Vinícius Graciano Elias

Servidor de E-mail - Postfix/Amavisd-new/SpamAssassin/ClamAV

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
2005

Vinícius Graciano Elias

Servidor de E-mail - Postfix/Amavisd-new/SpamAssassin/ClamAV

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 *abril 2005*

Prof. Ricardo Martins de Abreu Silva

Prof. Gustavo Guimarães Parma

Prof. Joaquim Quinteiro Uchôa
(Orientador)

Lavras
Minas Gerais - Brasil

Agradecimentos

Gostaria de agradecer a Administração do Tribunal Regional do Trabalho da 18 Região, especialmente ao Secretário de Tecnologia da Informação Sr. Humberto Magalhães Ayres, por me proporcionarem este curso. Também agradeço ao Prof. Joaquim Quintero Ûchoa, que em seu papel de orientador, mostrou-se sempre muito disposto, motivador e profissional. Não posso esquecer também a ajuda e compreensão de todos os meus familiares e amigos, que me apoiaram muito durante a confecção deste trabalho.

Resumo

O objetivo desta monografia é a instalação de um servidor de *e-mail* em ambiente linux, utilizando ferramentas livres, que englobe antivírus e antispam.

A minha esposa Wanessa, a minha família, amigos e a Comunidade Linux.

Sumário

Lista de Figuras

Capítulo 1

Introdução

A utilização de serviços de *e-mail* por uma empresa e seus funcionários, é cada vez mais importante e decisiva no mundo dos negócios. Para que seja possível esta interação com o mundo dos *e-mails*, as empresas optam por algumas soluções, como: a terceirização do serviço de *e-mail* por uma empresa especializada; a compra de um *software* de serviço de *e-mail* proprietário, e por último a construção de um servidor de e-mail através da junção de algumas ferramentas livres.

Dentro das propostas apresentadas, as soluções proprietárias na maioria das vezes são cobradas por conta de *e-mail* criada, sem falar que ainda deve-se pensar em ferramentas de controle de vírus e *spam*[1](LINDBERG, 1999).

A proposta dessa monografia será montar um servidor de *e-mail* em conjunto com controle de *spam* e vírus, utilizando-se ferramentas livres e conhecidas da Comunidade Linux. Assim o único gasto será o treinamento do pessoal que irá cuidar do serviço de *e-mail*.

Os primeiros capítulos desta monografia, irão apresentar um pouco das ferramentas que serão configuradas, a saber: Postfix, Amavisd-new, ClamAV e SpamAssassin. Como o objetivo é apenas instalar e fazer a configuração básica do servidor, que já englobe o antivírus e antispam, os capítulos que se seguem apenas ilustram um pouco o potencial de cada ferramenta.

[1]Termo pelo qual é comumente conhecido o envio, a uma grande quantidade de pessoas de uma vez, de mensagens eletrônicas, geralmente com cunho publicitário, mas não exclusivamente. O *spam* também é conhecido pela sigla inglesa UCE (*Unsolicited Commercial Email*, ou Mensagem Comercial Não-Solicitada).

Capítulo 2

Servidor de E-mail

2.1 Funcionamento do Correio Eletrônico

Saber como funciona o serviço de entrega e recebimento de *e-mail's* é essencial para correta configuração de um servidor de correio eletrônico. Para que uma pessoa possa enviar um *e-mail* para outra pessoa, é necessário a utilização de um *software* cliente de *e-mail*, conhecido como MUA (Agente de Mensagens do Usuário) ex.: *Thunderbird*[1]*, Eudora*[2]*, Outlook*[3]*, Pine*[4] *e Evolutions*[5].

Através do MUA, a pessoa escreve o seu *e-mail* e o encaminha ao MTA (Agente Transportador de Mensagens) ex.: *Postfix*[6]*, SendMail*[7]*, Qmail*[8] *e Exchange*[9], que está configurado em seu MUA. De posse da mensagem, o MTA do remetente irá verificar se o endereço de *e-mail* do destinatário lhe pertence (está no mesmo servidor de e-mail), neste caso a mensagem é simplesmente entregue na caixa postal do destinatário. Caso o MTA de destino não seja o mesmo, o MTA do remetente enviará a mensagem ao MTA do destinatário (em outro servidor).

A transferência do *e-mail* entre o MUA e o MTA é efetuada utilizando-se o protocolo SMTP (POSTEL, 1982), Protocolo Simples de Transferência de Mensa-

[1] http://www.mozilla.org/products/thunderbird/
[2] http://www.eudora.com/
[3] http://office.microsoft.com/
[4] http://www.washington.edu/pine/
[5] http://gnome.org/projects/evolution/
[6] http://www.postfix.org/
[7] http://www.sendmail.org/
[8] http://www.qmail.org/
[9] http://www.microsoft.com/exchange/

gens. O protocolo SMTP será utilizado também entre o MTA do remetente e o MTA do destinatário.

O servidor de *e-mail* do destinatário ao receber uma mensagem para um dos seus usuários simplesmente a coloca na caixa postal deste usuário. Dependendo do servidor de *e-mail*, a caixa postal pode ser um arquivo ou um diretório dentro do diretório particular do usuário (*home*[10]). Se o usuário possuir uma conta *shell*[11] neste servidor ele poderá ler os seus *e-mails* no servidor, seja diretamente ou por acesso remoto. Caso contrário o usuário deverá transferir suas mensagens para sua máquina a fim de lê-las utilizando o seu cliente de *e-mail* (MUA). A transferência de mensagens recebidas entre o servidor e o cliente de *e-mail* requer a utilização de outros programas e protocolos.

Para a troca de mensagens entre o servidor e o cliente de *e-mail* é utilizado o protocolo POP (MYERS; ROSE, 1996), Protocolo de "Agência" de Correio, que recebe este nome por agir como uma agência de correios, guardando as mensagens dos usuários em caixas postais e aguardando que estes venham buscá-las. Outro protocolo que pode ser utilizado para este mesmo fim é o IMAP (CRISPIN, 2003), Protocolo para Acesso de Mensagens via Internet, que implementa além das funcionalidades fornecidas pelo POP muitos outros recursos(ENGINEERING, 2003).

Os protocolos POP e IMAP são protocolos para recebimentos de mensagens, ao contrário do protocolo SMTP que serve para enviar mensagens, logo, possuem funcionalidades diferenciadas, como, por exemplo, autenticação do usuário. Para a utilização dos protocolos POP e IMAP (GRAY, 1995) faz-se necessária a instalação e configuração dos servidores apropriados, bem como a correta configuração, que vão ser responsáveis por atender as solicitações do cliente de *e-mail* por novas mensagens.

O recebimento de mensagens, se dá através da solicitação do MUA do usuário ao seu servidor de *e-mail*, que após a autenticação do usuário vai informar se existem mensagens em sua caixa postal e quantas são. A seguir o MUA solicita a transferência das mensagens para a máquina local (usando o protocolo POP), ou manipula as mensagens diretamente no servidor (usando o protocolo IMAP), finalizando assim o processo de troca de mensagens entre duas pessoas.

[10] Nome dado ao diretório padrão de um usuário do sistema operacional Linux.

[11] Tipo de conta do linux, que permite ao usuário se conectar ao servidor diretamente, utilizando linhas de comando. Exemplo de shell: *bash*.

A figura 2.1 mostra como seria o envio de um *e-mail* de um MTA a outro MTA, ou seja, o envio de um *e-mail* entre duas pessoas que possuem diferentes servidores de e-mail fisicamente.

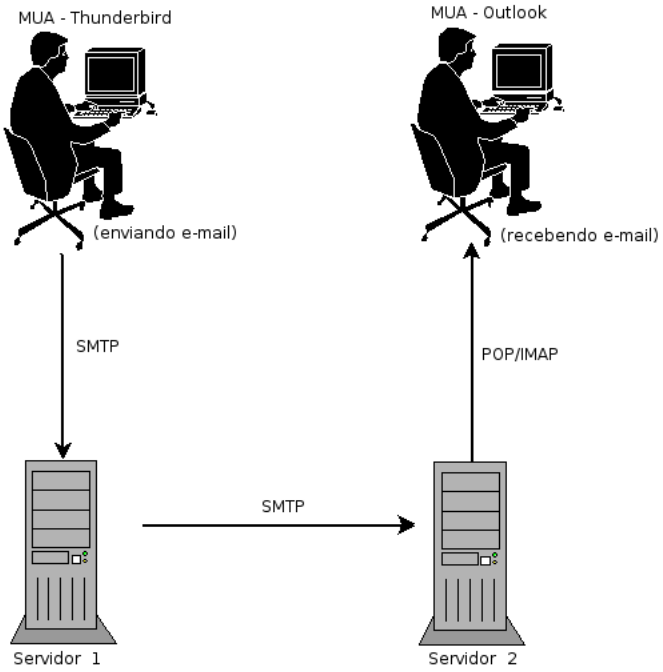

Figura 2.1: Transferências de Mensagens entre MTA's diferentes

2.2 E-mail

E-mail ou *email* é a forma resumida para "*electronic mail*", que é o método de se compor, enviar e receber mensagens através de sistemas eletrônicos de comunicação. Hoje a maioria desses sistemas de *e-mails* utilizam a Internet. O formato do e-mail para a Internet é definido atualmente na RFC 2822 (RESNICK, 2001), antes era descrito na RFC 822 (CROCKER, 1982).

Uma mensagem de e-mail consiste de duas partes essenciais:

- Cabeçalho (*Headers*) - Sumário da mensagem, remetente, destinatário, e outras informações a respeito do *e-mail*.

- Corpo (*Body*) - A própria mensagem, usualmente contendo um bloco de assinatura no final.

O cabeçalho contem pelo menos quatro campos:

- De(*From*) - O endereço de e-mail do remetente da mensagem.
- Para(*To*) - O endereço de e-mail do destinatário da mensagem.
- Assunto(*Subject*) - Um pequeno sumário do conteúdo da mensagem.
- Data (*Date*) - A data e hora local, de quando a mensagem foi enviada.

Outros campos comuns no cabeçalho, são:

- *Cc - Carbon copy*, lista de endereços de *e-mail* que receberão uma cópia da mensagem, sabendo da existência um dos outros.
- *Bcc - Blind carbon copy*, lista de endereços de *e-mail* que receberam cópia da mensagem, e saberão qual o endereço que estava no campo *Para*, mas não saberão os endereços do campo *Bcc*.
- *Received* - Informação de rastreamento gerada pelo servidor de *e-mail* que, previamente, entregou a mensagem.
- *Content-Type* - Informação de como a mensagem deve ser mostrada, usualmente um tipo MIME.

2.2.1 MIME

Praticamente todos os *e-mails* são transmitidos via SMTP no formato MIME (*Multipurpose Internet Mail Extensions*) (FREED INNOSOFT, 1996), um "padrão da Internet" para o formato de *e-mail*. O protocolo básico de transmissão de *e-mail* da Internet, o SMTP, suporta apenas caracteres *ASCII* de 7-bit, o que limita a transmissão de e-mails que possuam apenas caracteres da lingua inglesa.

MIME define mecanismos para enviar outros tipos de informação no *e-mail*, como caracteres em outras linguas além da inglesa (*non-ASCII* - caracteres não ASCII), conteúdo binário de de 8-bit, como arquivos contêndo imagens, sons, filmes e programas de computador.

2.3 Protocolos

2.3.1 SMTP

Em 1982 foi definido o protocolo cliente/servidor SMTP, onde o *host* que inicia o contato é o cliente e o *host* que é contatado é o servidor. O servidor só responde as perguntas feitas pelo cliente, sendo assim o cliente deve prestar bastante atenção as respostas do servidor, para saber como melhor interagir com o servidor.

Quando um *host* quer mandar um *e-mail* para algum lugar na Internet, este determina para onde o *e-mail* deve ir, e inicia o contato com o servidor de destino do *e-mail*. O remetente do *e-mail* é sempre o SMTP cliente, e o *host* que está escutando (*listening*) pelo tráfego SMTP é sempre o servidor.

O *host* remetente usa o *Domain Name System (DNS)* para determinar o endereço IP do *host* destinatário, e contata este usando o protocolo SMTP na porta TCP de número 25.

2.3.2 POP

Depois que o servidor SMTP (MTA) final recebeu a mensagem e a guardou, o receptor necessita poder alcançar a mensagem. Antigamente, e mesmo hoje em dia, os *e-mails* eram guardados em arquivos textos, e os e-mails eram lidos de fato lendo-se os arquivos de texto. É importante observar que mesmo utilizando o MIME, o que ocorre é apenas uma codificação dos dados binários da mensagem para o formato texto.

Agora se o receptor do e-mail não estiver conectado diretamente no computador onde está o arquivo texto, ler esse arquivo pode ficar muito difícil. Para aliviar esse problema foi criado o *Post Office Protocol (POP)*, que introduziu um modelo cliente/servidor para os MUA's. Assim os MUA's através do POP são capazes de pegar os e-mails do servidor MTA e guardá-los localmente na máquina onde estão instalados, para posterior leitura, sem a necessidade de estar conectado ao MTA o tempo todo.

A conexão usando o protocolo POP é feita através do fornecimento de usuário e senha pelo MUA ao MTA, estando corretos os dados informados pelo POP, este checa se existem mensagens na caixa postal do usuário, e então as transfere para o computador local, uma a uma. O cliente MUA tem ainda a opção de deixar as mensagens que foram lidas no servidor ou apagá-las depois que as mesmas forem transferidas.

2.3.3 IMAP

O POP funciona bem para muitos casos, é indicado para aqueles que possuem - por exemplo - uma linha discada *(dial-up)* com a Internet e não podem gastar horas de telefone e conta para ler seus e-mails.

Problemas do POP: não é possível ver uma mensagem no servidor antes de fazer o *download* da mesma; o receptor tem apenas uma caixa postal sem nenhuma estrutura hierárquica no servidor; se optar por deixar todas as suas mensagens no servidor, faz com que o download de novas mensagens se torne um tanto lento.

Para contornar estes problemas criou-se o *Internet Message Access Protocol (IMAP).* O protocolo IMAP dá a seus usuários muito mais poder que o protocolo POP, no controle de seus *e-mails*, como permitir deixar parte destes localmente e outra parte no servidor. Através do IMAP, por exemplo, é possível o compartilhamento de caixas postais entre usuários, bem como a criação de estruturas (pastas/-diretórios) no próprio servidor, para organizar as mensagens.

O IMAP é mais administrável e possuí a facilidade de permitir o acesso às mensagens de qualquer computador que possua uma conexão com a internet, já que as mensagens ficam guardadas no servidor de *e-mail*.

Capítulo 3

Postfix

3.1 Histórico

Em 1997, no Centro de Pesquisa Thomas J.Watson, nasce o servidor de *e-mail* VMailer, concebido pelo Ph.D. Wietse Zweitze Venema, enquanto trabalhava, para a gigante IBM (*International Business Machines Corp.*). Venema's é mais conhecido pelos programas que escreveu para proteger sistemas, de intrusos da Internet como o *Security Administrator Tool for Analyzing Networks* (SATAN) e *TCP Wrapper*.

O VMailer seria uma alternativa ao já conhecido servidor de *e-mail*, usado em 70% dos casos, Sendmail. O VMailer teria de ser rápido, fácil de se administrar e seguro, além de ser compatível suficientemente com o Sendmail, afim de atrair seus usuários, e garantir uma migração tranqüila.

A proposta de Venema's para o VMailer era que este servidor de *e-mail*, seria uma coleção de pequenos programas, relativamente simples e rápidos, que trabalhariam juntos para fazer o trabalho que o Sendmail já fazia. Diferentemente do Sendmail que é um grande programa monolítico, com uma linguagem de configuração/programação bastante confusa e complexa.

Em dezembro de 1997 Venema's desligou o Sendmail e iniciou os primeiros testes no VMailer, colocando-o como servidor do *porcupine.org*, seu site pessoal (LAIRD, 1998). Praticamente um ano depois, em dezembro de 1998, como um presente de Natal, a IBM e Wietse Zweitze Venema liberaram o código do VMailer na Internet, de onde o VMailer assumiria o nome conhecido de Postfix (MARKOFF, 1998).

3.2 Configuração

Os dois mais importantes arquivos de configuração do Postfix são: *main.cf* e *master.cf*, e estão localizados no diretório */etc/postfix/*. Estes arquivos devem pertencer ao usuário *root*. Para qualquer alteração feita nos arquivos citados acima, deve-se executar o comando: *postfix reload* para que as alterações sejam implantadas no servidor em execução.

3.2.1 O Arquivo main.cf

Formato

O arquivo de configuração *main.cf* do Postfix, especifica uma pequena porção de parâmetros que controlam o sistema de *e-mail* do Postfix. Os parâmetros que não forem especificados explicitamente neste arquivo, serão deixados com o seu valor padrão.

Cada linha lógica do *main.cf* está no formato [parâmetro = valor]. Os espaços, ao redor do "=" são ignorados, bem como os espaços ao final das linhas lógicas. Linhas em branco e linhas formadas apenas de espaços em brancos, também são ignoradas, bem como linhas que tenham como seu primeiro caracter o símbolo "#". O valor de um parâmetro pode fazer referencia a um outro parâmetro.

A expressão "$name","$name" ou "$(name)" são recursivamente trocadas pelo valor do parâmetro nomeado. A expressão "$name?value" muda para "value" quando "$name" não está vazio. A expressão "$name:value" muda para "value" quando "$name" está vazio. Os duas expressões anteriores são suportadas apartir da versão 2.2 do Postfix.

Para vizualizar todos os parâmetros e seus valores configurados no Postifix, é usado o comando *"postconf -d"*.

Alguns Parâmetros

- *myhostname* - Nome do servidor de *e-mail* na internet. O padrão é o uso do FQDN (*fully-qualified domain name*), Ex.: *myhostname = host.domain.tld*

- *mydomain* - Nome do domínio do servidor de *e-mail*. Padrão é o uso do FQDN menos o primeiro componente. Ex.: *mydomain = domain.tld*

- *myorigin* - Especifica o nome de domínio a ser usado nos *e-mails* enviados por este servidor de *e-mail*, o padrão é o uso da variável $myhostname. Ex.:

  - *myorigin = $myhostname (padrão: "user@$myhostname")*
  - *myorigin = $mydomain (mais usado: "user@$mydomain")*

- *mydestination* - Define quais os domínios que este servidor irá entregar *e-mails* localmente, ou seja, as caixas postais cujo destino são dele. O padrão é receber *e-mails* para a própria máquina. Ex.:

  - *mydestination = $myhostname localhost.$mydomain localhost*
  - *mydestination = $myhostname localhost.$mydomain localhost $mydomain*
  - *mydestination = $myhostname localhost.$mydomain localhost www.$mydomain ftp.$mydomain*

- *mynetworks_style* e *mynetworks* - Definem uma lista de clientes SMTP "confiáveis", que possuem mais privilégios que clientes "estranhos" (desconhecidos). Em particular clientes SMTP "confiáveis" estão autorizados a retransmitir suas mensagens atráves do Postfix. Caso seja usado o *mynetworks* o Postfix ignora o *mynetworks_style*, no *mynetworks* os *ip's* devem ser especificados de acordo com a notação CIDR-*Classless Inter-Domain Routing*(rede/mascara)(FULLER BARRNET, 1993) separados por espaço ou vírgula. Os possíveis valores para mynetworks_style são: *host* (autoriza apenas a própria maquina), *subnet* (autoriza clientes SMTP da sub-rede, onde a maquina está conectada) e *class* (autoriza os clientes SMTP que estejam na mesma classe de IP A/B/C da maquina servidora). Ex.:

  - *mynetworks_style = subnet (padrão: autoriza sub-redes)*
  - *mynetworks_style = host (seguro: autoriza apenas a máquina local)*
  - *mynetworks = 127.0.0.0/8 (seguro: autoriza apenas a máquina local)*
  - *mynetworks = 127.0.0.0/8 168.100.189.2/32 (autoriza máquina local, e outra)*

- *relay_domains* - Define os destinos remotos, para os quais o Postfix estará autorizado a retransmitir mensagens de clientes SMTP "estranhos"(clientes que não estajam listados em *mynetworks* ou *mynetworks_style*). Por padrão

o Postfix retransmitirá as mesagens de clientes SMTP "estranhos" apenas aos domínios especificados no parâmetro *relay_domains*. Ex.:

  - *relay_domains = $mydestination (padrão)*
  - *relay_domains = (seguro: não faz retransmissão para estranhos)*
  - *relay_domains = $mydomain (encaminha e-mails para meu domínio e sub-domínio)*

- *relayhost* - Por padrão o Postifix tenta entregar os *e-mail* diretamente à Internet. Dependendo da situação isto pode não ser possível, ou desejado. Então pode-se definir para onde o Postifix encaminhará os e-mails. Ex.:

  - *relayhost = (padrão: diretamente para a Internet)*
  - *relayhost = $mydomain (entrega via mailhub local)*
  - *relayhost = [mail.$mydomain] (entrega via mailhub local)*
  - *relayhost = [mail.isp.tld] (entrega via provedor do mailhub)*

- *notify_classes* - Define os níveis de notificações de problemas, que deverão ser encaminhas ao "postmaster"[1]. Ex.: *notify_classes = resource, software*.

- *proxy_interfaces* - Caso o servidor de *e-mail* esteja conectado a Internet através de um *proxy* ou de um NAT(*network address translator*), deve-se especificar este parâmetro. Ex.: *proxy_interfaces = 1.2.3.4 (endereço de rede do proxy/NAT)*.

- *inet_interfaces* - Define quais são as interfaces de rede que o Postfix deverá monitorar(*listen on*) para o recebimento de *e-mails*. Por padrão o Postfix é configurado para monitorar todas as suas interfaces de rede. Ex.: *inet_interfaces = all*.

Logs do Postfix

Os arquivos de log do Postfix são gerenciados pelo servidor syslog do linux. Para que o Postfix gere os logs corretamente, deve-se criar um arquivo para guardar os logs[2] (*/var/log/maillog*) e acrescentar ao arquivo de configuração do syslog (*/etc/syslog.conf*) as seguintes linhas: *1) mail.err /dev/console*; *2) mail.debug -/var/log/maillog*, não esquecendo de reiniciar o servidor ao final.

[1] Conta de *e-mail* do administrador do Postfix, deve ser criado um *alias* para um *e-mail* real, no arquivo */etc/aliases*.

[2] Registros de atividades

3.3 Arquitetura

Esta seção apresenta a arquitetura de recebimento e envio de email no Postfix. Ela encontra-se baseada em (POSTFIX.ORG, 2005b).

3.3.1 Como o Postfix recebe *e-mails*

Quando uma mensagem entra no sitema de correio do Postfix, a primeira parada é o *incoming queue* (fila de entrada). A figura 3.1 mostra como é processado um novo *e-mail*. Os nomes seguidos de números são comandos do Postfix ou programas servidores, enquanto que os nomes sombreados (sem números) são filas do Postfix.

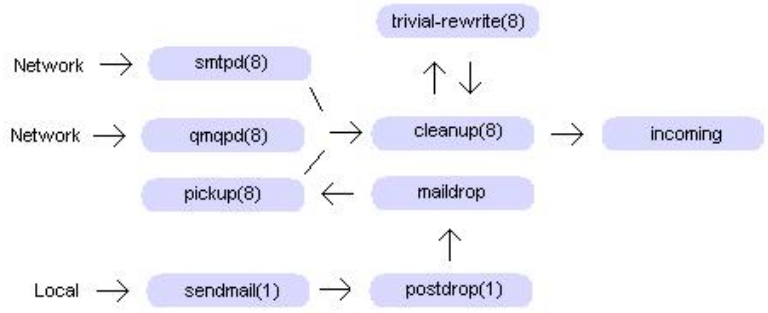

Figura 3.1: Postfix - Recebimento de mensagem

- Os *e-mails* da rede entram no Postfix por meio dos servidores *smtpd(8)* ou *qmqpd(8)*. Estes servidores removem os protocolos de encapsulamento (SMTP ou QMQP), fazem algumas checagens de "saneamento" para proteger o Postfix, e entregam o remetente, os recipientes e o conteúdo da mensagem para o servidor *cleanup(8)*.

- As submissões locais são recebidas com o comando de compatibilidade *sendmail(1)* do Postfix, e são enfileiradas na fila do *maildrop* pelo comando *postdrop(1)*. Este processo ocorre mesmo quando o sistema de correio do Postfix não estiver executando. O servidor local *pickup(8)*, pega as submissões locais, realiza algumas checagens de "saneamento" para proteger o Postfix, e entrega o remetente, os recipientes e o conteúdo da mensagem ao servidor *cleanup(8)*.

- Correio de fontes internas são entregues diretamente ao servidor *cleanup(8)*. Estas fontes não são mostradas na figura, e incluem: correio que é enviado

pelo agente de entrega *local(8)*, mensagens que são retornadas ao remetente pelo servidor *bounce(8)*, e notificações do postmaster sobre problemas com o Postfix.

- O servidor *cleanup(8)* implementa o estágio final do processamento antes que a mensagem seja enfileirada. Este acrescenta mensagens de cabeçalho e faz reescrita de endereços. Opcionalmente, o servidor *cleanup(8)*, pode ser configurado para fazer uma leve inspeção no documento usando expressões regulares. O servidor *cleanup(8)* coloca o resultado de seu processo como um único arquivo na *incoming queue* (fila de entrada), e notifica o gerente de filas (*queue manager*) da chegada de um novo *e-mail*.

- O servidor *trivial-rewrite(8)* reescreve o endereço para a forma padrão *user@fully.qualified.domain.*

3.3.2 Como o Postfix entrega *e-mails*

Assim que a mensagem alcança a *incoming queue* (fila de entrada) o próximo passo é entregá-la. A figura 3.2 mostra os principais componentes do aparato de entrega de *e-mails* do Postfix. Nomes seguidos de números são servidores ou comandos, enquanto nomes sombreados (sem números) representam filas do Postfix.

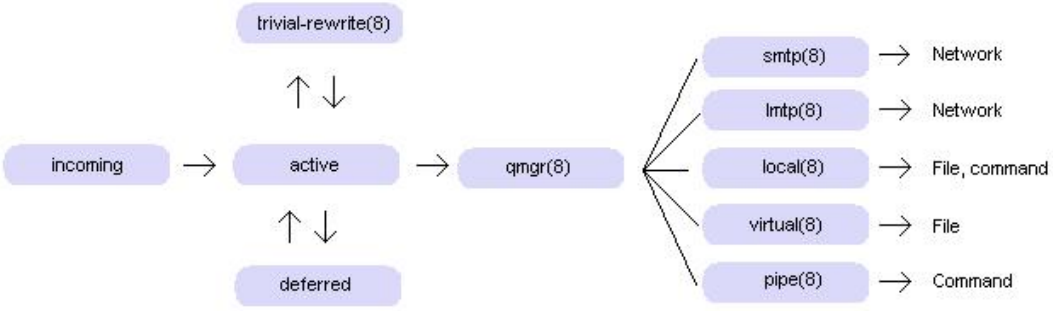

Figura 3.2: Postfix - Envio de mensagem

- O gerente de fila (o processo servidor *qmgr(8)* na figura) é o coração da entrega de correio do Postfix. Ele contata os agentes de entrega *smtp(8), lmtp(8), local(8), virtual(8), pipe(8), discard(8) ou error(8)*, e envia uma requisição de entrega para um ou mais endereços de recipientes. Os agentes de entrega *discard(8) e error(8)* são especiais: eles descartam ou pulam todos os *e-mails*, estes não são mostrados na figura acima.

- O gerente de fila mantem uma pequena fila ativa (*active queue*) com as mensagens que foram abertas para entrega. A fila ativa age como uma janela limitada em filas de entradas (*incoming*) ou adiadas (*deferred*) potencialmente grandes. A limitada fila ativa previne o gerente de filas de ficar sem memória quando tiver uma carga de *e-mails* muito pesada.

- O gerente de filas mantem uma fila de adiados (*deferred queue*) para o correio que não pode ser entregue, de modo que uma reserva grande do correio não retarde o acesso normal à fila.

- O servidor *trivial-rewrite(8)* resolve cada endereço recipiente de acordo com o seu endereço de classe local e remota.

- O cliente *smtp(8)* olha uma lista de possíveis correios para o host destino, organiza a lista por preferências, e testa cada servidor por vez até encontrar um que responda. Então encapsula o remetente, o recipiente e o conteúdo da mensagem como requerido pelo protocolo SMTP.

- O cliente *lmtp(8)* fala um protocolo similar ao SMTP, que é melhorado para entrega a caixas postais (*mailbox*) de servidores como o *Cyrus*.

- O agente de entrega *local(8)* entende *UNIX-style mailboxes*, arquivos *qmail-compatible maildir* , *Sendmail-style system-wide aliases(5) databases,* e arquivos *Sendmail-style per-user .forward*. Multiplos agentes de entrega local podem ser executados em paralelo, mas entrega paralela para o mesmo usuário usualmente é limitada.

- O agente de entrega *virtual(8)* é uma agente de entrega que só compreende caixas-postais no estilo Unix ou diretórios de *e-mail* no estilo *qmail*. Este agente de entrega pode entregar *e-mail* para múltiplos domínios, o que o faz especial para servir muitos domínios pequenos em uma única máquina.

3.4 Comentários Finais

Após a conclusão deste capítulo tem se a idéia de o quanto que o Postfix é robusto e modular, e de fácil configuração. Sendo muito rápido colocá-lo para funcionar, com apenas algumas modificações simples de configuração. Mais informações podem ser vistas em (POSTFIX.ORG, 2005e) e (DENT, 2003).

Capítulo 4

Aplicativos para Detecção de Vírus e Spam

4.1 Comentários Iniciais

Este capítulo destina-se a apresentação de 3 ferramentas que integram a solução proposta para o controle de vírus e *spam*. Estas ferramentas são: ClamAV[1] (Anti-Vírus de linha de comando), SpamAssassin[2] (Anti-Spam um projeto da *Apache Software Foundation*) e o Amavisd-new[3] (a interface entre o MTA e os verificadores de conteúdo).

4.2 ClamAV

4.2.1 Introdução

Clam AntiVirus é uma ferramenta de antivírus GPL para UNIX. O objetivo principal desse *software* é a sua integração com servidores de *e-mail* (varredura de anexos). O pacote provê um servidor multitarefa bastante flexível, um *scanner* de linha de comando, e uma ferramenta para atualização automática via Internet.

Apesar do objetivo principal do ClamAV ser a sua integração com servidores de *e-mail*, este também pode ser usado em *on-access scanning*[4], em sistemas Linux

[1] http://www.clamav.net/
[2] http://spamassassin.apache.org/
[3] http://www.ijs.si/software/amavisd/
[4] varreadura de vírus no acesso do arquivo

e FreeBSD, utilizando-se uma *thread* do *clamd* chamada Clamuko, o que pode ser visto em (KOJM, 2004).

Os programas são baseados em uma biblioteca compartilhada, distribuída com o pacote Clam Anti Vírus, a qual pode ser usada em qualquer *software*. O mais importante, o banco de dados do ClamAV esta sendo sempre atualizado, garantindo assim o *download* frequente de *updates*atualizações por seus usuários, o que será comentando na seção 4.2.2. Entre as empresas que utilizam o ClamAV, encontram-se SourceForge [5], Michigan State University[6] e FastMail[7] (ANTIVIRUS, 2005). Entre as características do ClamAV, encontram-se:

- *scanner* de linha de comando
- servidor rápido e muti-tarefa
- interface para Sendmail
- atualizador de bando de dados
- detecção de mais de 20000 vírus, *worms* e *trojans*
- suporte interno a arquivos RAR(2.0), Zip, Gzip, Bzip2, Tar e outros
- suporte interno a arquivos de *e-mails* dos tipos *mbox*, Maildir e *raw*.

4.2.2 Atualização Automática da Base de Vírus

O *freshclam* é a ferramenta de atualização padrão do banco de dados do Clam AntiVirus, e utiliza para isso endereço *database.clamav.net*. Esse endereço possui inúmeros servidores associados, via servidor DNS, o que funciona como um balanceamento de carga, selencionando automaticamente um banco de dados geograficamente localizado próximo ao cliente, para realizar os donwloads de *updates*. O *freshclam* pode operar de duas formas:

- interativo - linha de comando, explicitamente.
- servidor - sozinho e silenciosamente.

Um arquivo de *log*, deve ser criado para registro de *updates* do ClamAV, este de preferencia deve estar na pasta */var/log*, ter permissão igual a 600 (*chmod 600*

[5] http://sourceforge.net/forum/forum.php?forum_id=306329
[6] http://project.mail.msu.edu
[7] http://www.fastmail.fm

/var/log/clam-update.log), e pertencer ao usuário do ClamAV. Também deve se indicar o local e nome do arquivo de *log* criado na diretiva *UpdateLogFile* do arquivo de configuração *freshclam.conf*. Com estes requisitos satisfeitos, pode-se rodar o programa FreshClam no modo servidor: *freshclam -d*.

Outra forma é usar o *cron*. Acrescentando a linha *N * * * * /usr/local/bin/-freshclam –quiet* ao *crontab* do *root* ou do usuário do *clamav*, para checar por atualizações a cada hora. O caracter "N" deve ser um número entre 3 e 57. É recomendado não escolher multíplos de 10, devido ao grande número de clientes que já usam este intervalo de tempo. No arquivo de configuração existem variáveis que devem ser preenchidas, caso esteja-se usando um *proxy* para acesso a internet, conforme figura 4.1.

```
HTTPProxyPassword is enabled.
HTTPProxyServer myproxyserver.com
HTTPProxyPort 1234
HTTPProxyUsername myusername
HTTPProxyPassword mypass
```

Figura 4.1: Configuração de Proxy no *freshclam.conf*

Uma dica é colocar a deritiva *DNSDatabaseInfo current.cvd.clamav.net*, no arquivo de configuração *freshclam.conf*, isto garante a procura por um novo banco de dados, através de uma simples consulta DNS, o que possibilita que o *update* da base de dados seja feita em intervalos de até 4 vezes por hora.

4.3 SpamAssassin

4.3.1 Comentários Iniciais

Antes de existir o SpamAssassin, existia o "*filter.plx*" de Mark Jeftovic's, ambos escritos em Perl[8]. Este era um filtro de spam baseado em *context/keyword*, que utilizava as técnicas básicas utilizadas hoje no SpamAssassin: pontuação de spam, regras para pontuação, marcação de spam, e outros.

Junstin Mason usou o *filer.plx* por vários anos, acrescentando algumas funcionalidades, e trechos de código. Até que um dia decidiu fazer alguns melhoramentos e reescrever todo o código em uma Licença *open-source*, assim nasceu o

[8]PERL: Linguagem de de programação de scripts, muito utilizada no mundo Linux. Site: http://www.perl.org/.

SpamAssassin. (SPAMASSASSIN.APACHE.ORG, 2005)

SpamAssassin é um projeto maduro de código livre que serve como um filtro para reconhecer mensagens de *spam*, mais conhecidas como *unsolicited commercial e-mail* (UCE) - mensagem comercial não solicitada - O SpamAssassin usa uma variedade de mecanismos incluindo analises de texto e cabeçalho, filtro Bayesiano (KOHN, 2005), listas de DNS bloqueados, e filtros de banco de dados colaborativos. SpamAssassin é executado em um servidor, e filtra os *spams* antes que estes cheguem a *mailbox* do destinatário.

O SpamAssassin é um conjunto poderoso e flexivel de programas Perl. Diferentemente de antigos filtros de *spam*, o SpamAssassin usa a combinação de múltiplos resultados obtidos de vários tipos de verificação de *spam*, para pontuar e determinar se uma mensagem é ou não um *spam*. Entre as verificações primárias, encontram-se:

- Testes de cabeçalho
- Testes de frases no corpo do *e-mail* (MALTESTRETZ, 2004)
- *Whitelist/Blacklist* (FAYHE, 2004)
- *Whitelist/Blacklist* (KOHN MARCUS DRESSLER, 2004)
- Bancos de dados de identificação de *spam* (DCC, Pyzor, Razor2) (FORREST, 2004)
- Bloqueio de listas de DNS, conhecidas como RBL (*Realtime Blackhole List*) (MASON, 2005)

4.3.2 *Blacklist* e *Whitelist*

Blacklist e *Whitelist* são listas de endereços de remetentes de *e-mail*, onde a *Blacklist* é a lista de *spammers* [9] e a *Whitelist* é a lista de *hammers*[10]. Estas listas são uma maneira de dizer se uma mensagem é um *spam* ou um *ham* (não-*spam*), apenas examinando o endereço do remetente da mensagem e comparando-o com a *blacklist* e *whitelist*. Esta é uma forma rápida de verificação que descarta a verificação do conteúdo do *e-mail*.

[9] Remetentes de *spam's*
[10] Remetentes confiáveis

O problema é que o endereço do remetente é na maioria das vezes trocado em mensagens de *spam*, então o uso de whitelist pode ser uma forma perigosa de permitir o recebimento de correio. Já as *blacklist* tem sua utilidade: já que *spammers* não têm o desejo de ser um remetente em sua blacklist.

Deve ficar claro que estas listas (*black/white*) só afetam a verificação de *spam*. Elas não têm influência nas outras verificações como, de vírus, de *e-mails* banidos ou de cabeçalhos. *E-mails* infectados, de remetentes da *whitelist* serão bloqueados, se a política for bloquear vírus.

Outra coisa importante é que o endereço do remetente verificado é o que vem do protocolo SMTP. E este é conhecido como o endereço do remetente do envelope ou o caminho de retorno. Este endereço não necessariamente equivale ao endereço do autor do *e-mail* que está no cabeçalho (*From:*) ou ao endereço do remetente que está no cabeçalho (*Sender:*).

Isto fica mais óbvio com *e-mails* de listas de mensagens, onde os endereços dos remetentes dos envelopes são usualmente o endereço da pessoa que está enviando a mensagem. Este processo é mais rápido do que verificar o cabeçalho do *e-mail* a procura do remetente.

4.3.3 Pontuação de *SPAM* pelo SpamAssassin

Quando o *SpamAssassin* é chamado para analizar uma mensagem de *e-mail*, este após realizar vários testes, determina uma pontuação de *spam* para a mensagem, que é uma representação numérica do quão a mensagem deve ser considerada um *spam*. Quanto maior for este número, maior a chance da mensagem ser considerada um *spam*. Retornando assim a mensagem a quem o chamou (no caso deste trabalho, o Amavisd-new), para que este decida o que fazer com a mesma.

Números negativos ou pequenos, perto de zero, indicam uma mensagem limpa, normal, coloquialmente chamada de *ham*. A pontuação de *spam* é uma característica de toda a mensagem, e não depende das preferências do recipiente para qual se destina. O SpamAssassin é chamado apenas uma vez para cada mensagem, não importando para quantos recipientes (destinatários) a mensagem é destinada.

4.4 Amavisd-new

4.4.1 Introdução

Amavid-new[11] criado por Mark Martinec, Gerente de Sistemas do Instituto Jozef Stefan, na Eslovênia, é uma versão aperfeiçoada, após praticamente 3 anos de desenvolvimento, do Amavis[12]. O Amavis (*A Mail Virus Scanner*), por sua vez, surgiu em 1997 e foi mantido até 2000, por Mogens Kjaer, Carlsberg Laboratory, Jürgen Quade, Christian Bricart, Rainer Link, Lars Hecking e outros.

Amavisd-new é uma interface confiável e de alta-performance que fica entre um MTA e um ou mais verificadores de conteúdo: como anti-vírus, e/ou o SpamAssassin (verificador de *spam*). O Amavisd-new é "*plugado*" ao servidor de *e-mail*, e age assim que o mesmo tenta receber e/ou enviar uma mensagem. De posse da mensagem o Amavisd-new, desempacota, caso seja necessário, todos os anexos que fizerem parte da mesma e os encaminha aos serviços de controle de *spam* e anti-vírus que estiverem configurados, para que sejam tomadas as providências necessárias.

A comunicação do Amavisd-new e o Postfix é feita através do (E)SMTP (KLENSIN et al., 1995), LMTP (*Local Mail Transfer Protocol*) (MYERS, 1996) e até mesmo pelos programas ajudantes "*helper's*" (legados do Amavis). O Amavisd-new é colocado no encarregado central de entrega de correio, ou perto de um, não necessariamente no estágio final de entrega da mensagem.

O Amavisd-new é escrito em Perl assegurando confiabilidade, manutenibilidade e portabilidade elevadas. O arquivo de configuração do Amavisd-new é escrito diretamente em linguagem Perl, e encontra-se no diretório */etc* com o nome de *amavisd.conf*.

4.4.2 Pontuação de SPAM com Amavisd-new

Como já comentado na seção 4.3.3, quando o SpamAssassin pontua a mesagem, e a retorna ao Amavisd-new, este irá determinar o futuro curso da ação a ser tomada. O Amavisd-new compara a pontuação do *spam* com três níveis de valores numéricos: *tag, tag2 e kill*. Estes valores podem ser diferentes para cada recipiente, o que ocasiona ações futuras diferentes para cada recipiente. Se necessário, o processo

[11] http://www.ijs.si/software/amavisd/
[12] http://www.amavis.org/

de checagem do *e-mail* é quebrado em mais de uma transação para acatar as diferentes preferências de cada recipiente.

Nível *tag*

Se a pontuação de *spam* da mensagem for igual ou maior que o nível *tag*, campos de cabeçalho de *spam* (*X-Spam-Status, X-Spam-Level*) são inseridos para os recipientes locais; *"undef"* é interpretado como menor que qualquer pontuação de *spam*.

Nível *tag2*

Se a pontuação de *spam* da mensagem for igual ou maior que o nível *tag2*, os campos de cabeçalho de *spam* (*X-Spam-Status, X-Spam-Level, X-Spam-Flag e X-Spam-Report*) são inseridos para os recipientes locais, e *X-Spam-Flag e X-Spam-Status* recebem o valor *"YES"*.

Nível *kill*

Se a pontuação de *spam* da mensagem for igual ou maior que o nivel *kill*, a mensagem é bloqueada; e o remetente recebe uma notificação de não-entrege.

Alcançar o nível *kill* acarreta em:

- Mensagem vai para quarentena
- Administrador de *spam* recebe notificação
- O contador *ContentSpamMsgs* é incrementado
- É realizado o *Spam defanging*
- Remetente recebe uma notificação se o *warnspamsender* está como *true* (verdadeiro) e *$final_spam_destiny* está como *D_PASS*
- Se a mensagem não for entregue, o remetente recebe uma notificação de não-entregue

Por outro lado alcançar o nível *tag2*, apenas adiciona algumas marcas a mensagem, nas quais o receptor ou seu MUA podem decidir agir sobre estas ou não, como exemplo:

- Campo de cabeçalho de assunto é modificado
- Campos de cabeçalho *X-Spam-Flag e X-Spam-Status* recebem um *"Yes"*
- E o log de *e-mail* indica *"Passed SPAM"* ao invés de *"Passed CLEAN"*.

Se o receptor (ou seu MUA) decidir descartar um *e-mail* baseado na marcação do *tag2*, não existe maneira de recuperar este *e-mail* da quarentena, o remetente nunca será notificado, nem o administrador de *spam*. Para o MTA e o Amavisd-new, a mensagem foi entregue com sucesso, qualquer coisa que o MUA fizer com o *e-mail* é de sua própria responsabilidade.

4.4.3 Quarentena

O *e-mail* é posto em quarentena quando está infectado, foi banido, ou sua pontuação de *spam* para pelo menos um recipiente está acima ou igual ao nível *kill*. O parâmetro **quarantine_to* para cada recipiente (quando não está vazio), juntamente com o correspondente global **_quarantine_method*, determina o local da quarentena.

O parâmetro **_quarantine_method* pode ser considerado um ajuste estático, geralmente controlando o formato e a localização da quarentena no sistema. O parâmetro **quarantine_to* pode ser considerado a parte dinâmica da localização da quarentena, possivelmente afetada pelas configurações de cada recipiente. Isto serve para especificar a localização final, e.g. um arquivo ou uma *mailbox*[13] no sistema.

Dependendo da categoria do correio (tipo de *malware*), as seguintes variáveis especificam o método da quarentena.

- $virus_quarantine_method
- $spam_quarantine_method
- $banned_files_quarantine_method
- $bad_header'_quarantine_method

[13] caixa-postal de usuário

Uma maneira de desabilitar globalmente a quarentena é especificar estas variáveis como *"undef"* ou como uma *string* vazia.

Os métodos *local: e bsmtp:* são usados para quarentena. O *smtp:* atualmente não é usado para quarentena (é usado em envio e notificações). Dependendo do método especificado (*local/bsmtp/smtp*) o ajuste do **quarantine_to* para cada recipiente adota diferentes semânticas e sintaxes, possivelmente modificadas pela variável de configuração *$QUARANTINEDIR*.

O **quarantine_to* é atualmente limitado em sua funcionalidade, sendo usado apenas para desligar a quarentena para alguns usuários ou subdomínios locais. Em configurações comuns a localização da quarentena (e.g. um diretório ou uma *mailbox* dedicada) é a mesma para todos os recipientes. Se ao menos um recipiente especificar o **quarantine_to*, a mensagem será posta em quarentena no local especificado, não importando o número de recipientes.

4.4.4 Variáveis de Configuração

O comportamento do Amavisd-new é controlado por variáveis de configuração que estão no pacote Perl *Amavis::Conf*. No momento de inicialização do servidor Amavisd-new, estas variáveis são definidas como *"undef"*.

Após a inicialização do servidor, o arquivo de configuração *amavisd.conf*, que é um arquivo Perl, é lido para que as variáveis de configuração do Amavisd-new sejam ajustadas.

Como já foi dito, o arquivo de configuração do Amavisd-new é uma arquivo Perl, por esta razão o administrador do correio dever ter um certo conhecimento da sintaxe dessa linguagem, para que possa fazer alterações mais robustas nas configurações do Amavisd-new.

4.5 Comentários Finais

Após a conclusão deste capítulo, pode se ter uma idéia geral e explicativa de algumas características de cada ferramenta que será utilizada, e como estas irão interagir entre si. Verificou-se também, que as ferramentas utilizados no objetivo deste trabalho, são por demais, configuráveis e extensas, ao ponto de serem material suficiente para trabalhos em separado.

Sendo assim cabe ao leitor, interessado em ajustes finos, recorrer as referências deste trabalho, para que possa ajustar certas nuances e predicados, afim de satisfazer seus objetivos.

Como já dito na introdução deste trabalho, será demonstrada a configuração "padrão" para colocar o servidor para funcionar, o que será verificado no capítulo 5.

Capítulo 5

Instalação/configuração do Servidor

5.1 Introdução

O sistema operacional utilizado para a instalação do servidor foi o *United Linux for Suse*, não será abordado a instalação dos pacotes do Postfix, Amavisd-new, ClamAV e SpamAssassin, já que esta pode ser feita de várias maneiras de acordo com a distribuição a ser usada. As próximas seções partem do princípio de que a instalação dos programas foi efetuada, a máquina esta conectada a Internet e os serviços de POP e ou IMAP estejam executando.

5.2 Postfix (postfix-2.1.5)

5.2.1 Requesitos

Os pacotes binários do Postfix para as distribuições Linux mais populares, encontram-se em (POSTFIX.ORG, 2005f). Também pode ser feita a instalação atravéz do "*source*" em (POSTFIX.ORG, 2005d). O Postfix exige que os pacotes db*-devel e gcc estejam instalados.

5.2.2 Configuração

A configuração do Postfix e realizada em dois arquivos localizados no diretório */etc/postfix/*, o *master.cf* e o *main.cf*. Os parâmetros que devem ser alterados no arquivo *main.cf* podem ser vistos na figura 5.1. O arquivo *main.cf* é muito bem

documentado, e não serão necessarias muitas modificações para começar a usar o Postfix. A lista de todos os possíveis parâmetros do arquivo *main.cf* pode ser encontrada em (POSTFIX.ORG, 2005c). Uma descrição dos principais itens é listada a seguir:

1. *mail_owner* - Conta do linux responsável pelos processos servidores do Postfix e sua fila (*/var/spool/postfix*). Esta conta deve ser dedicada exclusivamente ao Postfix.

2. *myhostname* - Nome do servidor de *e-mail* na internet.

3. *mydomain* - Domínio do servidor de *e-mail*.

4. *myorigin* - Nome do domínio a ser anexado em contas deste servidor de *e-mail.*

5. *inet_interfaces* - Interface onde o servidor ira escutar por mensagens.

6. *mydestination* - Define quais os domínios que este servidor irá entregar *e-mails* localmente.

7. *mynetworks_style* - Define quais redes estão autorizados a retransmitir suas mensagens atráves do Postifix.

8. *alias_maps* - Define o local do banco de dados de *aliases* usados pelo sistema de entrega local.

9. *alias_database* - Define qual o banco de *aliases* será montado a partir do comando: *newaliases*

10. *mail_spool_directory* - Localização do diretório de *spool* de *e-mails*.

11. *sendmail_path* - Localização do comando *sendmail*.

12. *newaliases_path* - Localização do comando *newaliases*.

13. *mailq_path* - Localização do comando *mailq*.

14. *content_filter* - Especifica um filtro de conteúdo. Este está na própria máquina, escutando na porta 10024.

O arquivo *master.cf* não exige mudanças para o funcionamento geral do Postfix, este arquivo será comentado na seção 5.3.4 onde será configurado o Amavisd-new.

```
mail_owner = postfix
myhostname = united.linux.zuera
mydomain = linux.zuera
myorigin = $myhostname
inet_interfaces = all
mydestination = $myhostname, localhost.$mydomain, localhost
mynetworks_style = host
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
mail_spool_directory = /var/spool/mail
sendmail_path = /usr/sbin/sendmail
newaliases_path = /usr/bin/newaliases
mailq_path = /usr/bin/mailq
content_filter = smtp-amavis:[127.0.0.1]:10024
```

Figura 5.1: Parâmentros alterados no */etc/postfix/main.cf*

5.2.3 Executando o Postfix

Para inicializar o Postfix utiliza-se o comando: *postfix start*. O resultado do comando é mostrado na figura 5.2. Após inicializar o Postfix, é bom verificar se existem erros no arquivo de log, */var/log/mail*, conforme mostrado na figura 5.3.

```
  united: # postfix start
postfix/postfix-script: starting the Postfix mail system
```

Figura 5.2: Postfix - Iniciando o Postfix

```
united: # tail -f /var/log/mail
Mar 15 13:00:44 united postfix/postfix-script: starting the Postfix mail system
Mar 15 13:00:44 united postfix/master[1019]: daemon started -- version 2.1.5
```

Figura 5.3: Postfix - Saída do log

5.3 Amavisd-new (amavisd-new-2.2.0)

5.3.1 Requisitos

Para que a instalação do Amavisd-new seja realizada com sucesso, os módulos Perl da figura 5.4 devem ser instalados. Também é recomendável que o aplicativo *file(1)* versão 4.06 ou posterior esteja presente no sistema. Os pacotes e o *source* do Amavisd-new são encontrados em (AMAVISD, 2005).

```
Archive::Tar   (Archive-Tar-x.xx)
Archive::Zip   (Archive-Zip-x.xx) (1.14 or later should be used!)
Compress::Zlib (Compress-Zlib-x.xx)
Convert::TNEF  (Convert-TNEF-x.xx)
Convert::UUlib (Convert-UUlib-x.xxx) (stick to the new versions!)
MIME::Base64   (MIME-Base64-x.xx)
MIME::Parser   (MIME-Tools-x.xxxx) (latest version from CPAN - currently 5.415)
Mail::Internet (MailTools-1.58 or later have workarounds for Perl 5.8.0 bugs)
Net::Server    (Net-Server-x.xx)
Net::SMTP      (libnet-x.xx)     (use libnet-1.16 or latter for performance)
Digest::MD5    (Digest-MD5-x.xx)
IO::Stringy    (IO-stringy-x.xxx)
Time::HiRes    (Time-HiRes-x.xx) (use 1.49 or later, some older cause problems)
Unix::Syslog   (Unix-Syslog-x.xxx)
BerkeleyDB     with bdb library 3.2 or later (4.2 or later preferred)
```

Figura 5.4: Módulos Perl para o Amavisd-new

Os seguintes programas serão usados para desarquivar/decodificar as mensagens se os mesmos estiverem disponíveis no sistema: *compress, gzip, bzip2, nomarch (or arc), lha, arj (or unarj), rar (or unrar), zoo, cpio, lzop, freeze (ou unfreeze ou melt)*. Estes programas podem ser encontrados nos endereços apresentados na figura 5.5.

```
file:       ftp://ftp.astron.com/pub/file/
compress:   ftp://ftp.warwick.ac.uk/pub/compression/
gzip:       http://www.gzip.org/
bzip2:      http://sources.redhat.com/bzip2/
nomarch:    http://rus.members.beeb.net/nomarch.html
arc:        ftp://ftp.kiarchive.ru/pub/unix/arcers/
lha:        http://www2m.biglobe.ne.jp/~dolphin/lha/prog/
unarj:      ftp://ftp.kiarchive.ru/pub/unix/arcers/
arj:        http://testcase.newmail.ru/files/
```

Figura 5.5: Aplicativos auxiliares para o Amavisd-new

Não serão abordadas as instalações dos aplicativos da figura 5.5, apenas serão instalados os módulos Perl, que se fazem necessários. Um módulo "opcional" para o Amavisd-new que será utilizado é o *Mail::SpamAssassin*. Após sua instalação, o

filtro de *spam* SpamAssassin (WIKI.APACHE.ORG, 2005b), comentado na seção 4.3, já estará instalado e configurado para bloquear *spam*, juntamente com o Amavisd-new, o que facilita muito a instalação do servidor.

5.3.2 Instalando os módulos Perl, através do *CPAN shell*

Uma forma rápida e pratica de instalar módulos perl, é utilizar a sua ferramenta CPAN shell[1], conforme figura 5.6, para isto a máquina deverá possuir acesso a internet.

```
united:/ # perl -MCPAN -e shell
```

Figura 5.6: Iniciando o CPAN shell

O *script* de configuração do *CPAN Shell* é configurado praticamente teclando-se [ENTER] para a maioria de suas perguntas, e instalando os possíveis *softwares* que estiverem faltando, exemplo (*lynx, bzip*). A partir do *CPAN shell*, como pode ser observado na figura 5.7, será realizada a instalação dos módulos Perl.

```
united:/etc/postfix # perl -MCPAN -e shell

cpan shell -- CPAN exploration and modules installation (v1.76)
ReadLine support enabled

cpan>
```

Figura 5.7: CPAN shell

Para que a instalação através do *CPAN shell* seja realizada sem problemas, é indispensável que este esteja atualizado, o que pode ser feito executando-se os comandos mostrados na figura 5.8.

```
CPAN> install Bundle::CPAN
CPAN> reload CPAN
```

Figura 5.8: CPAN shell - Atualização

Um exemplo de instalação dos módulos perl pode ser visto na figura 5.9, a instalação dos módulos é feita através do comando *install* seguido do nome do

[1] *Comprehensive Perl Archive Network* (CPAN, 2005)

módulo. Para cada módulo a ser instalado o *CPAN shell*, verificará se este já encontra-se instalado e atualizado. Se o módulo não estiver instalado ou estiver desatualizado, o *CPAN shell* se encarregá de fazer o seu *download* e instalá-lo, bem como resolver as possíveis dependências entre módulos.

Durante o processo de instalação e/ou atualização dos módulos o *CPAN shell* poderá fazer perguntas, que serão respondidas teclando-se [ENTER], na maioria dos casos, pois o próprio *script*, já informa as respostas padrões a serem utilizadas.

Para cada módulo, o *CPAN shell* irá verificar se o módulo já encontra-se instalado e atualizado, caso seja necessário o *shell* fará o *download* do módulo e sua devida instalação, bem como instalar todos os arquivos necessários, caso verifique-se a necessidade, durante este processo o *script* poderá fazer perguntas, que devem ser respondidas teclando [ENTER], na maioria dos casos, pois o próprio *script*, já informa as respostas padrões utilizadas.

```
united:/etc/postfix # perl -MCPAN -e shell

cpan shell -- CPAN exploration and modules installation (v1.76)
ReadLine support enabled

cpan> install Archive::Tar
Caught SIGINT
CPAN: Storable loaded ok
Going to read /root/.cpan/Metadata
  Database was generated on Tue, 07 Dec 2004 08:00:38 GMT
Archive::Tar is up to date.

cpan>
```

Figura 5.9: Instalando um Módulo no CPAN shell

5.3.3 Configuração

O arquivo de configuração do Amavisd-new */etc/amavisd.conf*, deve ser alterado de acordo com a figura 5.10. Os parâmetros são relativamente simples, motivo pelo qual, são explicados como comentários na própria figura.

Antes de iniciar o Amavisd-new, é necessário testá-lo. Para isso utiliza-se o parâmetro *debug* na sua chamada, afim de se acompanhar o seu funcionamento. O Amavisd-new é inicializado em modo debug, utilizando o seguinte comando: *united:/ # /usr/local/sbin/amavisd -u amavis debug.*

```
# (nome da conta do usuário no qual o servidor será executado)
$daemon_user  = 'amavis';
# (nome do grupo do usuário no qual o servidor será executado)
$daemon_group = 'amavis';
# diretorio HOME do usuário amavis
$MYHOME   = '/var/amavis';
# diretório de trabalho, este deve ser criado manualmente
$TEMPBASE = ''$MYHOME/tmp'';
# váriavel de ambiente TMPDIR
$ENV{TMPDIR} = $TEMPBASE;
# diretório onde serão colocados os arquivos em quarentena
$QUARANTINEDIR = '/var/virusmails';
```

Figura 5.10: Variáveis à alterar no /etc/amavisd.conf.

O log[2] do Amavisd-new em modo de *debug* pode ser visto na figura 5.11.

```
amavisd[1140]: Found secondary av scanner ClamAV-clamscan at /usr/local/bin/clamscan
amavisd[1140]: No secondary av scanner: KasperskyLab kavscanner
amavisd[1140]: Creating db in /var/amavis/db/; BerkeleyDB 0.26, libdb 4.0
amavisd[1140]: SpamControl: initializing Mail::SpamAssassin
amavisd[1140]: SpamControl: done
amavisd[1140]: Net::Server: Beginning prefork (2 processes)
amavisd[1140]: Net::Server: Starting "2" children
amavisd[1163]: Net::Server: Child Preforked (1163)
amavisd[1140]: Net::Server: Parent ready for children.
amavisd[1163]: TIMING [total 113 ms] - bdb-open: 113 (100%), rundown: 0 (0%)
amavisd[1164]: Net::Server: Child Preforked (1164)
amavisd[1164]: TIMING [total 54 ms] - bdb-open: 53 (100%), rundown: 0 (0%)
```

Figura 5.11: Amavisd-new - Inicialização em modo debug

O Amavid-new está instalado e configurado, falta saber se este está funcionando corretamente. Para isto é feito um simples teste de *telnet* na porta SMTP 10024, porta padrão configurada no arquivo */etc/amavisd.conf*, conforme é demonstrado na figura 5.12.

5.3.4 Configurando o Postfix para se comunicar com o Amavisd-new

Para que o Postfix comunique-se com o Amavisd-new, cria-se no arquivo *master.cf* um filtro de conteúdo que foi especificado na última linha do arquivo *main.cf*. Este filtro é criado acrescentado-se as linhas da figura 5.13, no final do arquivo *master.cf*.

[2]Para que o *log* coubesse na página foram retiradas as datas, horas e caminho do mesmo

```
united:/ # telnet 127.0.0.1 10024
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
220 [127.0.0.1] ESMTP amavisd-new service ready
quit
221 2.0.0 [127.0.0.1] amavisd-new closing transmission channel
Connection closed by foreign host.
united:/ #
```

Figura 5.12: Amavisd-new - Testando porta SMTP 10024

Neste arquivo será configurado o Filtro de Conteúdo de nome (*smtp-amavis*), especificado na diretiva *content_filter* (POSTFIX.ORG, 2005a) do arquivo *main.cf*, conforme a figura 5.13. Na primeira linha o "n" indica que os serviços smtp e smtpd do postfix, não encontram-se em modo *chroot-ed*. As outras opções da primeira linha devem seguir o padrão dos serviços do Postfix.

As demais linhas são os parâmetros passados ao Postfix, pelo Amavisd-new, após o término, da manipulação da mensagem pelos verificadores de conteúdo configurados. O parâmetro mais importante que deve ser passado é o *-o content_filter=*, que deve estar vazio, evitando assim um *loop* infinito de verificações. Os demais parâmetros são parâmetros do *main.cf* que iram complementar e até mesmo substituir algumas configurações já existentes no *main.cf*. Para maiores informações sobre estes parâmetros consulte (POSTFIX.ORG, 2005c).

Sempre que forem realizadas modificações nos arquivos de configuração do Postfix, deve-se realizar o *postfix reload*, afim de que sejam aplicadas as alterações desejadas. Para validar as configurações feitas no arquivo *master.cf*, deve-se testar se o Postfix está aceitando conexões na porta do *content-filter*. Para isto será realizado um teste de *telnet* na porta 10025 do servidor. O que pode ser feito substituindo o valor 10024 da figura 5.12, pelo valor 10025.

5.4 ClamAV (clamav-0.80)

5.4.1 Requisitos

Os seguintes pacotes são necessários: *zlib, zlib-devel e compilador gcc*. Os pacotes *bzip2, bzip2-devel library e GNU MP 3* são altamente recomendados, porém não necessários. Os pacotes de instalação do ClamAV podem ser encontrados em (CLAMAV, 2005).

```
# ==========================================================================
# service type  private unpriv  chroot  wakeup  maxproc command + args
#               (yes)   (yes)   (yes)   (never) (100)
# ==========================================================================

smtp-amavis unix - - n - 2 smtp
-o smtp_data_done_timeout=1200
-o smtp_send_xforward_command=yes
-o disable_dns_lookups=yes

127.0.0.1:10025 inet n - n - - smtpd
-o content_filter=
-o local_recipient_maps=
-o relay_recipient_maps=
-o smtpd_restriction_classes=
-o smtpd_client_restrictions=
-o smtpd_helo_restrictions=
-o smtpd_sender_restrictions=
-o smtpd_recipient_restrictions=permit_mynetworks,reject
-o mynetworks=127.0.0.0/8
-o strict_rfc821_envelopes=yes
-o smtpd_error_sleep_time=0
-o smtpd_soft_error_limit=1001
-o smtpd_hard_error_limit=1000
-o smtpd_client_connection_count_limit=0
-o smtpd_client_connection_rate_limit=0
-o receive_override_options=no_header_body_checks
```

Figura 5.13: Alterações no final do */etc/postfix/master.cf*

5.4.2 Configuração dos arquivos *amavisd.conf* e *clamd.conf*

Note na figura 5.14 que o parâmetro "User" recebe o nome do mesmo usuário do Amavisd-new. O local especificado no parâmetro "LocalSocket" do arquivo *clamd.conf* deve ser o mesmo especificado no arquivo *amavisd.conf* de acordo com a figura 5.15.

```
LogFile /var/log/clamav/clamav.log
PidFile /var/run/clamav/clamd.pid
LocalSocket /tmp/clamd
FixStaleSocket
User amavis
```

Figura 5.14: ClamAV - Arquivo clamd.conf

```
# ### http://www.clamav.net/
 ['ClamAV-clamd',
   \&ask_daemon, ["CONTSCAN {}\n", "/tmp/clamd"],
   qr/\bOK$/, qr/\bFOUND$/,
   qr/^.*?: (?!Infected Archive)(.*) FOUND$/ ],
# # NOTE: run clamd under the same user as amavisd;  match the socket
# # name (LocalSocket) in clamav.conf to the socket name in this entry
# # When running chrooted one may prefer: ["CONTSCAN {}\n","$MYHOME/clamd"],
```

Figura 5.15: Arquivo - amavisd.conf

5.4.3 Executando o Postfix/Amavisd-new/ClamAV

Finalizadas as alterações nos arquivos de configuração do servidor, deve-se reinicializar o Postfix e Amavisd-new, para que os mesmos recebam as novas configurações, conforme figura 5.16.

```
--> obs: Para parar o Amavisd-new, precione CTRL+C,
         na tela onde o mesmo foi iniciado anteriormente.

united:/ # clamd
united:/ # /usr/local/sbin/amavisd -u amavis debug

--> em outra tela (sessão)

united:/ # postfix reload
```

Figura 5.16: Inicializando o Servidor.

5.5 Finalizando a Configuração e Testando

Observa-se que, até o dado momento, não foi necessário fazer configurações para o SpamAssassin, pois este foi instalado como um módulo Perl, como comentando na seção 5.3.1. Por isso suas diretivas já encontram-se configuradas com os valores padrões para operação imediata. Será realizado apenas o treinamento do SpamAssassin, o que poderá ser visto na seção 5.5.2.

Estando configurados o Postfix, o Amavisd-new com o SpamAssassin e o ClamAV, falta fazer com que o Postfix envie todas as mensagens que receber para o Amavisd-new. Assim o Amavisd-new se encarregará de fazer a inspeção de conteúdo com o auxilío do ClamAV e do SpamAssassin. Para isto usa-se a di-

retiva *content_filter* no *main.cf*, uma forma rápida de se fazer isto, é excutar o comando *postconf -e 'content_filter=smtp-amavis:[127.0.0.1]:10024'*. Após essa última configuração, faz-se necessária a reinicialização dos serviços, conforme figura 5.17.

```
--> ClamAV
united:/ # clamd

--> Amavisd-new/Spamassassin
united:/ # /usr/local/sbin/amavisd -u amavis

--> Postfix
united:/ # postfix start
```

Figura 5.17: Iniciando Postfix/Amavisd-new/ClamAV/Spamassassin

5.5.1 Teste de envio de vírus e spam ao Servidor

Um teste de mensagem normal, através de *telnet* na porta 10024, é demonstrado na figura 5.18. Um teste de mensagem contendo uma assinatura do "teste" de vírus *EICAR*[3], pode ser visto na figura 5.19. A resposta enviada pelo servidor, depende do conteúdo das variáveis *$final_virus_destiny e *virus_lovers**, do arquivo *amavisd.conf*, e poderá ser, algo parecido com:

- **** 550 5.7.1 Message content rejected, id=16968-01 - VIRUS: EICAR-AV-Test*
- **** 250 2.5.0 Ok, but 1 BOUNCE*
- **** 250 2.7.1 Ok, discarded, id=16984-01 - VIRUS: EICAR-AV-Test*
- **** 250 2.6.0 Ok, id=17041-01, from MTA: 250 Ok: queued as 3F1841A5F5*

5.5.2 Treinando o SpamAssassin

Antes do SpamAssassin começar a marcar as mensagens como *spam*, ele deve ser treinado com mensagens de *spam* e de *ham*. O ideal é ter dois arquivos com no mínimo 200 mensagens cada, sendo um arquivo de *spam's* e outro de *ham's*. Para se treinar o SpamAssassin, utiliza-se os comandos:

```
sa-learn --showdots --mbox --spam nomedoarquivodespam
sa-learn --showdots --mbox --ham  nomedoarquivodeham
```

[3] http://www.eicar.org/anti_virus_test_file.htm

```
-->  $ telnet 127.0.0.1 10024
     Trying 127.0.0.1...
     Connected to 127.0.0.1.
     Escape character is '^]'.
     220 [127.0.0.1] ESMTP amavisd-new service ready
-->  MAIL FROM:<test@example.com>
     250 2.1.0 Sender test@example.com OK
-->  RCPT TO:<postmaster>
     250 2.1.5 Recipient postmaster OK
-->  DATA
     354 End data with <CR><LF>.<CR><LF>
-->  Subject: test1
-->
-->  test1
-->  .

***  250 2.6.0 Ok, id=31859-01, from MTA: 250 Ok: queued as 90B7F16F
```

Figura 5.18: Testando o Servidor - Mensagem Normal

```
united:~ # telnet 127.0.0.1 10024
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
220 [127.0.0.1] ESMTP amavisd-new service ready
MAIL FROM:<teste@eicar.com>
250 2.1.0 Sender teste@eicar.com OK
RCPT TO:<vinny@united.linux.zuera>
250 2.1.5 Recipient vinny@united.linux.zuera OK
DATA
354 End data with <CR><LF>.<CR><LF>
Subject: teste de vírus EICAR

X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
.
250 2.7.1 Ok, discarded, id=01485-01 - VIRUS: Eicar-Test-Signature
```

Figura 5.19: Testando o Servidor - Mensagem com *EICAR*.

Para simular o teste de envio de *spam* ao servidor, serão trocados os valores das variáveis *$sa_tag2_level_deflt = 6.31;* e *$sa_kill_level_deflt = 6.31;*, encontradas no arquivo */etc/amavisd.conf*, para os valores 3 e 4, respectivamente. Em seguida será enviada ao servidor uma mensagem de *spam* com pontuação igual a 3.94, este valor irá ultrapassar o nível *tag2* que foi configurado.

Isto permitirá que a mensagem seja entregue na caixa postal do destinatário, com "marcações de *spam*" em seu cabeçalho, como pode ser visto na figura 5.20. Se o servidor receber uma mensagem com pontuação acima do nível *kill*, esta men-

sagem não será entregue, e o remetente da mensagem receberá a seguinte resposta [*250 2.5.0 Ok, id=01416-01, BOUNCE*].

```
From TESTE@MAIL.COM  Wed Mar 17 09:42:52 1999
Return-Path: <TESTE@MAIL.COM>
X-Original-To: vinny@united.linux.zuera
Delivered-To: vinny@united.linux.zuera
Received: from localhost (localhost [127.0.0.1])
        by united.linux.zuera (Postfix) with ESMTP id 522E617281
        for <vinny@united.linux.zuera>; Wed, 17 Mar 1999 09:42:52 -0800 (PST)
Received: from unknown ([127.0.0.1])
 by localhost (united [127.0.0.1]) (amavisd-new, port 10024) with SMTP
 id 01484-01 for <vinny@united.linux.zuera>;
 Wed, 17 Mar 1999 09:41:55 -0800 (PST)
Subject: ***SPAM*** teste
X-Virus-Scanned: amavisd-new at linux.zuera
X-Spam-Status: Yes, hits=3.942 tagged_above=2 required=3 tests=ALL_TRUSTED,
 EMAIL_ROT13, HTML_30_40, HTML_IMAGE_ONLY_24, HTML_MESSAGE,
 HTML_MISSING_CTYPE, JOIN_MILLIONS, MISSING_DATE, OBSCURED_EMAIL
X-Spam-Level: ***
X-Spam-Flag: YES
Message-Id: <19990317174252.522E617281@united.linux.zuera>
Date: Wed, 17 Mar 1999 09:42:52 -0800 (PST)
From: TESTE@MAIL.COM
To: undisclosed-recipients:;
```

Figura 5.20: Cabeçalho de Mensagem de nível tag2

Capítulo 6

Conclusão

Após a conclusão desta, ficou demonstrado que com alguns aplicativos livres e um pouco de tempo, estudo e paciência, é possível de se ter uma grande ferramenta de controle de *e-mail* disponível para a empresa gratuitamente.

Foram realizados os testes de *spam* e de vírus, e como se observou para o teste do *spam*, será necessário que o administrador do servidor faça ajustes e configurações de acordo com a realidade de sua empresa, além de estar sempre treinando o SpamAssassin. Este treinamento pode ser feito manualmente, como foi mostrado na seção 5.5.2, ou pode ser feito automaticamente, o que pode ser implementado com um pouco de leitura em (JOUSSET, 2004) e (WIKI.APACHE.ORG, 2005a).

O propósito foi alcançado, instalar e configurar um servidor de *e-mail's* com controle de *spam* e controle de vírus. O próximo passo é realizar ajustes e refinamentos do servidor para que este supra as necessidades de cada empresa. Uma das sugestões é estudar as referências do SpamAssassin, afim de configurá-lo mais ao perfil de cada usuário, pois o módulo perl *Mail::SpamAssassin* instalado vem padrão para todos os usuários. E o que pode ser *spam* para alguns pode não ser para outros.

Bom o trabalho não para por aqui, através da montagem e preparação deste servidor foram descobertas várias formas e maneiras diferentes de chegar a um resultado igual e até possivelmente melhor, utilizando-se de outras ferramentas encontradas ao longo do caminho de pesquisa. Um exemplo seria a utilização do módulo Mail::ClamAV um módulo perl para o ClamAV, que reduziria muito o tempo de instalação e manutenção do servidor.

Referências Bibliográficas

AMAVISD. *Download*. Febreary 2005. Disponível em: <http://www.ijs.si-/software/amavisd>.

ANTIVIRUS, C. *Who's using ClamAV*. April 2005. Disponível em: <http://clamav.net/whos.htm>.

CLAMAV. *ClamAV - Binary packages*. April 2005. Disponível em: <http://www-.clamav.net/binary.htm>.

CPAN. *Comprehensive Perl Archive Network*. April 2005. Disponível em: <http://www.cpan.org>.

CRISPIN, M. *INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1*. March 2003. RFC 3501. Disponível em: <ftp://ftp.rfc-editor.org/in-notes/rfc3501.txt>.

CROCKER, D. H. *STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES*. August 1982. RFC 822. Disponível em: <ftp://ftp.rfc-editor.org/in-notes/rfc822.txt>.

DENT, K. D. *Postfix : The Definitive Guide*. 1. ed. [S.l.]: O'Reilly, 2003.

ENGINEERING, C. of. What are pop and imap ? *Oregon State University*, September 2003. Disponível em: <http://engr.oregonstate.edu/computing/email-/6>.

FAYHE, O. D. *Auto Whitelist*. December 2004. Wiki. Disponível em: <http://wiki.apache.org/spamassassin/AutoWhitelis>.

FORREST, D. *Network Tests*. October 2004. Wiki. Disponível em: <http://wiki-.apache.org/spamassassin/NetworkTest>.

FREED INNOSOFT, N. B. N. *Multipurpose Internet Mail Extensions,(MIME) Part One: Format of Internet Message Bodies*. November 1996. RFC 3501. Disponível em: <ftp://ftp.rfc-editor.org/in-notes/rfc2045.txt>.

FULLER BARRNET, K. V. V. *Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy*. September 1993. RFC 1519. Disponível em: <ftp://ftp.rfc-editor.org/in-notes/pdfrfc/rfc1519.txt.pdf>.

GRAY, T. *Message Access Paradigms and Protocols*. September 1995. Disponível em: <http://www.imap.org/papers/imap.vs.pop.htm>.

JOUSSET, A. *Setup of special aliases in Postfix to forward spams and hams*. April 2004. Disponível em: <http://jousset.org/pub/sa-postfix.en.htm>.

KLENSIN, N. F. J. et al. *SMTP Service Extensions*. November 1995. RFC 1869. Disponível em: <ftp://ftp.rfc-editor.org/in-notes/rfc1869.txt>.

KOHN, D. *Bayes Introduction*. March 2005. Wiki. Disponível em: <http://wiki-.apache.org/spamassassin/BayesInSpamAssassi>.

KOHN MARCUS DRESSLER, X. D. *Manual Whitelist*. July 2004. Wiki. Disponível em: <http://wiki.apache.org/spamassassin/ManualWhitelis>.

KOJM, T. *On-access scanning*. October 2004. Disponível em: <http://www-.clamav.net/doc/0.80/html/node18.htm>.

LAIRD, C. Venema aims to make network software safe. *developer.com*, October 1998.

LINDBERG, G. *Anti-Spam Recommendations for SMTP MTAs*. February 1999. RFC 2505. Disponível em: <ftp://ftp.rfc-editor.org/in-notes/rfc2505.txt>.

MALTESTRETZ. *SpamAssassinRules*. June 2004. Wiki. Disponível em: <http://wiki.apache.org/spamassassin/SpamAssassinRule>.

MARKOFF, J. Sharing software, ibm to release mail program blueprint. *New York Times Company*, December 1998.

MASON, J. *DNS Blocklists*. February 2005. Wiki. Disponível em: <http://wiki-.apache.org/spamassassin/DnsBlocklist>.

MYERS, C. M. J. *Local Mail Transfer Protocol*. October 1996. RFC 2033. Disponível em: <ftp://ftp.rfc-editor.org/in-notes/rfc2033.txt>.

MYERS, C. M. J.; ROSE, M. *Post Office Protocol - Version 3*. May 1996. RFC 1939. Disponível em: <ftp://ftp.rfc-editor.org/in-notes/rfc1939.txt>.

POSTEL, J. B. *SIMPLE MAIL TRANSFER PROTOCOL*. August 1982. RFC 821. Disponível em: <ftp://ftp.rfc-editor.org/in-notes/rfc821.txt>.

POSTFIX.ORG. *Postfix After-Queue Content Filter*. April 2005. Disponível em: <http://www.postfix.org/FILTER_README.htm>.

POSTFIX.ORG. *Postfix Architecture Overview*. April 2005. Disponível em: <http://www.postfix.org/OVERVIEW.htm>.

POSTFIX.ORG. *Postfix Configuration Parameters*. April 2005. Disponível em: <http://www.postfix.org/postconf.5.htm>.

POSTFIX.ORG. *Postfix Download Sites - Mirrors*. April 2005. Disponível em: <http://www.postfix.org/download.htm>.

POSTFIX.ORG. *Postfix Howtos and FAQs*. April 2005. Disponível em: <http://www.postfix.org/docs.htm>.

POSTFIX.ORG. *Postfix Packages and Ports*. April 2005. Disponível em: <http://www.postfix.org/packages.htm>.

RESNICK, P. *Internet Message Format*. April 2001. RFC 2822. Disponível em: <ftp://ftp.rfc-editor.org/in-notes/rfc2822.txt>.

SPAMASSASSIN.APACHE.ORG. *SpamAssassin Prehistory*. April 2005. Disponível em: <http://spamassassin.apache.org/prehistory>.

WIKI.APACHE.ORG. *Setting up Site-Wide Bayesian Filtering*. February 2005. Wiki. Disponível em: <http://wiki.apache.org/spamassassin-/SiteWideBayesSetu>.

WIKI.APACHE.ORG. *SpamAssassin Integration with Postfix, using Amavis*. March 2005. Wiki. Disponível em: <http://wiki.apache.org/spamassassin-/IntegratedInPostfixWithAmavi>.