UNIVERSIDADE ESTADUAL DE SANTA CRUZ
DEPARTAMENTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
COLEGIADO DE CÊNCIA DA COMPUTAÇÃO

PLANO DE SEGURANÇA E PLANEJAMENTO DA REDE DE COMPUTADORES DA WAYTEC TECNOLOGIA EM COMUNICAÇÃO LTDA.

ANDERSON LUIZ SOUZA MOREIRA

TRABALHO DE CONCLUSÃO DE CURSO

Ilhéus – Bahia,
Agosto - 2003

UNIVERSIDADE ESTADUAL DE SANTA CRUZ
DEPARTAMENTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
COLEGIADO DE CÊNCIA DA COMPUTAÇÃO

PLANO DE SEGURANÇA E PLANEJAMENTO DA REDE DE COMPUTADORES DA WAYTEC TECNOLOGIA EM COMUNICAÇÃO LTDA.

ANDERSON LUIZ SOUZA MOREIRA

Trabalho de conclusão de estágio apresentado ao Colegiado de Ciência da Computação para obtenção dos créditos das disciplinas Estágio Supervisionado I e II e validação para o Trabalho de Conclusão de Curso.

ORIENTADOR: TEODORO PIRES

Ilhéus – Bahia,
Agosto – 2003

Este trabalho é dedicado a minha família, que mesmo estando longe sempre torceram pelo meu sucesso, especialmente para Fátima e Carminha.

SUMÁRIO

1. INTRODUÇÃO

Vivemos numa época em que a informação é um dos recursos mais valiosos dentro das organizações, de forma que obtê-la é tão importante quanto protegê-la. Com um computador interligado a outros computadores através de uma rede de comunicação, ou seja, com o desenvolvimento das tecnologias de redes de computadores, obter e compartilhar tal informação tornou-se muito mais fácil, ao mesmo tempo em que protegê-la contra o acesso indevido tornou-se uma tarefa muito mais complexa. O conceito de redes abertas para tornar disponível a informação, tornou também mais difícil o controle de acesso a um sistema e aos dados nele contidos, de modo que a segurança em redes de computadores tem sido largamente estudada e muito se tem feito no sentido de desenvolver novas técnicas que viabilizem a implementação do nível de segurança requerido.

Este trabalho é o relatório de experiências feitas na Waytec Tecnologia em Comunicação Ltda., local do Estágio Supervisionado I e II e encontra-se dividido em duas partes. Nesta primeira parte é apresentado um estudo feito durante o Estágio I sobre as principais formas de ataque utilizadas contra um sistema de computação e as principais formas de defesa no sentido de se implementar uma estratégia de segurança. Na segunda parte, durante o Estágio II, um exemplo de implementação de uma estratégia de segurança é fornecido através da utilização de *firewalls* e a descrição de como foi organizada a rede da Empresa Waytec, para um modelo de rede segura.

2. SEGURANÇA EM REDES DE COMPUTADORES

Segurança em redes de computadores é um vasto tema que tem sido muito estudado nos últimos anos. Isto porque, apesar dela ser tão importante, garanti-la não é uma tarefa tão simples.

Com a evolução das tecnologias de rede e o crescimento da Internet - a Internet não é segura – a necessidade de se aprimorar e desenvolver mecanismos de segurança aumentou, já que diferentes e “brilhantes” formas de atacar um sistema para conseguir o acesso não autorizado ao mesmo e aos dados nele contidos, também surgiram. Se a disponibilidade da informação aumentou, consegui-la de forma indevida também tornou-se mais fácil.

Mas porque segurança é necessária? “A segurança está relacionada à necessidade de proteção contra o acesso ou manipulação, intencional ou não, de informações confidenciais por elementos não autorizados, e a utilização não autorizada do computador ou seus dispositivos periféricos” [SOARES 95]. Ou seja, é preciso controlar o acesso ao uso de recursos (CPU, disco...), bem como à informação, a fim de que se obtenha: (1) **confidencialidade** - garantia contra o acesso não autorizado aos dados armazenados no sistema; (2) **integridade** - garantia da incorruptibilidade dos dados armazenados no sistema, ou seja, precisão das informações; e (3) **disponibilidade** - garantia de que o sistema estará funcionando a contento e, consequentemente, que a informação nele armazenada estará disponível quando necessário.

Sendo assim, segurança tem se tornado um dos interesses primários dos administradores de rede, principalmente quando imagina-se a possibilidade de se ter informações e recursos expostos a intrusos da Internet.

A segurança pode ser analisada de várias formas. Dois de seus mais importantes aspectos são a **perda de dados** e os **intrusos**. Algumas das razões mais comuns para a perda de dados são as seguintes:

- Fatalidades ou acidentes naturais: incêndios, inundações, terremotos, ratos roendo fitas ou disquetes;
- Erros de *hardware* ou de *software*: falhas no processamento, erros de comunicação, *bugs* em programas, discos ou fitas ilegíveis;
- Erros humanos: entrada de dados incorreta, montagem errada de fita ou de disco, perda de uma fita ou de um disco, executar o programa errado, erros de configuração.

Um política de *backup* eficiente pode resolver a maioria desses problemas.

O outro aspecto diz respeito à ação de intrusos no sistema. Estamos falando dos *hacker*s e dos *cracker*s. Cabe aqui diferenciá-los. Um *hacker* é “uma pessoa que sente prazer em aprender os detalhes dos sistemas de programação e sugar o máximo de suas capacidades” - ao oposto da maioria que prefere aprender o mínimo necessário. Um *cracker* “usa seus conhecimentos sobre sistemas computacionais para fins ilícitos”. Então, sob o ponto de vista conceitual, são os *crackers* quem representa o perigo e é com eles que devemos nos preocupar. Muitas são as estratégias de ataque utilizadas pelos *crack*ers a fim de obter acesso indevido aos sistemas.

Ao se decidir por implantar uma estratégia de segurança em um sistema de computadores deve-se considerar a segurança em dois níveis: segurança local e

segurança remota. No primeiro caso, trata-se de proteger o sistema contra ataques dos seus próprios usuários, aqueles que têm acesso direto ao sistema. No segundo caso, a idéia é proteger toda a rede interna de intrusos externos. Neste ínterim, deve-se considerar falhas de segurança que vão desde vulnerabilidades em aplicações do próprio sistema operacional até falhas de projeto de protocolos de comunicação como o TCP/IP. Vejamos a seguir as principais formas de ataque utilizadas por *crackers e hackers* para burlar a segurança de um sistema.

2.1. Formas de Ataque

Um ataque consiste de uma violação intencional da segurança de um sistema e pode resultar na destruição de informação ou de outros recursos; modificação ou deturpação da informação; roubo, remoção ou perda de informação ou de outros recursos; revelação de informação; e interrupção de serviços.

Quanto a sua origem, um ataque pode ser de origem externa, realizado por alguém em outro *site* - e aí diz-se que o ataque é remoto - ou de origem interna (ataque local), realizado por alguém da própria rede interna (pessoa com acesso físico às instalações).

Um intruso normalmente segue alguns passos no sentido de realizar um ataque.

Primeiramente ele constrói um verdadeiro banco de dados com informações sobre o sistema a ser atacado. Existem muitas maneiras de coletar “informação útil”, como por exemplo: servidores DNS (*Domain Name Service*) podem acessar uma lista de endereços IP dos *hosts* (máquinas) da rede e seus respectivos nomes, o protocolo *finger* pode revelar informações detalhadas sobre os usuários do sistema;

o programa *ping* pode ser usado para localizar um *host* particular e determinar sua disponibilidade.

Em seguida, o intruso pode vasculhar o sistema em busca de suas vulnerabilidades.

Ferramentas como SATAN e ISS (que serão analisadas mais tarde) podem ajudar nesse sentido.

Utilizando os resultados obtidos, resta, ao intruso, explorar estas vulnerabilidades para atingir seus objetivos. Muitas são as formas encontradas para burlar a segurança de um sistema e diversas são as estratégias utilizadas. Vejamos a seguir as principais delas.

2.1.1. Back Doors

Back Doors ou *Trap Doors* são pedaços de código escritos em aplicações ou sistemas operacionais, que realizam serviços não previstos em sua especificação, por exemplo, versões alteradas de *login*. *Back Doors* são normalmente inseridas em aplicações que possuem, por exemplo, procedimentos de autenticação onde o usuário é forçado a entrar com múltiplas senhas diferentes antes de poder executar estas aplicações. O desenvolvedor da aplicação pode utilizar uma *back door* para evitar todos os passos de autenticação enquanto *depurando* o programa. *Back doors*, entretanto, são tipicamente utilizadas para obter acesso não autorizado ao sistema.

Um dos exemplos mais famosos de *back door* no UNIX foi a opção DEBUG do programa *sendmail* que, quando habilitada, permitia o acesso remoto a um computador através da rede sem que fosse necessária uma autenticação prévia. Versões mais novas do *sendmail* (a partir da 8.6.10) já corrigiram o erro.

Verificar regularmente a integridade de arquivos importantes pode ajudar a evitar *back doors* no sistema.

2.1.2. Bombas Lógicas

Bombas lógicas consistem de características ocultas em programas, que ficam dormentes por um extenso período de tempo e se tornam ativas sob certas condições. Neste ponto, elas realizam uma função que não é a função pretendida pelo programa no qual elas estão contidas.

Condições que podem ativar uma bomba lógica incluem a presença ou ausência de certos arquivos, ou um dia particular da semana. Uma vez ativada, uma bomba lógica pode destruir ou alterar dados ou causar a queda (*crash*) de máquinas. O vírus *sexta-feira 13* pode ser visto como uma bomba lógica.

2.1.3. Vírus

Vírus são pedaços de código inseridos em outro código executável; quando aquele programa executa, o código do vírus também é executado. Quando executado, o vírus copia a si mesmo para outros programas e, com isso, procura infectar tantos arquivos quanto possível. É imensa a variedade de vírus que existem e os efeitos causados vão desde uma mensagem inocente no rodapé do aplicativo que executa o arquivo infectado (vírus CAP para arquivos .DOC) até a queda total do sistema.

Vírus não executam sozinhos. Eles precisam de um programa hospedeiro para serem ativados, tal qual um vírus em sua acepção da palavra, que precisa de um outro ser vivo para desempenhar suas funções vitais.

Executar vacinas regularmente, não utilizar códigos binários suspeitos e não deixar diretórios como */bin, /usr/bin* e */usr/ucb* desprotegidos, são medidas que protegem o sistema contra a ação de vírus.

2.1.4. Worms

Worms, ou vermes, são programas que executam independentemente e "viajam" de uma máquina para outra através de conexões de rede; *worms* podem ter

porções deles próprios executando em diferentes máquinas. Sem sombra de dúvida, um dos ataques mais famosos no âmbito das redes de computadores é o Verme da Internet, escrito por Robert Tappan Morris, um estudante da *Cornell University*. Explorando erros do UNIX de Berkeley, Morris derrubou em questão de horas todos os sistemas Sun e VAX da Internet.

O Verme da Internet tentava três métodos para infectar máquinas espalhadas pela rede. O primeiro método consistia em executar um *shell* (interpretador de comandos) remoto, usando o comando *rsh*. Algumas máquinas corrompiam outras máquinas que executavam *rsh* sem necessidade de qualquer identificação.

O método 2 explorava um *bug* (erro) do programa *finger*. O worm chamava o *finger* com um parâmetro de 536 bytes, especialmente preparado. Esta longa cadeia não cabia no *buffer* do programa servidor (*daemon) finger* e era, então, escrita na pilha deste servidor. O *bug* explorado aqui era a falha do servidor em identificar uma condição de *overflow* quando da passagem de seus parâmetros. Uma vez que a sua pilha tinha sido adulterada, o servidor não conseguia voltar para o programa principal e passava a executar um procedimento dentro dos 536 bytes armazenados na pilha. Ao executar um comando */bin/sh* com sucesso, o *worm* tinha um *shell* executando na máquina-alvo de seu ataque. Este método utilizado pelo Verme da Internet também configura um outro tipo de ataque popularmente conhecido como ***buffer overflow*.**

O terceiro método dependia de um *bug* do sistema de correio eletrônico, que permitia, através do *sendmail*, que o *worm* enviasse uma cópia de si mesmo para outras máquinas e a executasse.

Desta forma, o Verme da Internet infestou milhares de máquinas em toda a rede e ficou classicamente conhecido como um dos maiores ataques de toda a história da Computação.

2.1.5. Cavalos de Tróia

O conceito é análogo ao cavalo de Tróia da mitolo*g*ia grega. Programas assemelham-se com o programa que o usuário deseja executar (um jogo, um editor) e enquanto aparentam fazer o que o usuário quer, eles estão realmente fazendo algo totalmente diferente.

Durante a edição de um texto, por exemplo, enquanto o usuário digita inocentemente seu documento, o editor pode realmente estar removendo arquivos, reformatando um disco ou alterando informação.

2.1.6. Bactéria

Também conhecidos como *rabbits*, as bactérias são programas cuja única proposta é a replicação. A bactéria se reproduz exponencialmente, consumindo memória, CPU, disco, impossibilitando a utilização desses recursos por outros usuários. Máquinas sem limite de quotas ou limite de uso de recursos são especialmente susceptíveis a esta forma de ataque.

2.1.7. Flood

Esta forma de ataque, assim como o IP *spoofing* descrito a seguir, explora características de projeto do protocolo de comunicação utilizado na Internet: o TCP/IP (*Transmission Control Protocol/ Internet Protocol*).

O *flood* é um tipo de ataque que se utiliza de informações contidas no cabeçalho de pacotes TCP/IP, como mensagens ICMP[1] (*Internet Control Messages Protocol*), para causar a queda do sistema (*denial of service attack*). Por exemplo, *floods* ICMP_ECHO[2] exploram o fato de que a maioria dos núcleos dos sistemas operacionais simplesmente respondem a pacotes de requisição ICMP_ECHO um após o outro, sem parar mais. Ou seja, ao encher o sistema de pacotes que carregam mensagens ICMP desse tipo, o intruso tomará para si todos os recursos operacionais da máquina, configurando um tipo de ataque que levará à indisponibilidade de qualquer outro serviço.

2.1.8. IP Spoofing

Neste tipo de ataque, o intruso transmite pacotes a partir da rede externa que fingem ser originários de um *host* interno - os pacotes falsamente contém o endereço IP fonte que especifica um *host* interno da rede (um *host* confiável, por exemplo). O intruso, então, espera que uma verificação simples de endereço seja feita, de modo a permitir que pacotes de *hosts* internos confiáveis sejam aceitos ao mesmo tempo em que pacotes de outros *hosts* sejam descartados.

Uma vez tendo entrado no sistema, o intruso pode executar um simples comando para deixar uma *back door* e está configurado o ataque (na verdade, o IP S*poofing* não é o ataque, mas é apenas um passo no sentido de configurá-lo).

Uma boa técnica para detectar um ataque como este é descartar pacotes que chegam pela interface externa do roteador com endereços IP fonte que especificam *hosts* internos.

---

[1] ICMP é um protocolo que atua junto ao IP na pilha de protocolos TCP/IP, de modo a permitir a troca de mensagens de erro e controle entre *hosts* e *gateways*.

[2] Mensagens ICMP_ECHO são utilizadas para testar a comunicação entre dois sistemas.

3. SEGURANÇA EM AMBIENTES LINUX

O LINUX, plataforma para desenvolvimento das pesquisas realizadas durante o estágio, foi feito baseado no UNIX a qual não foi projetado com aspectos relacionados à segurança em mente - o que não significa que ele não implemente qualquer nível de segurança. Na verdade, alguns mecanismos de segurança estão disponíveis mas, ao longo de sua história, o UNIX tem sido vítima de vários e famosos tipos de ataques. E, se construir um sistema seguro é difícil, adicionar características que provejam o nível de segurança requerido é muito mais.

O LINUX então foi concebido para ser um sistema aberto, porém mais seguro que o UNIX: o usuário *root* determina acessos a seus arquivos, apenas ele tem acesso à configuração do sistema, um usuário não pode obter informações sobre o outro, há o conceito de diretórios compartilhados (*/tmp*, por exemplo). O LINUX foi escolhido pois suporta as seguintes características de um servidor de rede:

- Todos os protocolos TCP/IP;
- Protocolos NetBIOS SMB (Server Message Block) da Microsoft;
- Clientes NetWare;
- Acesso remoto, incluindo acesso *dial-up* com PPP e SLIP;
- Servidor FTP, incluindo *anonymous* FTP;
- Servidor de correio eletrônico TCP/IP completo;
- Protocolos POP (*Post Office Protocol*) e IMAP (*Internet Message Access Protocol*) para serviços de caixa de correspondência;
- Servidor Web;
- Kit de desenvolvimento JAVA nativo;

- Servidor de aplicações;
- Um ambiente completo de desenvolvimento de programas: C, C++, Perl e outras linguagens de programação.

Nenhuma dessas opções tem custo adicional. São todas incluindas como parte da distribuição básica do LINUX.

Com a adição da grande quantidade de serviços de rede ao sistema, como *login* remoto, RPC (*Remote Procedure Call*), NFS (*Network File System*) e correio eletrônico, e a utilização de estações *diskless*, controlar a segurança tornou-se ainda mais difícil. Se por um lado tais características aumentaram a utilidade e usabilidade do sistema, por outro lado aumentaram, também, as suas vulnerabilidades de forma a possibilitar o acesso não autorizado ao mesmo.

A segurança no LINUX pode ser analisada em três níveis: segurança de contas e segurança de rede, onde preocupa-se em proteger o sistema contra o acesso não autorizado (seja ele local ou remoto), e segurança do sistema de arquivos, onde preocupa-se em evitar o acesso não autorizado, por usuários legítimos ou *crackers*, aos dados armazenados no sistema. Ao longo do estágio, aspectos nos três níveis foram considerados embora o enfoque tenha sido dado à segurança no nível de rede, culminando com a implantação de um *firewall* no sistema.

3.1. Segurança de Contas

Uma das maneiras mais fáceis de um *cracker* conseguir o acesso não autorizado ao sistema é através da conta de algum dos usuários deste sistema. Descobrir senhas alheias não é muito difícil, principalmente porque os usuários teimam em utilizar senhas fáceis e que podem ser facilmente descobertas.

3.1.1. Senhas

Senhas são o mecanismo-chave para implementar segurança no LINUX. É através delas que usuários são autenticados de forma a poderem utilizar o sistema de acordo com seus direitos e permissões.

Manter a segurança de senhas do sistema é muito importante, pois uma vez que um *cracker* tenha descoberto uma dessas senhas ele poderá utilizar os recursos do sistema segundo as propriedades do usuário cuja senha foi descoberta.

O problema aqui reside no fato de que os usuários insistem em utilizar senhas frágeis que, mesmo criptografadas, podem ser facilmente quebradas em tempo hábil por programas desenvolvidos para esse fim, os *cracks*. Então é preciso cuidar para que senhas sejam seguramente escolhidas. Isto depende da realização de um trabalho de conscientização junto ao usuário, único responsável por sua própria senha.

De forma a construir boas senhas alguns conselhos podem ser seguidos:

- Não usar o próprio *login* como senha de forma alguma (duplicado, em letras maiúsculas, ao reverso, etc);
- Não usar combinações com o próprio nome (primeiro nome, último nome, etc);
- Não usar senhas com menos de seis caracteres;
- Não usar palavras de dicionários;
- Misturar letras maiúsculas e minúsculas e caracteres não-alfabéticos, por exemplo, dígitos e/ou pontuação;
- Utilizar uma senha que seja fácil de digitar e memorizar, não sendo necessário escrevê-la em nenhum lugar.

A utilização de senhas seguras ajudará em muito a melhorar o nível de segurança do sistema. Adicionalmente, mecanismos de *password aging* podem ser utilizados, forçando os usuários a mudarem suas senhas periodicamente.

3.1.2. O arquivo /etc/passwd

É através do arquivo */etc/passwd* que é feita a autenticação de usuários no LINUX. Cada linha contém o *login* do usuário, sua senha criptografada, seu UID (*user identification*), seu GID (*group identification*), algum comentário a respeito do usuário, seu diretório de trabalho (*home*) e o interpretador de comandos (*shell*) utilizado. Veja o exemplo para o usuário *root*:

root:$t*p!fh;pl&):0:0:Super-usuário:/bin/ksh

Verificar continuamente o conteúdo deste arquivo é importante para garantir a integridade do sistema de contas. Por exemplo, contas sem senhas ou contas com UID de *root* (UID=0) quando encontradas neste arquivo, evidenciam sérios problemas de segurança. Existem vários pacotes de *software* disponíveis que podem ser utilizados para detectar problemas de segurança gerados pela corrupção do arquivo */etc/passwd*.

Um outro aspecto diz respeito ao fato de que qualquer pessoa tem acesso a este arquivo (qualquer um pode lê-lo) e, conseqüentemente, as senhas criptografadas estão acessíveis a qualquer usuário. Sendo assim, um *cracker* pode muito bem tentar criptografar dicionários selecionados e comparar os resultados com o conteúdo do arquivo */etc/passwd*. Se alguma coincidência ocorrer, a senha terá sido descoberta.

Versões mais novas do LINUX “escondem” estas senhas criptografadas num outro arquivo (*/etc/shadow*, por exemplo), implementando o que chamamos de *shadow passwords*. Este arquivo, de propriedade do administrador, não é acessível para outros usuários. O arquivo */etc/passwd* é, então, modificado e apenas um “*” é colo-

cado no campo de cada linha reservado à senha do usuário. Desta forma, ninguém mais terá acesso às senhas, nem mesmo criptografadas.

3.1.3. Contas Compartilhadas

Contas compartilhadas são um furo de segurança e, portanto, devem ser evitadas. O fato de várias pessoas utilizarem uma mesma conta e, consequentemente, compartilharem uma mesma senha, pode comprometer em muito a segurança do sistema.

Uma forma mais eficiente de prover o compartilhamento de dados entre usuários é através do conceito de grupos. Usuários que pertencem a um mesmo grupo têm um mesmo GID e, assim, podem acessar os mesmos recursos, ou seja, podem ter acesso a qualquer informação possuída pelo grupo como um todo. Cada usuário preserva seu próprio *login* e sua própria senha.

3.1.4. Expiração de Contas

A existência de contas velhas ou não utilizadas no sistema também pode representar um problema de segurança. Isto porque se um *cracker* tiver acesso a uma destas contas ele poderá permanecer durante muito tempo no sistema sem ser notado.

Associar datas de expiração a contas - digamos, um ano depois delas terem sido criadas - evita o acúmulo de contas inutilizadas no sistema. Usuários devem ser notificados antes que as contas sejam desativadas e aquelas contas efetivamente usadas devem, então, ter suas datas de expiração renovadas.

3.1.5. NIS (Network Information Service)

O NIS permite que muitas máquinas compartilhem arquivos de senhas através da rede, arquivos estes que estão normalmente armazenados numa única máquina (tais arquivos são chamados mapas NIS). Isto permite que um usuário possa

se conectar ao sistema a partir de qualquer máquina da rede, já que a autenticação é feita a partir de um único conjunto de arquivos. O NIS, entretanto, contém algumas vulnerabilidades conhecidas.

Para utilizar esse serviço o cliente NIS deve adicionar, no arquivo */etc/passwd*, a seguintelinha:

+::0:0:::

e no arquivo de grupo, */etc/group*, a linha

+:

Se o “+” for esquecido ou apagado, ter-se-á uma conta sem *login*, sem senha e com UID de *root*! O sistema inteiro estará aberto a qualquer tipo de ataque.

Uma vez que qualquer *host* tem acesso aos mapas NIS, programas como os *cracks* podem ser executados a partir de qualquer ponto da rede, utilizando estes mapas como entrada de modo a quebrar as senhas dos usuários do sistema. Além disso, se o NIS é utilizado, uma vez que um intruso tenha acesso a uma conta em uma máquina do sistema, ele terá acesso a todas as outras máquinas automatica-mente. Se a rede está conectada à Internet e sérias preocupações com relação à segurança estão envolvidas, o NIS não deve ser utilizado.

O NIS+, desenvolvido pela *Sun Microsystems*, foi projetado para corrigir as deficiências do NIS (e incluir suas próprias deficiências). Ao contrário do NIS, ele pode ser empregado em grandes redes e, entre outras coisas, inclui características de segurança. Ele foi desenvolvido sobre o sistema seguro de RPC da *Sun*, que

permite autenticação baseada na chamada criptografia de chave pública (voltaremos a falar sobre este método de criptografia na seção 6.2.3).

Servidores NIS+ podem ser configurados para requerer “credenciais criptografadas” ou, se for mais conveniente, eles podem seguir as convenções usuais do LINUX. Assim como um arquivo, qualquer objeto NIS+ tem um proprietário e um grupo proprietário. Permissões sobre os objetos são atribuídas separadamente para o proprietário, o grupo e os outros usuários. Credenciais podem estar associadas a máquinas e a usuários. De uma forma geral, é assim que o NIS+ tenta suprir as principais deficiências do NIS com relação à segurança, ao mesmo tempo em que preserva suas funcionalidades.

3.2. Segurança do Sistema de Arquivos

Para implementar a segurança dos dados armazenados no sistema, o LINUX trabalha com permissões de arquivos. Nove bits controlam quem pode ler (r), escrever (w) ou executar (x) um arquivo. Os três primeiros bits estão associados ao proprietário do arquivo; os três bits seguintes, ao grupo ao qual o arquivo pertence; e os três últimos bits aos outros usuários. Ou seja, a cada arquivo está associado um conjunto de permissões que permite gerenciar o que cada usuário pode fazer com ele.

Outros três bits afetam a operação de arquivos executáveis:

- *setuid* - quando um programa for executado com o bit *setuid* ligado, ele é executado com as permissões do dono do arquivo e não com as permissões de quem o está executando.

- *setgid* - análogo ao *setuid*. O programa executa com as permissões do grupo ao qual pertence o arquivo.
- *sticky bit* - mantém o arquivo executável em memória após a sua execução (em desuso) e faz controle de escrita em diretórios de uso geral (quando ligado, apenas o dono do arquivo pode removê-lo).

Acesso	Significado p/Arquivos	Significado p/Diretórios
r	permite ler conteúdo de arquivo	permite listar diretório
w	permite modificar/remover arquivo	permite alterar conteúdo de diretório (criar/remover arquivos)
x	permite executar arquivo	permite entrar no diretório

Tabela 1: Bits de permissão de arquivos no LINUX

E isto é tudo que o LINUX oferece em termos de segurança dos sistemas de arquivos.

Vejamos alguns pontos que devem ser considerados dentro deste contexto.

3.2.1 Setuid Programs

Programas que executam com o bit *setuid* ligado representam um problema de segurança, principalmente se este bit está associado ao super-usuário *root*. Escrever programas com *setuid* de *root* ligado dá permissões de super-usuário a qualquer usuário do sistema enquanto estes programas estiverem sendo executados e isto pode ser explorado por um intruso a fim de realizar um ataque.

É preferível criar um pseudo-usuário (incluindo nova entrada no arquivo */etc/passwd*), dando-lhe as permissões necessárias à execução de um determinado

programa a ter que associar *setuid* de *root* àquele programa. O bit *setuid* é então ligado com relação àquele usuário.

3.2.2. Umask

Um arquivo é tipicamente criado com todas as permissões habilitadas (rwxrw-xrwx ou 111111111, ou ainda 777). Isto, entretanto, não é usualmente desejável em nenhum sistema. A *umask*, ou valor de máscara, especifica as permissões *default* para criação de arquivos, de forma que se o administrador deseja que os arquivos criados tenham as permissões rwxr-xr-x ou 755, o valor de máscara deve ser 022 - correspondente aos bits que devem ser desligados.

É preciso atentar para este valor de modo que dados não possam ser indevidamente acessados.

3.2.3. Arquivos Criptografados

O *crypt* é o comando padrão utilizado nos sistemas UNIX, mas não no LINUX, para criptografar arquivos mas ele não é muito seguro. De fato, existem muitos programas capazes de descriptografar arquivos criptografados com o *crypt*. Arquivos suficientemente grandes podem ser “quebrados” em algumas horas.

No LINUX os arquivos podem ser criptografados utilizando o protocolo SSH, que cria chaves baseadas em 1024 bits.

O DES (*Data Encryption Standard*) está disponível em alguns sistemas e oferece um nível de segurança um pouco melhor que aquele oferecido pelo *crypt*. Entretanto, assim como o *crypt*, ele utiliza um algoritmo de criptografia baseado em chave privada (ou algoritmo simétrico), em que uma mesma chave é utilizada tanto para codificação como para decodificação. Sendo assim, transmissor e receptor precisam conhecer a chave secreta e única utilizada. Se a chave precisa ser transmitida

através da rede, não há nenhuma garantia de que ela não será interceptada por um intruso.

Outros algoritmos, baseados no conceito de chave pública e também chamados assimétricos, podem ser utilizados. Neste caso, os métodos baseiam-se na utilização de chaves distintas: uma para codificação e outra para decodificação. A mensagem é criptografada com uma chave pública (disponível em uma base de dados), mas só poderá ser descriptografada através de uma chave privada (secreta) em poder do destinatário. O RSA, cujo nome deriva dos nomes de seus criadores - R. L. **R**ivest, A. **S**hamir e L. **A**dleman - é um exemplo de algoritmo assimétrico.

A criptografia é um dos mais antigos mecanismos utilizados para enviar informação através de um meio de comunicação não confiável. Mas apesar das garantias, se alguma informação não deve ser descoberta, é melhor que ela não esteja armazenada no sistema.

3.2.4. Arquivos de Dispositivos

Arquivos de dispositivo constituem uma abstração para utilização de dispositivos no LINUX. Eles são utilizados por outros programas para acessar discos ou memória. Existe uma grande quantidade destes arquivos no sistema, usualmente localizados abaixo do diretório */dev*.

É importante que estes arquivos estejam apropriadamente protegidos. Caso contrário ele podem ser utilizados por intrusos para atacar o sistema.

3.3. Segurança de Rede

Com a interligação de redes à Internet e a adição de serviços de rede como *telnet* e FTP ao sistema, garantir um bom nível de segurança tornou-se uma tarefa muito mais complexa. A exploração de falhas em aplicações TCP/IP, por exemplo,

tem possibilitado diferentes formas de ataque. É preciso proteger-se do resto do mundo.

3.3.1. *Hosts* Confiáveis

Executar comandos remotamente sem que seja necessária qualquer autenticação: esta é a idéia por trás do conceito de *hosts* confiáveis. Neste contexto, um conjunto de *hosts* pode ser especificado e, então, a execução de qualquer comando a partir destas máquinas é feita sem que seja necessário digitar qualquer senha. Uma solução conveniente mas também extremamente insegura.

Os arquivos *hosts.equiv* e *.rhosts* são utilizados pelo sistema para implementar este conceito. Cada linha de *hosts.equiv* especifica um *host* confiável. Se o usuário tem uma conta com mesmo *login* num destes *hosts*, nenhuma senha é requerida para permitir o acesso remoto ao mesmo. *hosts.equiv* é um arquivo que está sob controle exclusivo do administrador do sistema.

No arquivo *.rhosts* combinações *host*-usuário são especificadas, preferivelmente a *hosts* em geral. Arquivos *.rhosts* são controlados por cada usuário, ou seja, cada usuário pode ter um arquivo *.rhosts* em sua conta, de forma que cada usuário pode construir seu próprio conjunto de *hosts* confiáveis. Por exemplo, suponha que o arquivo *~anderson/.rhosts*, na máquina **abara**, contenha as seguintes linhas:

caruru.waytec.com.br

vatapa.waytec.com.br admin

Com este arquivo *.rhosts*, um usuári com *login* **anderson** na máquina **caruru**, poderia executar comandos remotamente (*rlogin* e *rsh*, por exemplo) na conta de **anderson** na máquina **abara** sem digitar uma senha. A segunda linha indica que um

usuário com *login* **admin** no *host* **vatapa** poderia também usar a conta de **anderson** na máquina **abara** sem que fosse necessária qualquer autenticação.

A execução remota de comandos como *rlogin* e *rsh*, baseia-se no conceito de *hosts* confiáveis, de modo que arquivos *hosts.equiv* e *.rhosts* podem representar um perigo para o sistema em termos de segurança (nenhuma autenticação é realizada se a execução é feita a partir de um destes *hosts*). *Hosts* externos nunca devem ser considerados confiáveis e arquivos *.rhosts* devem ser evitados.

3.3.2. NFS (Network File System)

O NFS (*Network File System*), serviço de arquivos distribuído baseado no modelo cliente-servidor, foi projetado para permitir que várias máquinas pudessem compartilhar arquivos através da rede. Com ele, usuários podem ler e alterar arquivos de qualquer ponto da rede sem necessitar de uma senha.

Por *default*, ele não se preocupa com qualquer aspecto de segurança e, de fato, qualquer máquina na Internet pode acessar arquivos locais a menos que você a impeça. Há, entretanto, alguns mecanismos disponíveis para tornar o NFS mais seguro.

O arquivo */etc/exports* é um arquivo de configuração do NFS que lista quais sistemas de arquivos podem ser exportados a partir de um determinado servidor NFS para outras máquinas. Por exemplo,

```
/home -access=poka, poka, lili
```

seria um exemplo de linha do arquivo */etc/exports*, em que as máquinas poka, poka e lili teriam permissão para “montar” o sistema de arquivos */home* a partir do servidor de arquivos NFS (máquina **abara**, no caso do Laboratório). Se a palavra-

chave *access* é omitida, qualquer *host* de qualquer lugar da rede pode montar o sistema de arquivos via NFS. É necessário, portanto, controlar o acesso através do arquivo */etc/exports*, de modo que usuários não autorizados não tenham acesso a informações locais.

Grupos de rede também podem ser definidos no arquivo */etc/netgroup*. Por exemplo, a linha

grupo_segurança (anderson, ,) (lilian, ,)

Define um grupo chamado **grupo_segurança**, formado pelos usuários **anderson** e **lilian**.

Cada membro de um grupo de rede é definido como uma tripla: (*host*, usuário, domínio). O campo domínio é normalmente omitido. Se *host* ou *usuário* é omitido, qualquer *host* ou usuário pode ser combinado com a tripla em questão. Então a tripla (anderson, ,) indica usuária **anderson** em qualquer *host*, enquanto (, lili,) indica qualquer usuário no *host* lili. A partir daí, o sistema de arquivos poderá ser montado por todos os membros constituintes do grupo de rede especificado, como em

/var/security -access=security_group

Onde o sistema de arquivos */var/security* pode ser montado por todos os membros do grupo **grupo_segurança**.

3.3.3. FTP (File Transfer Protocol)

É preciso tomar cuidado com o FTP (*File Transfer Protocol*), ou melhor, com a sua configuração. A utilização deste protocolo permite a conexão remota para trans-

ferência de arquivos e, através do FTP "*anonymous*" usuários que não têm conta no sistema local podem transferir arquivos a partir de um diretório específico.

É preciso cuidar para que, uma vez dentro deste diretório, usuários mal-intencionados não possam comprometer a segurança do sistema. Criar um pseudo-usuário chamado "ftp", cuja conta deve ser desativada com um "*" no campo de senha no arquivo */etc/passwd*, e fazer com que apenas um diretório público (*~ftp/pub*, digamos) seja acessível a qualquer usuário, colocando nele os arquivos que podem ser transferidos são procedimentos padrão quando da configuração deste serviço.

Limitar FTP a alguns *hosts* do sistema também pode contribuir para evitar a ação de intrusos ao mesmo tempo em que facilita um monitoramento da segurança - possíveis violações estão restritas a estes *hosts*.

3.3.4. TFTP (Trivial File Transfer Protocol)

O TFTP (*Trivial File Transfer Protocol*), versão simplificada do FTP, é usado para permitir que estações sem disco (*diskless workstations*) possam ser inicializadas através da rede. Tal protocolo utiliza mecanismos baseados em UDP (*Unit Datagram Protocol*), que por si só não garante qualquer nível de segurança, e não faz qualquer tipo de autenticação. Assim sendo, muitas implementações do TFTP possuem furos de segurança.

O serviço TFTP deve ser configurado de forma que apenas arquivos de inicialização possam ser realmente transferidos. Caso contrário, poderia-se planejar o seguinte ataque:

```
# Exemplo de ataque via tftp
% tftp
tftp> connect yourhost
tftp> get /etc/passwd
tftp> quit
%
```

Ou seja, sem qualquer dificuldade um intruso teria transferido o arquivo de senhas para sua máquina. A partir daí, ele poderia executar um programa chamado *crack*, que tentaria quebrar as senhas utilizadas pelos usuários do sistema.

3.3.5. finger

O utilitário *finger* permite que um usuário obtenha informações sobre outros usuários como nome completo e diretório *home*. Tal informação pode ser obtida, inclusive, por usuários remotos e pode ser utilizada, por exemplo, para descobrir senhas - afinal, alguns usuários teimam em usar sobrenomes ou equivalentes como senhas.

Versões mais antigas do *finger* tiveram *bugs* explorados pelo Verme da Internet, como foi descrito na seção 5.1.4.

3.3.6. telnet

A aplicação te*lnet* provê o que chamamos de “serviço de terminal remoto virtual”. Através desse serviço, usuários podem utilizar terminais remotos como se estivessem sentados a sua frente, tendo acesso aos recursos da máquina segundo suas permissões. Para tanto, são necessários um *login* e uma senha. A disponibilidade desse serviço, entretanto, pode acarretar riscos de segurança ao sistema.

No momento em que a conexão está sendo estabelecida, senha e *login* trafegam através da rede. Intrusos podem estar “escutando” o meio de transmissão a fim de capturar a senha e *login* necessários para estabelecimento desta conexão, de modo que usuários não autorizados poderiam entrar no sistema via *telnet* para atacá-lo.

O serviço *telnet* deve ser restrito a algumas máquinas de modo a minimizar os estragos decorrentes de possíveis violações do sistema.

3.3.7. rlogin e rsh

A execução remota de comando é um dos principais problemas do UNIX em termos de segurança. Ela baseia-se no conceito de *hosts* confiáveis para dispensar qualquer autenticação.

Através do comando *rlogin*, um usuário que tenha conta com mesmo *login* no *host* remoto pode conectar-se a este último uma vez que tenha digitado uma senha e, então, terá disponível uma nova sessão. Se a requisição procede de um *host* confiável, nenhuma autenticação é necessária.

O comando *rsh* permite a execução remota de um único comando e trabalha apenas em cima do conceito de *hosts* confiáveis, executando segundo as permissões atribuídas em arquivos *.rhosts*. Portanto, *hosts* considerados confiáveis devem estar acima de qualquer suspeita (se é que isso é possível).

A execução de comandos remotos é uma das principais vias de acesso para intrusos no sistema.

3.3.8. Correio Eletrônico

O correio eletrônico é um dos serviços mais populares da Internet. O programa *sendmail*, que implementa o protocolo SMTP (*Simple Mail Transfer Protocol*) e que é utilizado para prestação deste serviço, já teve muitos *bugs* explorados e já serviu como porta de entrada para muitos intrusos. Por exemplo, o *sendmail* quando compilado com a opção DEBUG habilitada, permitia acesso irrestrito de usuários remotos ao sistema. Ao executar com as permissões de super-usuário, um ataque podia comprometer todo o sistema.

É preciso estar atento para as correções efetuadas sobre este tipo de programa. *Sendmail*, *rlogin*, *finger* e assim por diante, são serviços que devem estar executando em suas versões mais novas a fim de que o sistema possa estar imune a pelo menos os erros conhecidos até o momento.

4. FERRAMENTAS DE SEGURANÇA

Nesta seção, descreveremos algumas das principais ferramentas e pacotes disponívies na Internet para prover segurança. Ferramentas de segurança podem ser classificadas de acordo com o objetivo a que se propõem: ferramentas de análise do sistema ou ferramentas de autenticação.

4.1. Ferramentas de Análise

Ferramentas de análise são usadas para obter informações sobre o estado do sistema, permitindo monitorar o seu nível de segurança. Algumas ferramentas constituem verdadeiros pacotes de auditoria e checam furos de segurança bem conhecidos. Outras verificam a integridade do sistema de arquivos. Vejamos alguns exemplos.

4.1.1. COPS

O COPS (Computer Oracle and Password System) é um pacote de *software* que procura por erros de configuração no sistema UNIX. Ele verifica modos/permissões de arquivos, fragilidade de senhas, consistência do arquivo */etc/passwd* (o COPS verifica, por exemplo, a existência de entradas duplicadas e contas sem senha), arquivos com SUID de *root* ligado e datas de arquivos binários importantes, entre outros. O COPS não precisa executar como *root*.

4.1.2. CRACK

O *crack* é utilizado para testar as senhas dos usuários de um sistema. Um conjunto de regras é aplicado e vários dicionários são usados, inclusive aqueles for-

necidos durante a instalação do *software*. Ele escolhe as senhas a serem testadas a partir do arquivo */etc/passwd* e possui arquivos de configurações que permitem fazer várias tentativas a partir dos dicionários fornecidos.

A idéia é executar um *crack* antes que o intruso o faça, de modo a alertar os usuários sobre o perigo que suas próprias senhas representam ao sistema como um todo.

4.1.3. Tripwire

O *tripwire* é um monitor de integridade de sistemas de arquivos LINUX. Ele usa diversas rotinas para detectar alterações nos arquivos, bem como para monitorar informações mantidas pelo sistema.

A configuração do *tripwire* permite especificar arquivos e diretórios a serem monitorados, ou não, e também permite especificar arquivos onde alterações limitadas são permitidas sem gerar alerta. O *tripwire* também monitora alterações em permissões, *links* e tamanhos de arquivos e diretórios, permitindo, também, detectar adições e remoções de arquivos nos diretórios controlados.

Uma vez instalado em um "sistema limpo", ele permite identificar modificações não autorizadas em arquivos, detectando a inserção de *back doors* ou bombas lógicas, bem como a existência de vírus. O tripwire não requer privilégios de *root* e não modifica o sistema.

4.1.4 ISS

O ISS (*Internet Security Scanner*) é um *software* que testa a vulnerabilidade da rede. Ele vasculha todos os computadores dentro de uma faixa de endereços IP especificada para determinar o nível de segurança oferecido por cada um, levando em conta as vulnerabilidades mais comumente encontradas em um sistema.

O ISS verifica, por exemplo: a existência de contas *default*, que não devem ter senhas triviais; a configuração do *anonymous* FTP, que deve ser feita cuidadosamente, já que, através desse serviço, usuários sem conta em um determinado *site* têm acesso, embora restrito, a certos diretórios do sistema; a configuração do *sendmail* - comandos *sendmail*, como *wiz* e *debug* devem ser desabilitados.

4.1.5. IcmpInfo

Este *software* monitora mensagens ICMP (*Internet Control Message Protocol*) recebidas de um *host* a fim de detectar atividades de rede suspeitas. Pacotes com mensagens ICMP do tipo *redirect* ou *destination unreachable* podem estar forçando um redirecionamento de rotas, fazendo com que pacotes passem através de "roteadores piratas".

4.1.6. SATAN

Outra ferramenta que pode auxiliar no processo de monitoração do estado de segurança do sistema é o SATAN (*Security Analysis Tool for Auditing Network*). Muita informação sobre *hosts* remotos pode ser obitda ao examinar serviços de rede como *finger*, NFS, NIS, FTP e TFTP.

Executando o SATAN numa determinada máquina, todos os *hosts* diretamente conectados a ela podem ser examinados.

Entre os problemas mais encontrados estão:

- sistemas de arquivos (NFS) exportados sem restrições
- utilização de antigas versões de *sendmail* (anteriores à versão 8.6.10)
- diretório *anonymous* FTP aberto para escrita

Para cada tipo de problema encontrado existe um tutorial explicando o que pode ser feito - corrigir um erro de configuração, instalar um pacote para correção de

um *bug*, usar algum meio para restringir acesso ou desabilitar um serviço - e o seu impacto.

4.1.7. TCP Wrapper

O TCP *Wrapper* é usado para monitorar e registrar as conexões IP entrantes em um sistema. Ele consiste basicamente em interceptar conexões IP, tomando a decisão de permitir/negar determinado serviço de acordo com um conjunto de arquivos de configuração.

Quando chega uma requisição para determinado serviço, o programa servidor *inetd*[3] chama o *tcpd*, servidor do TCP*Wrapper*, que após ter registrado devidamente a requisicão, ativa o servidor apropriado. Isto é configurado no arquivo de configuração do *inetd,* o *inetd.conf*. Por exemplo:

```
# Exemplo de chamada do servidor tcpd no arquivo inetd.conf
telnet stream tcp nowait root /etc/tcpd telnetd
```

Qualquer conexão solicitada tem sua hora e origem registradas. Se o controle de acesso está ligado, uma lista será verificada para ver se a origem da conexão pode ou não acessar aquele serviço. Em caso afirmativo, o *tcpd* chama o servidor real para processar a requisição.

O controle de acesso é basicamente implementado a partir de dois arquivos: */etc/hosts.allow* e */etc/hosts.deny*. No primeiro, definem-se quais serviços são permitidos para quais *hosts* da rede. Analogamente, no arquivo */etc/hosts.deny*, definem-se quais serviços são negados.

---

[3] O *inetd* (*internet daemon*) é um programa utilizado para inicialização de serviços TCP/IP. De acordo com a requisição feita, o *inetd* inicializa o servidor apropriado para realização do serviço.

Um exemplo de entrada no arquivo *hosts.deny* seria:

Exemplo de configuração no arquivo *hosts.deny*

telnetd: 150.165.75.21: echo Permission denied

Tão logo realizem seu trabalho, estes servidores “morrem”. Os serviços inicializados pelo *inetd* são chamados serviços inicializados por demanda.

Neste caso, o acesso via *telnet* à máquina onde este arquivo foi configurado, seria negado para conexões oriundas na máquina cujo endereço IP é 150.165.75.21. Apenas os serviços inicializados pelo *inetd* podem ser controlados pelo TCP *Wrapper*.

4.2. Ferramentas de Autenticação

4.2.1. Npasswd

O *npasswd*, assim como *passwd+*, são programas que servem como substitutos para o programa de troca de senhas *default* do LINUX (*passwd*). Eles implementam uma validação da senha digitada, verificando a sua fragilidade (via comparação com dicionários) e outras características atribuídas pelo administrador (tamanho mínimo, número máximo de caracteres repetidos, e outras).

4.2.2. TIS Internet Firewall Toolkit (FWTK)

O TIS é um conjunto de programas e práticas de configuração projetados para facilitar a construção de *firewalls*. Ele inclui:

- um servidor de autenticação que provê vários mecanismos para verificar fragilidade de senhas;

- servidores *proxies* para uma variedade de protocolos: FTP, HTTP, *Gopher*, *rlogin*, *telnet* e X11 - vide mais sobre serviços *proxies* na seção 8.3;
- um servidor *proxy* genérico para protocolos TCP;
- um *wrapper* para servidores SMTP;
- um *wrapper* para servidores inicializados pelo *inetd*, como o *telnetd* e *ftpd*.

Todos os componentes do TIS compartilham um arquivo de configuração comum, o *netperm-table*. Nele são configurados o *netacl* (*wrapper* para servidores inicializados pelo servidor *inetd*), servidores *proxies* que devem ser disponibilizados no *firewall* e servidor de autenticação, chamado *authsrv*.

O programa *netacl* é configurado de modo a permitir um acesso limitado ao *firewall*, tal qual um TCP *Wrapper*. Na verdade, ele é incluído no pacote como uma implementação mínima de um TCP *Wrapper*. Servidores *proxies* podem ser configurados de acordo com um conjunto de permissões, de modo que um controle de acesso por cada serviço disponível também pode ser mantido.

O servidor de autenticação, o *authsrv*, suporta múltiplos mecanismos de autenticação. Para cada usuário, um mecanismo pode ser escolhido e o serviço de autenticação dever ser associado a cada serviço *proxy* que deseje utilizá-lo. Veja o exemplo a seguir:

```
# Exemplo de utilização do serviço de autenticação com o FTP proxy
ftp-gw: authserver localhost 7777
ftp-gw: denial-msg /usr/local/etc/ftp-deny.txt
ftp-gw: welcome-msg /usr/local/etc/ftp-welcome.txt
ftp-gw: help-msg /usr/local/etc/ftp-help.txt
```

ftp-gw: permit-*hosts* 150.165.75.*

ftp-gw: permit-*hosts* * -auth

ftp-gw: timeout 3600

Neste exemplo, o serviço de autenticação está sendo utilizado com o *proxy* FTP (ftp-gw).

A quinta linha indica que *hosts* na rede 150.165.75.0 podem usar o *proxy* FTP sem que seja necessário autenticação. Todos os outros acessos devem ser devidamente autenticados (parâmetro - *auth*) e caso o acesso não seja permitido, uma mensagem de negação de serviço, */usr/local/etc/ftpdeny.txt*, é mostrada e o cliente é desconectado.

O pacote é projetado de forma que é possível utilizar apenas algumas das funcionalidades oferecidas. Voltaremos a falar sobre o FWTK nas próximas seções.

5. FIREWALLS

Como foi dito anteriormente, ataques podem ter origem interna, quando feitos por usuários do próprio *site*, ou externa, quando executados por alguém em outro *site*. No sentido de aumentar a segurança de redes ligadas à Internet, ou seja, no sentido de proteger a rede interna contra ataques externos, o *firewall* é um mecanismo que tem sido muito utilizado.

O *firewall* é uma espécie de barreira de proteção que interliga a rede interna à Internet; é uma coleção de componentes colocada entre duas redes, que coletivamente possui as seguintes propriedades [SOARES 95]:

- todo o tráfego de dentro para fora da rede passa através do *firewall*, e vice-versa;
- só o tráfego autorizado pela política de segurança deve passar pelo *firewall*;
- o *firewall* deve ser à prova de violações.

Ao forçar a passagem de qualquer tipo de tráfego através do *firewall*, cada *host* do sistema na rede privada fica protegido contra ataques externos, uma vez que, para atingi-los é preciso, primeiramente, "vencer" o *firewall*. Trata-se de criar uma rede interna não-acessível diretamente, onde apenas o *firewall* se comunica com o mundo. O *gateway* do *firewall* é normalmente conhecido como *bastion host*, sendo a única máquina dentro da rede interna diretamente acessível a partir da Internet.

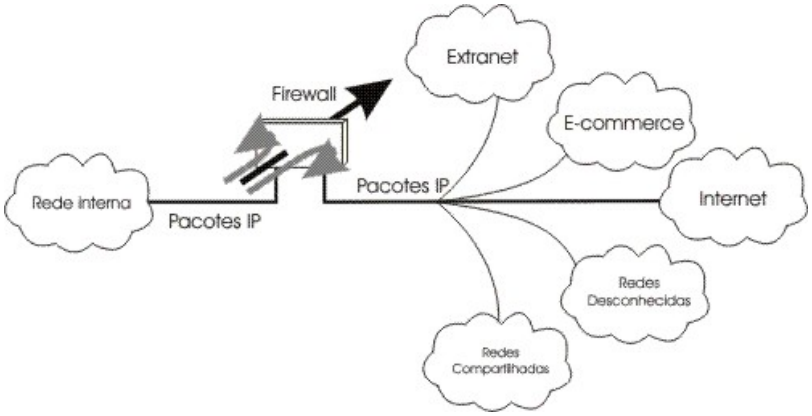


Figura 1: *Firewall* - uma barreira de proteção

Com a utilização de um *firewall*, muitos são os benefícios obtidos:

Centralização da segurança da rede. Para que esta "barreira" seja efetiva, todo o tráfego de informações que entra ou que sai da rede tem que passar por ela, ou seja, todas as conexões que vêm de fora para a rede, e vice-versa, têm que ser filtradas em um único ponto, considerado como sendo o ponto forte de defesa do sistema que mantém os indesejáveis intrusos fora da rede protegida.

Desta forma, a gerência de segurança da rede é simplificada pelo *firewall* que centraliza a defesa, preferivelmente a distribuí-la por todos os *hosts* em toda a rede privada.

Determinação dos serviços possíveis de serem utilizados. Este mecanismo de proteção também determina quais serviços internos podem ser acessados de fora, quais usuários externos têm a permissão de acessar os serviços permitidos internamente na rede, e quais serviços externos podem ser acessados pelos usuários internos.

Cobrança por utilização de recursos. Se esta questão faz parte das diretrizes da empresa, o *firewall* torna-se útil, pois é um ponto onde o administrador de rede pode verificar as taxas de utilização de serviços externos (Internet) de cada usuário.

Disponibilização de servidores Web e FTP. O *firewall* é também um ponto ideal para se disponibilizar servidores da Internet como um servidor *Web* e um servidor FTP. Este é configurado de forma a permitir acesso apenas a esses servidores, enquanto proíbe acesso, de usuários externos, aos outros recursos da rede protegida.

Entretanto, uma vez que um intruso tenha conseguido passar através do *firewall*, nenhuma proteção é provida, de modo que é muito importante dispensar uma cuidadosa administração, por parte dos administradores do sistema, à(s) máquina(s) que implementa(m) o *firewall*. O *firewall* é, por exemplo, um bom lugar para implementar mecanismos mais sofisticados de autenticação.

5.1. Projetando um Sistema de Firewall

Ao se decidir por implantar um *firewall* na rede, alguns aspectos precisam ser considerados, como por exemplo, qual a postura do *firewall* e quais serão os seus componentes ou blocos constituintes.

Há duas posturas diametricalmente opostas:

- *Tudo que não é especificamente permitido é negado*. Esta postura assume que o *firewall* bloqueará todo o tráfego e que cada serviço deve ser explicitamente habilitado.
- *Tudo que não é especificamente negado é permitido*. Aqui, o *firewall* deve assumir que todo o tráfego pode ser repassado, a menos que ele seja explicitamente proibido.

No primeiro caso, tem-se um nível mais alto de segurança desde que apenas os serviços selecionados cuidadosamente são suportados. No segundo caso, entretanto, tem-se uma maior flexibilidade, uma vez que uma maior disponibilidade de

serviços é provida aos usuários. A idéia é que, para diminuir os riscos, a configuração dos *firewalls* deve ser minimizada, excluindo tudo que não seja estritamente necessário.

Um outro aspecto diz respeito a qual arquitetura do *firewall* adotar. Há três arquiteturas básicas: *dual homed host*, *screened host* e *screened subnet*. Veremos cada uma delas com mais detalhes nas próximas seções. Também é necessário decidir quais funcionalidades implementar no *firewall*, levando em conta duas opções básicas: filtros IP e servidores *proxies*.

As alternativas mostradas não são mutuamente exclusivas. A escolha certa normalmente resulta na combinação de diferentes técnicas para solucionar diferentes problemas.

5.2. Filtros de Pacotes

A idéia aqui é simples: o roteador retransmitirá pacotes TCP/IP entre as duas redes interligadas segundo um conjunto de regras de filtragem. Tais regras são baseadas na informação contida no cabeçalho dos pacotes, de forma que, para cada datagrama, o roteador examinará seu cabeçalho, tomando a decisão de permitir/negar sua passagem de acordo com as regras definidas. As informações do cabeçalho relevantes e que são utilizadas por esta técnica de filtragem de pacotes incluem: endereço IP fonte, endereço IP destino, protocolo utilizado (se TCP ou UDP, por exemplo), porta TCP/UDP fonte, porta TCP/UDP destino, tipo de mensagem ICMP, interface de entrada e interface de saída.

Alguns exemplos de regras de filtragem:

- permitir sessões de *telnet* e FTP oriundas de usuários externos, apenas para uma lista de *hosts* internos específicos

- negar todo o tráfego entrante oriundo de redes externas específicas

Algumas vantagens são obtidas ao se implementar filtros de pacotes no roteador. Primeiro, o planejamento e a configuração dos filtros é simples. Pouco tempo é gasto para configurar as regras de filtragem. Depois, se o tráfego através do roteador é moderado, um filtro de pacotes causará pouco impacto sobre a desempenho do sistema ao mesmo tempo em que oferece flexibilidade e transparência para os usuários da rede.

Entretanto, muitas são as desvantagens:

1. Devido a esta técnica dispor de um tratamento pacote a pacote - no nível de rede não tem como estabelecer uma conexão entre pacotes que entram e que saem e que, portanto, estão relacionados através dessa conexão - o sistema perderá em desempenho, pois será gasto mais tempo de CPU para extrair as informações necessárias do cabeçalho de cada pacote que chega ao *firewall*.
2. Esta técnica não provê nenhum tipo de proteção aos dados, nem qualquer proteção no nível de protocolo de aplicação. Por exemplo, quando o tráfego do protocolo de correio eletrônico, SMTP, é permitido entre clientes externos e servidores internos, o *firewall* não tem como proteger o servidor de um ataque de um cliente que envia uma enorme quantidade de mensagens (ataque conhecido como flood). Tais ataques só poderiam ser prevenidos tomando-se medidas de segurança adicionais internas aos servidores de correio eletrônico.
3. Os registros de *log* são muito limitados.

4. Existem poucas facilidades para verificar a validade das regras de filtragem utilizadas pelo *firewall*, podendo tornar o sistema vulnerável a ataques.

5.3. Serviços Proxies

Os serviços *proxies*, ao invés de basear-se num mecanismo de propósito geral, como os filtros de pacotes, constituem aplicações especiais, desenvolvidas especificamente para funcionar de forma segura. Para cada serviço existe um servidor *proxy* disponível (*proxy*-FTP, *proxy-telnet*, e qualquer outro serviço que o administrador deseje disponibilizar), de modo que o cliente não mais se conecta diretamente com o servidor externo, mas com o servidor *proxy* no *firewall*.

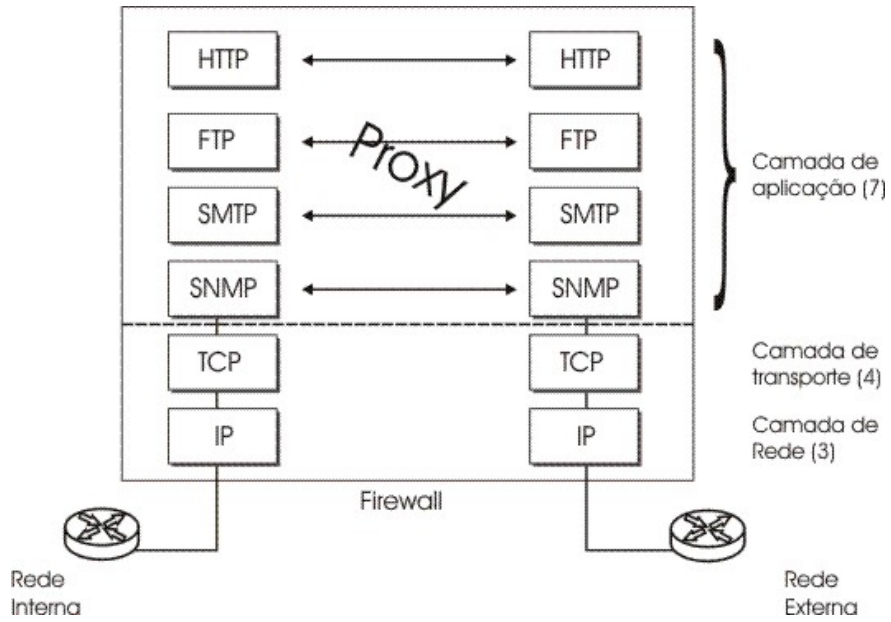


Figura 2: Serviços *proxies* - o cliente conecta-se ao servidor *proxy* que, por sua vez, repassa a conexão para o servidor real

Um serviço *proxy* requer dois componentes: um servidor *proxy* e um cliente *proxy.* O cliente *proxy*, ao invés de conectar-se diretamente com o servidor “real”,

conecta-se com o servidor *proxy*. O servidor *proxy* avalia a requisição e decide o que fazer. Se a requisição deve ser atendida, o servidor *proxy* contacta o servidor real e passa a retransmitir dados entre cliente *proxy* e servidor real, como se a conexão se originasse na máquina onde o serviço *proxy* é executado. Os *proxies*, então, refazem conexões no nível de aplicação e atuam, assim, como g*ateways* de serviços, sendo também chamados *gateways* de aplicação.

A idéia por trás dos serviços *proxies* é tornar a rede interna invisível ao mundo externo com a utilização de endereços IP virtuais, não externamente roteáveis, e mesmo assim possibilitar que usuários internos continuem utilizando serviços externos via servidores *proxies*, já que estes últimos refazem conexões como se elas se originassem no *firewall* (o *firewall* possui um endereço IP válido). Isto provê um alto nível de segurança ao sistema ao mesmo tempo em que não compromete a obtenção de serviços externos à rede.

Novamente, há vantagens e desvantagens ao se utilizar um *gateway* de aplicação.

Primeiramente, esta técnica provê o nível mais elevado de segurança (construção de redes internas virtuais) e já que o *gateway* de aplicação, como o próprio nome diz, provê segurança no nível de aplicação, prováveis falhas no protocolo IP não são relevantes; servidores *proxies* provêem *caching* de dados, guardando dados sobre as ligações (*links*) mais frequentes; registros de *log* podem ser mais completos, uma vez que os servidores *proxies* entendem melhor o protocolo cliente/servidor a eles relacionado, tornando-se possível gerar registros de *log* com mais detalhes como o número de bytes transferidos entre cliente e servidor, duração, número de conexões, entre outros; por fim, as regras de filtragem utilizadas por esta técnica são mais fáceis de serem mantidas e testadas.

Entretanto, não há um servidor *proxy* genérico. O *firewall* tem que ter um servidor *proxy* para cada serviço provido pela rede (*proxy-telnet*, *proxy*-FTP e assim por diante). Em conseqüência disso, tem-se que nem todos os serviços podem ser suportados por um *firewall* baseado em servidores *proxies*. Servidores *proxies* que suportem o NFS, por exemplo, são bastante raros.

Note ainda, que para que os servidores *proxies* possam efetivamente proteger a rede interna, é preciso desabilitar o roteamento direto de pacotes pelo *firewall*. Caso contrário, os pacotes externos poderão ignorar os servidores *proxies* e contactar diretamente máquinas internas, já que servidores *proxies* executam no nível de aplicação e o roteamento seria feito ainda no nível de rede da pilha de protocolos.

5.4. Arquiteturas de Firewall

Há três tipos básicos de arquitetura de *firewall*: arquitetura *dual-homed host*, arquitetura *screened host* e arquitetura *screened subnet.* Vejamos como se apresenta cada uma delas.

5.4.1. Arquitetura Dual-Homed Host

Este tipo de arquitetura baseia-se num *host dual-homed*, um computador que tem pelo menos duas interfaces de rede. O *host dual-homed* é colocado entre a Internet e a rede interna e passa, então, a interceptar todo o tráfego entre as duas redes conectadas. Pacotes IP de uma rede não são diretamente roteados para a outra rede. *Hosts* internos comunicam-se com o *firewall* que comunica-se com a Internet, e vice-versa. A rede interna é totalmente invisível ao mundo externo, que por sua vez

só enxerga o *firewall*. Tem-se aqui o conceito de uma rede interna virtual com a utilização de serviços *proxies* no *firewall*.

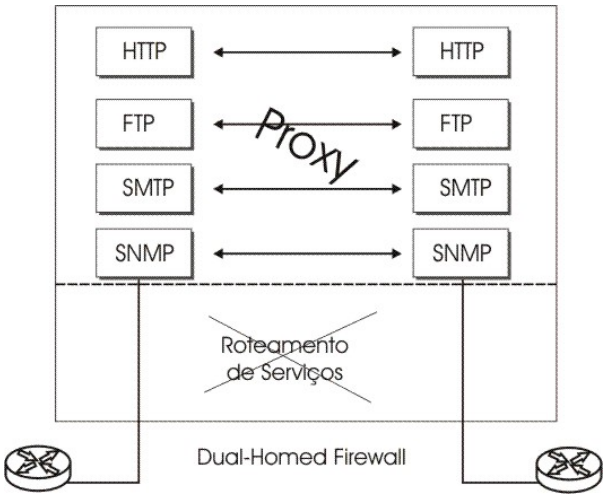


Figura 3: Arquitetura *dual-homed host*

Hosts dual-homed provêem um alto nível de controle. Se, por exemplo, algum pacote na rede interna possui um endereço-fonte externo à rede local, algum problema de segurança está acontecendo.

5.4.2. Arquitetura Screened-Host

A arquitetura *screened-host* implementa o sistema de *firewall* em mais de uma máquina.

Há um roteador separado que interliga a Internet à rede interna, desempenhando suas funções específicas de roteamento, e uma outra máquina, chamada *bastion-host*, que é a única máquina dentro da rede interna que comunica-se com a Internet diretamente.

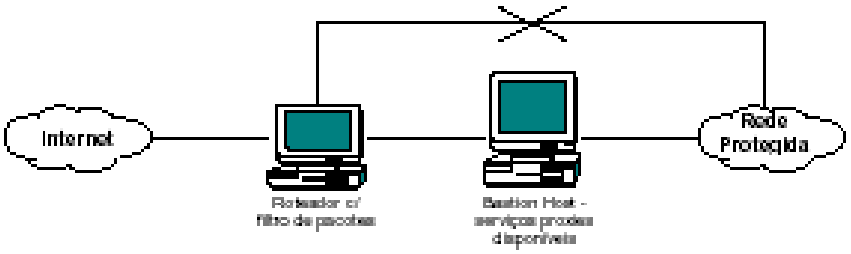


Figura 4: Arquitetura *Screened-Host*

O filtro de pacotes no roteador é configurado de forma que qualquer tráfego vindo da Internet seja enviado apenas para o *bastion host*. Qualquer sistema externo que tente acessar a rede interna, terá que fazê-lo através do *bastion host* que, justamente por isto, deverá ser cuidadosamente protegido. Qualquer *host* interno que deseje comunicar-se com o mundo externo também terá que fazê-lo através do *bastion host*, onde estarão os servidores *proxies*.

Alguns aspectos que devem ser considerados no sentido de prover um melhor nível de segurança ao *bastion host* incluem:

- O *bastion host* deve executar uma versão do sistema operacional tão restrita (*stripped down*) e segura quanto possível - módulos adicionais só devem ser instalados se estritamente necessários.
- Só instalar serviços estritamente essenciais no *bastion host* (*telnet*, FTP, DNS, SMTP), pois serviços não instalados não podem ser atacados.
- Cada *proxy* é configurado para suportar apenas um subconjunto de operações que podem ser realizadas em *hosts* específicos da rede (por exemplo, limitar *telnet* ao *bastion-host* ou utilizar o *proxy*-FTP apenas para fazer *download* de arquivos).

- Os *proxies* são independentes entre si. Se há algum problema com a operação de qualquer *proxy*, ou se uma vulnerabilidade é descoberta, este serviço pode ser desabilitado sem afetar os demais serviços. Além disso, novos serviços podem ser facilmente adicionados ao sistema.
- Cada *proxy* executa como um usuário não-privilegiado em um diretório seguro no *bastion-host*.
- Pacotes de auditoria como COPS, Tiger e Tripwire devem ser instalados no *bastion host.*

Os *logs* gerados são úteis para monitorar o seu nível de segurança.

Alternativamente, a configuração de um filtro IP no roteador pode permitir outros *hosts* internos abrirem conexões diretas para certos serviços. Observe, entretanto, que isto diminui o nível de segurança do sistema a partir do momento em que expõe mais *hosts* ao mundo externo.

5.4.3. Arquitetura Screened-Subnet

Vimos que na arquitetura *screened-host* o *bastion host* funciona como intermediário entre rede interna e Internet, protegendo a primeira contra ataques externos. Mas uma vez que um intruso tenha conseguido passar pelo *bastion host*, ele terá conseguido penetrar inteiramente no sistema, pois nenhum mecanismo de segurança adicional é implementado entre o *bastion host* e os demais *hosts* internos da rede.

A arquitetura *screened-subnet* implementa um nível extra de segurança. Há neste caso, dois roteadores além do *bastion host*. Um deles está entre a Internet e o *bastion host* (roteador externo) e o outro está entre o *bastion host* e a rede interna (roteador interno). Para conseguir acesso a rede interna com este tipo de arquitetu-

ra, um intruso terá que passar pelos dois roteadores. Ainda que ele passe pelo *bastion host*, terá que enfrentar mais um roteador.

- Cada *proxy* é configurado para suportar apenas um subconjunto de operações que podem ser realizadas em *hosts* específicos da rede (por exemplo, limitar *telnet* ao *bastion-host* ou utilizar o *proxy*-FTP apenas para fazer *download* de arquivos).
- Os *proxies* são independentes entre si. Se há algum problema com a operação de qualquer *proxy*, ou se uma vulnerabilidade é descoberta, este serviço pode ser desabilitado sem afetar os demais serviços. Além disso, novos serviços podem ser facilmente adicionados ao sistema.
- Cada *proxy* executa como um usuário não-privilegiado em um diretório seguro no *bastion-host*.
- Pacotes de auditoria como COPS, Tiger e Tripwire devem ser instalados no *bastion host.*

Os *logs* gerados são úteis para monitorar o seu nível de segurança.

Alternativamente, a configuração de um filtro IP no roteador pode permitir outros *hosts* internos abrirem conexões diretas para certos serviços. Observe, entretanto, que isto diminui o nível de segurança do sistema a partir do momento em que expõe mais *hosts* ao mundo externo.

5.4.3. Arquitetura Screened-Subnet

Vimos que na arquitetura *screened-host* o *bastion host* funciona como intermediário entre rede interna e Internet, protegendo a primeira contra ataques externos. Mas uma vez que um intruso tenha conseguido passar pelo *bastion host*, ele terá conseguido penetrar inteiramente no sistema, pois nenhum mecanismo de se-

gurança adicional é implementado entre o *bastion host* e os demais *hosts* internos da rede.

A arquitetura *screened-subnet* implementa um nível extra de segurança. Há neste caso, dois roteadores além do *bastion host*. Um deles está entre a Internet e o *bastion host* (roteador externo) e o outro está entre o *bastion host* e a rede interna (roteador interno). Para conseguir acesso a rede interna com este tipo de arquitetura, um intruso terá que passar pelos dois roteadores. Ainda que ele passe pelo *bastion host*, terá que enfrentar mais um roteador.

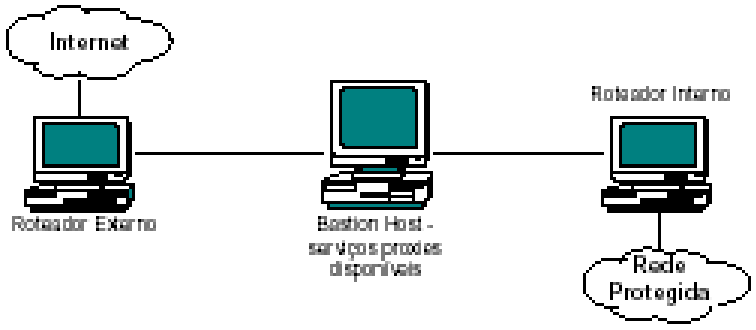


Figura 5: Arquitetura *Screened-Subnet*

A filtragem de pacotes é mais intensa no roteador interno, mesmo porque o roteador externo normalmente foge ao escopo da administração local da rede (ele seria, por exemplo, um provedor Internet). É no roteador interno que as regras para limitação de serviços são efetivamente definidas. No roteador externo, as regras são mais genéricas e visam a proteger apenas o *bastion host* e o roteador interno.

Na segunda parte deste trabalho, apresentamos um exemplo de estratégia de segurança que pode ser implementado segundo as políticas locais de um determinado sistema. Neste exemplo, *firewalls* são utilizados, além de medidas adicionais de segurança, como uma política de segurança nos servidores de arquivos utilizando

serviços SAMBA com autenticação windows, um servidor *proxy* utilizando o SQUID e um servidor de mensagens eletrônicas utilizando anti-vírus de mensagens.

6. PLANEJAMENTO E INSTALAÇÃO

O LINUX oferece mais flexibilidade de escolha do que a maioria dos outros sistemas operacionais. Muitos sistemas UNIX limitam sua escolha de fornecedores de hardware porque o sistema só funciona em um tipo de *hardware*. O *Windows* da *Microsoft* limita sua escolha de fornecedores de sistemas operacionais, pois só está disponível pela *Microsoft*. Com o LINUX, pode se escolher entre diferentes distribuições e todas funcionam com uma vasta quantidade de diferentes *hardwares*.

6.1. Distribuição LINUX

O LINUX é um sistema operacional baseado no UNIX eficiente e com várias características. Mas, até mesmo para os profissionais de UNIX, o mundo LINUX pode parecer confuso.

Profissionais de informática que estão aprendendo UNIX ouvem com freqüência que existem muitas versões de UNIX. Com isso, querem dizer que Solaris, Irix, AIX, BSD e HP UNIX são todos diferentes. Cada um possui um processo de instalação diferentes. Cada um tem uma localização diferente para arquivos de inclusão ou de sistema e cada tem suas próprias ferramentas. Para profissionais de sistema que se conformam com um revendedor quer usa apenas um fornecedor de sistema operacional, esta variedade pode ser assustadora.

Assim como o UNIX, existem muitas versões de LINUX. As diferenças entre elas são superficiais mas extremamente importantes, pois as diferenças ocorrem freqüentemente naquelas áreas onde o administrador está mais envolvido: instalação, localização de arquivos de inclusão, chaves de configuração, ferramentas administrativas e interface de usuário.

Facilidade de manutenção e administração são características importantes na redução dos custos totais dos direitos. Durante o estágio foi escolhida a distribuição Conectiva variando da versão 6 a 8, pois, o suporte oferecido é profissional, aparenta possuir estabilidade no produto e suas versões estão voltadas para o público empresarial.

6.1. Escolhendo o Hardware

O LINUX roda praticamente em qualquer *hardware* de PC, desde o obsoleto 486 até o processador mais atual. Existem atualmente três famílias de procesadores tipo Pentium usados nos novos sistemas LINUX. São: os processadores AMD, Cyrix e Pentium da Intel. Estes processadores oferecem uma grande variedade de preços e performances. Na Waytec, o *hardware* foi escolhido com base no preço e estabilidade decidido pelo gerente da área. Foi dividida as tarefas entre os sitemas da seguinte forma:

Servidor DNS: Necessita muito pouco recurso de CPU ou capacidade de disco. O escolhido foi um AMD K6-2, com velocidade de 400 MHz, 64 MB de memória RAM, 500 MB de disco e placa de rede 3COM.

Servidor de Correio: Requer um disco de capacidade considerável, mas pouco recurso de CPU. O número de usuários de e-mail que determina o tamanho do disco que precisa. No caso da Waytec usa um AMD Athon XP, com velocidade de 1.100 MHz, 256 de memória RAM, 30 GB de disco do tipo SCSI (*Small Computer System Interface*) para melhoria na performance e duas placas de rede 3COM.

Servidor Proxy: Usa um AMD K6-2, com velocidade de 500 MHZ, 128 MB de memória RAM, disco de 10 GB de espaço para um bom *cache* de páginas e duas placas de rede 3COM.

Servidor de Arquivos: Usa um AMD K6-2, com velocidade de 500 MHZ, 256 MB de memória RAM, um disco SCSI de 30 GB, um disco IDE de 20 GB, uma unidade de fita DAT DDS 4 - 20/40 e uma placa de rede Intel.

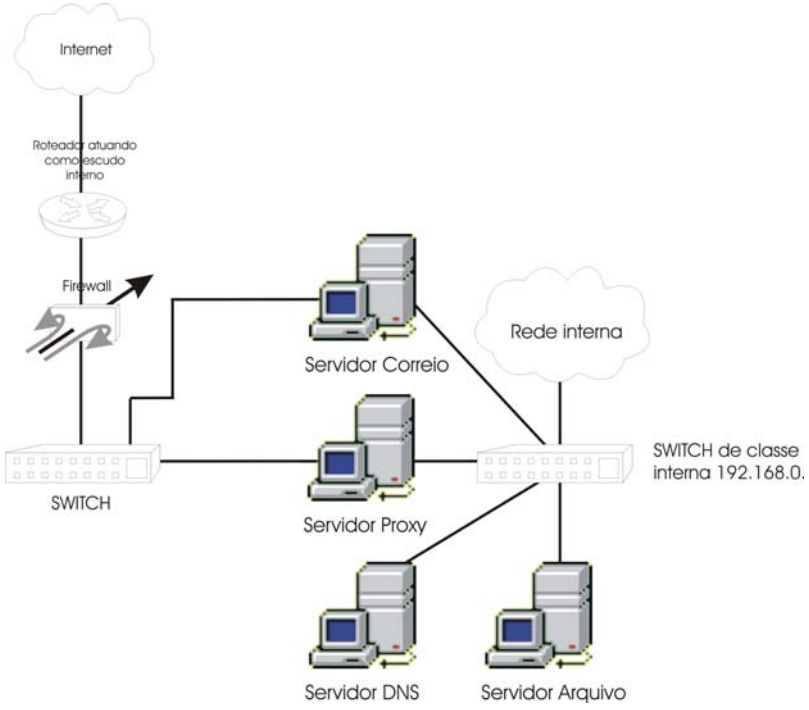
Máquinas Clientes: A rede é composta por 52 usuários, com máquina variando de AMD K6-2 500 MHZ a Pentium 4 de 2,0 GHZ e todas com o sistema operacional *Windows* da *Microsoft*.

Além de CPU, memória, disco e velocidade algumas características foram observadas:

- Verificou o hardware no "*Linux Hardware Compatibility HOWTO"* de Patrick Reijnen [16] e a página da Internet do fabricante da distribuição [17] para ver se existe alguma atualização ou correção para o hardware selecionado. Não foram encontradas atualizações.
- Um servidor de rede não necessita de gráficos de grande performance. É usada a console do servidor para aplicações administrativas. Uma placa de vídeo barata é a utilizada (SiS 530).
- Não foram comprados equipamentos dos quais não necessitará. Como placas de som.

7. DESENHO E PLANEJAMENTO DA REDE

O desenho da rede ficou definido como a seguir:



7.2. Estrutura Adotada para as Redes Locais na Waytec

No projeto de um ambiente de rede local, a associação dos diversos dispositivos eletrônicos e a elaboração do projeto físico compreendem a consideração de diversos aspectos importantes de distâncias, escolha do meio, definição de infraestrutura de dutos, desempenho do sistema, localização das estações etc., que possuem influência direta no custo final da rede a ser implantada. Dessa forma, todas as definições e recomendações deste documento devem ser criteriosamente avaliadas, e implantadas por profissionais com conhecimentos específicos.

7.2.1. Tecnologias recomendadas

Dentre as tecnologias de LAN existentes, este documento recomenda para uso interno às edificações da Waytec, cobrindo uma larga faixa de aplicações, a utilização do padrão 802.3 do IEEE (*Institute of Electrical and Electronic Engineers*), também conhecido como padrão *Ethernet* e as suas variações de alta velocidade (*fast e giga ethernet*), todas baseadas no método CSMA/CD (*Carrier Sense Multiple Access with Collision Detection*).

7.2.2. Equipamentos

O mais simples dos equipamentos capazes de operacionalizar uma rede física, em concordância com as especificações anteriores, é conhecido como HUB que em conjunto com as placas de rede das estações, torna possível o intercâmbio de dados. Os HUBs devem ter características mínimas de desempenho, capacidade de empilhamento. Gerenciamento por SNMP e de segurança, tais como proteção contra intrusão e contra interceptação serão opcionais.

Proteção contra intrusão significa que em cada porta do HUB só será permitida a ligação de estações com o endereço físico *Ethernet* (*MAC address*) configurado na porta do equipamento; proteção contra interceptação significa que um dado transmitido só será reconhecido e válido na porta configurada com o endereço físico *Ethernet* de destino (enviado junto com o cabeçalho da mensagem); nas demais portas a mensagem não é reconhecida evitando-se assim, a monitoração do tráfego.

7.2.3. Infra-estrutura e cabeamento

Quando se lida com projetos de cabeamento deve-se considerar os efeitos de agentes propagantes de chama e de fumaça. Muitas instalações possuem espaços para o transporte de ar em sistemas de condicionamento ambiental pelo forro ou piso, conhecidos pelo termo em inglês, plenum. Assim, essas áreas possuem comu-

nicação com diversos ambientes e são fontes propagantes de fumaça na ocorrência de um acidente. Para evitar catástrofes, existem técnicas e materiais adequados para serem aplicados nas instalações de cabeamento:

- Cabos com capas externas do tipo Plenum; são capas em Teflon, ao invés do tradicional PVC, que apresentam diversas classificações NEC (*National Electric Code*) de acordo com a aplicação. Dessas, a especificação Riser indica que o cabo possui baixa propagação de chama na vertical sendo especialmente indicado para cabeamento tronco; para o cabeamento horizontal podem ser utilizadas as especificações CM ou CMX. Essas especificações são gravadas ao longo do cabo e especialmente nos cabos de origem americana e européia.
- Utilização de *fire stopping*, isto é, produtos que retêm o fogo e são facilmente removidos quando necessário. As áreas indicados para aplicação desses produtos são aberturas feitas para instalação de infraestrutura em paredes ou piso (prumadas verticais, *shafts*, passagens feitas através dos ambientes pelas eletrocalhas, etc..). Existem em duas categorias: os mecânicos e não mecânicos.

No primeiro caso, os produtos consistem de materiais anti-infiamáveis prémanufaturados que se ajustam perfeitamente aos cabos, calhas ou eletrodutos existentes. No segundo caso, eles apresentam diversos formatos e texturas e adaptam-se a aberturas irregulares. Na segunda opção podemos destacar os seguintes produtos: *Fire Rated Mortar, Silicone Foam e Firestop Pillows*.

7.2.4. Infra-estrutura

A infra-estrutura, neste documento, representa o conjunto de componentes necessários ao encaminhamento e passagem dos cabos, para aplicações multimídia, em todo os pontos da edificação, assim como os produtos necessários à instalação dos componentes ativos do sistema que compõem uma rede local.

Fazem parte dessa classificação os seguintes materiais: eletrocalhas, eletrodutos, caixas de passagem, gabinetes, suportes de fixação, buchas, parafusos, etc. As edificações são dinâmicas, e durante a vida de um prédio são executadas diversas reformas, assim deve almejar que um projeto de infra-estrutura seja suficientemente capaz de preservar o investimento e garantir condições técnicas de alterações e/ou expansões durante cerca de 15 anos. Como existem diversas opções de arquitetura e engenharia utilizada na construção de um prédio, o sistema mais utilizado no mercado e os principais requisitos da norma TIA/EIA 569-A de fevereiro de 1998. Foi adotado como recomendação para o modelo básico de infra-estrutura o sistema composto por eletrocalhas e eletrodutos. Esse sistema de encaminhamento de cabos permite uma excelente flexibilidade e capacidade de expansão com custo reduzido,

Outros sistemas como o de dutos de piso ou rodapé falso, ainda que atendam as normas TIA/EIA 569-A, não estão regulamentados neste documento e devem ser criteriosamente analisados, antes da execução do projeto, pois apresentam sérias desvantagens de expansão e podem, ainda, resultar em interferências e redução no desempenho nas redes locais instaladas. A opção de piso elevado, utilizada geralmente em salas de processamento corporativo (antigos CPD), é uma excelente opção para locais com alterações constantes de layout e imprevisibilidade. Deverá atender à especificação do ítem 4.3 da TIA/EIA 569-A e o CCE e os CIs devem ser consultados para auxiliar no projeto. Os eletrodutos e eletrocalhas a serem utilizados

devem obrigatoriamente ser do tipo metálico rígido, dando preferência para tratamento com zincagem a quente (pószincagem) ou alternativamente, a frio (galvanização eletrolítica). Todo o conjunto (eletrocalha, eletroduto e acessórios) deve ser aterrado em um único ponto ou seja, no(s) Armário(s) de Telecomunicações ou Sala de Equipamentos. O aterramento deverá atender aos requisitos da norma TIA/EIA 607 (*Commercial Building Grounding and Bonding Requirements for Telecommunications*) Caso seja opção da unidade, após a instalação, executar um acabamento alternativo com pintura em esmalte sintético ou similar, recomenda-se utilizar a cor cinza-escuro.

7.2.5. Estrutura mínima exigida para as LANs na Waytec

Como resumo dos padrões anteriores, foi resimudo os componentes mínimos necessários em qualquer rede local na Waytec.

- Método de acesso CSMA/CD, rede local IEEE 802.3 (*Ethernet*) e suas variações de alta velocidade;
- topologia da rede física em estrela hierárquica com um nível;
- rede física com estruturação TIA/EIA 568-A em par-trançado, 4 pares 100 ohms;
- utilização de painéis de conexão, cabos, tomadas RJ45 e outros componentes de cabeamento compatíveis com TIA/EIA 568-A cat 5e *Power Sum NEXT*,
- codificação de pinagem em conformidade com T568-A;
- infra-estrutura exclusiva para encaminhamento e proteção de cabos;
- utilização de gabinetes, *racks* e *brackets* para a instalação dos componentes;
- testes de certificação e desempenho da rede física obrigatórios;

- documentação da rede lógica e física (*as-Built*) obrigatório;
- projeto lógico e físico levando em conta flexibilidade de crescimento e de alterações, utilízando-se para dimensionamento a regra básica de 2 pontos por 10 m2 de Área de Trabalho;
- utilização de equipamentos empilháveis.

7.2.6. Identificação dos componentes de uma rede local

A identificação dos componentes de uma rede local na Waytec é obrigatória para os componentes passivos e recomendada para os ativos. A seguir, é descrito o padrão de identificação obrigatório, em concordância com a norma TIA/EIA 606. Esta identificação é válida para qualquer componente do sistema, independente do meio físico.

A identificação sempre conterá no máximo nove caracteres alfa-numéricos. Esses nove caracteres são divididos em sub-grupos que variam de acordo com as funções propostas. As etiquetas de identificação a serem instaladas junto aos componentes deverão ser legíveis (executadas em impressora), duradouras (não descolar ou desprender facilmente) e práticas (facilitar a manutenção).

Identificação dos Armários de Telecomunicações :

Cada Armário de Telecomunicações é identificado por um sub-grupo de três caracteres que indicam a localidade, onde os dois primeiros caracteres informam o nível topográfico (ou andar) e o terceiro (uma letra), um determinado armário naquele andar.

Observação: Antes de iniciar a identificação dos pontos, ou durante o projeto, verificou-se cuidadosamente a instalação predial em vista de localizar o pavimento de menor cota topográfica (nível de referência). Esse local ainda que não venha a

ser contemplado com ponto de um sistema de cabeamento estruturado deverá ser identificado como sendo o nível de referência, cabendo ao mesmo, se necessário, a identificação com o dígito "00".

Exemplo: 01B-XX-XX = Armário de Telecomunicações "B" do 1º andar.

Identificação de painel de conexão em Armário de Telecomunicações :

Em cada Armário de Telecomunicações de um andar haverá, no mínimo, um painel de conexão com 24 posições (número de portas de referência). A identificação desse painel será composta por dois dígitos numéricos que o localizam no sentido de cima para baixo no gabinete, *rack* ou *bracket*.

Exemplo: 01B-02-XX = segundo painel de conexão do Armário de Telecomunicações "B" do 1º andar.

Identificação do Ponto de Telecomunicações (tomada RJ45 na Área de Trabalho) :

Um ponto de telecomunicação em uma Área de Trabalho sempre é terminado em um painel de conexão instalado em um Armário de Telecomunicações. Esse painel, independente do número de tomadas RJ45 existente (24, 48 ou 72), será sempre referendado como agrupamento de 24 conectores RJ45. Assim, a identificação do ponto será correspondente à posição do cabo UTP em uma das vinte e quatro posições existentes em um painel.

Exemplo: 01B-02-23 = posição número 23 do painel de conexão número dois no Armário de Telecomunicações "B" do 1º andar. Dessa forma, no espelho da caixa de superfície na Área de Trabalho, junto à tomada RJ45 correspondente, deverá ser instalada a etiqueta com a identificação do ponto como sendo 01B-02-23.

Identificação do Ponto de Telecomunicações em painel de conexão :

O painel de conexão no armário deverá possuir identificação nas tomadas RJ45 de forma a garantir a identificação do outro extremo do cabo. Existem duas situações possíveis: cabos pertencentes ao sistema de cabeamento tronco ou cabos do sistema horizontal. Para cabos pertencentes ao cabeamento tronco, terminados em outro painel de conexão, é obrigatória a identificação, que será semelhante à utilizada no caso de um ponto de telecomunicação ou seja, localização do armário, painel e posição da tomada.

Exemplo: 00A-05-01 = posição número 01 do painel de conexão número cinco no Armário de Telecomunicações "A" do pavimento térreo. Para cabos pertencentes ao sistema de cabeamento horizontal, isto é, oriundos de Áreas de Trabalho, a identificação é recomendada, mas é necessário que a edificação possua implantado um sistema de identificação de toda as áreas, que seja conhecido e confiável (por exemplo, número de sala, numeração sequencial, etc.), de forma que cada local possa ser identificado de forma inequívoca e precisa. Caso isso aconteça, a identificação na tomada RJ45 do painel será composta por um código de nove caracteres alfanuméricos, dividido em três partes:

- os seis primeiros caracteres alfanuméricos indicam o andar/sala ou número sequencial da área onde está o espelho com a(s) tomada(s) RJ45, conforme sistema próprio de identificação da edificação;
- a segunda, com dois dígitos, indica o espelho;
- a terceira e última, com um dígito, indica a posição da tomada RJ45 no espelho.

Exemplo: 02C401-05-1 = primeira posição da tomada RJ45 do espelho 05 na sala C401 no 2º andar.

Observações:

1. Em um espelho com mais de uma tomada RJ45 deve-se padronizar a identificação das tomadas RJ45. Como sugestão, considerar a primeira tomada como sendo a posição superior esquerda e na sequência, executar um movimento esquerda-direita e de cima para baixo para a numeração sequencial das demais.

2. Se houver mais de uma caixa de superfície (ou espelho) instalada na mesma área deve-se identificá-la no canto esquerdo superior com o número sequencial apropriado;

3. Obrigatoriamente, as caixas com tomadas múltiplas (cabeamento por zona - TSB- 75) deverão ser identificadas junto aos painéis de conexão aos quais estão ligadas. Nesses casos, se o local não possuir identificação, sugere-se incluir as iniciais "MTO" (*Multi-user Telecommunication Outlet*) no local da sala. Exemplo: 02MTO-05-01 indicando primeira posição da tomada múltipla 05 do segundo andar.

8. INSTALAÇÃO DOS SERVIDORES

8.1. Instalação do Servidor SAMBA

Foi instalado o LINUX Conectiva 7.0. A versão do Samba é a 2.2.3a.

Após instalação do LINUX, baixar a versão do Samba, pela página[18] ou pegar os RPM´s do CD de instalação do LINUX para eliminar as dependências do Samba e instalar na seguinte ordem (lembrando que o símbolo "**#**" é o *shell prompt* e não deve ser digitado, e usuário iniciado como *root*):

 # rpm -ivh setup-2.0.2-28cl.noarch.rpm

 # rpm -ivh setuptool-1.5-5cl.i386.rpm

 # rpm -ivh glibc-2.2.3-18cl.i386.rpm (observe que para este pacote todos os rpm´s glib devem ser instalados, para complementar as extenstes do arquivo).

 # rpm -ivh ipxutils-2.2.0.16.a-6cl.i386.rpm

 # rpm -ivh ncpfs-2.2.0.16.a-6cl.i386 - -nodeps (deve-se utilizar o comando "nodeps", pois o arquivo anexado libncp.so.2.3 já é nativo do Conectiva e o instalador considera como não instalado).

 # rpm -ivh mt-st-0.6-1cl.i386.rpm

Depois de instalar os pacotes, instalar o Samba também via rpm:

 # rpm -ivh samba-2.2.3a-xxxxxxx.rpm - -nodeps (deve-se utilizar novamente o comando "nodeps", pois a biblioteca do samba readline.so.3 está instalada).

Editar o arquivo de configuração do samba: */etc/samba/smb.conf*

Segue um modelo deste arquivo (Modelo o qual está sendo usado na Empresa):

```
# Samba config file created using Linux CL7
# from UNKNOWN (192.168.0.44)
# Date: 2002/01/03 16:05:06
[global]
      workgroup = intranet
      netbios name = waytecnet-srv
      server string = Servidor de Arquivos
      encrypt passwords = Yes
      map to quest = Bad Password
      passwd program = /usr/bin/passawd
      unix password sync = Yes
      max log size = 50
      time server = Yes
      encrypt passwords = Yes
      log file = /var/log/samba/log.%m
      max log size = 50
      socket options = TCP_NODELAY SO_RCVBUF=8192 SO_SNDBUF=8192
      preferred master = Yes
      domain master = Yes
      local master = Yes
      security = user
      domain logons = Yes
      logon path = \\%N\profiles\%u
      wins support = Yes
      hosts allow = 192.168.0.
      admin users = anderson lilian admin
      SO_SNDBUF=8192
      Logon script = netlogon.bat
      domain logons = Yes
      os level = 64
      dns proxy = No

[trab]
      comment = Area de trabalho da empresa
      path = /trab
      browseable = Yes
      read only = No
      writable = Yes
      guest ok = Yes
      force create mode = 0770
      force directory mode = 0770

[publico]
      comment = Area de Troca de arquivos
      path = /publico
      browseable = Yes
      read only = No
      guest ok = Yes
      force create mode = 0777
      force directory mode = 0777

[sgq]
      comment = Area do sistema de garantia da qualidade
      path = /sgq
      browseable = Yes
      writable = Yes
      read only = No
      guest ok = Yes
```

```
[netlogon]
      commet = NetLogon SHARE
      path = /home/samba/netlogon
      guest account =

[samba]
      commet = login tracking share
      path = /home/samba/samba
      root preexec = /usr/local/bin/netlogon.sh %u
      root postexec = /usr/local/bin/netlogoff.sh %u
```

Precisa configurar o SAMBA como controlador de domínio primário. Por isso utiliza a função *domain máster* na seção *global*.

A rede é dividida em 3 áreas:

- Uma pública para troca de arquivos entre as áreas da Empresa, qualquer usuário pode ter acesso a qualquer arquivo ou pasta e poder controlar;
- Uma área de trabalho, essa sim já é privada para cada setor sendo que um usuário só terá acesso a área do seu setor e terá uma área de uso particular.
- Uma área para armazenar os documentos da qualidade, que é de responsabilidade da área competente com acesso somente ao pessoal da qualidade.

O compartilhamento *netlogon* é o lugar onde as estações *Windows* pegam o *script* executado quando o usuário faz *logon* no controlador de domínio. Precisa deste compartilhamento para colocar um *script* de *logon* que vai dizer à estação *Windows* para montar um outro compartilhamento, que vai ser usado para rastrear o endereço IP do usuário.

Observe a linha:

```
logon script = netlogon.bat
```

Esta linha dirá à estação *Windows* para pegar e executar o *script* chamado netlogon.bat. Este script deve ser colocado no compartilhamento *netlogon*. Precisa, então, de um *script* chamado netlogon.bat para as estações *Windows*. Foi usado o exemplo listado a seguir e gravado com o nome de NETLOGON.BAT no diretório do compatilhamento netlogon, neste caso, em */home/samba/netlogon/netologon.bat*.

```
REM NETLOGON.BAT
net use p: \\waytecnet-srv\publico /persistent:yes
net use g: \\waytecnet-srv\trab /persistent:yes
net use q: \\waytecnet-srv\sgq /persistent:yes
net use z: \\waytecnet-srv\samba /yes
```

Este *script* vai dizer à estação *Windows* para montar especificamente o compartilhamento SAMBA, e assim, pode-se rastrear o usuário e o endereço IP de sua estação através da saída do programa *smbstatus*, que faz parte do pacote do SAMBA.

Precisa de um compartilhamento de rastreamento que, neste exemplo, foi chamado de samba:

```
[samba]
      comment = login tracking share
      path = /home/samba/samba
      root preexec = /usr/local/bin/net/netlogon.sh %u
      root postexec = /usr/local/bin/net/netlogoff.sh %u
```

Note as linhas *root preexec* e *root postexec*. Elas dizem ao *daemon* do SAMBA para executar os *scripts* indicados quando um usuário monta e desmonta o com-

partilhamento. Neste caso, estamos passando o nome do usuário como parâmetro para o script executado.

Os *scripts* netlogon.sh e netlogoff.sh listados a seguir:

```
# !/bin/sh
# netlogon.sh
# usage:
# netlogon.sh <username>
smbstatus  | grep $1  | grep samba  | gawk `// {print sbstr ($6,2, length
($6) – 2)}´  >  /var/run/smbgate/$1
IPTABLES= ´/usr/sbin/iptables´
EXTIPF=´ppp0´
COMMAND= ´-A´
ADDRESS= `cat /var/run/smbgate/$1`

/etc/smbgate/users/$1  $COMMAND  $ADDRESS  $EXTIF
```

Este *script* (netlogon.sh) será executado quando o usuário fizer *login* e filtrará a saída do programa smbstatus, extraindo o endereço IP do usuário, que será escrito em um arquivo no diretório /var/run/smbgate. O arquivo levará o nome do usuário e será usado depois, quando o usuário dizer *logoff*. O endereço IP extraído será passado como argumento para o script no diretório */etc/smbgate/users* com o nome do usuário, e este *script* vai finalmente atualizar o firewall.

```
$ !/bin/sh
# netlogoff.sh

# usage:
# netlogoff.sh  <username>

IPTABLES= ´/usr/sbin/iptables´
EXTIF=´ppp0
´COMMAND= ´-D´
ADDRESS= `cat /var/run/smbgate/$1`

/etc/smbgate/users/$1 $COMMAND  $ADDRESS  $EXTIF
rm –f /var/run/smbgate/$1
```

Este *script* (netlogon.sh) será executado quando o usuário fizer *logoff* e pegará o endereço do arquivo */var/run/smbgate/user*. Este endereço será passado como

argumento para o script */etc/smbgate/users/user*, que atualizará o *firewall*, restaurando as regras para a forma de repouso.

O que segue é um script de exemplo para /etc/smbgate/user/user:

```
# !/bin/sh
COMMAND=$1
ADDRESS=$2
EXTIF=$3
IPTABLES= ´/usr/sbin/iptables´
$IPTABLES $COMMAND POSTROUTING –t nat –s $ADDRRESS
-o
$EXTIF – j MASQUERADE
```

Para iniciar o samba deve-se usar o comando no *prompt*:

cds

smb start

8.2. Instalação do Servidor DNS

Os servidores de DNS da Waytec estão rodando sobre sistema operacional LINUX Conectiva 6 e com o software BIND 8.

A seguir estão listados os passos para configurar um novo domínio nos servidores de DNS.

VATAPA – 200.223.147.98

No final do arquivo no /ETC/NAMED.CONF é adicionado as seguintes linhas:

****

zone "nome_do_dominio.com.br"{
type máster;
file "nome_do_dominio.com.br";

****

2 – No final do arquivo /ETC/RESOLV.CONF é adicionado as linhas:

domain nome_do_dominio.com.br
nameserver endereço_ip_do_servidor_para_onde_o_dominio_aponta

3 – É criado um arquivo novo no diretório /VAR/NAMED/nome_do_dominio.com.br, com as seguintes configurações:

****

@ IN SOA vatapa.waytecnet.com.br. hostmas-ter.vatapa.waytecnet.com.br. (
2000092501 ; serial – **Esse número sempre é diferente, e tem que ser igual no serv. secundário**
3600 ; refresh
900 ; retry
1209600 ; expire
43200 ; default_ttl
)
@ IN MX 5 acaraje.waytecnet.com.br.
@ IN NS vatapa.waytecnet.com.br.
@ IN A 200.223.147.98
mail IN A 200.223.147.98 – IP DE MAIL.DOMINIO.COM.BR
webserver IN A 200.223.147.99 –I P DE WEBSERVER.DOMINIO.COM.BR
mail IN CNAME mail.dominio.com.br.
www IN CNAME webserver.dominio.com.br.

Esse arquivo informa ao DNS qual o servidor WEB e o servidor de MAIL do domínio a ser registrado.

ACARAJE – 200.223.147.99

1 – No arquivo /ETC/NAMED.CONF é adicionado as seguintes linhas:

****

zone "nome_do_dominio.com.br" {
type slave;
file "sec/nome_do_dominio.com.br";
masters{
200.223.147.99;
200.223.147.98;
};
};
****

2 – No arquivo /ETC/RESOLV.CONF

domain nome_do_dominio
nameserver endereço_ip_do_servidor_para_onde_o_dominio_aponta

3 – Faça um novo arquivo no diretório /VAR/NAMED/SEC/nome_do_dominio.com.br com as seguintes configurações:

```
**********************************************************************
; BIND version named 8.1.2 sáb abr 17 22:07:24 EST 1999
; BIND version root@buildmaster.conectiva.com.br:/usr/src/rpm/BUILD/src/bin/named
; zone 'waytecnet.com.br'   last serial 0
; from 200.223.147.98   at Sat Jan 29 15:56:17 2000
$ORIGIN com.br.
domínio         IN      SOA     vatapa.waytecnet.com.br.  hostmaster.vatapa.waytecnet.com.br.
(
                2000092501* 3600 900 1209600 43200 )
                IN      NS      vatapa.waytecnet.com.br.
                IN      MX      5 acaraje.waytecnet.com.br.
                IN      A       200.223.147.98
$ORIGIN domínio.com.br.
mail            IN      CNAME       mai1l.dominio.com.br.
www             IN      CNAME       webserver.dominio.com.br.
mail1           IN      A       200.223.147.101     – IP DE MAIL.DOMINIO.COM.BR
webserver       IN      A       200.223.147.101 – IP DE WEBSERVER.DOMINIO.COM.BR
**********************************************************************
```

obs * - Numero serial idêntico ao do servidor MASTER

Para que o servidor de nomes funcione com as novas configurações, e necessário parar o serviço NAMED e reiniciá-lo. Deve digitar as seguintes linhas de comando:

```
# cds
#./named stop
#./named start
```

Para verificar se esta tudo funcionando perfeitamente, é ir no site da registro BR [19] na área de pesquisa, tem onde digita o domínio e os endereços IP´s para verificar se esta tudo funcionando.

8.3. Instalação do Servidor de Correio

Para a instalação do servidor de correio foi utilizado o Postfix por sua facilidade de uso e configuração, além que é nativamente protegido contra relay, ao contrário do Sendmail. O Postfix é um *software* que permite o envio e o recebimento de mensagens. Quando utilizado com o Amavis ele faz verificação das mensagens por vírus ou *scripts* maliciosos. O antivírus escolhido foi o HBEDV, que necessita de uma chave de cadastro na página do desenvolvedor para sua utilização [20]. Após o registro recebe por e-mail um arquivo chamado *hbedv.key*. Que deve ser guardado para ser usado posteriormente..

Para a instalação deve seguir os seguintes passos:

1) Primeiro instalar o Conectiva Linux 8+. Após a instalação verificar se os pacotes do postfix estão instalados na máquina (o símbolo “#” significa o *prompt* de comando):

```
# rpm -qa | grep postfix
```

O resultado na tela deve ser:

```
postfix-doc-20010228pl02-6cl
linuxconf-postfixconf-1.25r3-24cl
postfix-20010228pl02-6cl
```

Caso não apareça alguns dos pacotes instale a partir do CD de instalação do Conectiva.

Antes de fazer o processo de configuração aconselha-se a fazer uma cópia do arquivo original onde serão feitas as alterações, pois se for necessário refazer a configuração desde o início por algum motivo de erro não será preciso reinstalar os pacotes. Para fazer a cópia do arquivo original utilize o comando:

```
# cp /etc/postfix/main.cf /tmp
```

Agora é preciso instalar as dependências necessárias para os arquivos do Antivírus e do Amavis. Pegar os RPM´s do CD de instalação do Linux e instalar: setup, setuptool, flex, bison, m4, autoconf, automake, freetype2-devel, make, libtool, kernel-headers, kernel-source, binutils, gcc, g++, byacc, ncurses4, ncurses-devel, libstdc++-devel, glib-devel, glibc, glibc-devel, libsigc++-devel, nasm, arc, lha, tnef, unarj, unrar, zôo, wget.

Agora deve ser instalado o perl, que pode ser baixado no na página do desenvolvedor [21], a versão utilizada foi o perl 5.8.0:

Deve-se logar no sistema como *root* e instalar o perl:

Entrar diretório /usr/local e descompactar o arquivo stable.tar.gz com o comando:

```
# cd /usr/local
# tar zxpvf <local_aonde_esta_o_arquivo>/stable.tar.gz
# cd perl5.8.0/
# rm -f config.sh Policy.sh
# sh Configure -de
# make
# make test
# make install
```

Depois ir no shell e digitar:

```
# perl -MCPAN -e shell
```

Se for a primeira vez que utiliza a *shell* CPAN, irão ser feitas algumas perguntas, deve se escolher os valores padrões (as que estão entre parênteses).

Depois disso irá aparecer o *shell* de *login* do CPAN:

```
cpan>
```

Agora deve-se instalar os módulos que não foram encontrado no sistema, nesse exemplo vai ser demonstrada a instalação de todos os módulos:

```
cpan> install Unix::Syslog
cpan> install Convert::UUlib
cpan> install Convert::TNEF
cpan> install Compress::Zlib
cpan> install Archive::Tar
cpan> install Archive::Zip
cpan> install G/GB/GBARR/MailTools-1.15.tar.gz
cpan> install MIME::Tools
cpan> install Bundle::libnet
```

Depois da instalação dos compactadores, módulos perl e outras dependências necessárias para o Sistema, o próximo passo é instalar o antivírus. Antes de fazer isso deve-se criar o usuário “vscan”:

```
# adduser vscan
```

Descompacte o arquivo avlxsrv.tgz no diretório /usr/local com o comando:

```
# tar zxpvf avlxsrv.tgz
```

No diretório antivir-2.0.5-server executar:

./install

Durante a instalação será necessário responder algumas perguntas, como por exemplo, se deseja que seja feita automaticamente o *update* e coisas do tipo, se usa proxy, foi deixado por conta do crontab acrescentando as seguintes linhas:

45 6 * * * /usr/lib/AntiVir/antivir - -update -q

Isso diz que todos os dias durante todos os meses, às 6:45h vai ser executado o comando antivir –update –q. Após a instalação terá que copiar o arquivo *hbedv.key* para o diretório /usr/lib/AntiVir/,depois executar o arquivo de update da BIOS de anti-vírus:

antivir –update

8.3.1. Configurando o postfix - editando os arquivos:

Abrir o arquivo de configuração digitando:

vi /etc/postfix/main.cf

Os primeiros passos é para configurar o básico do Postfix para que o mesmo inicie:

Na linha “myhostname”, substituir pelo nome da máquina/host:

myhostname=abara.waytec.com.br

Na linha “mydomain”, substituir pelo seu domínio:

mydomain=waytec.com.br

Esta opção especifica como ficará a terminação do e-mail após o símbolo de @, basta descomentar a opção:

myorigin=$mydomain

Agora deve cadastrar o domínio que o servidor de e-mail é responsável:

mydestination = $myhostname, localhost.$mydomain, $mydomain, waytec.com.br

Informar qual a rede que o postfix irá trabalhar:

mynetworks_style = subnet

mynetworks = 192.168.0.0/24, 127.0.0.0/8

Configurar qual o pacote que usará para transportar as mensagens:

default_transport = smtp

Pelo padrão o postfix permite *relay* apenas de e-mails das redes informadas na opção "*mynetworks*", e de seus domínios. Para habilitar para outros IP´s acrescente as seguintes linhas ao *main.cf*:

smtpd_recipient_restrictions = permit_mynetworks
check_client_access hash:/etc/postfix/client_access
check_relay_domains

Salva e sai do arquivo e cria o arquivo /etc/postfix/client_access

```
# touch /etc/postfix/client_access
# vi /etc/postfix/client_access
```

E adicionar as seguintes linhas ao arquivo criado:

```
200.223.147.101 OK
200.223.147.99 OK
200.221.94.108 OK
200.207.152.10 OK
192.168.0.1 OK
192.168.0.2 OK
192.168.0.3 OK
192.168.0.4 OK
192.168.0.5 OK
192.168.0.6 OK
```

Observar que o exemplo do arquivo descrito acima deve seguir até a última máquina que pode ser disponível na rede – neste caso 192.168.0.254. Salvar o arquivo e gerar o mapas de clientes com o comando:

```
# postmap /etc/postfix/client_access
```

Para bloquear apenas uma máquina da rede retirar a opção “*permit_mynetworks*”, deixando a linha assim:

smtpd_recipient_restrictions = check_client_access hash:/etc/postfix/client_access
check_relay_domains

E colocar no arquivo /etc/postfix/client_access as máquinas que desejam bloquear com o sufixo REJECT ao invés de OK.

Agora abrir o arquivo *main.cf* e colocar a seguinte linha

relay_domain = $mydestination, /etc/postfix/relay-domains

Salvar o arquivo, sair e criar o arquivo “*relay-domains*” com a seguinte estrutura:

 # touch /etc/postfix/relay-domains
 # vi /etc/postfix/relay-domains

192.168.0
200.207.152.10
200.223.147.101
200.221.94.108
200.171.54.119
200.223.147.99

Para a criação de *aliases* descomente as seguintes linhas do *main.cf*:

alias_maps=hash:/etc/aliases
alias_database=hash:/etc/postfix/aliases

Salvar o arquivo e criar o mapa de *aliases* com o comando no *prompt*:

 # postalias /etc/aliases

Os *aliases* servem para que um usuário receba e-mail´s com outro nome, para administração do servidor. O arquivo /etc/aliases terá como exemplo a seguinte estrutura:

admin: anderson, admin

Esse exemplo mostra ao postfix que ao receber uma mensagem para admin deverá enviar o mesmo para anderson@waytec.com.br e uma cópia para ele mesmo. Depois de criar o *aliases* deve-se sempre executar o comando para criação de mapas de *aliases*:

Para configurar domínios virtuais no sistema, editar o arquivo "/etc/postfix/virtual" e colocar os domínios virtuais do sistema da seguinte forma:

targetair.com.br Dominio Target

anderson@targetair.com.br ricardo

A estrutura do arquivo é bem simples, a primeira linha informa que o nome do domínio virtual seguido de um comentário qualquer. A segunda linha informa o e-mail do usuário virtual seguido do usuário real.

Para que os domínios virtuais funcionem deve ser colocada a seguinte linha ao *main.cf*:

virtual_maps = hash:/etc/postfix/virtual

Salvar o arquivo e criar o mapa de domínios com o comando:

```
# postmap /etc/postfix/virtual
```

8.3.2. Configurando o POP-3:

Descomente a seguinte linha do arquivo “/etc/inetd.conf”:

```
pop-3          stream tcp     nowait          root     /usr/sbin/tcpd          ipop3d
```

Salve o arquivo e reinicie o serviço do inetd:

```
# cds
# ./inet stop
# ./inet start
```

Após as configurações feitas inicie o servidor postfix:

```
# cds
# ./posfix start
```

O próximo passo é a configuração do Amavis, baixe o pacote no endereço do desenvolvedor e descompactar o pacote no diretório “/usr/local/” com o comando:

```
# cd /usr/local
# tar zxpvf amavis-perl-11.tar.gz
```

Agora configure o sistema:

 # cd amavis-perl-11
 # ./configure - -enable-postfix - -enable-smtp
 # make
 # make install

Troque as permissões do diretório “/var/amavis” para 777 e coloque no final do “/etc/postfix/main.cf” as seguintes linhas:

content_filter = vscan:
soft_bounce = yes

e no “/etc/postfix/master.cf”:

vscan unix

8.4. Instalação do Servidor Proxy

O Servidor Proxy é um software que permite o acesso a diversos serviços da Internet por uma rede interna, fazendo a interface entre esta e o mundo externo. Permite também implementar controles e estatísticas sobre os acessos.

Para instalar o Squid 2.4.7 que vem no CD 1 do Conectiva Linux 8+:

a) Na máquina onde será instalado o Squid, monte o CD 1 do Conectiva Linux 8+ com o seguinte comando no shell:

 # mount -t iso9660 /dev/cdrom /mnt/cdrom

b) Mova-se para o diretório /mnt/cdrom/conectiva/RPMS. Digite o comando :

 # rpm -ivh squid-2.4.7-1U8_3cl.rpm

b) Certifique-se que após a instalação deverão estar criados os seguintes arquivos e diretórios :

- diretório /etc/squid/ : diretório de configuraçno do squid ;

- arquivo /etc/squid/squid.conf : arquivo mestre de configuraçno do proxy.

A seguir são listados os arquivos pertencentes a configuração do Squid e seus conteúdos. Poucas mudanças são necessárias a configuração original, por isso, dentro dos arquivos, as linhas que devem ser mudadas são:

Arquivo /etc/squid/squid.conf:

```
http_port 3128
cache_mem 8 MB
cache_log /var/log/squid/cache.log
cache_store_log /var/log/squid/store.log
dns_nameservers 200.223.147.98 200.254.67.2 200.19.119.9 200.255.253.234
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 SSL_ports port 443 563
acl Safe_ports port 80        # http
acl Safe_ports port 21        # ftp
acl Safe_ports port 443 563   # https, snews
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 CONNECT method CONNECT
http_access allow manager localhost
http_access deny manager
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
http_access allow all
icp_access allow all
miss_access allow all
```

A inicialização do Squid se dá automaticamente na inicialização do Linux. Para verificar seu funcionamento, basta configurar uma estação cliente para acessá-lo da seguinte maneira :

a) Na configuração de Proxy do navegador, coloque o endereço IP do servidor Proxy e a porta configurada para acesso HTTP do squid (3128 é a padrão) para os seguintes serviços : HTTP, SECURITY, FTP e SOCKS

Para criar regras de acesso a WEB no squid e bastante simples, basta apenas criar 2 arquivos no diretório /etc/squid, chamados PALAVRAS (palavras que serão bloqueadas na web) e SITES (sites que serão bloqueados). Depois vá no arquivo de configuração do squid (/etc/squid/squid.conf) e na seção ACL acrescente as seguintes variáveis:

```
acl way_tec src 192.168.0.0 - (ip da rede interna waytec)
acl ip_net src 192.168.0.19 192.168.0.58 - (coloque aqui os ips que nao es-
tarao restritos)
acl ip_net 1 src 192.168.0.89 (coloque aqui os ips barrados sem precisar
que o squid localize)
acl sites_net dstdomain “/etc/squid/sites”
acl proibir_palavras url_regex “/etc/squid/palavras”
```

Logo abaixo no arquivo squid.conf, localiza a linha:

```
http_access allow all
```

Comente essa linha e adicione as seguintes linhas:

```
http_access deny sites_net ip_net1
http_access allow p_net
http_access deny proibir_palavras
http_access allow way_tec
```

Parar o squid e iniciar novamente:

#cds
#squid reload

9. CONCLUSÃO

Observou se ao final do período de estágio que algumas tarefas em andamento, necessárias para manter um servidor operacional e confiável. O estágio focalizou a construção de servidores LINUX, desde o início de avaliação do ambiente, mostrando as possíveis falhas de segurança, até o ponto de se poder utilizá-lo por um ambiente operacional 24 horas. Selecionamos o *hardware* e o *software*, instalamos o sistema, configuramos e observamos a viabilidade do projeto. Além disso, viu como um sistema LINUX oferece serviços de configuração e disponibiliza compartilhamento de arquivos, compatíveis com todos os tipos de clientes fazendo o sistema operacional possam ser integrado à rede dos departamentos para, desta maneira, simplificar a manutenção.

Quando um projeto deste tipo é configurado e mantido pos um profissional capacitado de informática, torna-se uma excelente plataforma para servidor de Internet ou para departamento.

Grande parte da discussão a respeito do software livre no ambiente corporativo tem sido ao redor das vantagens (ou desvantagens) de custo, argumentação estimulada pelos fabricantes de softwares proprietários – a quem interessa que os custos sejam analisados isoladamente, fora de um contexto estratégico.

Observou também que o LINUX oferece maior segurança, pois permite que problemas sejam encontrados e resolvidos, ao invés de permanecerem ocultos durante anos.

De todas as vantagens estratégicas, porém, o fator mais importante que deveria estimular a adoção de soluções baseadas em padrões abertos e LINUX é a redução do poder do fornecedor.

Do ponto de vista legal, o LINUX oferece grandes vantagens sobre o modelo proprietário, a começar pelo fato que suas licenças não procuram restringir os direitos do usuário, e permitem que inúmeros fornecedores prestem serviços comerciais com base no software.

A conclusão mais importante que este trabalho apresenta é que, mais do que simplesmente uma questão de custo, as empresas devem começar a analisar o impacto estratégico da escolha de um fornecedor ou de uma solução, e que muitas vezes, o LINUX é a melhor opção.

10. BIBLIOGRAFIA

[1] TANEMBAUM, A. S., **Sistemas Operacionais Modernos**, Prentice-Hall do Brasil LTDA, 1995.

[2] [SOARES 95] Soares, L. F. G., Lemos, G., Colcher, S., **Redes de Computadores: das LAN's, MAN's e WAN's às redes ATM**, 2a edição, Campus, 1995.

[3] Nemeth, E., Snyder, S., Hein, T. R., ***UNIX System Administration Handbook***, 2nd ed., Prentice Hall, 1989.

[4] Chapman, D. B., Zwick, E. D., ***Building Internet Firewalls***, O'Reilly & Associates, Inc., 1995.

[5] Garfinkel, S., Spafford, G., ***Pratical UNIX Security***, O'Reilly & Associates, Inc., 1994.

[6] Stern, H., ***Managing NFS and NIS***, O'Reilly & Associates, Inc., 1991

[7] Hunt, C., ***TCP/IP Network Administration***,. O'Reilly & Associates, Inc., 1991.

[8] RNP Publicações. **LAN – Uma Introdução Completa a Redes Locais**, Parte 6 – p. 297-309.

[9] Steele, G. L., Jr., D. R. Woods, Finkel, R. A., Crispin M. R., Stallman R. M., Goodfellow G. S., ***The Hacker's Dictionary***. New York: Harper and Row, 1988.

[10] Ritchie, D. M., ***On the Security of UNIX****,* 1975.

[11] Curry, D. A., ***Improving The Security of Your UNIX System.***, Systems Programmer. Information and Telecommunications Sciences and Techno*log*y Division.

[12] Champine, G. A., ***Distributed Computer Systems: Impact on management, design and analysis***. North-Holland Publishing Company, 1980.

[13] Semeria, C. ***Internet Firewalls and Security: A technology overview***. Published by 3COM Corporation, 1997.

[14] Cobb, S., ***Internet Firewalls***. Published by National Computer Security Association - NCSA, 1997.

[15] Steve, M.; Slater D. E., ***Issues in Firewall Security***. Published by Netsight Consulting, 1996.

[16] <http://sunsite.unc.edu/mdw/HOWTO/Hardware-HOWTO.html> acesso em 20 de junho de 2001.

[17] <http://www.conectiva.com.br/hardaware> acesso em 20 de junho de 2001.

[18] <http://www.samba.org> acesso em 18 de agosto de 2001

[19] <http://registro.br> acesso em 15 de agosto de 2002

[20] <http://www.antivir.de> acesso em 15 de agosto de 2002

[21] <http://www.perl.org> acesso em 23 de dezembro de 2001