freebsd-ou-linux-.html>.

TFBSD. *The FreeBSD Project*. P.O. Box 20247, Boulder, CO 80308, USA: The FreeBSD Foundation, 2010. Disponível em: <http://www.freebsd.org>.

TFBSD2. *IPFW - FreeBSD Handbook, Capítulo 30 Firewalls*. P.O. Box 20247, Boulder, CO 80308, USA: The FreeBSD Foundation, 2010. Disponível em: <http://www.freebsd.org/doc/en/books/handbook/firewalls-ipfw.html>.

TRACANELLI, M. G. P. *IPFW – HOWTO*. 2002. Disponível em: <http://www-.freebsdbrasil.com.br/fbsdbrerline files/File/public_docs/ipfw-howto.pdf>.

UCHÔA, J. Q. *Segurança Computacional*. 2. ed. Lavras: UFLA-FAEPE, 2005.

Apêndice A

A.1 Arquivo de Configuração do Squid

```
#        WELCOME TO Squid 2.6.STABLE16 - Itabuna GATEWAY
#   Squid.conf 2.6 Funcional -> ok

#   Descricao - Squid com:

#        Sistemas de bloqueios de Ips da rede e sites da Web
#   Permitindo acesso a alguns sites por determinados IPs 13/12/2009
#   Codigo Limpo by Richer em 13/12/2009
#   Squid.conf base para qualquer servidor !

# NETWORK OPTIONS

# ------------------------------------------------------------------------------

#  TAG: http_port

http_port 3128 transparent

# OPTIONS WHICH AFFECT THE NEIGHBOR SELECTION ALGORITHM
newline
# ------------------------------------------------------------------------------

#  TAG: hierarchy_stoplist

hierarchy_stoplist cgi-bin ?

acl QUERY urlpath_regex cgi-bin \\?
```

```
cache deny QUERY

# TAG: broken_vary_encoding

acl apache rep_header Server \^{}Apache

broken_vary_encoding allow apache

# OPTIONS WHICH AFFECT THE CACHE SIZE

# -----------------------------------------------

#  TAG: cache_mem        (bytes)

cache_mem 1300 MB

#  TAG: maximum_object_size        (bytes)

maximum_object_size 200 MB

#  TAG: minimum_object_size

minimum_object_size 0 KB

#  TAG: maximum object size in memory (bytes)

maximum_object_size_in_memory\ 32 KB

# LOGFILE PATHNAMES AND CACHE DIRECTORIES

# -----------------------------------------------
#  TAG: cache_dir

# colocando cache de 100 Gigas!

cache_dir ufs /usr/local/squid/cache 100000 16 256

#  TAG: access_log
access_log /usr/local/squid/logs/access.log

#  TAG: cache_log
cache_log /usr/local/squid/logs/cache.log

#  TAG: cache_store_log
```

```
cache_store_log none

# OPTIONS FOR TUNING THE CACHE

#  TAG: refresh_pattern

#Suggested default:

refresh_pattern \^ftp:                1440        20\%\hspace*{1cm}        10080

refresh_pattern \^gopher:            1440        0\%\hspace*{1cm}        1440

refresh_pattern .                0        20\%\hspace*{1cm}        4320

# ACCESS CONTROLS (CONTROLE DE ACESSO)

# ---------------

#  TAG: acl

#        Defining an Access List

#Recommended minimum configuration:

acl all src 0.0.0.0/0.0.0.0

acl manager proto cache_object

acl localhost src 127.0.0.1/255.255.255.255
acl to_localhost dst 127.0.0.0/8
acl SSL_ports port 443
acl Safe_ports port 80                # http
acl Safe_ports port 21            # ftp
acl Safe_ports port 443        # https
acl Safe_ports port 70                # gopher
acl Safe_ports port 210                # wais
acl Safe_ports port 1025-65535        # unregistered ports
acl Safe_ports port 280                # http-mgmt
acl Safe_ports port 488                # gss-http
acl Safe_ports port 591                # filemaker
acl Safe_ports port 777                # multiling http
acl Safe_ports port 3001        # Setor Licitacao
acl Safe_ports port 1972      # Porta da ADi a pedido de Neto
acl Safe_ports port 23        # Porta telnet
```

```
acl CONNECT method CONNECT

# Only allow cachemgr access from localhost

http_access allow manager localhost

http_access deny manager

# Deny requests to unknown ports

http_access deny !Safe_ports

# Deny CONNECT to other than SSL ports

http_access deny CONNECT !SSL_ports

#client_netmask 255.255.0.0

#
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS

# Descrição do Controle de Acesso: Politica DROP

# Liberando acesso a internet para IPs cadastrados

acl acesso_restrito src "/usr/local/etc/squid/acesso-net"

# Nível 1 - Sites permitidos para navegacao (apenas .ba.gov)

acl sites url_regex -i "/usr/local/etc/squid/sites_permitidos"

acl bloqueados url_regex -i "/usr/local/etc/squid/bloqueados"

acl liberados url_regex -i "/usr/local/etc/Squid/sites_liberados"

# Nível 2 - Grupo_Secretario_diretores_dti

acl usuarios_sec_dir_dti src "/usr/local/etc/squid/ips_acesso_normal"

# Nível 3 - Acesso-Full

acl acesso_full src "/usr/local/etc/squid/acesso-total"

http_access allow localhost
```

```
http_access allow acesso_full

http_access deny bloqueados

http_access allow usuarios_sec_dir_dti

http_access allow acesso_restrito

# Bloqueando todos acessos externos a esse PROXY!

http_access deny all

#  TAG: http_reply_access

http_reply_access allow all

#Allow ICP queries from everyone

icp_access allow all

#  TAG: coredump_dir

coredump_dir /usr/local/squid/cache

#  TAG: error_directory

error_directory /usr/local/etc/squid/errors/Portuguese
```

A.2 Arquivo de Configuração do SARG

```
# sarg.conf
#
# TAG:  language
# sarg.conf
#
# TAG:  language

language Portuguese

# TAG:  access_log file
#       Where is the access.log file
#       sarg -l file
```

```
#
access_log /usr/local/squid/logs/access.log

# TAG: graphs yes|no
#        Use graphics where is possible.
#           graph_days_bytes_bar_color blue|green|yellow|orange|brown|red
#
#graphs yes
#graph_days_bytes_bar_color orange

# TAG:        title
#         Especify the title for html page.
#
title "Relatorio de Acesso a Internet"

# TAG:        font_face
#         Especify the font for html page.
#
font_face Tahoma

# TAG:        header_color
#         Especify the header color
#
header_color darkblue

# TAG:        header_bgcolor
#         Especify the header bgcolor
#
header_bgcolor blanchedalmond

# TAG:        font_size
#         Especify the text font size
#
font_size 9px

# TAG:        header_font_size
#         Especify the header font size
#
header_font_size 9px

# TAG:        title_font_size
#         Especify the title font size
#
title_font_size 11px
```

```
# TAG:        background_color
# TAG:        background_color
#        Html page background color\#
background_color white

# TAG:        text_color
#        Html page text color
#
text_color #000000

# TAG:        text_bgcolor
#        Html page text background color
#
text_bgcolor lavender

# TAG:        title_color
#        Html page title color
#
title_color green

# TAG:        logo_image
#        Html page logo.
#
logo_image none

# TAG:        logo_text
#        Html page logo text.
#
#logo_text ""

# TAG:        logo_text_color
#        Html page logo texti color.
#
logo_text_color #000000

# TAG:        logo_image_size
#        Html page logo image size.
#       width height
#
image_size 80 45

#TAG:        background_image
#        Html page background image
```

```
#
background_image none

# TAG:  password
#       User password file used by Squid authentication scheme
#       If used, generate reports just for that users.
#
password none

# TAG:  temporary_dir
#       Temporary directory name for work files
#       sarg -w dir
#
temporary_dir /tmp

# TAG:  output_dir
#       The reports will be saved in that directory
#       sarg -o dir
#
output_dir /usr/local/www/data/relatorios

# TAG:  output_email
#       Email address to send the reports. If you use this tag, no html reports will be generate
#       sarg -e email
#
#output_email none

# TAG:  resolve_ip yes/no
#       Convert ip address to dns name
#       sarg -n
resolve_ip no

# TAG:  user_ip yes/no
#       Use Ip Address instead userid in reports.
#       sarg -p
user_ip yes

# TAG:  topuser_sort_field field normal/reverse
#       Sort field for the Topuser Report.
#       Allowed fields: USER CONNECT BYTES TIME
#
topuser_sort_field BYTES reverse

# TAG:  user_sort_field field normal/reverse
```

```
#       Sort field for the User Report.
#       Allowed fields: SITE CONNECT BYTES TIME
#
user_sort_field BYTES reverse

# TAG:  exclude_users file
#       users within the file will be excluded from reports.
#       you can use indexonly to have only index.html file.
#
#exclude_users none

# TAG:  exclude_hosts file
#       Hosts, domains or subnets will be excluded from reports.
#
#       Eg.: 192.168.10.10 - exclude ip address only
#            192.168.10.0  - exclude full C class
#            s1.acme.foo   - exclude hostname only
#            acme.foo      - exclude full domain name
#
#exclude_hosts none

# TAG:  date_format
date_format e

# TAG: lastlog n
lastlog 0

# TAG: remove_temp_files yes
remove_temp_files yes

# TAG: index yes|no|only
index yes

# TAG: overwrite_report yes|no
overwrite_report yes

# TAG: records_without_userid ignore|ip|everybody
records_without_userid ip

# TAG: use_comma no|yes
use_comma no

# TAG: topsites_num n
#      How many sites in topsites report.
```

```
#
topsites_num 100

# TAG: topsites_sort_order CONNECT|BYTES A|D
topsites_sort_order BYTES D

# TAG: index_sort_order A/D
index_sort_order D

# TAG: max_elapsed milliseconds
#      If elapsed time is recorded in log is greater than max_elapsed use 0 for elapsed time.
#      Use 0 for no checking
#
max_elapsed 28800000
# 8 Hours

# TAG: report_type type

report_type topusers topsites sites_users users_sites date_time denied auth_failures site_user_t

# TAG: usertab filename
#      You can change the "userid" or the "ip address" to be a real user name on the reports.
#      Table syntax:
#                  userid name   or   ip address name
#      Eg:
#                SirIsaac Isaac Newton
#                vinci Leonardo da Vinci
#                192.168.10.1 Karol Wojtyla
#
#      Each line must be terminated with '\n'
#
#usertab none

# TAG: long_url yes|no
long_url no

# TAG: charset name
charset Latin1

# TAG: show_successful_message yes|no
#      Shows "Successful report generated on dir" at end of process.
#
show_successful_message yes
```

```
# TAG: show_read_statistics yes|no
#      Shows some reading statistics.
#
#show_read_statistics yes

# TAG: topuser_fields
#      Which fields must be in Topuser report.
#
topuser_fields NUM DATE_TIME USERID CONNECT BYTES %BYTES IN-CACHE-OUT USED_TIME MILISEC %TIME TO

# TAG: user_report_fields
#      Which fields must be in User report.
#
user_report_fields CONNECT BYTES %BYTES IN-CACHE-OUT USED_TIME MILISEC %TIME TOTAL AVERAGE

# TAG: bytes_in_sites_users_report yes|no
#      Bytes field must be in Site & Users Report ?
#
#bytes_in_sites_users_report no

# TAG: topuser_num n
#      How many users in topsites report. 0 = no limit
#
topuser_num 0

# TAG: site_user_time_date_type list|table
#      generate reports for site_user_time_date in list or table format
#
site_user_time_date_type table

# TAG: datafile file
#      Save the report results in a file to populate some database
#
#datafile none

# TAG: datafile_delimiter ";"
#      ascii character to use as a field separator in datafile
#
#datafile_delimiter ";"

# TAG: datafile_fields all
#      Which data fields must be in datafile
#      user;date;time;url;connect;bytes;in_cache;out_cache;elapsed
#
```

```
#datafile_fields user;date;time;url;connect;bytes;in_cache;out_cache;elapsed

# TAG: datafile_url ip|name
#      Saves the URL as ip or name in datafile
#
#datafile ip

# TAG: weekdays
#      The weekdays to take account ( Sunday->0, Saturday->6 )
# Example:
#weekdays 1-3,5
# Default:
#weekdays 0-6

# TAG: hours
#      The hours to take account
# Example:
#hours 7-12,14,16,18-20
# Default:
#hours 0-23

# TAG: dansguardian_conf file
#      DansGuardian.conf file path
#      Generate reports from DansGuardian logs.
#      Use 'none' to disable it.
#      dansguardian_conf /usr/dansguardian/dansguardian.conf
#
#dansguardian_conf none
# TAG: show_sarg_info yes|no
#      shows sarg information and site path on each report bottom
#
#show_sarg_info yes

# TAG: show_sarg_logo yes|no
#      shows sarg logo
#
#show_sarg_logo yes

# TAG: www_document_root dir
#     Where is your Web DocumentRoot
#     Sarg will create sarg-php directory with some PHP modules:
#     - sarg-squidguard-block.php - add urls from user reports to squidGuard DB
#
#www_document_root /var/www/html
```

```
# TAG: user_authentication yes|no
#     Allow user authentication in User Reports using .htaccess
#     Parameters:
#        AuthUserFile         - where the user password file is
#        AuthName        - authentication realm. Eg "Members Only"
#        AuthType        - authenticaion type - basic
#        Require                - authorized users to see the report.
#
#
# user_authentication no
# AuthUserFile /usr/local/sarg/passwd
# AuthName "SARG, Restricted Access"
# AuthType Basic
# Require user admin
```