ROSSINI TEIXEIRA DE MORAES

UM ESTUDO SOBRE FERRAMENTAS DE SEGURANÇA PARA SERVIDORES LINUX EM INTRANETS

Trabalho de conclusão de curso apresentado ao Curso de Ciência da Computação.

UNIVERSIDADE PRESIDENTE ANTÔNIO CARLOS

Orientador: Prof.. Elio Lovisi Filho
Co-orientador: Prof. Emerson Rodrigo Alves Tavares

BARBACENA
2003

ROSSINI TEIXEIRA DE MORAES

UM ESTUDO SOBRE FERRAMENTAS DE SEGURANÇA PARA SERVIDORES LINUX EM INTRANETS

Este trabalho de conclusão de curso foi julgado adequado à obtenção do grau de Licenciado em Ciência da Computação e aprovado em sua forma final pelo Curso de Ciência da Computação da Universidade Presidente Antônio Carlos.

Barbacena – MG, 25 de junho de 2003

_____

Prof. Elio Lovisi Filho - Orientador do Trabalho

_____

Prof. Emerson Rodrigo Alves Tavares - Membro da Banca Examinadora

_____

Prof. Eduardo Macedo Bhering - Membro da Banca Examinadora

AGRADECIMENTOS

Agradeço primeiramente ao meu Orientador que com sua competência me ajudou a concluir este importante trabalho. Agradeço a Deus e aos meus pais, por me apoiarem e por me darem forças para chegar onde cheguei. Agradeço também a todos os colegas que de certa forma ajudaram nesse trabalho.

RESUMO

Este trabalho consiste em mostrar os níveis de segurança que um sistema operacional Linux, para que os seus usuários e administradores não sejam prejudicados se ocorrer alguma falha de segurança. Algumas das técnicas que este trabalho descreve podem servir também como procedimento de segurança para vários sistemas operacionais de mercado, salientando a importância da segurança dos mesmos.

Sabendo que o Linux ao ser instalado, vem com configurações padrão, isto é, sem qualquer atribuição de segurança, ele pode ser considerado, em alguns pontos, vulnerável se for conectado a algum tipo de rede.

Neste trabalho serão descritas as principais ferramentas de segurança que podem amenizar ou sanar algumas vulnerabilidades do sistema para aplicações voltadas a intranet. Além de ferramentas, algumas técnicas de configuração do sistema operacional também serão mostradas com o intuito de auxiliar o administrador na detecção da falhas de segurança.

SUMÁRIO

LISTAS 1

LISTAS 2

1 INTRODUÇÃO

1.1 OBJETIVO

O Linux é um sistema operacional derivado do Unix, desenvolvido por Linus Torvalds, um estudante de Ciência da Computação da Universidade de Helsinki, na Finlândia, onde lançou a primeira versão do Sistema (Versão 1.0) em março de 1992, sendo esse o primeiro lançamento oficial do Sistema.

Embora o que vemos hoje seja um sistema operacional amplo e robusto, desenvolvido com a ajuda de programadores de todo o mundo, na sua versão original o Linux era somente um pequeno sistema operacional. O Linux hoje, tem um grande suporte a *hardware* e seu desempenho é bom, dando a muitos computadores um poder comparável ao de computadores de porte médio, como os de grandes sistemas como SPARC e SUN Microsystems.

Os sistemas da família Unix têm sido considerados os melhores sistemas multitarefa, pois são capazes de executar grandes números de aplicativos simultâneos, tornando-se ideais para servidores e estações de trabalho. [DANESH 2000]

Ainda mais importante que um sistema multitarefa, o Linux é também um sistema operacional multiusuário. Ele permite a vários usuários utilizar simultaneamente os recursos de multitarefa do sistema operacional. A grande vantagem disso é que o Linux pode ser empregado como servidor de aplicativos.

O Linux pode ser usado com praticamente qualquer tipo de aplicativo e entre eles estão aplicativos de processamento de texto, sendo os principais *World Perfect, Star Office e Applixware*, Linguagens de programação, destacando-se os mais importantes compiladores das mais diversas linguagens de programação; vários ambientes *X-Window*, que é a interface

gráfica do sistema; ferramentas para internet, incluindo servidor *Web*, servidor de correio e de *newsgroup*; Banco de Dados, proporcionando uma robusta plataforma para rodar aplicativos de banco de dados cliente-servidor, incluindo suporte e *MySQL, Oracle e Sybase* e os importantes *Softwares* de compatibilidade com DOS e *Windows*, que são eles, *DOSEmu* e *Vmware*.

O *kernel* do Linux é distribuído pela GNU (*General Public License*). Essa Licença, desenvolvida pela *Free Software Foundation*, que promove a distribuição e o desenvolvimento aberto de *software*. Ao contrário de outras licenças, a licença GNU permite que qualquer um possa distribuir o *software*, em outras palavras, qualquer um pode pegar esse *software*, alterá-lo e redistribuí-lo, mas não pode impedir que outros façam o mesmo.

Por ser distribuído pela GNU, é de se esperar que muitos fornecedores diferentes produzam diversas distribuições do Sistema. Essas distribuições são um conjunto de pacotes documentados, desenvolvidos por diferentes empresas que distribuem-nas através de CDs ou gratuitamente em seus sites. Hoje em dia existem várias distribuições que se diferenciam pelos instaladores e pelo números de aplicações. É muito grande o número atual de distribuições, as mais conhecidas são: *Slackware, Red Hat, Open Linux, Debian, SuSe, Corel Linux, Mandrake* e *Conectiva*, sendo essa última, uma distribuição desenvolvida por um empresa brasileira. [DANESH 2000]

1.2 MOTIVAÇÃO

Muitas pessoas não acreditam na segurança do Linux, pelo fato dos programas fonte do sistema estarem disponíveis para que qualquer um possa fazer alterações. Esse tipo de contribuição pode ajudar, apontando algum ponto vulnerável que facilite a invasão do sistema e a ação de intrusos. Então, o que se deve considerar é que em quaisquer contribuições, o sistema será sempre analisado por equipes capacitadas que fazem uma avaliação, dizendo se ela é válida ou não.Os que contribuem com êxito acabam participando de lista de colaboradores no desenvolvimento do sistema operacional Linux.

Quando o sistema operacional Linux é instalado, ele vem com configurações padrão, que por sua vez podem estar desprotegidas contra algum tipo de intruso e sem ferramentas de segurança instaladas, ficando parcialmente vulnerável se conectado a uma rede. Então, para se precaver contra ataques ao sistema, será preciso ter um administrador bem atento às vulnerabilidades do mesmo. È importante que ele esteja sempre buscando configurações e atualizações que corrijam os problemas recentemente descobertos.

Naturalmente os computadores são conectados em rede e acabam correndo o risco de serem invadidos, seja por uma falha de administração do sistema ou por mais uma vulnerabilidade descoberta. Quando isso ocorrer devido a uma falha do administrador, ele terá que analisar onde se encontra a falha utilizando de ferramentas para descobrir tal problema e corrigí-lo(s).

Na ocorrência de um problema recentemente descoberto, os fabricantes de *software* em geral disponibilizam atualizações para corrigí-los de mês em mês, o que é muito tempo, dependendo da gravidade do problema. Para problemas mais graves são disponibilizadas pequenas atualizações semanais para corrigí-los. No caso do Linux, existem equipes de plantão que trabalham vinte e quatro horas nos sete dias da semana, para prestar suporte a descobertas de novas vulnerabilidades que podem vir a ocorrer, podendo resolver estas em poucas horas e divulgando a solução do problema para a comunidade Linux.

Um fator que aumenta a segurança dos sistemas é a criptografia, pois ela evita que alguém fique ouvindo o que se passa em uma rede. Através dela, consegue-se codificar e decodificar informações com chaves públicas fazendo com que apenas o destinatário possa ler a informação enviada no pacote criptografado.

Se tem um sistema no qual se pode realizar um ajuste mais fino em segurança, utilizando de muitas listas de discussões que ajudam em sua administração e dá um suporte mais ágil para resolver problemas, com isso conseguimos com muito empenho uma segurança mais rígida e mais confiável do que se enxerga em outros sistemas existentes no mercado.

De todos os problemas de segurança existentes no Linux, um dos mais importantes atualmente é o fato de a maioria dos administradores não conhecerem detalhes importantes

de sua instalação e configuração, logo vão efetuando de forma padronizada a instalação e configuração de todos os pacotes da distribuição.

Como o Linux está crescendo na área de servidores, muitas empresas estão adotando o Linux, pois, além de ser um sistema de baixo custo e manutenção por ser um sistema "gratuito", ele é um dos sistemas mais poderosos que oferece uma maior segurança se comparado com outros sistemas, desde que bem configurado. [DANESH 2000]

Foi com o intuito de alerta para estas vulnerabilidades que neste trabalho tenta-se implementar um servidor de intranet Linux, buscando de todas as formas possíveis a implementação de um servidor relativamente seguro, longe de falhas e vulnerabilidades, apontando todos os problemas e as respectivas soluções para corrigí-los.

1.3 VISÃO GERAL

Este trabalho apresenta no capítulo 2 uma visão geral de aspectos básicos de segurança em servidores Linux, abrangendo segurança física, administração básica do sistema, gerenciamento de usuários e permissões de arquivos.

No terceiro capítulo, é apresentado de uma forma mais detalhada, técnicas de invasão, abrangendo vulnerabilidades do sistema Linux, *Spoofing*, *Scanners*, *Sniffres*, a segurança no núcleo do sistema Linux (Linux *Kernel*), ataques de Recusa de Serviço, detecção de invasão, uma idéia dos dois tipos de *firewall* e auditoria em *logs*.

No quarto capítulo, é apresentada uma gama de ferramentas para que administradores de sistemas de Intranet possam deixar o servidor Linux relativamente seguro. Além das ferramentas, são mostradas técnicas de configuração do próprio sistema operacional para mantê-lo seguro.

2 ASPÉCTOS BÁSICOS DE SEGURANÇA

2.1 A SEGURANÇA FÍSICA

Um ponto freqüentemente subestimado, quando se trata de segurança, é a maior vulnerabilidade dos servidores ao ataque físico ao que do ataque remoto. Os responsáveis por isso geralmente são:

Usuários maliciosos;

Vândalos;

Ladrões;

Invasores;

É mais provável que o servidor seja atacado por uma pessoa com uma ferramenta qualquer, como um "machado" do que por um utilitário de *spoofing*, mas, quando isso ocorre, os efeitos podem ser desastrosos. Se o seu sistema remoto está sendo invadido por algum *hacker*, é possível sempre reinicializar, reinstalar ou re-configurar o sistema, mas, quando o sistema é danificado ou comprometido fisicamente, aí sim, aparecem os problemas.

[ANÔNIMO 2000]

2.1.1 A LOCALIZAÇÃO FÍSICA DO SERVIDOR E O ACESSO FÍSICO

Suponha-se que foram dados 10 segundos para um usuário malicioso ficar sozinho com o servidor. Apesar de ser pouco tempo, ele poderia gerar algum dano no sistema, realizando ataques de recusa de serviços primitivos como desconectar cabos, desligar o *hardware* de rede ou reinicializando o servidor.

Atos como esses são incomuns em certas instalações. Nesses casos, a preocupação principal deve ser os usuários locais que tem autorização, pelo menos limitada para acessar o sistema. Estima-se que 80% de todas as invasões ocorram por usuários internos. A razão é que esse pessoal está munido de informações privilegiadas que invasores remotos freqüentemente não podem ter. [ANÔNIMO 2000]

2.1.2 CENTRO DE OPERAÇÃO DE REDES (*NETWORK OPERATIONS CENTER* – NOC)

O *Network Operation Center* é uma área restrita que abriga os servidores. Eles geralmente são presos por rebites a um *rack*, ou seguros ao *hardware* essencial de rede.

O local ideal para o NOC deve ser uma sala separada em que poucas pessoas têm acesso. As pessoas autorizadas devem possuir chaves, se possível chaves-cartão que registrem a entrada de usuários, mesmo autorizados, a certas horas do dia. [ANÔNIMO 2000]

Enfim, é de grande necessidade manter um *log* (um tipo de documentação) de acesso escrito e obrigar a todos, até mesmo aos autorizados, a assiná-lo quando entrarem ou saírem.

Além disso, é importante verificar se o NOC possui alguns requisitos como:

- Instalar o NOC dentro de um espaço onde o público não tenha acesso;
- O caminho até o NOC não deve possuir nenhuma parede de vidro;
- As portas que levam a ele devem ser blindadas por completo.

É importante também que diretivas escritas proibindo o acesso ao NOC devem ser colocadas nos contratos com os empregados. Com isso, eles saberão que, se violarem essa diretiva, poderão sofrer alguma penalidade. [ANÔNIMO 2000]

2.2 ADMINISTRAÇÃO BÁSICA DO SISTEMA LINUX

Toda a força da administração do sistema Linux está no usuário principal ou o administrador (raiz) *root*. Conectado como *root*, um administrador tem o total acesso ao sistema podendo controlar usuários, grupos e arquivos individuais.

Ao adicionar no sistema uma conta de usuário, elas são organizadas em grupos de acordo com as respectivas tarefas e necessidades de acesso. Com isso, é atribuído a cada usuário o direito de acesso individual. [DANESH 2000]

2.2.1 CONTAS DE USUÁRIOS

Todo usuário do sistema Linux precisa de uma conta própria. Não é aconselhável que se use o usuário *root* para propósitos pessoais, exceto quando for absolutamente necessário. Há várias razões para isso. [DANESH 2000] :

- Primeiro, com a conta de *root* tem-se o poder absoluto, pode-se mudar permissões de arquivos e restrições de acesso. Essas mudanças no sistema são de extrema importância, pois, se realizá-las indiscriminadamente, pode causar danos irreparáveis no sistema.

- Segundo, pode-se deixar vulnerável o sistema para inúmeras ameaças de segurança. Por exemplo, supondo que um usuário utilize a internet rodando o navegador *Netscape Communicator*. Se o suporte de linguagem estiver ativado em escala completa, o navegador processará um mini-aplicativo de Java, e com esse mini-aplicativo poderá herdar privilégios de acesso e utilizá-los para atacar o sistema.

2.2.2 CRIANDO E GERENCIANDO CONTAS

2.2.2.1 Diretivas de conta

No sentido geral, uma conta consiste em dois elementos:

- Autorização para efetuar *login* (conectar-se);

- Autorização para acessar serviços.

A autorização para efetuar *login* é um privilégio. Esse privilégio nunca deve ser concedido sem que haja um objetivo concreto.

É importante, se possível, fornecer aos usuários serviços críticos sem lhes dar acesso de *Shell* (*prompt* de comandos). Esse acesso ocorre quando usuários fazem um *telnet* remoto para um *Shell* local em um servidor. Isso gera problemas. Quanto mais usuários com acesso de *Shell* houver, mais provavelmente o sistema terá brechas de segurança interna. [CISNEIROS 1998]

Se for necessário conceder o acesso de *Shell* a usuários numa rede Linux, é importante reduzir os riscos observando os seguintes itens:

- Deve-se disponibilizar uma máquina somente para acesso de *Shell*.
- Restringir-se essa máquina somente para utilização do mesmo.
- Manter-se fora dela os serviços não-essenciais de rede.
- Particionar as unidades de disco levando em conta uma futura recuperação de informação. Pois, as máquinas de *Shell* sofrem desastres regularmente.
- Deve-se proibir relacionamentos de confiança entre *Shell* e outras máquinas.
- Deve-se colocar diretórios com arquivos sigilosos (/tmp, /home, /var) em partições separadas e alterar os binários "suid" (*Super User Identification*) para uma partição que o Linux não monte automaticamente, ativando-a como "no setuid".
- Deve-se redirecionar os *logs* direto para o servidor de *logs*.

Se a configuração for apenas em uma máquina, se aplicam as mesmas regras. Deve-se conceder o acesso a *Shell* para os que realmente precisam deles. Com isso, deve-se ser cauteloso ao conceder acesso de *Shell* para uma pessoa estranha, seja uma *hacker* ou um *cracker*. Caso contrário, mudanças podem ser feitas na máquina, ou o IP (*Internet Protocol*) pode ser censurado por algo que o administrador não fez. [CISNEIROS 1998]

2.2.2.2 Estrutura da conta

Uma conta de usuário consiste no seguinte:

- Um nome de usuário e uma senha válida;
- Um diretório inicial (*home*);
- Acesso de *Shell*.

Quando um usuário tenta conectar o Linux verifica se esses itens estão corretos examinando o arquivo “passwd”. Esse arquivo encontra-se o diretório /etc.

Esse arquivo consiste em entradas de contas de usuários. Por exemplo:

```
root:x:0:0::/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/log:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/:
news:x:9:13:news:/usr/lib/news:
uucp:x:10:14:uucp:/var/spool/uucppublic:
operator:x:11:0:operator:/root:/bin/bash
games:x:12:100:games:/usr/games:
ftp:x:14:50::/home/ftp:
smmsp:x:25:25:smmsp:/var/spool/clientmqueue:
mysql:x:27:27:MySQL:/var/lib/mysql:/bin/bash
rpc:x:32:32:RPC portmap user:/:/bin/false
gdm:x:42:42:GDM:/var/state/gdm:/bin/bash
pop:x:90:90:POP:/:
nobody:x:99:99:nobody:/:
orbital:x:1000:100:Rossini Moraes:/home/orbital:/bin/bash
```

Exemplo 2 – 1: O arquivo /etc/passwd.

Será verificado cada conta utilizando a conta atribuída ao usuário orbital (a última linha):

orbital: Nome do usuário;

x: Senha criptografada;

1000: User ID (UID);

100: Group ID (GID);

Rossini Moraes: Nome Verdadeiro;

/home/orbital: Diretório do usuário;

/bin/bash: Usuário de *shell*

Durante o processo de criação de uma conta, a ferramenta de criação também deve criar diretórios, incluindo o novo diretório inicial (*home directory*) do usuário, em geral /home/usuário.

Além disso, se a conta for adicionada manualmente, será necessário copiar arquivos de inicialização padrão (localizados em /etc/skel) para o novo usuário, e configurar corretamente as permissões. [DANESH 2000]

OBS: em algumas distribuições, o .profile é denominado local.profile

2.2.3 ADICIONANDO USUÁRIOS COM FERRAMENTAS GRÁFICAS

Existem várias ferramentas gráficas de administração do Linux, mas elas podem variar de distribuição para distribuição. Um exemplo é o *usercfg* (programa gráfico utilizado para adição de usuários) que está disponível com o *Red Hat* e *Conectiva.*

O *usercfg* é uma ferramenta de gerenciamento de conta independente e, para utiliza-la é necessário que esteja instalado à biblioteca Python [CISNEIROS 1998].

Com a instalação concluída, basta digitar o comando *usercfg* no *Shell*.

E com isso o *usercfg* armazenará as informações em /etc/passwd.

2.2.4 ADICIONANDO USUÁRIOS COM O ADDUSER

Quando o uso de ferramentas gráficas torna o sistema uma pouco lento, os utilitários de linha de comando são mais rápidos.

Uma ferramenta de gerenciamento de contas de linha de comando é o *adduser*.

```
$ useradd John
```

O *useradd* faz todos os passos que o *usercfg*, porém só não configura a senha.

```
Lookin for first available UID... 508
Lookin for first available UID... 509
Addin login: John…done.
Creating home directory: /home/John…done.
Creating mailbox: /var/spool/mail/John…done.
Don´t forget to set the password.
```

Exemplo 2 – 2: Resultado do comando "usercfg"

Para configurar a senha para o usuário John, usa-se o comando "passwd" mais o nome de usuário.

```
$ passwd john
```

Em reposta será pedido a senha e a confirmação:

```
Enter new Unix password:
Enter new Unix password:
```

2.2.5 CONTROLE DE ACESSO

O controle de acesso concede ou nega acesso de usuários a recursos de sistema, o que inclui arquivos, diretórios, volumes, unidades, serviços, *hosts*, redes e assim por diante.

2.2.6 PERMISSÕES

Existem três tipos básicos de permissões:

- *Read* (Leitura) – Permite aos usuários a leitura de um arquivo especificado.
- *Write* (Escrita) – Permite aos usuários alterar um arquivo especificado.
- *Execute* (Execução) – Permite aos usuários executar um arquivo especificado.

Quando são atribuídas permissões, o Linux grava um registro delas. O status de permissão de cada arquivo é expresso em *tokens*. Os *tokens* de permissão são:

- r – acesso a leitura
- w– acesso a escrita
- x– acesso a execução

Exemplo de uma listagem de arquivos mostrando suas permissões. Essa listagem é feita com o comando "ls –l". Serão listados como exemplo os arquivos do diretório /home/john.

```
root@orbital:/etc# ls -l
-rw-r--r--    1 root     root        15067 Fev 24  2002 a2ps.cfg
-rw-r--r--    1 root     root         2561 Fev 24  2002 a2ps-site.cfg
-rw-r--r--    1 root     root          139 Mai  9 07:09 acentos.conf
drwxr-xr-x    3 root     root         4096 Mai 16  2002 acpi/
-rw-r--r--    1 root     root           47 Jun 18 15:11 adjtime
-rw-r--r--    1 root     root         1707 Mar  6  2002 afpd.conf
drwxr-xr-x    7 root     root         4096 Mai 22 00:53 apache/
-rw-r--r--    1 root     root         1117 Mar  6  2002 Ap-
pleVolumes.default
-rw-r--r--    1 root     root          993 Mar  6  2002 AppleVolumes.system
```

```
drwxr-xr-x    2 root     root         4096 Mai  8 22:34 apsfilter/
-rw-r--r--    1 root     root          899 Mar  6  2002 atalkd.conf
-rw-r--r--    1 root     root            0 Mai 12  1994 at.deny
-rw-r--r--    1 root     root         1229 Fev 26  2002 bootptab
```

Exemplo 2 – 3: Exemplo do comando "ls –l"

Permissões do arquivo parse_out.pl

A coluna de permissão armazena 10 caracteres.

1	234	567	8910
_	rwx	rwx	r- x
Diretório ou arquivo	Permissões de proprietário	Permissões de grupos	Permissões de todo mundo

O primeiro caractere referente a permissão especifica o tipo de recurso. Por exemplo:

- representa um arquivo.

b representa um arquivo especial de bloco.

c representa um arquivo especial de caractere.

d representa um diretório.

l representa um link simbólico

Os nove caracteres restantes são divididos em três grupos de três:

- Permissões de proprietário – essas permissões mostram o acesso de proprietário ao arquivo.
- Permissões de grupo – essas permissões mostram os arquivos de grupo de acesso.
- Permissões globais – essas permissões mostram quais direitos todos os demais usuários têm para acessar esse arquivo.

2.2.7 ALTERANDO PERMISSÕES DE ARQUIVO

O *chmod* é utilizado par configurar permissões para um usuário individual em um arquivo ou diretório. O *chmod* utiliza três operadores.

- O operador – remove permissões;
- O operador + adiciona permissões;
- O operador = atribui permissões.

Para atribuir permissões com o *chmod* é utilizado letras (r, w, x) em arquivos e diretórios. Outro método é utilizar o sistema *octal*.

2.2.7.1 O sistema octal

Neste sistema, números apresentam permissões.

Exemplo do esquema de números *octais*.

0000	Equivalente a ---, ou nenhuma permissão.
0001	Equivalente a –x, ou permissão de execução pra o proprietário.
0002	Equivalente a –w-, ou permissão de gravação para o proprietário.
0004	Equivalente a r--, ou permissão de leitura para o proprietário
0010	Permissão de execução para o grupo, onde o segundo conjunto de três é - - x.
0020	Permissão de gravação para o grupo: -w-
0040	Permissão de leitura para o grupo: r--
0100	Permissão de execução para todo mundo: - - x
0200	Permissão de gravação para todo mundo: -w-
0400	Permissão de leitura para todo mundo: r--
1000	Restringe a proprietários do arquivo o direito de excluir diretórios ou arquivos. È chamado bit de aderência (*sticky bit*)
2000	Aplica o bit SETGID.
4000	Aplica o bit SETUID.

Tabela 2 – 1: O Sistema Octal

Pode-se também reduzir as permissões de proprietário, grupo e outras a um número de três dígitos, utilizando estes valores.

0 – nenhuma permissão

1 – execução

2 – gravação

3 – gravação e execução (não é muito utilizado)

4 – leitura

5 – leitura e execução

6 – leitura e gravação

7 – leitura, gravação e execução

Exemplo da utilização do *chmod*:

```
chmod 751 myscript.cgi
```

- O proprietário pode ler, gravar e executar (7);
- O grupo pode ler e executar (5);
- Os outros usuários podem somente executar (1).

Arquivos com permissões especiais

Existem duas permissões especiais de arquivo:

- SGID (2000 octal)
- SUID (4000 octal)

Os arquivos com permissões SGID (*Super Group Identification*) ou SUID (*Super User Identification*) são especiais pois, se um programa ou arquivo estiver configurado como SUID de *root*, ele sempre executará como *root* mesmo se um usuário comum o estiver utilizando. Por essa razão, os arquivos SGID e SUID são um perigo para segurança.

Localizando arquivos com permissões de SUID no sistema:

```
find / -perm +4000
```

2.2.8 UM VISÃO DE GRUPOS DE USUÁRIOS

O Linux configura automaticamente privilégios em arquivos possuídos por *root*, protegendo assim esses arquivos de outros usuários. Ocasionalmente pode ser necessário proteger um grupo de usuários (e seus arquivos) de outros. [DANESH 2000]

2.2.8.1 Visão do arquivo de grupos

Os dados de grupos são armazenados em /etc/group.

O arquivo consiste no registro dos grupos. Cada linha armazena um registro e cada registro é dividido em quatro campos delimitados por vírgula.

```
users::100:amd,marty,John
```

- Nome do grupo;
- Senha do grupo;
- ID do grupo (GID);
- Usuários do grupo.

2.2.8.2 Criando grupos

Para incluir um grupo, usa-se o comando *groupadd*, apresentado abaixo:

```
groupadd nomegrupo
```

Pode-se também criar grupos com o ID especificado.

```
groupadd –g 503 nomegrupo
```

A opção “-g” seria a identificação do usuário.

O resultado no arquivo /etc/group é o seguinte:

```
nomegrupo::503:
```

Não existe um programa padrão para se incluir usuários em grupos. O modo mais simples é editar o arquivo /etc/group.

```
nomegrupo::503:amd,ddr,fiscal
```

2.2.8.3 Excluindo grupos

Para excluir um grupo, usa-se o comando *groupdel*.

```
groupdel nomegrupo
```

Apesar de simples, esse comando não exclui um grupo por completo. Os arquivos pertencentes ao grupo não são excluídos. Se o grupo a ser excluído é um grupo principal, então ele não será excluído.

2.3 ATAQUES DE USUÁRIOS

2.3.1 ATAQUES DE SENHAS

Esse termo descreve várias atividades incluindo qualquer ação realizada para quebrar, decriptar ou excluir senhas, ou de outro modo superar mecanismos de segurança de senha.

Na escala das ameaças de segurança, os ataques de senhas são primitivos (ou seja, os mais antigos). De fato, *crackear* senha é a primeira coisa que os invasores aprendem. Hoje é possível quebrar senhas do Linux utilizando ferramentas automatizadas. [ANÔNIMO 2000]

A segurança pobre de senha resulta em comprometimento total do sistema. Invasores que ganham inicialmente apenas acesso limitado podem expandir com rapidez esse a-

cesso atacando a segurança de senha. Por simples ataques de senha, os invasores obtém acesso *root* e detêm controle da máquina.

2.3.2 CÓDIGO MALICIOSO

Define-se como:

- Código não autorizado (contido dentro de um programa legítimo) que desempenha funções desconhecidas para usuário;
- Um programa legítimo que foi alterado pelo posicionamento de código não autorizado dentro dele que realiza funções desconhecidas e provavelmente indesejáveis; [SIDDIQUI 2002]
- Qualquer programa que pareça realizar uma função necessária e desejável mas (por causa do código não-autorizado) realiza funções desconhecidas e indesejáveis pelo usuário;
- Código não autorizado projetado para ocultar-se e destruir seus dados.

Os tipos mais comuns de código malicioso são:

- Cavalos de Tróia (programas de computador utilizados para fins destrutivos);
- Vírus (programas de computador utilizados para fins destrutivos).

2.4 SISTEMA DE ARQUIVOS

Assim como em outros pontos que compõem um sistema operacional, é estabelecido um padrão para controle do sistema de arquivos. Após escolhido qual sistema de arqui-

vos será utilizado, deve-se ficar atento a alguns pontos importantes para a segurança dos dados:

[ABSOLUTA 1998a]

- Se for utilizado o sistema de arquivos NFS, deve-se estar certo de que a configuração é adequada, como por exemplo, verificar se foram aplicadas as restrições de acesso que deixam o sistema mais seguro;
- O nome do computador deve ser informado por completo;
- Verificar a existência de programas que tenham SUID/SGID (*Super Identification/Super Group Identification*) ligados, mas, se possível, não utilizá-los;

O sistema de arquivos armazena um ambiente totalmente heterogêneo com arquivos de diversos tipos, como arquivos de programas, arquivos de configuração do sistema e arquivos de dados. Alguns cuidados são indispensáveis para tentar evitar problemas: [ANDRADE 1996]

- Fazer cópias de segurança regularmente;
- Certificar da integridade das cópias de segurança, para que em uma emergência elas estejam efetivamente íntegras para uso;
- Utilizar ferramentas de verificação de integridade de programas e arquivos como por exemplo o “checksum md4/5”;
- Ter certeza que os arquivos e diretórios têm as permissões adequadas para garantir a segurança do sistema;
- Controlar as permissões de alterações ou gravações nos dispositivos dos terminais;
- Não utilizar SUID/SGID em scripts (*shell* ou “perl”);

- Remover todos os interpretadores de comandos que não estiverem sendo utilizados (csh, zsh, ash, etc);

- Manter sempre uma listagem dos programas que utilizam SUID/SGID e compará-los a cada verificação;

- Excluir todos os utilitários e pacotes de programas do sistema que não são necessários.

3 SEGURANÇA EM REDES LINUX

3.1 TÉCNICAS DE INVASÃO

Muitas redes corporativas hoje em dia são implementadas com o objetivo de tirar o maior proveito dos serviços que as redes disponibilizam, esquecendo o aspecto da segurança dos dados que nela trafegam e a importância disso. É em umas dessas redes que o intruso poderá causar algum estrago ou roubar informações. [GERLACH 1999]

3.1.1 VULNERABILIDADES DE UM SISTEMA DE REDE

Em um sistema de rede, recomenda-se ter algum sistema de filtragem que estabelece um certo nível de segurança. Um intruso, ao atacar um servidor, está na verdade tentando se infiltrar na rede interna e ele terá sucesso se o *firewall* (programa para filtragem de pacotes) desta rede não estiver adequadamente configurado. É recomendado, então, para evitar estes tipos de intrusões, se ter sistemas de *firewall* bem configurados e executar alguma ferramenta que permita apenas computadores confiáveis acessarem à rede. [GERLACH 1999]

Nessa situação, o intruso pode determinar em quais computadores foram estabelecidas conexões, verificando assim qual dos computadores contém confiança suficiente para dar condições favoráveis ao seu ataque.

A ferramenta “TCP Wrapers” (Ferramenta que registra em *Log* solicitações de sessão, ou seja, permitir ou negar conexões de *telnet*, *ftp*, dentre outros), estando instalada nos computadores de rede, poderá determinar quais computadores poderão ou não estabelecer conexões através de portas como: *ftp* (21), *ssh* (22), *smtp* (25), *named* (53), pop3 (110), *imap* (143), *rsh* (514), *rlogin*(513), *lpd*(515).

È possível um intruso usar um *exploit* (nome usado para generalizar programas de invasão) CGI (*Common Gatway Interface*) para visualizar o arquivo "hosts.allow" do computador e então verificar para qual domínio tem acesso as portas de *telnet* e FTP. Assim, ele pode tentar obter permissão a um domínio que seja confiável para aquele computador. Desta forma, ele obtém acesso facilmente. Em função disso, se recomenda verificar o grau de confiança que se tem com os outros computadores e quanto estes estão seguros perante algum ataque.

Corporações que não estruturam adequadamente a sua rede, com certeza possuirão vários computadores e roteadores mal configurados. Isto pode ajudar um intruso a ganhar o acesso à rede interna.

Outro ponto importante a ser observado é quanto à configuração dos servidores de DNS (*Domain Name Server*). O fato de se ter servidores de DNS mal configurados, permite com facilidade o mapeamento de rede interna. Empresas tiveram sucesso em testes de penetração, onde foi possível mapear a rede interna, utilizando servidores de DNS mal configurações. Devido a problemas como este, não se aconselha colocar servidores de DNS entre computadores de rede interna e externa.

Nos servidores de correio eletrônico, onde há conexões entre rede interna a externa para que ocorra a troca de mensagens, deve haver algum sistema de filtragem para evitar o ataque de um intruso. [WRESKI 2000c]

Os ataques a uma rede corporativa podem ser reduzidos consideravelmente se as portas de "smtp", "name" e "portmapper" forem configuradas adequadamente. Deve ser reforçada a segurança nos roteadores de filtragem, evitando assim, ataques de "SMMP-Scanning" (*Simple Network protocol-Scanning*) e password *crackers*. Pois, se um intruso tentar se infiltrar, o roteador poderá servir como ponte para mais acessos não autorizados. [GERLACH 1999]

Através do processo-servidor *finger* pode-se conseguir informações para identificar pontos de uma rede que são facilmente explorados por intrusos. As informações podem ser sobre usuários, computadores e sistema operacionais.

3.1.2 COLETA DE INFORMAÇÕES

A primeira etapa que um invasor executa em sua invasão, é tentar obter informações sobre a rede interna e externa para verificar quais computadores estão de alguma forma conectados à internet. Essas informações podem ser descobertas utilizando as seguintes técnicas: [GERLACH 1999]

- Utilização de "nslookup" para execução de comandos "ls <domínio da rede>";
- Visualização do código HTML (Hyper Text Markup Language) nos servidores de *Web*, identificando os computadores;
- Visualização dos documentos gravados nos servidores de FTP;
- Através do comando *finger*, identificando usuários de computadores externos.

Utilizando os itens descritos anteriormente, fica mais fácil para o intruso obter uma lista dos computadores com suas relações de confiança e entendê-las. Ao tentar a invasão, o intruso pode cometer algum erro, deixando alguma pista em *logs*. Se, por acaso, houver desconfiança de estar sendo atacado e se tiverem os registros de *logs* do sistema habilitados, então pode-se verificá-los para tentar descobrir o que está acontecendo.

3.1.3 IDENTIFICAÇÃO DE PONTOS VULNERÁVEIS DA REDE

Quando o intruso obtiver a relação de informações que identifica os computadores internos e externos, ele poderá utilizar ferramentas disponíveis para determinar as vulnerabilidades individuais de cada computador. Alguns programas podem ser utilizados para descobrir

as vulnerabilidades em determinado computador, tais como: “ADMhack”, “mscan”, “nmap”, entre outros. [GERLACH 1999]

Geralmente os programas que o intruso utiliza têm origem de máquinas com conexões com taxas de comunicação altas. Através dessas máquinas pode-se efetuar o ataque rapidamente, antes que seja notada a sua presença. A ferramenta “ADMhack”, para ser executada, exige o acesso *root* ao sistema e obtendo este acesso o intruso escolherá uma máquina, através da qual conseguirá obter acesso a outros computadores.

Outros *scans*, como o “mscan” e “nmap” já não exigem o acesso *root* para serem executados, porém são lentos e não conseguem se ocultar, como o acaso do “ADMhack”. O “mscan” e o “nmap”, também existem para outras plataformas.

O “ADMhack e o “mscan” exploram os seguintes testes em computadores remotos:

- “TCP portscan”;
- “dump” do servido RPC executado pelo “portmapper”;
- lista de dados exportados por “nfsd”;
- lista os compartilhamentos pelo samba ou *netbios*;
- solicitações *finger* para verificar a existência de contas padrão;
- verificações de vulnerabilidades em CGIs;
- localização de versões vulneráveis de *daemons* do tipo “sendmail”, “POP3”, “RPC staus”, “RPC mountd” e “IMAP” (*Internet Message Access Protocol*).

3.1.4 SNIFFERS

Após o intruso ter invadido e se apossado de um determinado computador da rede, o próximo passo será instalar *backdoors, trojans* em seguida *sniffers*. *Sniffers* são programas que trabalham, geralmente em rede *Ethernet*, utilizando uma interface de rede do computador em modo promíscuo. Esta interface "escuta" tudo o que passa pelo barramento e envia para o arquivo de *log* do *sniffer*.

Através dos *sniffers*, será capturado todo e qualquer dado que trafegar pelo barramento *Ethernet* onde estiver conectado o *sniffer*. Dados como nomes de usuários, senhas, seja senhas com "shadow" ou não, e informações que identificam os computadores podem ser coletadas da rede. O arquivo de *log*, que é armazenado no computador local que foi instalado pelo invasor, é regularmente consultado pelo mesmo.

Os *trojans* e *backdoors* costumam estar camuflados em arquivos binários do sistema (comentados no texto) para não serem descobertos. Os *sniffers* também utilizam este procedimento, instalando um *backdoor* no binário "ps" do sistema, por exemplo, para não ser percebido. Eles executam em *background* no sistema gerando como resultados os *logs*.

Existe uma ferramenta chamada "ifstatus", que detecta a existência de interfaces de rede promíscua para o caso de execução de um *sniffer*. [BAUER 2002]

3.1.5 SCANNERS

Um *scanner* é uma ferramenta de segurança que detecta vulnerabilidades de sistema.

Seu objetivo é varrer o /etc/passwd, procurando campos de senhas vazios. Para cada campo vazio que ele encontra, ele adverte o usuário via correio eletrônico. Embora isso seja rudimentar, demonstra concisamente o conceito de *scanner*. [UNICAMP 2001]

Existem vários tipos de *scanners*, mas todos se enquadram em uma das duas categorias.

- *Scanners* de sistema;
- *Scanners* de rede.

3.1.5.1 *Scanner* de Sistema

Varrem o *host* local, procurando vulnerabilidades óbvias de segurança que surgem de descuidos, pequenos deslizes e problemas de configuração que até usuários experientes às vezes perdem, por exemplo, permissões de arquivos excessivas, contas-padrão, entradas duplicadas de UID.

3.1.5.2 *Scanner* de Rede

Os *scanner* de rede testam *hosts* sobre conexões de rede, quase da mesma forma que um *cracker*. Eles investigam serviços e portas disponíveis, procurando as fraquezas mais conhecidas que invasores remotos podem explorar. [CISNEIROS 1998]

3.1.6 SPOOFING

O *spoofing* ocorre quando invasores autenticam uma máquina em nome de outra forjando pacotes de um *host* confiável (*trusted*). Nos últimos anos, essa definição foi expandida para abranger qualquer método de subverter a relação de confiança ou autenticação baseada em endereço ou em nome de *host*. [ANÔNIMO 2000]

3.2 SERVIÇOS

Dentre os vários serviços disponíveis no Linux, para alguns deles deve-se ficar atento a detalhes como, por exemplo, configuração e atualização. No sistema Linux existem algumas falhas de programas e serviços ou de configurações dos mesmos que permitem que alguém, através de um acesso remoto, possa obter várias informações para chegar a algum objetivo, seja para roubo dessas informações, como para concorrência ou para o planejamento de invasões futuras.

Portanto, deve-se sempre manter o sigilo do sistema. Um exemplo de um programa que pode fornecer informações sobre o usuário que está utilizando o sistema é o *finger*. Através de um *finger* (tcp/79) podem ser extraídas várias informações sobre o sistema, como: quando tal usuário se encontra com sua sessão aberta no sistema, o que ele está executando, e quem está utilizando o sistema. Estes dados são suficientes para o invasor obter senhas, qual a conexão que ele poderá utilizar e em que momento. Se não puder evitá-lo, deve-se sempre mantê-lo atualizado. [WRESKI 2000b]

Um outro exemplo é o "netstat" "Tcp/15". Este acaba dando informações sobre conexões atuais, como endereços, portas, serviços de nomes, etc. Estes serviços podem ser dispensáveis na maioria dos casos.

Outro serviço no qual deve-se tomar cuidado, em termos de ocultar informações, é o "systat" (tcp/11). Ele deixa visíveis todos os processos que estão em execução no momento em um determinando computador, então, se este não for desabilitado, qualquer usuário poderá usá-lo e saber o que está sendo executado naquele momento.

Deve-se também analisar os serviços que serão habilitados antes de conectar um computador a um ambiente de rede. Com isso, se reduz a preocupação com a segurança do sistema, focalizando apenas nos serviços úteis. [WRESKI 2000b]

Deve-se desabilitar todo e qualquer serviço RPC (*Remote Procedure Call*) desnecessário. O RPC tem sido em sua ausência um serviço inseguro, e já existem outros serviços

mais seguros que o substituem. Utilizando o comando “rpcinfo –p hostname” pode-se listar quais serviços RPC que estão executando no computador.

A melhor forma de configurar o sistema é habilitar apenas o que se necessita para atender o propósito final. Deve-se então verificar os serviços disparados no servidor, excluindo aqueles que não são necessários. Porque quanto mais serviços estiverem habilitados em um computador, haverá mais possibilidade de que exista um programa para explorar uma fraqueza de conjunto de serviços. Esses programas são ferramentas utilizadas pelo intruso que o auxiliam no momento do ataque.

3.2.1 SMTP

O SMTP (*Simple Mail Transport Protocol*) é um dos mais importantes serviços utilizados atualmente, e infelizmente é também um dos mais vulneráveis. Ele é responsável por entregar *e-mail*s através da rede que trabalham com TCP/IP.

No Linux, um programa que serve de interface para o SMTP é o “sendmail”. Quando for utilizado o “sendmail”, é importante utilizar sempre a versão mais atual, porque existem muitos problemas conhecidos. As versões mais recentes dos “sendmail” trazem melhorias para prevenir as mensagens não-solícitas (*spam*) que contêm a característica *ant-relaying* habilitado já na instalação padrão. Além disso, estas versões mais novas do “sendmail” previnem contra a utilização de servidores da rede como repetidores de *mail* e a soma de novos filtros que ajudam a prevenir contra o *spam*.

3.2.2 FTP

Atualmente, os servidores de FTP são os mais vulneráveis, independentes do sistema operacional utilizado. Dentre eles, os mais utilizados hoje no Linux são o “wu_ftp” e o “proFTP”. Quando é lançada uma versão mais nova de um destes programas, logo em seguida

é anunciado que o servidor foi atacado, O FTP deve ser utilizado apenas em extrema necessidade e, mesmo assim, em casos que não onerem o funcionamento do sistema pela ocorrência de algum tipo de invasão. Portando, aconselha-se utilizar, se for o caso, um desses dois serviços na suas últimas versões.

Ao instalar o "wu_ftp", este vem no padrão para o usuário ter acesso no diretório *home* e para que ele consiga acessar toda a máquina com permissão de leitura. Como conseqüência dessa permissão, o usuário poderá entrar no sistema e buscar informações que não lhe dizem respeito, e até mesmo procurar por falhas. [WRESKI 2000a]

3.2.3 TELNET

O *telnet* possibilita que usuários acessem um computador remotamente, bastando o serviço estar disponível. Por não utilizar nenhum tipo de criptografia ele acaba se tornando muito vulnerável, pois pode estar sendo utilizada uma ferramenta na rede para coletar dados ou senhas. O que geralmente é feito pelos administradores é desabilitar este serviço.

Através do *daemon* "ttysnoop" pode-se habilitar terminais para acessarem conexões de *telnet* e também associar a cada um deles um terminal do console a fim de ser monitorado. [CISNEIROS 1998]

3.2.4 HTTP

Foi para plataformas Unix que a *Web* apareceu, portanto existe uma ampla variedade de servidores HTTP disponíveis.

A maioria dos servidores *Web* (HTTP) Linux é gratuita. Os mais conhecidos são:

- NCSA httpd
- Apache

- AOLServer
- Boa
- WN
- WEC/Cern

3.2.4.1 O servidor *Web* Apache

O Apache é *software* de servidor mais usado. Significando "A Patchy Server" (um servidor patchy), o Apache nasceu do trabalho realizado para consertar o NCSA httpd, um dos servidores *Web* originais, para corrigir alguns problemas e incluir funcionalidade.

Desde então o Apache tem surgido como o servidor não-comercial de escolha para sistemas Unix. Mais recentemente, ele foi portado para o *Windows*, e pode ser usado como servidor *Web* em sistemas *Windows* NT.

O Apache oferece numerosos recursos que o tornam atraente para administradores de sistema Unix. Além de usar uma configuração baseada nos arquivos de configuração originais do "NCSA httpd", o Apache está disponível com o código-fonte completo e é desenvolvido coletivamente, como muitos outros aplicativos populares para o ambiente Linux.

O Apache oferece sua própria API (Interfaces de Programa para Aplicativos – Oferecem um meio de escrever programas que se integram fortemente com o servidor *Web* e geralmente não exigem novos processos para cada solicitação), que pode ser usada como uma alternativa à CGI (que também é suportada pelo Apache). Além disso, a API pode ser usada para produzir módulos de *plug-in* que servem para vários propósitos. Entre os módulos disponíveis estão:

- Sistemas de autenticação alternativos, incluindo a autenticação de servidores NIS ou de banco de dados LDAP (*Lightweight Directory Access Protocol*)

- Ambientes de script no lado do servidor, que têm a mesma função do Microsoft *Active Server Pages* ou do *Netscape LiveWire*, incluindo PHP.

- Módulos projetados para melhorar o desempenho da CGI tradicional. Por exemplo, o módulo *FastCGI* usa atalhos para minimizar o tempo que leva para executar um programa CGI e retorna o resultado. O módulo "perl" permite que scripts "perl" sejam executados em um único processo e compilados apenas na primeira execução, fazendo com que a CGI baseada em "perl" desempenhe quase com a mesma rapidez que os programas CGI compilador e alguns aplicativos *Web* baseados em API

3.3 SEGURANÇA NO NÚCLEO (*KERNEL*)

Quando se tem vários usuários que acessam as máquinas pela rede, convém ter um núcleo atualizado e testado pelo fornecedor. Estas versões atualizadas garantem que o sistema não seja pego de surpresa por algum intruso, decorrente de alguma falha que esteja embutida no núcleo. É ele que controla os vários serviços disponíveis na rede dos computadores. Para manter a segurança do sistema a nível de núcleo, é aconselhável sempre atualizá-lo pela sua versão mais atual, mantê-lo o mais enxuto possível, habilitando somente o necessário, e deve-se ter o máximo de cuidado nos serviços que serão habilitados. Com isso, o sistema ficará mais rápido, por ter um código mais atualizado e menos sujeito a falhas. [CISNEIROS 1998]

Após a decisão de quais serviços e opções que ficarão habilitadas, sugere-se a compilação do núcleo.

Alguns desenvolvedores de novas versões de núcleo têm se preocupado muito no momento do desenvolvimento, com a segurança em aplicações que tenham SUID. Portanto, não é aconselhável utilizar versões do núcleo ainda experimentais. Além disso, é necessário seguir alguns cuidados.

- Manter acesso restrito a ligações (*links*) no diretório “tmp”;
- Manter acesso restrito a dutos (*pipes*) no diretório “tmp”;
- Manter o acesso restrito para o diretório “proc”;

São várias as opções disponíveis para melhorar a segurança do sistema Linux a-través de diretório /proc, pseudo-sistema de arquivos. Alguns dos arquivos em /proc/sys estão diretamente relacionados à segurança. Muitas destas opções estão disponíveis no arquivo /proc/sys/net/ipv4, o qual contém: [WRESKI 2000a]

- icmp_echo_ignore_all: ignora todas as requisições de “ICMP (*Internet Control Message Protocol*) ECHO”. A habilitação desta opção irá prevenir este computador de responder requisições de *pings*;
- icmp_echo_ignore_broadcasts: ignorar requisições de “ICMP ECHO” com broadcast/multicast como endereço de destino. A rede poderia ser usada para dar início aos ataques do tipo DoS, inundando outros computadores com pacotes;
- ip_forward: habilita ou desabilita o reenvio de pacotes entre interfaces de rede. O valor padrão depende e o núcleo foi configurado como computa-dor ou roteador.
- ip_masq_debug: habilita ou desabilita a depuração do mascaramento de IP (*Internet Protocol*);
- tpc_syncookies: proteção de “SYN atack”. Envia “syncookies” quando a fila de “SYN backlog” de um *socket overflows* fica sobrecarregada. Onde

os “syncoockies” soa métodos de segurança que se utilizam de “cookies” para checar a validação de IP solicitada e a existência deste;

- rp_filter: determina se a verificação de endereços está habilitada. Habilitando esta opção em todos os roteadores, será possível prevenir ataques de *spoofing* na rede interna;

- accept_source_route: determina se os pacotes roteados são aceitos ou recusados. Deve ser desabilitado até que se tenha uma forte razão para habilitá-lo.

3.4 ATAQUES DE RECUSA DE SERVIÇO

Um ataque de recusa de serviço (DoS) é qualquer ação iniciada por alguma pessoa que incapacitaria o *hardware* ou o *software* de um *host*, ou ambos, tornando o sistema inacessível e, portanto, negando serviço para usuários legítimos (ou mesmo ilegítimos). Em um ataque de DoS, o objetivo do invasor é simples e direto: derrubar o *host* da rede. Exceto quando as equipes de segurança testam *hosts* vulneráveis, os ataques de DoS são sempre maliciosos e, hoje em dia, ilegais.

A recusa de serviço é um problema persistente por duas razões: em primeiro lugar, os ataques de DoS são rápidos e fáceis, e eles geram um resultado notável e imediato. Portanto, são populares e proliferam entre “*crackers*”. Como isso, os administradores de sistema devem esperar freqüentes ataques de DoS.

Há uma razão mais importante para que ataques de DoS permaneçam problemáticos. A maioria dos ataques explora erros, limitações ou inconsistência em implementações de TCP/IP de fornecedores existentes, até que os fornecedores corrijam o problema. Nesse ínterim, todos os *hosts* afetados permanecem vulneráveis.

Um exemplo típico foi o ataque de “teardrop”, que enviava pacotes de “UDP” malformados para *hosts Windows*. Os alvos examinavam os cabeçalhos de pacotes malforma-

dos, engasgavam-se com eles e geravam um exceção fatal. Quando esse ataque surgiu, a Microsoft rapidamente reexaminou sua pilha de TCP/IP e gerou um correção e colocou as atualizações disponíveis para o público.

Entretanto, as coisas não são sempre tão simples assim. Quando se dispões do código fonte do Sistema Operacional, como usuários de Linux, à medida que surgem novos ataques de DoS, é de extrema importância corrigir o *software*, re-configurar o *hardware* ou filtrar as "portas" atacadas, dependendo da situação. [ANÔNIMO 2000]

3.4.1 RISCOS IMPOSTOS POR ATAQUES DE RECUSA DE SERVIÇO

Houve um tempo em que as pessoas viam os ataques de DoS como meros aborrecimentos. Eram problemas, certamente, mas não eram necessariamente críticos. Algumas pessoas ainda mantêm essa visão, argumentando que a maioria dos ataques de DoS eliminavam somente certos serviços que são fáceis de reiniciar. Mas, isso não é mais o ponto de vista predominante. Em vez disso, ataques de DoS agora estão sendo vistos de outra forma, principalmente porque os hábitos de computação da sociedade alteraram-se rapidamente. Os servidores hoje são os ingredientes essenciais no comércio eletrônico e outros serviços críticos. Nesse novo ambiente, incentivar ataques de DoS pode diminuir ou mesmo interromper lucros. De fato, poucas organizações podem ter recursos para um mal-sincronizado ataque de DoS. [ANÔNIMO 2000]

3.4.2 ATAQUES DE DOS CONTRA *HARDWARE* DE REDE

A metodologia de DoS contra *hardware* de rede tem paralelos muito próximos de suas variedades comuns. De fato, em muitos casos, o mesmo ataque de DoS inutiliza tanto o *software* quanto o *hardware*. Alguns exemplos.

- Os invasores enviam solicitações de conexão a partir de endereços de IP forjados, inexistentes. Como a unidade receptora não pode resolver esses endereços, a sessão pode ficar pendente. (Essa condição pode incapacitar um único serviço ou porta, ou o sistema inteiro.)

- Os invasores comprometem todas as sessões disponíveis, evitando assim o alcance do roteador remotamente. Sendo assim, é necessário que o sistema seja reinicializado manualmente.

- Os invasores exploram estouros em rotinas de *login*, causando a queda da unidade ou reinicializando-a . Com isso, pode ser necessário a reinicialização do sistema.

- Os invasores inundam o sistema com pacotes ou estruturas peculiarmente malformados. O sistema não pode processá-lo corretamente e se auto-bloqueia.

3.5 FIREWALL

O *Firewall* consiste em um conjunto de componentes organizados de uma forma a garantir certos requisitos de segurança. Os componentes básicos para a construção de um *firewall* são:

- *Packet Filters*: são responsáveis pela filtragem (exame) dos pacotes que trafegam entre dois segmentos de rede.

- *Bastion Host*: computador responsável pela segurança de um ou mais recursos (serviços) de rede.

3.5.1 *BASTION HOST*

Bastion Host é qualquer máquina configurada para desempenhar algum papel crítico na segurança da rede interna. Essa máquina promove serviços permitidos segundo a política de segurança da empresa.

Segundo Marcus Ranum (um dos responsáveis pelo termo *Bastion*), *bastions* são áreas críticas de defesa, geralmente apresentando paredes fortes para desencorajar os invasores.

Um *bastion host* deve ter uma estrutura simples, de forma que seja fácil de garantir a segurança. É importante que se esteja preparado para o fato de que o *bastion host* seja comprometido, considerando que ele provavelmente será alvo de ataques.

Dependendo do seu tipo, o *bastion host* tem responsabilidades diferentes do *packet filter*. Alguns autores enfatizam que enquanto o *packet filter* atua em um nível mais baixo, o *bastion host* se encarrega de todos os níveis (referentes ao modelo OSI). Na realidade, um *host* pode acumular tanto as funções de filtragem de pacotes como também pode prover alguns serviços. Neste caso, ele seria um *packet filter* e *bastion host* simultaneamente. Independentemente de qual seja a nomenclatura adotada, o que se deve ter em mente é o papel que estes dois componentes desempenham filtragem e provêem serviços. Traçando um paralelo destes dois componentes em relação ao modelo OSI, o *packet filter* realiza algum exame dos pacotes até o nível de transporte da camada OSI enquanto o *bastion host* se encarrega dos níveis superiores, basicamente o nível de aplicação.

Este tipo de máquina também recebe a denominação de "*aplication gateway*" porque funciona como um *gateway* no nível de aplicação da camada OSI. Os servidores disponíveis nos *bastion host* são denominados de "proxy servers", ou seja, servidores por procuração que atuam como intermediários entre o cliente e o servidor. Neste caso, os servidores só podem ser providos via *bastion host*, obrigando o cliente (por exemplo, via regra de filtragem nos roteadores) a acessar estas máquinas. Portanto, este é um mecanismo que garante que o

bastion host seja desviado (a não ser que o roteador seja comprometido e as regras de filtragem alteradas). [BAUER 2002]

3.5.1.1 Tipos Especiais de *Bastion Hosts*

Dependendo da localização do *bastion host* dentro do *firewall,* tem-se alguns tipos de máquinas com funções diferentes na segurança, os quais são:

- *Dual homed host*: trata-se de um computador com duas interfaces de rede conectadas cada uma a segmentos diferentes de rede. Uma das características fundamentais dessa configuração é que o roteamento direto (IP forwarding) é desabilitado e, portanto, todo o roteamento é realizado a nível de aplicação. Neste caso, todos os serviços segurados podem ser fornecidos via procuração (*proxys servers*) e somente o tráfego referente aos serviços habilitados via *proxy* e aqueles especificados pelas regras de filtragem circulam entre os dois segmentos de rede conectados ao *bastion host*.

- *Victim machines*: estas máquinas abrigam serviços que não são considerados fáceis de serem segurados. A máquina é configurada basicamente somente com os serviços fornecidos, para garantir que nada mais significativo esteja à disposição do atacante, caso a máquina seja comprometida. Geralmente uma "subnet" possui uma ou mais máquinas deste tipo.

- *Internal bastion hosts*: são aquelas máquinas com maior interação com as máquinas internas (por exemplo uma máquina que recebe o *e-mail* e o reenvia a um servidor de correio eletrônico residente na rede interna)

Caso o *bastion host* esteja localizado junto a sub-rede interna, ele pode ser facilmente desviado na ocorrência do roteador externo ser comprometido. O mesmo não ocorreria caso este *bastion host* fosse do tipo dual *homed* e estivesse entre o roteador externo e a rede interna. Há recomendação para não utilizar a configuração do dual *homed host* fundida com o roteador interno porque todo o tráfego interno pode ser capturado, caso o *bastion host* seja

comprometido. Em contrapartida, a fusão do *bastion host* com o roteador externo pode ser realizada sem que ocorram grandes problemas (caso exista um roteador externo). [BAUER 2002]

3.5.2 PACKET FILTERS

Como tornou-se comum hoje as redes internas (intranet) estarem conectadas à internet através de um servidor, é de extrema importância dar-se atenção aos protocolos IP, TCP, ICMP e UDP. Estes são os principais protocolos a nível de rede e transporte (Modelo OSI) que são considerados e examinados ao se estabelecer regras de filtragem em um *packet filter* para a internet. Este mecanismo de filtragem a nível de roteador possibilita que se controle o tipo de tráfego de rede que pode existir em qualquer segmento de rede, conseqüentemente, pode-se controlar o tipo de serviço que podem existir no segmento de rede. Serviços que comprometem a segurança da rede podem, portanto, ser restringidos.

O *packet filter* não se encarrega de examinar nenhum protocolo de nível superior ao nível de transporte, como por exemplo, o nível de aplicação que fica como tarefa dos aplicativos *gatways* (*proxys servers*). Portanto, qualquer falha de segurança a nível de aplicação não pode ser evitada, utilizando somente um *packet filter*. O componente que realiza a filtragem de pacotes geralmente é um roteador dedicado, mas pode ser um *host* de propósito geral configurado como roteador e recebe a denominação de *screening router*. Há ferramentas *freeware* (ferramentas de livre distribuição) para se configurar um simples computador como um roteador, com recursos de filtragem de pacotes, consistindo em uma alternativa de baixo custo.

Frisando que a filtragem dos pacotes não considera protocolos acima do nível de transporte, não é tomada nenhuma decisão baseada no conteúdo dos pacotes, ou seja, nos dados dos pacotes propriamente ditos. A filtragem que a maioria dos *screening routers* realizam são baseadas nas seguintes informações:

- Endereço IP fonte;

- Endereço IP destino;
- Protocolo: se o pacote é TCP, UDP ou ICMP;
- Portas TCP ou UDP fontes;
- Portas TCP ou UDP destino;
- Tipo de mensagem ICMP (se for o caso)

No protocolo TCP existe um *flag* denominado ACK que é utilizado para a confirmação de pacotes e também pode ser utilizado para detectar se o pacote é o primeiro de uma solicitação de conexão. Quando o *flag* não estiver setado significa que o pacote se refere a um solicitação de conexão e, caso contrário, o pacote corresponde a alguma conexão já existente. Desta forma, o *packet filter* pode bloquear um serviço *inbound* (de fora para dentro, ou seja, o servidor está na rede interna) apenas não permitindo o fluxo de pacotes com o ACK setado, destinado a um servidor interno associado a *port* (por exemplo, a port 23 do *telnet*) do serviço bloqueado. Em protocolos não orientados a conexão, por exemplo o protocolo UDP, não é possível tomar nenhuma decisão deste tipo, ou seja, nestes protocolos, nunca se sabe se o pacote que está chegando é o primeiro que o servidor está recebendo. Para fazer uma filtragem correta dos pacotes é importante saber se o protocolo é bidirecional (pacotes fluem nos dois sentidos, cliente para servidor e vice-versa) ou unidirecional. Não se pode confundir serviços *inbound* (a rede interna provendo algum serviço) e serviços *outbound* (o cliente está na rede interna e o servidor na internet) com pacotes *inbound* (pacotes que chegam na rede interna) e pacotes *outbound* (pacotes que saem da rede interna), ou seja, ambos os serviços apresentam pacotes *inbound* e *outbound* caso o protocolo seja bidirecional. [CISNEIROS 1998]

3.6 REALIZAÇÃO DE AUDITORIA EM ARQUIVOS DE *LOGS*

Como em qualquer sistema operacional de rede, sempre soa necessárias auditorias periódicas, que têm o objetivo de conferir determinadas falhas de sistema. Através dessas au-

ditorias, são avaliadas as necessidades de serem aplicadas novas técnicas de segurança. As falhas nas configurações dos serviços são uma das causas mais comuns nos sistemas. [SLO-HA 1999]

É através de sistemas de *logs*, simples arquivos de textos, que um administrador poderá analisar a ocorrência de um incidente e poderá concluir se é ou não um ataque. A seguir são listados algumas discas sobre *logs* do sistema:

- Através do /etc/syslog.conf se tem uma grande variedade de opções para ativar *logs* para auditoria de sistemas;
- Consultar os arquivos de *logs* para auditoria periodicamente;
- Consultar os arquivos gerados pelos *daemons* como o xferlog (arquivo de *log* de transferências ftp), "syslog" (syslogd), *messages* (syslogd), *access log* (httpd);
- Verificar o arquivo de *log* "sulog".

Para o caso de análise de algum problema no sistema ou da desconfiança de algum invasor estar tentando entrar no sistema, existe o serviço "syslog" comentado no parágrafo anterior, que registra através do *daemon* "syslogd", os vários eventos que acontecem no sistema em arquivos de *logs*. Desta forma, o administrador conseguirá monitorar através destes, o que acontece no sistema. Há tempo o "klogd" que é o processo-servidor responsável por registrar tudo que acontece a nível de núcleo. Esses registros em arquivos de *logs* podem auxiliar na resolução de problemas.

A maneira que o *daemon* "syslogd" registra informações, em formato de *log* para os arquivos específicos, é de fácil entendimento para o caso de uma posterior análise. [ANÔNIMO 2000]

Exemplo de algumas linhas do arquivo de configuração de *logs*.

```
# Monitora as tentativas de autenticações
auth.*;authpriv.*      /var/log/authlog
```

```
# Monitora todas as mensagens do núcleo
kern.*                  /var/log/kernlog
# Monitora todas as tendências e mensagens de erro
.warn;.err              /var/log/syslog
# Envia uma cópia para o “loghost” remoto
Configure “syslogd init”
# Script para executar com –r –s domain.com opcoes no servidor de log.
# se deve construir um alto nível de segurança no servidor de logs !
*.info                  @localhost
auth.*;authpriv.*       @localhost
```

Exemplo 3 – 1: Configuração do arquivo de *LOG*.

Além disso, deve ser restringido o acesso aos usuários ao diretório de log.

```
# Registra as permisoes no arquivo “logrotate.d”
chmod 751 /var/log /etc/logrotate.d
# Registra as permissões no arquivo “syslog.conf”
chmod 640 /etc/syslog.conf /etc/logrotate.conf
# Registra as permissões nos arquivis de log no diretório de logs
chmod 640 /var/log/*.log
```

Exemplo 3 – 2: Restrição de acesso ao diretório de *LOG*.

Há alguns diretórios que geralmente são utilizados para armazenar *logs* em sistemas UNIX: /usr/adm, /var/adm, /var/log, /etc e /etc/security, sendo que podem haver variações na localização destes conforme a versão e distribuição.

Qualquer serviço de rede, que seja inicializado pelo *daemon* “inetd” como (popd, imapd, ftpd, telnetd, sshd.), pode enviar mensagens de log, e estes *logs* poderão ser gerenciados por ferramentas como a “TCP Wrappers”.

Seguem alguns cuidados gerais:

- Criar uma política de *logs* séria. Verificar qual tipo de informação estará disponível para se registrar nos *logs*, quais mecanismos fazem tal registro, onde será executada a terefa e onde os *logs* serão armazenados;

- Verificar se as informações necessárias são registradas nos *logs* pelos mecanismos disponíveis pelo sistema;

- Habilitar todos os mecanismos de log disponíveis, e, aos poucos, ir filtrando quais são mais importantes para possíveis análises de detecção de intrusão;

- Monitorar regularmente todos os *logs* verificando registros não comuns e evidenciando possíveis ataques;

- Analisar toda e qualquer ocorrência estranha que ocorra no sistema, tal como: reinicialização do sistema;

- Se houver confirmação de intrusão pelos *logs*, deve-se compartilhar estas mensagens com a equipe de segurança ou com os administradores das redes comprometidas;

- Aconselha-se utilizar algum "script" ou ferramenta que auxilie o analisador, reduzindo o grande volume de informações e não retirando o essencial para a análise;

- Ao analisar sinais de intrusão, deve-se ter certeza que a intrusão esteja sendo realizada antes de passar a informação adiante;

- Efetuar cópias de segurança dos arquivos de *logs* regularmente, se possível em mídias não regraváveis;

- Mudar o local dos *logs* de tempos em tempos, para evitar que o intruso os localize com facilidade;

- Utilizar servidores de tempo para sincronizar todos os dispositivos da rede, inclusive computadores normais, para que se unifique os *timestamps* dos *logs*;

- Aconselha-se a instalação de uma máquina que deve centralizar os registros de *logs* e ter uma melhor segurança;

- Para o auxílio e automatização das tarefas, deve-se incluir utilitários para auxiliar em tarefas, como:

  - A procura de padrões de intrusão;

  - Para isolar *logs* que se destacam entre outros;

3.7 DETECÇÃO DE INVASÃO

A detecção de invasão é uma prática de utilizar ferramentas automatizadas para detectar tentativas de invasão em tempo real. Essas ferramentas são chamadas *Intrusion Detection Systems* (IDS).

3.7.1 CONCEITOS BÁSICOS DE DETECÇÃO DE INVASÃO

Em sistemas de detecção de invasão baseados em regras, há duas abordagens: preempitiva e reacionária. A diferença está relacionada principalmente com o tempo:

- Na abordagem preempitiva, a ferramenta de detecção de invasão ouve o tráfego de rede. Quando a atividade suspeita é observada (uma inundação de pacotes particulares, por exemplo), o sistema toma a ação apropriada.

- Na abordagem reacionária, a ferramenta de detecção de invasão observa os *logs* em vez de ouvir o tráfego de rede. Novamente, quando atividade suspeita é observada, o sistema toma a ação apropriada.

Essa diferença pode parecer sutil, mas não é. A abordagem reacionária é simplesmente um passo à frente do registro em *logs* padrão; ela faz um alerta para o ataque que acabou de acontecer. [ANÔNIMO 2000]

Ao contrário, a abordagem preempitiva realmente permite que o sistema emita uma resposta enquanto um invasor está montando o ataque. Além disso, certos sistemas permitem que os operadores testemunham e monitorem ao vivo um ataque em progresso.

É possível alcançar o modelo reacionário simplesmente utilizando ferramentas de segurança padrão do Linux. Teoricamente, é possível construir um sistema de detecção e resposta a pseudo-invasão assim:

- Utilizar "LogSurfer" (ferramenta de segurança) para observar certa atividade predefinida e ameaçadora nos *logs*.
- Um script que adiciona o endereço do invasor (ou mesmo sua rede inteira) a "hosts.deny" de modo que "tcpd" recuse conexões futuras.

Esse tipo de detecção é do tipo rápida. O invasor foi detectado e seu endereço agora será recusado. Entretanto, essa abordagem tem muitas deficiências. Uma é que os endereços de origem não são confiáveis. Eles são facilmente forjados, então um invasor pode continuar tentando, utilizando um endereço de origem diferente a cada vez.

Mas os modelos preemptivos também têm deficiências. Um, é que eles utilizam recursos intensamente. Isso realmente representa dois problemas: um, devido à interatividade inerente desses sistemas e outro, devido a limitações de *hardware* e *software*.

Primeiro, se um invasor sabe que é possível estar escutando um sistema preemptivo de detecção de invasão, ele pode fazer várias suposições. Uma, é que os IDS (*Intrusion Detection Systems*) empreenderá uma ação idêntica quando defrontado com um ataque idêntico. Portanto, inundando o *host* com múltiplas instancias do mesmo ataque de diferentes ende-

reços, ele pode realizar um ataque de saturação de processo e talvez derrube ou incapacite os IDS.

Segundo, dependendo da capacidade do processador e das limitações de memória, pode ser forçado a escolher análise de tráfego à análise de conteúdo. A análise de tráfego consome menos recursos porque está em processamento cabeçalhos de pacote e não de conteúdo. Isso protege contra muitos ataques, mas não todos. Não por muito tempo. Um número substancial de assinaturas de ataque está escondido no conteúdo do pacote e a simples análise de tráfego é insuficiente para esses casos. [ANÔNIMO 2000]

Por fim, ambas as abordagens podem gerar falso-positivo, o que pode ter conseqüências sérias. Por exemplo, muitas pessoas instruem o sistema de detecção de invasão a emitir um bip quando um ataque está em progresso. Depois de muitos falsos-positivos, os membros da equipe técnica começaram a ignorar esses avisos. Ou seja, os avisos não farão mais sentido.

De fato, a detecção de invasão não está cem por cento evoluída. Independentemente do sistema escolhido, pode-se ter um ou mais problemas. Além disso, a maioria dos sistemas de detecção de invasão disponíveis publicamente são mais um conjunto de ferramentas do que soluções finais.

4 FERRAMENTAS DE SEGURANÇA

Neste capítulo, serão apresentadas ferramentas para tentar deixar um servidor de intranet Linux menos vulnerável possível. Além de ferramentas, serão apresentadas soluções que são configuradas no próprio sistema operacional.

4.1 FERRAMENTAS PARA CONTROLE DE *SNIFFERS*

Os ataques de *sniffer* são difíceis de detectar e impedir porque os *sniffers* são programas passivos. Eles não geram uma trilha de evidência (*logs*) e, quando utilizado adequadamente, eles não utilizam uma grande quantidade de disco e recursos de memória.

A sabedoria convencional manda que para procurar um *sniffer* deve-se determinar se quaisquer interfaces de rede estão no modo promíscuo. Para isso, existem algumas ferramentas.

- ifconfig
- ifstatus

Será apresentado agora um exame de cada programa.

4.1.1 IFCONFIG

Pode-se detectar rapidamente uma interface no modo promíscuo em um *host* local utilizando “ifconfig”, uma ferramenta para configurar parâmetros de interface de rede. Para executar o “ifconfig”, é só emitir o comando em um promtp, assim:

```
$ ifconfig
```

Em resposta, “ifconfig” informará o status de todas as interfaces.

4.1.2 IFSTATUS

O utilitário “ifstatus” verifica todas as interfaces de rede no sistema e informa qualquer que esteja no modo de depuração ou promíscuo.

O “ifstatus” detecta *sniffers* no *host* local. Depois de descarregá-lo, descompactá-lo com o comando “tar”, é necessário editar as primeiras linhas operativas do arquivo *Makefile* do “ifstatus”, pois, por padrão, o “ifstatus” é configurado para compilar para o sistema “Solaris” da “SUN”.

Eis a primeira parte de “ifstatus” *Makefile*:

```
# To build “ifstatus”, you need to edit the OSNAME and LIBS variables
# below, as fallows:
#
# Define OSNAME to one of the following:
#
# OSNAME          Operations System
# -------------   ----------------------
# BSD             4.3DSB or similar (try this if your o/s is not listed)
# HPUX            Hewlett-Packard HP-UX 9.0x (may work on 8.0x too)
# SUNOS4          Sun Microsystem SunOS 4.1.x (may work on 4.0.x too)
# SUNOS55         Sun Microsystem SunOS 5.5 (Solaris 2.5)
# SUNOS56         Sun Microsystem SunDS 5.6 (Solaris 2.6)
#
# Define LIBS to one of the following:
#
# OSNAME          LIBS
# -------------   ----------------------
# BSD             (empty)
# HPUX            (empty)
# SUNOS4          (empty)
# SUNOS55         -lkvm –left –lnsl –lsocket
# SUNOS56         -lkvm –left –lnsl –lsocket
#
OSNAME=       SUNOS55
LIBS=              -lkvm –left –lnsl –lsocket
```

Exemplo 4 – 1: Exemplo do arquivo *Makefile.*

É necessário alterar as seguintes linhas

```
OSNAME=              SUNOS55
LIBS=              -lkvm –left –lnsl –lsocket
```

Para isto:

```
OSNAME=              BSD
#LIBS=              -lkvm –left –lnsl –lsocket
```

É interessante observar que a linha LIBS não foi alterada, mas sim comentada, isto é, foi colocado o símbolo "#" na frente da linha.

Depois de feitas as alterações, é só salvar as modificações e rodar o comando *make* no *prompt*. Com isso, o "ifstatus" irá compilar sem problemas.

Para um exemplo de uso do "ifstatus" foi executado o "Linux_sniffer" (utilitário de *sniffer* pra Linux) no *host* local e foi chamado o comando "ifstatus".

Eis a saída do "ifstatus":

```
WARNING:  LINUX2.SAMSHACKER.NET INTERFACE eth0 IS IN PROMISCUO MODE.
```

Tanto o "ifconfig" quanto o "ifstatus" são bons para detectar *sniffers* em um *host* local. Mas se a rede for muito grande, além de verificar cada máquina individualmente, será necessário uma ferramenta para detectar *sniffers* através de uma sub-rede. Existem ferramentas que foram projetadas especificamente para esse propósito e o mais conhecido é o "NEPED".

4.1.3 NEPED: *NETWORK PROMISCUOUS ETHERNET DETECTOR*

NEPED pode detectar atividade de *sniffer* em uma sub-rede.

A ferramenta NEPED varrerá a sub-rede, procurando interfaces no modo promíscuo. Em *kernels* do Linux anteriores à versão 2.0.36, "NEPED" forja essas interfaces explorando um defeito na implementação de "arp" no Linux. "NEPED" envia uma solicitação de "arp" e extrai uma resposta da estação de trabalho de *sniffing*.

Infelizmente, NEPED tem limitações. Primeiro, em *kernels* posteriores, implementações de "arp" do Linux foram corrigidas, então as estações de trabalho de *sniffing* não mais responderão a solicitação de "arp" errantes. Além disso, é possível configurar o sistema para ignorar solicitações de "arp" e, nesse estado, ele escapará de uma varredura de "NEPED". Apesar de limitações, o "NEPED" é uma ferramenta bem útil.

4.1.4 OUTRAS DEFESAS MAIS GENÉRICAS CONTRA *SNIFFERS*

Localizar um *sniffer* em uma rede é muito difícil. Não é necessário esperar um ataque de *sniffer* para combatê-los. De fato, pode-se tomar uma medida preventiva muito eficaz desde o início, quando se estabelece uma rede: empregar criptografia.

Sessões criptográficas reduzem significativamente seu o risco de *sniffer*. Em vez de preocupar-se com dados sofrendo ataques de *sniffers*, pode-se simplesmente embaralha-os além do reconhecimento. As vantagens dessa abordagem são obvias: mesmo se o invasor capturar os dados, eles serão inúteis para ele. Entretanto, há desvantagens nessa abordagem também.

Os usuários podem resistir a utilizar criptografia. Eles podem achá-la muito incômoda. Por exemplo, é difícil fazer os usuários utilizarem S/Key (um tipo de sistema de senha) toda vez que eles efetuem *login* no servidor. É necessário com isso, localizar um meio termo, utilizando aplicativos que suportem criptografia forte e que também sejam amigáveis ao usuário.

4.2 FERRAMENTAS PARA CONTROLE DE *SCANNERS*

4.2.1 DEFENDENDO-SE CONTRA ATAQUES DE *SCANNER*

Os *scanners* são altamente benéficos quando eles estão em boas mãos. Entretanto, qualquer um pode obtê-los, incluindo *crackers*. E, enquanto os *scanners* não dão acesso imediato aos invasores ao servidor, sua mera existência já inspira preocupação.

Os *scanners* coletam importantes informações do servidor. Só por essa razão, deve-se se familiarizar com a detecção de *scanner*.

4.2.2 COURTNEY (DETECTOR DE *SCANNERS* SAINT E SATAN)

Requitos: Perl 5 +, tcpdump, libpcap-0.0

O “courtney” é um script escrito em “perl” que, em conjunção com tcpdump, detecta varredura de “SAINT” e “SATAN”. Ele registra em *logs* os avisos no formato “syslogd ALERT” padrão e a notificação é visível em /var/log/messages.

Para instalar a ferramenta “courtney”, é só descompactar o arquivo com o comando “tar” em um *prompt*. “courtney” irá descompactar os arquivos em courtney-1.x/, que deve conter os seguintes arquivos:

DISCLAIMER
INSTALL
README
Courtney.pl

Para executar “courtney”, é preciso emitir o seguinte comando:

```
$ courtney.pl &
```

Será mostrada a seguinte mensagem:

```
Tcpdump: listening on eth0
```

Para esse exemplo, executou-se o “courtney” em uma máquina com nome Linux2, ele iniciou uma varredura de “SAINT”. À medida que a varredura progredia, “courtney” registrava a atividade. É exibido o trecho de /var/log/messages no Linux2 (a maquina da vítima):

```
may 30 23:51:57 Linux2 syslog: error: cannot execute
/usr/sbin/ipop3d: No such file or directory
may 30 23:51:57 Linux2 root: courtney[6197]: NORMAL_ATTACK
from gns –target Linux2.example.net
```

Exemplo 4 – 2: Exemplo de um *log* gerado pela ferramenta “courtney”.

A abordagem do “courtney” é simples e direta. Entretanto, ele oferece várias opções de linha de comando para personalização.

Várias opções de linha de comando “courtney”	
Opção	Propósito
-c	Essa opção adiciona somente saída local de STDOUT de *hostname* de ataque.
-d	Essa opção inicializa a depuração.
-i [interface]	Essa opção altera o tipo de interface em que tcpdump executa.
-l	Essa opção desativa o registro de syslog.
-m [usuário@host]	Essa opção faz com com o “courtney” remeta os resultados para usuário@host.

Tabela 4 –1: Opções da Ferramenta “courtney”

4.2.3 ICMPINFO (DETECTOR DE VARREDURA/BOMBA ICMP)

O “icmpinfo” detecta atividade de ICMP suspeita, como bombas e varreduras. Para utilizá-lo é só descompactar o arquivo com o comando “tar”. O “icmpinfo” irá descompactar em icmpinfo-1.11, que deve conter os seguintes arquivos:

CHANGES
CHECKSUMS.asc
DOC
LICENSE
Makefile
NocTools.Infos
README
TODO
Defs.h
Err.c
Icmpinfo.c
Icmpinfo.man
Linux_ip_icmp.h
Print.c
Recevping.c

Para instalá-lo é só rodar o comando *make* em um *prompt*.

```
$ make
```

Com isso, o “icmpinfo” está pronto para ser executado. Para esse exemplo, foi executado o comando “icmpinfo” com a opção –vv para capturar *ping* e o tráfego *traceroute*:

```
$ icmpinfo –vv
```

É registrado uma saída em /var/log/messages.

```
may 31 23:45:27 ICMP_Dest_Unreachable[Port] < 172.16.0.2
[Linux2.example.net]
> 172.16.0.2 [Linux2.example.net] sp=34304 dp=33435
seq=0x00140000 sz=36(+20)
may 31 23:45:27 ICMP_Dest_Unreachable[Port] < 172.16.0.2
[Linux2.example.net]
> 172.16.0.2 [Linux2.example.net] sp=34304 dp=33436
seq=0x00140000 sz=36(+20)
may 31 23:45:27 ICMP_Dest_Unreachable[Port] < 172.16.0.2
[Linux2.example.net]
> 172.16.0.2 [Linux2.example.net] sp=34304 dp=33437
```

```
seq=0x00140000 sz=36(+20)
```

Exemplo 4 – 3: Exemplo de um *log* gerado pela ferramenta "IcmpInfo".

O "icmpinfo" observa tanto o tráfego entrante como o tráfego que sai e é completamente configurável.

Várias opções de comando de linha "icmpinfo":

Opção	Propósito
-l	Essa opção envia a saída de "icmpinfo" para *logs* de syslog.
-p [port]	Essa opção omite a porta.
-s	Essa opção captura o endereço da interface receptora. Ela é utilizada quando se tem mais de uma interface. Essa recurso ajuda a descobrir qual delas recebeu o quê.
-v	Essa opção captura todo o tráfego ICMP.
-vv	Essa opção captura além do tráfego ICMP, *pings*.

Tabela 4 – 2: Opções da Ferramenta IcmpInfo

4.2.4 KLAXON

O "klaxon" é uma ferramenta sofisticada que detecta varreduras de porta pelo serviço. Ele substitui serviços de TCP e UDP em /etc/inetd.conf, deixando-o assim:

```
rexec       stream      tcp   nowait      root  /etc/local/klaxon klaxon
rexec
link        stream      tcp   nowait      root  /etc/local/klaxon klaxon
link
supdup      stream      tcp   nowait      root  /etc/local/klaxon klaxon
supdup
tcpmux      stream      tcp   nowait      root  /etc/local/klaxon klaxon
tcpmux
```

Exemplo 4 – 4: O arquivo /etc/inted.conf.

O "klaxon" detecta varreduras e registra a atividade em *logs* (os primeiros 128 bytes de cada investigação).

4.3 FERRAMENTAS PARA CONTROLE DE *SPOOFING*

4.3.1 SERVIÇOS VULNERÁVEIS A SPOONFING DE IP

Spoofing de IP afeta somente certas máquinas executando certos serviços. Os serviços conhecidos por serem vulneráveis incluem.

- RPC (*Remote Procedure Call Services*)
- X Window System
- Serviços R

4.3.2 IMPEDINDO ATAQUES DE *SPOOFING* DE IP

A defesa mais segura contra *spoofing* de IP é evitar a utilização do endereço de origem para autenticação. Hoje, não há absolutamente nenhuma razão para essa autenticação porque existem soluções criptográficas adequadas.

Uma posição freqüentemente citada é que se a geração de números de seqüência de TCP fosse fortalecida em todos os sistemas operacionais afetados, talvez as soluções criptográficas fossem desnecessárias.

Infelizmente, essa visão não é realista. Independente de que semente seja utilizada, permanece o fato de que, capturando exemplos de números seqüenciados, os invasores finalmente determinarão o algoritmo básico ou outras informações vitais.

Bons números de seqüência não são um substituto à autenticação criptográfica. No melhor dos casos, são uma medida paliativa. Um interpretador que possa observar as mensagens inicias de uma conexão, pode determinar seu estado de números de seqüência pela personificação dessa conexão.

Por outro lado, se tiver uma razão urgente para não instituir a autenticação criptográfica no nível de sistema, pode-se ainda tomar medidas menos eficientes, mas marginalmente confiáveis, incluindo:

- Configurar a rede (roteador) para rejeitar pacotes da Internet que reivindicam a origem de um endereço local. (Essa regra tem que ser imposta explicitamente. Executar meramente um *firewall* de forma automática não protege a rede contra ataques de *spoofing*. Se for permitido o acesso de endereços internos pela parte externa do *firewall*, a rede ainda estará vulnerável.)

- Se realmente forem autorizadas conexões externas de *hosts* confiáveis, tem-se que ativar sessões de criptografia no roteador. Isso impedirá que os invasores capturem tráfego de rede para amostragem (e os impedem de se autenticarem).

Também é possível detectar *spoofing* por meio de procedimento de registro em *log* (mesmo em tempo real). Fazer uma comparação em conexões entre *hosts* confiáveis é uma boa solução.

4.3.3 IMPEDINDO ATAQUES DE *SPOOFING* DE ARP

Spoofing de “ARP” é uma variação no tema de *spoofing* de IP e explora uma fraqueza semelhante. No “ARP”, a autenticação também está baseada em endereço. A diferença é que o “ARP” confia no endereço de *hardware*.

Há várias maneiras de derrotar *spoofing* de “ARP”, mas a mais eficiente é “gravar” os mapeamentos de endereço. Mas isso é quase inviável, pois é cansativo e demorado.

Muitos sistemas operacionais entretanto têm recursos para tornar as entradas no cachê ARP “estáticas” de modo que elas não expiram a cada poucos minutos. É recomendável a utilização desse recurso pra impedir *spoofing* de “ARP”, mas isso exigirá a atualização do cachê manualmente toda vez que um endereço de *hardware* mudar.

Apesar do tempo extra gasto, o esforço é bastante válido. A maneira mais fácil de configurar tabelas “ARP” estáticas é no roteador. Entretanto, isso também é possível com a utilização do comando “arp”.

4.3.4 ARP: UMA FERRAMENTA PARA MANIPULAR TABELAS DE ROTEAMENTO

A tabela resume opções de linha de comando de “arp” e o que elas fazem:

Opção	Função
-a [hostname]	Especifica um *host* particular que seria consultado.
-d [hostname]	Exclui a entrada para o *host* especificado.
-s [hostname] [tipo_de_endereço]	Especifica o tipo de endereço de *hardware* para o host específico.
-t [tipo]	Especifica o tipo de entrada a ser procurada. Tipos válidos podem ser ether, x25, arcnet e pronet (Proteon ProNET Token Ring).

Tabela 4 – 3: Opções da Ferramenta ARP

4.4 FERRAMENTAS DE *FIREWALL*

4.4.1 PACKET FILTERS

4.4.1.1 *Firewall* IPTABLES

O `iptables` é um *firewall* a nível de pacotes e funciona baseado no endereço/porta de origem/destino do pacote, prioridade, etc. Ele funciona através da comparação de regras para saber se um pacote tem ou não permissão para passar. Em *firewalls* mais restritivos, o pacote é bloqueado e registrado para que o administrador do sistema tenha conhecimento sobre o que está acontecendo em seu sistema.

Ele também pode ser usado para modificar e monitorar o tráfego da rede, fazer NAT (*masquerading, source nat, destination nat*), redirecionamento de pacotes, marcação de pacotes, modificar a prioridade de pacotes que chegam/saem do seu sistema, contagem de bytes, dividir tráfego entre máquinas, criar proteções *anti-spoofing*, contra *syn flood*, DoS.

O tráfego vindo de máquinas desconhecidas da rede pode também ser bloqueado/registrado através do uso de simples regras. As possibilidades oferecidas pelos recursos de filtragem `iptables` como todas as ferramentas UNIX maduras dependem da "imaginação", pois ele garante uma grande flexibilidade na manipulação das regras de acesso ao sistema, precisando apenas conhecer quais interfaces o sistema possui, o que deseja bloquear, o que tem acesso garantido, quais serviços devem estar acessíveis para cada rede, e iniciar a construção do *firewall*.

O `iptables` ainda tem a vantagem de ser modularizável. Funções podem ser adicionadas ao *firewall* ampliando as possibilidades oferecidas. Este é um *firewall* que tem possibilidades de gerenciar tanto a segurança em máquinas isoladas como roteamento em grandes organizações, onde a passagem de tráfego entre redes deve ser minuciosamente controlada.

4.4.1.2 Características do *firewall* iptables

- Especificação de portas/endereço de origem/destino
- Suporte a protocolos TCP/UDP/ICMP (incluindo tipos de mensagens icmp)
- Suporte a interfaces de origem/destino de pacotes
- Manipula serviços de *proxy* na rede
- Tratamento de tráfego dividido em *chains* (para melhor controle do tráfego que entra/sai da máquina e tráfego redirecionado).
- Permite um número ilimitado de regras por *chain*
- Muito rápido, estável e seguro
- Possui mecanismos internos para rejeitar automaticamente pacotes duvidosos ou mal formados.
- Suporte a módulos externos para expansão das funcionalidades padrões oferecidas pelo código de *firewall*
- Suporte completo a roteamento de pacotes, tratadas em uma área diferente de tráfegos padrões.
- Suporte a especificação de tipo de serviço para priorizar o tráfego de determinados tipos de pacotes.
- Permite especificar exceções para as regras ou parte das regras
- Suporte a detecção de fragmentos

- Permite enviar alertas personalizados ao `syslog` sobre o tráfego aceito/bloqueado.
- Redirecionamento de portas
- *Masquerading*
- Suporte a *SNAT* (modificação do endereço de origem das máquinas para um único IP ou faixa de IP's).
- Suporte a *DNAT* (modificação do endereço de destino das máquinas para um único IP ou fixa de IP's)
- Contagem de pacotes que atravessaram uma interface/regra
- Limitação de passagem de pacotes/conferência de regra (muito útil para criar proteções contra, *syn flood*, *ping flood*, DoS, etc).

4.4.1.3 Arquivos de *logs* criados pelo iptables

Todo tráfego que for registrado pelo `iptables` é registrado por padrão no arquivo `/var/log/kern.log`.

4.4.1.4 O que proteger ?

Antes de iniciar a construção do *firewall* é bom pensar nos seguintes pontos:

- Quais serviços precisam ser protegidos;
- Serviços que devem ter acesso garantido a usuários externos e quais serão bloqueados a determinadas máquinas.

É recomendável bloquear o acesso a todas portas menores que 1024 por executarem serviços que rodam com privilégio de usuário `root`, e autorizar somente o acesso às portas que realmente deseja (configuração restritiva nesta faixa de portas).

Serviços com autenticação em texto plano e potencialmente inseguros como *rlogin, telnet, ftp, NFS, DNS, LDAP, SMTP RCP, X-Window* são serviços que devem se ter acesso garantido somente para máquinas/redes que você confia. Estes serviços podem não se só usados para tentativa de acesso ao seu sistema, mas também como forma de atacar outras pessoas aproveitando-se de problemas de configuração.

A configuração do *firewall* ajuda a prevenir isso. Mesmo se um serviço estiver mal configurado e tentando enviar seus pacotes para fora, será impedido. Da mesma forma se uma máquina *Windows* da rede for infectada por um *trojan* não haverá pânico: o *firewall* poderá estar configurado para bloquear qualquer tentativa de conexão vinda da internet (*cracker*) para as máquinas da rede.

- Que máquinas terão acesso livre e quais serão restritas.
- Que serviços deverão ter prioridade no processamento.
- Que máquinas/redes não deverão ter acesso a certas/todas máquinas.
- O volume de tráfego que o servidor manipulará. Através disso você pode ter que balancear o tráfego entre outras máquinas, configurar proteções contra DoS, *syn flood*, etc. 0
- O que tem permissão para passar de uma rede para outra (em máquinas que atuam como roteadores/*gateways* de uma rede interna).

4.4.1.5 O que são regras?

As regras são como comandos passados ao `iptables` para que ele realize uma determinada ação (como bloquear ou deixar passar um pacote) de acordo com o endereço/porta

de origem/destino, interface de origem/destino, etc. As regras são armazenadas dentro dos *chains* e processadas na ordem que são inseridas.

As regras são armazenadas no *kernel*, o que significa que, quando o computador for reiniciado tudo o que fez será perdido. Por este motivo, elas deverão ser gravadas em um arquivo para serem carregadas a cada inicialização.

Um exemplo de regra: `iptables -A INPUT -s 123.123.123.1 -j DROP`.

4.4.1.6 O que são *chains*?

Os *chains* são locais onde as regras do *firewall* definidas pelo usuário são armazenadas para operação do *firewall*. Existem dois tipos de *chains*: os embutidos (como os *chains INPUT*, *OUTPUT* e *FORWARD*) e os criados pelo usuário. Os nomes dos *chains* embutidos devem ser especificados sempre em maiúsculas (os nomes dos *chains* são case-sensitive, ou seja, o *chain* `input` é completamente diferente de `INPUT`).

4.4.1.7 O que são tabelas?

Tabelas são os locais usados para armazenar os *chains* e conjunto de regras de um mesmo conjunto que cada um possui. As tabelas podem ser referenciadas com a opção *-t tabela* e existem 3 tabelas disponíveis no `iptables`:

- `filter` - Esta é a tabela padrão, contém 3 *chains* padrões:
    1. `INPUT` - Consultado para dados que chegam a máquina
    2. `OUTPUT` - Consultado para dados que saem da máquina
    3. `FORWARD` - Consultado para dados que são redirecionados para outra interface de rede ou outra máquina.

Os *chains INPUT* e *OUTPUT* somente são atravessados por conexões indo/se originando de localhost. **OBS**: Para conexões locais, somente os *chains INPUT* e *OUTPUT* são consultados na tabela filter.

- `nat` - Usada para dados que geram outra conexão (*masquerading, source nat, destination nat, port forwarding, proxy* transparente são alguns exemplos). Possui 3 *chains* padrões:

1. `PREROUTING` - Consultado quando os pacotes precisam ser modificados logo que chegam. É o *chain* ideal para realização de DNAT e redirecionamento de portas.

2. `OUTPUT` - Consultado quando os pacotes gerados localmente precisam ser modificados antes de serem roteados. Este *chain* somente é consultado para conexões que se originam de IPs de interfaces locais.

3. `POSTROUTING` - Consultado quando os pacotes precisam ser modificados após o tratamento de roteamento. É o *chain* ideal para realização de SNAT e IP *Masquerading*.

- `mangle` - Utilizada para alterações especiais de pacotes (como modificar o tipo de serviço). Possui 2 *chains* padrões:

1. `PREROUTING` - Consultado quando os pacotes precisam ser modificados logo que chegam.

2. `OUTPUT` - Consultado quando pacotes gerados localmente precisam ser modificados antes de serem roteados.

4.4.1.8 Habilitando o suporte ao iptables no *kernel*

Para usar toda a funcionalidade do *firewall* `iptables`, permitindo fazer o controle do que tem ou não permissão de acessar sua máquina, fazer Masquerading/NAT na rede, serão necessários os seguintes componentes compilados no *kernel* (os módulos experimentais fora ignorados intencionalmente):

```
     *
     * Network Options:
     *

     Network packet filtering (replaces ipchains) [Y/m/n/?]
      Network packet filtering debugging [Y/m/n/?]
```

e na Subseção:

```
     *
     *   IP: Netfilter Configuration
     *
     Connection tracking (required for masq/NAT) (CONFIG_IP_NF_CONNTRACK)
[M/n/y/?]
       FTP protocol support (CONFIG_IP_NF_FTP) [M/n/?]
       IRC protocol support (CONFIG_IP_NF_IRC) [M/n/?]
     IP tables support (required for filtering/masq/NAT) (CON-
FIG_IP_NF_IPTABLES) [Y/m/n/?]
       limit match support (CONFIG_IP_NF_MATCH_LIMIT) [Y/m/n/?]
       MAC address match support (CONFIG_IP_NF_MATCH_MAC) [M/n/y/?]
       netfilter MARK match support (CONFIG_IP_NF_MATCH_MARK) [M/n/y/?]
       Multiple port match support (CONFIG_IP_NF_MATCH_MULTIPORT) [M/n/y/?]
       TOS match support (CONFIG_IP_NF_MATCH_TOS) [M/n/y/?]
       LENGTH match support (CONFIG_IP_NF_MATCH_LENGTH) [M/n/y/?]
       TTL match support (CONFIG_IP_NF_TTL) [M/n/y/?]
       tcpmss match support (CONFIG_IP_NF_MATCH_TCPMSS) [M/n/y/?]
       Connection state match support (CONFIG_IP_NF_MATCH_STATE) [M/n/?]
       Packet filtering (CONFIG_IP_NF_FILTER) [M/n/y/?]
         REJECT target support (CONFIG_IP_NF_TARGET_REJECT) [M/n/?]
       Full NAT (CONFIG_IP_NF_NAT) [M/n/?]
         MASQUERADE target support (CONFIG_IP_NF_TARGET_MASQUERADE) [M/n/?]
         REDIRECT target support (CONFIG_IP_NF_TARGET_REDIRECT) [M/n/?]
       Packet mangling (CONFIG_IP_NF_MANGLE) [M/n/y/?]
         TOS target support (CONFIG_IP_NF_TARGET_TOS) [M/n/?]
         MARK target support (CONFIG_IP_NF_TARGET_MARK) [M/n/?]
       LOG target support (CONFIG_IP_NF_TARGET_LOG) [M/n/y/?]
       TCPMSS target support (CONFIG_IP_NF_TARGET_TCPMSS) [M/n/y/?]
```

Exemplo 4 – 5: Configuração do "IPTABLES" no *Kernel*.

Esta configuração permite que não se tenha problemas para iniciar o uso e configuração do seu *firewall iptables*. Ela ativa os módulos necessários para utilização de todos os recursos do *firewall iptables*.

4.4.1.9 Regras do IPTABLES

A tabela a seguir mostra as principais regras do *Firewall* IPTABLES.

Regra	Função
-A	Usada para adicionar novas regras no final do *chain*
-L	Usada para listar as regras existente
-D	Usada para apagar uma regra existente
-I	Usada para inserir uma nova regra. Deve-se entra com o número da regra.
-R	Usada para substituir uma regra existente. Deve-se entrar com o número da regra que deseja-se substituir
-N	Usada para criar um novo *chain*
-E	Usada para renomear uma *chain* existente
-F	Usada para limpar todas as regras de um *chain* criado por um usuário
-X	Usada para apagar uma *chain* criado por um usuário.
-Z	Usada para zerar os contadores de bytes. Estes campos podem ser visualizados com o comando –L.
-P	Especifica um policiamento padrão de um *chain*.
-s	Especifica um endereço de origem.
-d	Especifica d% endereço de destino.
-o	Especifica uma interface de saída (out)
-i	Especifica uma interface de entrada (in)
--sport	Especifica uma porta ou faixas de portas de origem
--dport	Especifica um porta ou faixa de portas de destino

Tabela 4 – 4: Opções da Ferramenta IPTABLES

4.5 VERIFICAÇÃO DE SEGURANÇA NOS SERVIÇOS *FTP, SMTP, TELNET, HTTP*

4.5.1 FTP

O “ftpd” oferece recursos marginais de segurança incluindo controle de acesso de rede baseado no *host* e no usuário. Pode-se implementar esses recursos utilizando três arquivos.

- /etc/ftpusers
- /etc/ftphosts
- /etc/ftpaccess

Será examinado cada um deles.

4.5.1.1 ftpusers: os usuários restringidos acessam arquivos

O “ftpusers” é o arquivo de acesso de usuários restritos. Qualquer usuário cujo nome apareça aqui tem acesso de *login* de FTP negado.

Exemplo de um arquivo /etc/ftpusers com o comando “more”.

```
$ more /etc/ftpusers
root
```

```
bin
daemon
adm
lp
sync
shutdown
halt
mail
news
uucp
operator
games
nobody
```

Exemplo 4 – 6: O arquivo /etc/ftpusers.

Por padrão todos os *logins* de sistemas devem estar desativados.

Para negar completamente o acesso de FTP a um usuário, basta inserir o *login* do usuário em /etc/ftpusers. Cada nome deve entrar em uma linha separada.

4.5.1.2 ftphosts

O “ftphosts” é arquivo de acesso individual de usuário/host do ftpd.

O arquivo “ftphosts” é utilizado para permitir ou negar acesso para certas contas a partir de usuários *hosts.*

Se for feita uma consulta em /etc/ftphosts com o comando “more”, de provavelmente estará vazio ou estará assim:

```
$ more /etc/ftphosts
# Example host access file
#
# Everything after a ´#´ is treated as comment,
# empty lines are ignored
```

Exemplo 4 –7: O arquivo /etc/ftphosts.

Para especificar uma regra para permitir ou negar usuários específicos a partir de *hosts* específicos, utiliza-se a seguinte sintaxe.

```
allow [username] [host or host pattern] [host ou host pattern]
deny [username] [host or host pattern] [host ou host pattern]
```

Tomando como exemplo que um determinado administrador de redes deseja negar o acesso ao usuário “mwagner” a partir de “empresa.com.br” mas permitir acesso do usuário “jspart” a partir de “empresa2.com.br”. Tem-se a seguinte diretiva:

```
# Everything after a ´#´ is treated as comment,
# empty lines are ignored
deny  mwagner     empresa1.com.br
allow jsprat            empresa2.com.br
```

Nesse caso, uma vez que os dois usuários estão vindo de redes diferentes, não é necessário preocupar-se com a ordem de *allow*/*deny*. O “ftpd” processa as diretivas *deny* e *alow* seqüencialmente e não localiza nenhuma contradição entre elas.

Entretanto, tomando como exemplo que um administrador de rede queira negar todo o acesso ao usuário “jsprat” em “empresa2.com.br” exceto a partir de “rh.empresa2.com.br”. Então, torna-se importante a ordem de *allow*/*deny*. Tem-se a seguinte diretiva:

```
deny  jsprat            *.empresa2.com.br
allow jsprat            rh.empresa2.com.br
```

Nesse ponto, “jsprat” seria absolutamente incapaz de efetuar *login* uma vez que “ftpd” processaria e aceitaria primeiro a diretiva *deny* (e daria a ela precedência sobre a diretiva *allow*). Portanto, para a regra funcionar corretamente, seria necessário inverter a ordem de *allow*/*deny*:

```
allow jsprat            rh.empresa2.com.br
deny  jsprat            *.empresa2.com.br
```

Com isso, “ftpd” processaria a diretiva *allow* primeiro e assim o usuário “jsprat” poderia efetuar *login* a partir de “rh”.

O argumento [host or host pattern] [host ou host pattern] pode ser um *hostname,* um endereço de IP. Por exemplo, todas as entradas a seguir são válidas:

```
gerencia.empresa.com.br
*.empresa.com.br
192.168.1.21
```

Adicionalmente, pode-se empilhar *hosts* e padrões de *hosts* separando-os com espaços em branco. Portanto, todas as entradas a seguir também são válidas:

```
gerencia.empresa.com.br rh.empresa.com.br
*.empresa.com.br  *.empresa2.com.br
192.168.1.* *.adm.empresa1.com.br
```

4.5.1.3 ftpaccess: o arquivo de configuração de ftpd

O "ftpaccess" é arquivo de configuração de núcleo do "ftpd". Por meio das diretivas nesse arquivo, pode-se controlar o funcionamento do "ftpd".

O seguinte é um exemplo do "ftpaccess".

```
$ more /etc/ftpaccess
class all   real,guest,anonymous          *
email root@localhost
loginfails 5
readme            README*     login
readme            README*     cwd=*
message     /home/ftp/welcome.msg   login
message     /home/ftp/.message      cwd=*
compress          yes               all
tar               yes               all
chmod             no                guest,anonymous
delete            no                guest,anonymous
overwrite         no                guest,anonymous
rename            no                guest,anonymous
log transfers anonymous,real  inbound,outbound
shutdown /etc/shutmsg
passwd-check      rfc822 warn
```

Exemplo 4 – 8: O arquivo /etc/ftpaccess.

Cada linha inicia com uma diretiva e termina com várias opções. A tabela resume diretivas relacionadas com a segurança de "ftpaccess".

Comando	Resultado
autogroup [grupo|classe]	Atribui dinamicamente direitos especiais de grupo e proprietário para selecionar usuários que sejam membros de uma classe predefinida.
banner [caminho]	Especifica o caminho para uma mensagem de informações. Essa mensagem informacional exibirá quando usuários se conectarem.
chmod [yes|no][tipo]	Especifica se usuários que pertencem a um tipo particular podem executar chmod no servidor.
class [classe|tipo|endereço]	Especifica classes especiais de usuários. Pode-se utilizar essas classes (em conjunto com a diretiva auto-group) para permitir aos membros de classes direitos e privilégios adicionais. Uma completa definição de classe consiste em pelo menos três partes: rótulo da classe (nome da classe em particular), o tipo da classe (anonymous, guest e assim por diante) e o endereço IP.
delete [yes|no] [tipo]	Especifica se usuários que pertencem a um tipo particular podem executar delete no servidor.
deny [endereço] [mensagem]	Essa diretiva é usada para definir *hosts* a partir dos quais ftpd não aceitará conexões. Uma definição completa de *deny* consiste na diretiva *deny*, os endereços indesejáveis e uma mensagem que será exibida para *hosts* que têm acesso negado.
email [nome-do-usuário]	Especifica o endereço de *e-mail* do mantenedor do FTP.
guestgroup [nome-do-grupo]	Restringe usuários reais ao FTP no estilo anônimo. Isto é, quando um logon é efetuado, nenhum usuário pode mudar para o diretório acima da árvore de diretórios de FTP público.
limit [classe|N|tempo|msg]	Limita classe de usuários particulares a N usuários em certos momentos (e especifica uma mensagem para exibir para clientes entrantes quando esse limite foi alcançado)
log commands [tipo]	Especifica que ftpd deve registrar em *log* todos os comandos de usuários de *tipo*.
log transfers [tipo]	Especifica que ftpd deve registrar em *log* todas as transferências feitas por usuários.
loginfails [N]	Especifica quantas vezes um usuário suces-

	sivamente pode ter *logins* mal sucedidos antes de ftpd enviar uma mensagem para *log*.
message [caminho|quando]	Especifica um caminho para uma mensagem informativa que será impressa, depois que os usuários efetuarem logon.
overwrite [yes|no] [type]	Especifica se os usuários que pertencem a um tipo particular podem sobrescreve arquivos.
passwd-check [opções]	Especifica o nível em que ftpd deve verificar as senhas.
private [yes]	Permite que usuários obtenham acesso aprimorado ou aumentado depois que eles efetuam logon emitindo valores USER e PASS adicionais.
rename [yes|no] [tipo]	Especifica se usuários que pertencem a um tipo particular podem executar rename no servidor.
umask [yes|no] [tipo]	Especificar os usuários que pertencem a um tipo particular podem executar umask no servidor.
upload [dir] [opções]	Especificar árvores de diretório em que usuários não podem carregar arquivos.

Tabela 4 – 5: Opções do arquivo /etc/ftpaccess

É apresentado novamente o arquivo /etc/ftpaccess

```
$ more /etc/ftpaccess
class all   real,guest,anonymous         *
email root@localhost
loginfails 5
readme            README*     login
readme            README*     cwd=*
message     /home/ftp/welcome.msg   login
message     /home/ftp/.message      cwd=*
compress          yes               all
tar               yes               all
chmod             no                guest,anonymous
delete            no                guest,anonymous
overwrite         no                guest,anonymous
rename            no                guest,anonymous
log transfers anonymous,real  inbound,outbound
shutdown /etc/shutmsg
passwd-check      rfc822 warn
```

Exemplo 4 – 9: O arquivo /etc/ftpaccess. (2)

Pode-se ver que os membros da classe *guest* e *anonymous* não podem dar *chmod*, excluir, sobrescrever ou renomear arquivos:

```
chmod           no              guest,anonymous
delete          no              guest,anonymous
overwrite       no              guest,anonymous
rename          no              guest,anonymous
```

Além disso, transferências feitas para os usuários da classe real e *anonymous* são registrados em *log* em ambas as direções.

```
log transfers anonymous,real  inbound,outbound
```

4.5.2 SMTP

4.5.2.1 Proteção do serviço sendmail

Embora os ataques de “sendmail” possam ameaçar um servidor, aqueles verdadeiramente eficientes raramente surgem. A melhor defesa contra tais ataques é permanecer atualizado. Além disso, não há passos genéricos que se possam tomar para se proteger contra eles. Entretanto, há vários passos que se pode tomar para proteger os serviços de “sendmail”.

4.5.2.2 Protegendo-se contra transmissão não-autorizada

Transmissão não-autorizada (*unauthorized relaying*) é um problema irritante, especialmente em instalações mais antigas do Linux. (As versões do “sendmail” anteriores a 8.9 têm transmissão ativada por padrão). Por outro lado, as distribuições atuais de Linux incluem “sendmail” 8.9.x.

Versões do “sendmail” 8.9.x, podem facilmente ser configuradas para transmitir mensagens somente para *hosts* autorizados. Naturalmente, não se sabe exatamente para que servem as transmissões, mas elas são exigidas. Por exemplo, um gerenciador de intranets para uma empresa relativamente grande com várias redes. E possível que se vá querer controlar correio eletrônico de um único servidor. Para realizar isso é necessário configurar transmissões.

Suponha-se que uma organização controla “ourtoys.com”, “ourgames.com” e “ourweapons.com”. Para servir correio para os três, será necessário uma configuração de servidor que possa transmitir para todos os três.

Configura-se a transmissão editando o arquivo /etc/mail/access para incluir todos esses domínios participantes. Dependendo da distribuição do Linux, esse arquivo pode ter o nome um pouco diferente.

Eis um exemplo:

```
# Check the /usr/doc/sendmail-8.9.3/README.cf file for a description
# of the formatof this file. (search for access_db in that file)
# The /usr/doc/sendmail-9.8.3/README.cf is part of the sendmail-doc
# pachage
#
# by default we allow relaying from localhost…
localhost.localdomain   RELAY
localhost               RELAY
ourtoys.com             RELAY
ourgames.com            RELAY
ourweapons.com          RELAY
```

Exemplo 4 – 10: Configuração do arquivo /etc/mail/access.

Depois de adicionar os domínios, é necessário reconstruir o arquivo de banco de dados binário que corresponda ao arquivo de texto que acabou de ser alterado. Para fazer isso, basta rodar o comando *make* de /etc/mail em um *prompt*:

```
$ cd /etc/mail
$ make
```

Pode-se também utilizar /etc/mail/access para bloquear correio recebido a partir de um particular nome de domínio, sub-rede ou nome de usuário. Para que isso seja feito, é necessário utilizar um palavra-chave diferente de *RELAY*, dependendo do que se quer. Palavras-chaves válidas são como as que seguem:

REJECT

Essa entrada mais comumente utilizada para bloquear mensagens indesejáveis ou remetentes. A palavra-chave *REJECT* rebaterá a mensagem entrante como sendo impossível de enviar (*undeliverable*).

OK

Se uma entrada é definida como certa, o correio a partir da regra *OK*, permitirá, mesmo que outra regra negá-lo. Por exemplo, se quiser bloquear todo correio entrante do domínio empresa.com.br mas permitir mensagens de uma máquina específica nesse domínio, como rh.empresa.com.br, é estabelecida assim:

```
empresa.com.br    REJECT
rh.empresa.com.br OK
```

Com essa configuração, o servidor rejeitará mensagens de qualquer máquina no domínio “empresa.com.Br” exceto “rh”.

DISCARD

Freqüentemente, não se quer retornar mensagem de erro para um remetente. Por exemplo, um usuário qualquer esteja inundando uma determinada rede, e o administrador não quer que eles saibam por que o servidor está descartando as mensagens deles. Em tais casos, discard todas as mensagens entrantes com o comando *DISCARD*. O resultado é idêntico ao *REJECT*, mas nenhum erro é gerado.

```
### Error Message
```

Para retornar uma mensagem de erro personalizada para as mensagens *reject*, utiliza-se o código de resposta de erro RFC 821 (em geral 550) seguido pelo texto personalizado. Isso é idêntico a palavra-chave *REJECT*, mas permite que seja configurado o próprio código de resposta. Por exemplo:

```
empresa.com.br    550   Mail  from  empresa ins not acceptable.
```

Com esse conjunto de regras, as mensagens “empresa.com.Br” são rejeitadas com a mensagem “Mail from empresa in not acceptable” (“Correio de empresa não é aceitável”)

Pode-se também utilizar um endereço particular de remetente com essas diretivas, em vez de bloquear uma sub-rede inteira. Por exemplo, pode-se querer bloquear correio eletrônico de nome “RH”, utiliza-se então “RH@” como entrada, seguida por uma diretiva apropriada como *REJECT*, *DISCARD* etc.

4.5.2.3 Desativando EXPN e VRFY

Dois comandos SMTP que vazam informações são EXPN (*expand*) e VRFY (*verify*). Os *crackers* utilizam esses comandos pra identificar usuários válidos e expandir lista de distribuição.

Exemplo do EXPN em ação:

```
$ telnet 127.0.0.1
Trying 127.0.0.1...
```

```
Connected to localhost.
Escape character is '^]'.
220 rh.empresa.com.br ESMTP Sendmail 8.9.3/8.9.3;
Sun, 15 Jun 2003 19:35:31
-0400
EXPN
501 Argument requerid
EXPN samplelist
250 <jray@rh.empresa.com.br>
250 <jackd@rh.empresa.com.br>
250 <maddy@rh.empresa.com.br>
quit
221 rh.empresa.com.br closing connection
```

Exemplo 4 – 11: Utilização do comando EXPN.

Uma expansão de lista de exemplos revela três destinatários. Todas são contas válidas em rh.empresa.com.br e agora são potenciais alvos de ataque. Desativar EXPN e VRFY é uma decisão inteligente.

Para fazer isso, edita-se o arquivo /etc/sendmail.cf e adiciona-se algumas diretivas "noexpn" e "novrfy" às configurações *PrivacyOptions*. As opções definidas corretamente devem ser semelhantes a isso no arquivo de configurações:

```
# provacy flags
0 PrivacyOptions=authwarnings,noexpn,novrfy
```

Basta reiniciar o "sendmail" (/etc/rc.d/init.d/sendmail restart) e os comandos EXPN/VRFY serão desativados. É possível testar isso manualmente emitindo *telnet* para o *host* local (127.0.0.1) do servidor de "sendmail".

4.5.3 TELNET

4.5.3.1 Sistema de *telnet* seguros

Existem muitas implementações de "*telnet* seguras" incluindo as seguintes:

- deslogin
- SRA *Telnet* da Texas A&M University

4.5.3.2 deslogin

A ferramenta “deslogin” de David A. Barret fornece um serviço de *login* de rede com autenticação segura (ao contrário de *telnet* ou *rlogin*). Dados em trânsito são criptografados utilizando DES e portanto são blindados contra intercepção eletrônica.

Para intalar o “deslogin” serão necessários dois programas:

- O núcleo “deslogin”,
- A cifra de criptografia.

4.5.3.3 Instalando a distribuição deslogin

Os arquivos utilizados são deslogin-1.3.tar.gz e cipher-3.0.tar.Z

```
$ tar –xzvf cipher-3.0.tar.Z
$ tar –xzvf deslogin-1.3.tar.gz
```

O pacote de cifra descompactará cipher-3.0/, e “deslogin” descompactará deslogin-1.3/.

4.5.3.4 Instalando o pacote cipher

Em seguida, altere cipher-3.0/ e construa o pacote de cifra:

```
make
```

Depois de verificar se *make* foi bem-sucedido (nenhum erro além dos avisos sobre chaves de 16 bits), instale o pacote:

```
/cipher-3.0 $ make install
cp cipher /usr/local/bin
ln /usr/local/bin/cipher /usr/local/bin/decipher
cp cipher.1 /usr/local/man/man1
cp btoa atob /usr/local/bin
cp btoa.1 /usr/local/man/man1
ln /usr/local/man/man1/btoa.1 /usr/local/man/man1/atob.1
```

4.5.3.5 Instalando o componente deslogin

Mude para o deslogin-1.3/ e abra *Makefile* para edição. Aqui, será necessário configurar *Makefile* para o próprio sistema. Por exemplo, muda-se a linha 50 para refletir seu *Shell*. Por padrão, a linha 50 é

```
SHELL=/bin/sh
```

Em seguida, pode-se querer alterar linhas 92, 93 e 94 para especificar arquivos e diretórios de *log* alternativos. As configurações-padrão são:

```
USER_FILE=\"/usr/local/etc/deslogin.users\"
LOG_FILE=\"/usr/adm/deslogin.log\"
GW_LOG_FILE=\"/usr/adm/deslogingw.log\"
```

É necessário remover o caractere de comentário das linhas 271-274 de modo que "deslogin" construa utilizando Linux gcc. As linhas, cujos caracteres de comentário devem ser removidos, são semelhantes às seguintes:

```
# CC              = gcc –ansi
# CFLAGS          = $ (DEBUG) –DLinux –D_LINUX_SOURCE
# LDFLAGS         = $ (DEBUG)
# NSTCFLAGS = $ (DEBUG) –DLinux
```

Agora pode-se fazer o pacote “deslogin”. Basta executar o comando *make*:

```
make
```

Essa compilação morrerá com o seguinte erro:

```
make:***No rule to make target ‘desblock.o’, needed by ‘deslogin’.
Stop
```

Basta ignorar o erro e executar um *make* novamente:

```
make
```

O “deslogin” agora construirá e solicitará uma frase de senha de criptografia.

```
Input Default UserFile PassPhrase:
```

Depois de inserir a frase de senha, o comando *make* terminará. Nesse ponto, basta instalar o pacote:

```
make install
```

4.5.3.6 Configuração do deslogin

Antes de utilizar “deslogin”, será necessário estabelecer várias opções de configuração. Primeiro, tem-se que copiar o arquivo netlogind.users de exemplo para /usr/local/etc/.

Exemplo do arquivo:

```
#
# Netlogind user database
#
```

```
# Whitespace separated list of username/passphrase pairs.
# Note that whitespace may appear in the passphrase so it´s last.
# The empty passphrase is all allowed
# Ascii values greater than 127 are illegal.
#
# For added security, this file may be encrypted with the cipher program
# And the same key given to netlogind when it´s invoked interactively.
# In an case, make sure that it´s not readable by other than root and the
# Netlogind program´s owner (group).
#
martha      martha´s passphrase
john        simple, but secure
```

Exemplo 4 – 12: Configuração padrão da ferramenta “deslogin”.

O formato de arquivo é simples, cada linha deve ter um nome de usuário com 8 caracteres (ou menos), uma tabulação e uma frase de senha constituindo em qualquer número de caracteres de 7 bits. (As linhas iniciando com # são comentários).

O “deslogind” (o servidor de “deslogin”) aceita várias opções de linha de comando, que estão resumidas na tabela.

Opção	O que faz
-c	Especifica se “deslogin” deve usar o arquivo-padrão de usuário (e não solicitar uma chave de cifra).
-d	Permite depuração.
-f\fIuserFile	Especifica um arquivo alternativo de usuários (normalmente deslogin.users) ou netlogind.users em /usr/local/etc.
-i\fIinactiveSecs	Especifica o número de segundos em que a sessão pode ficar inativa antes de o servidor derrubar o cliente.
-k	Especifica uma frase que “deslogind” utiliza para decriptar o arquivo de usuário.
-l\fIlogFile	Especifica um arquivo de *log* alternativo. O padrão é /usr/adm/netlogind.log ou /usr/adm/deslogind.
-n	Utilizada quando o arquivo de usuário não estiver criptografado. Essa opção diz ao “deslogind” para não se preocupar em procurar uma chave de cifra.
-p\fIport	Especifica as portas em que houver solicitações.
-t\fIloginSecs	Especifica o número de segundos a esperar por uma resposta depois de um pedido de

	conexão de usuário. Se nenhuma resposta for recebida, o servidor derruba o cliente

Tabela 4 – 6: Opções da Ferramenta deslogin

O "deslogind" não permite que usuários passem variáveis de ambiente para o final do servidor. (Ao contrário de *telnet*, o "deslogin" proíbe ataques de ambiente). Entretanto, "deslogin" configura as seguintes variáveis de ambiente ao efetuar *login*:

- HOME
- LOGNAME
- MAIL
- PATH
- RHOSTNAME (nome de *host* remoto)
- SHELL
- TERM
- TZ
- USER

4.5.3.7 O cliente de deslogin

O cliente de "deslogin" é fácil de utilizar. Emita o comando "deslogin" mais seu nome de usuário, o *hostname* remoto e a porta, como mostrado a seguir:

```
$ deslogin user@Linux6.empresa.com.br:2010
```

Em reposta, o sistema solicitará uma frase de senha:

```
Pass Phrase:
```

E por fim, se inserir a frase de senha correta, “deslogind” irá efetuar o login:

```
Linux6 $
```

4.5.4 HTTP

O “Apache” é o servidor mais popular de “http” do mundo e fornece muitos mecanismos de segurança predefinidos, incluindo:

- Controle de acesso de rede baseado em *host*.
- Controle sobre onde usuários locais podem executar *scripts* CGI.
- Controle sobre como usuários locais podem anular suas configurações.

4.5.4.1 Controlando acesso: access.conf

O “Apache” fornece controle de acesso de rede baseado em *host* via “access.conf”. Dependendo de sua distribuição do Linux, “access.conf” pode ser encontrado em vários diretórios, mas é provável de seja no /etc/apache/.

Eis um exemplo do arquivo “access.conf” padrão de uma instalação padrão:

```
# access.conf: configuração de acesso global Docs On-Line em
# http://www.apache.org
# Este arquivo define configurações de servidor que afetam os tipos de
# serviços que são permitidos, e em que circunstâncias. Cada diretório a
# que o Apache tem acesso pode ser configurado com relação a que serviços
# e recursos são permitidos ou não nesse diretório (e seus
# subdiretórios). Primeiro configuramos o “padrão” para ser um conjunto
# muito restritivo de permissões.

<Directory />
Options None
AllowOverride None
</Directory>

# Note que desse ponto em diante você deve especificamente ativar os
# recursos particulares que serão permitidos – então se algo não estiver
# funcionando como esperava, certifique-se de que você o ativou
# especificamente abaixo.
# Isso deve ser alterado para o que você configurou como DocumentRoot.

<Directory /home/httpd/html>

# Isso também pode ser “None”, “All”, ou qualquer combinação de
# “Indexes”, “Includes”, “FollowSymLinks”, “ExecCGI” ou “MultiViews”.

# Note que “MultiViews” deve ser nomeado explicitamente --- “Options All”
# não fornece isso a você.

# Options Indexes followSymLinks
Options None

# Isso controla quais opçõess os arquivos .htaccess nos diretórios podem
# anular. Também pode ser “All” ou qualquer combinação de “Options”,
# “FileInfo”, “AuthConfig” e “Limit”.

AllowOverride None

# Controla quem pode obter material desse servidor.

Order allow,deny
Allow from all

</Directory>

# /usr/local/etc/httpd/cgi-bin deve ser alterado o nome do diretório onde
# seu ScriptAliased CGI reside, se você configurou isso.

# <Directory /usr/local/etc/httpd/cgi-bin>
<Directory /home/httpd/cgi-bin>
AllowOverride None
# Options None
Options ExecCGI
</Directory>

# Permite relatórios de status do servidor, com o URL de
```

```
# http://servername/server-status
# Mude o “your_domain.com” para corresponder com o domínio que você quer
# ativar.

# <Location /server-status>
# SetHandler server-status

# Order deny,allow
# Deny from all
# Allow from .your_domain.com
# </Location>

# Há relatos de pessoas tentando abusar de um artigo bug (antes da versão
# 1.1). Esse bug envolvia um script de CGI distribuído como uma parte do
# Apache. Removendo o caractere de comentário para ativar essas linhas,
# você pode redirecionar esses ataques para um script de registro em log
# em phf.apache.org. Ou você pode registra-los por sua própria conta,
# utilizando o script support/phf_abuse_log.cgi.

# <Location /cgi-bin/phf*>
# deny from all
# ErrorDocument 403 http://phf_abuse_log.cgi
# </Location>

# Você pode colocar depois disso qualquer outro diretório ou localização
# sobre os quais você deseja obter informações de acesso.
```

Exemplo 4 – 13: O arquivo /etc/apache/access.conf.

Para estabelecer regrar de controle de acesso de rede, basta concentrar-se nessas diretivas:

```
order allow,deny
allow from all
```

As diretivas oferecem três tipos de controle:

- *allow* – A diretiva *allow* controla que *hosts* (se houver algum) podem co-nectar-se e oferecer escolhas: *all, none,* ou lista (onde lista é uma lista de *hosts* aprovados).

- *deny* – A diretiva *deny* controla que *hosts* (se houver algum) não podem conectar-se e oferecetrês escolhas: *all none*, ou lista (onde lista é uma lista de *hosts* não aprovados).

- order – A diretiva *order* controla a ordem em que as regras *allow*/*deny* são aplicadas e oferece três escolhas: *allow*, *deny*, *allow* ou mutual-failure. (mutual-failure é uma opção especial que especifica que uma conexão deve passar por ambas as regras anteriores, *allow* e *deny*.)

Utilizando essas diretivas em conjunto, pode-se aplicar controle de acesso de várias maneiras:

- Inclusivamente – Nomeia-se explicitamente todos os *hosts* autorizados.
- Exclusivamente – Nomeia-se explicitamente todos os *hosts* não-autorizados.
- Inclusivamente e exclusivamente – Combinação dos dois.

4.5.4.2 Permitindo explicitamente *hosts* autorizados

Suponha-se que um *host* qualquer "Linux1.domain.net", e que seja necessário restringir todo o tráfego externo. Sua seção de controle de acesso poderia ser semelhante a isto:

```
order deny, allow
allow from Linux1.domain.net
deny from all
```

Na avaliação de uma solicitação de conexão, o servidor primeiro processa negações e rejeita todo mundo. Em seguida, ele verifica os *hosts* aprovados e localiza "Linux1.domain.net". Nesse cenário, somente solicitações de conexões provenientes de "Linux1.domain.net" são autorizadas.

Naturalmente, esse cenário também é um pouco restritivo. Provavelmente, pode-se querer permitir que pelo menos algumas máquinas no domínio se conectassem. Neste caso, pode-se tornar as regras levemente mais liberais utilizando uma lista de *hosts*, assim:

```
order deny, allow
allow from Linux1.domain.net Linux2.domain.net Linux3.domain.net
deny from all
```

Nesse novo cenário, não apenas “Linux1.domain.net” pode se conectar, como também “Linux2.domain.net” e “Linux3.domain.net”. Entretanto, outras máquinas no domínio são excluídas. (Por exemplo, o servidor rejeita conexões de “Linux4.domain.net” e “Linux5.domain.net”).

Talvez pretende-se permitir todas as conexões iniciadas do domínio e rejeitar somente aquelas vindas de redes estrangeiras. Para fazer isto, pode-se configurar as diretivas de controle de acesso assim:

```
order deny, allow
allow from domain.net
deny from all
```

Assim, qualquer máquina no domínio “domain.net” pode conectar-se.

4.5.4.3 Bloqueando explicitamente *hosts* indesejáveis

Suponha-se que se queira bloquear conexões de “hackers.domain.net” mas ainda permitir conexões de todo mundo. Basta configurar as diretivas desta forma:

```
order deny,allow
allow from all
deny from hackers.domain.net
```

Isso bloqueia somente “hackers.domain.net” e concederia a outros *hosts* acesso aberto. Naturalmente, na prática isso provavelmente seria uma abordagem não-realista. Pode-se existir contas em outras máquinas dentro de “domain.net”. portanto, talvez seja necessário bloquear o domínio inteiro, dessa forma:

```
order deny, allow
allow from all
deny from domain.net
```

4.6 DEFENDENDO-SE CONTRA ATAQUES DE RECUSA DE SERVIÇO

Não há defesa genérica contra ataques de recusa de serviço. Entretanto, pode-se aumentar significativamente a resistência de uma rede seguindo estes passos:

- Desativar endereçamento de transmissão.
- Filtar tráfego entrante de ICMP, *PING* e UDP.
- Para servidores sem *firewall*, re-configurar o período de tempo limite antes de uma conexão aberta mas não resolvida ser derrubada. (Em geral, esse período de tempo é de aproximadamente alguns minutos a dez segundos). Isso reduzirá os riscos impostos por ataques de fila de conexões, onde *crackers* inundam a fila de conexões do sistema com solicitações de abertura de conexão.
- Se o roteador suportar interpretação TCP, é recomendável sua utilização. A interceptação TCP ocorre quando o roteador intercepta e valida as conexões de TCP. As conexões que não conseguem se converter em um estado estabelecido depois de um tempo razoável são derrubadas. Além disso, as solicitações de conexão de *hosts* inatingíveis são derrubadas. Em ambos os casos, o servidor aceita apenas conexões válidas, completamente abertas. Isso reduzirá ataques de SYN.
- Deixar sempre atualizado o *kernel* com *patches* e atualizações.

- Utilizar filtros de pacotes para descartar endereços de origem suspeitos (uma defesa comum contra *spoofing*). Por exemplo, a rede nunca deve aceitar pacotes vindos da internet que reivindicam ser originários de dentro da rede.

4.7 REALIZAÇÃO DE AUDITORIA EM ARQUIVOS DE *LOGS*

4.7.1 O SISTEMA E MENSAGENS DE *KERNEL*

O sistema e mensagens de *kernel* são tratados por dois *daemons*:

"syslogd": registra o tipo de conexão que muitos programas utilizam. Valores típicos que "syslogd" intercepta incluem o nome de programa, tipo de recurso, prioridade e mensagem de *log* de estoque.

"klogd": intercepta e registra em *logs* as mensagens de *kernel*.

O arquivo de saída com ações de "syslogd" e "klogd" é o /var/log/messages.

4.7.1.1 /var/log/messages: gravando mensagens do sistema e do *kernel*

As mensagens de diagnostico do sistema e do *kernel* aparecem na ordem em que elas são recebidas.

```
Jun 20 21:05:39 orbital kernel: bttv0: Bt878 (rev 2) at 00:09.0, irq: 5,
latency: 64, memory: 0xcddfc000
Jun 20 21:05:39 orbital kernel: bttv0: model: BT878(Prolink PV-BT878P+4E
(P)
[insmod option]
Jun 20 21:05:39 orbital kernel: bttv0: i2c: checking for TDA9875 @ 0xb0...
```

```
not found
Jun 20 21:05:39 orbital kernel: bttv0: i2c: checking for TDA7432 @ 0x8a...
foundJun 20 21:05:39 orbital named[448]: command channel listening on
127.0.0.1#953
Jun 20 21:05:39 orbital named[448]: zone 0.0.127.in-addr.arpa/IN: loaded
serial
1997022700
Jun 20 21:05:39 orbital named[448]: zone localhost/IN: loaded serial 42
Jun 20 21:05:39 orbital named[448]: running
Jun 20 21:05:39 orbital kernel: tvaudio: TV audio decoder + audio/video mux
driver
Jun 20 21:05:39 orbital kernel: tvaudio: known chips:
tda9840,tda9873h,tda9850,tda9855,tea6300,tea6420,tda8425,pic16c54 (PV951)
```

Exemplo 4 – 14: *Log* gerado pelo "klogd".

4.7.1.2 syslog.conf: personalizando o syslog

O arquivo syslog.conf é o arquivo de configuração principal para "syslogd" que registra em *log* mensagens de sistema em sistemas *nix. Esse arquivo especifica regras para registro em *log*.

Em syslog.conf, pode-se definir regras com dois campos:

- O campo *selector*: O que registra em *log*
- O campo *action*: Onde registra em *log*

4.7.1.3 O campo *Selector*

No campo *Selector*, deve-se especificar pelo menos um ou dois valores:

- O *type* de mensagem
- A *priority* de mensagem

O *type* de mensagem é chamado de *facility* (recurso) e deve ser um dos seguintes:

- auth: recurso de segurança que monitora a autenticação de usuário em vários serviços como FTP e *login*. (Essencialmente, o recurso *auth* monitora qualquer ação de usuário que requeira um nome de usuário e senha para efetuar *login* ou utilizar o recurso-alvo.)
- authpriv: É um recurso de segurança que monitora mensagens de segurança/autorização.
- *cron*: É um *daemon* que executa comandos agendados.
- *daemon*: Monitora mensagens adicionais do *daemon* de sistema.
- kern: Rastreia mensagens do *kernel*.
- lpr: Rastreia mensagens do sistema de impressora.
- *mail*: Rastreia mensagens do sistema de correio.
- *news*: Rastreia mensagens do sistema de *news*.
- uucp: Rastreia mensagens do subsistema de cópia de UNIX para UNIX.

Exemplo de uma regra que especifica que o sistema deve enviar todas as mensagens do *kernel* para o console:

```
kern.*                          /dev/console
```

Aqui, a *facility* é *kernel* e a *action* será registrado em /dev/console. Pode-se também registrar todas as mensagens de *kernel* para /var/log/messages:

```
kern.*                          /var/log/messages
```

A segunda metade do campo *Selector* é *Priority*, que não é sempre necessário, a menos que se queira refinar sua saída. *Priority* deve ser um dos seguintes:

alert: Os alertas indicam sérios problemas que demandam atenção imediata.

crit: Mensagens crit (críticas) indicam problemas fatais.

err: Mensagens de erro.

warning: São avisos-padrão (por exemplo, o sistema não pode realizar a tarefa solicitada).

Por exemplo, se for necessário conectar mensagens de erro de um sistema de notícias, deve-se criar uma regra como a seguinte:

```
# Save news errors of level err and higher
# in a special file.
news.err                /var/log/spooler
```

Seus valores são:

- Sua *facility* = *news*
- Sua *priority* = err (mensagens de erro)
- Sua *action* = registra estas em /var/log/spooler

4.7.1.4 O campo de ação

No campo de ação, especifica-se o que “syslog” deve fazer com as mensagens solicitadas. Uma possível escolha é registrar em *logs* as mensagens em um arquivo particular. Outras escolhas incluem as seguintes:

- O terminal ou console
- Uma máquina remota (se ela estiver executando “syslogd”)
- Usuários específicos
- Todos os usuários

Por exemplo, se for necessário enviar as mensagens de *kernel* para um *host* remoto Linux3 (“syslogd” em execução). Pode-se criar uma regra como a seguinte:

```
kern.*                          @Linux3
```

O syslog.conf fornecido com o Linux oferece várias possibilidades predefinidas:

```
# more /etc/syslog.conf

# /etc/syslog.conf
# For info about the format of this file, see "man syslog.conf"
# and /usr/doc/sysklogd/README.Linux.

# Uncomment this to see kernel messages on the console.
#kern.*                                         /dev/console

# Log anything 'info' or higher, but lower than 'warn'.
# Exclude authpriv, cron, mail, and news.  These are logged elsewhere.
*.info;*.!warn;\
      authpriv.none;cron.none;mail.none;news.none    /var/log/messages

# Log anything 'warn' or higher.
# Exclude authpriv, cron, mail, and news.  These are logged elsewhere.
*.warn;\
      authpriv.none;cron.none;mail.none;news.none    /var/log/syslog
```

```
# Debugging information is logged here.
*.=debug                          /var/log/debug

# Private authentication message logging:
authpriv.*                        /var/log/secure

# Cron related logs:
cron.*                            /var/log/cron

# Mail related logs:
mail.*                            /var/log/maillog

# Emergency level messages go to all users:
*.emerg                           *

# This log is for news and uucp errors:
uucp,news.crit                    /var/log/spooler

# Uncomment these if you'd like INN to keep logs on everything.
# You won't need this if you don't run INN (the InterNetNews daemon).
#news.=crit                       /var/log/news/news.crit
#news.=err                        /var/log/news/news.err
#news.notice                      /var/log/news/news.notice
```

Exemplo 4 – 15: O arquivo /etc/sysgd.conf.

5 CONSIDERAÇÕES FINAIS

5.1 CONCLUSÃO

Quando surgiram os primeiros sistemas operacionais não existia necessidade de uma segurança muito rígida. Com o passar do tempo, houve a necessidade de ligar máquinas em rede, assim, a segurança tornou um pré-requisito importantíssimo.

Por razões de segurança a utilização do sistema operacional Linux está cada vez mais sendo utilizado em grandes servidores. Algumas aplicações como, provedores de acesso à internet, servidores em empresas, utilizam o sistema operacional Linux buscando principalmente estabilidade de funcionamento e a segurança que ele fornece. Mesmo utilizando um sistema como o Linux, tem-se a necessidade de configura-lo e atualiza-lo pelo fato de sua configuração padrão não ser totalmente segura.

Para prevenir de certos problemas deve-se manter o sistema operacional atualizado, evitar a ativação de serviços desprotegidos, e, principalmente, manter-se sempre atualizado sobre novas ferramentas de segurança. Isso pode ser feito através de fóruns e listas de discussões sobre Linux.

Este trabalho foi desenvolvido com o intuito de mostrar as principais ferramentas de segurança para um servidor de Intranet Linux, para que se possa evitar invasões e perda de informações sigilosas.

As ferramentas e as técnicas de segurança apresentadas neste trabalho servem como base para uma boa configuração de qualquer servidor intranet que esteja rodando Linux. Além da proteção do sistema, é apresentado neste trabalho formas de verificação de *Logs*, onde pode-se descobrir o autor da tentativa de invasão ou roubo de informação.

5.2 RECOMENDAÇÕES

O trabalho desenvolvido foi voltado para servidores de Intranet, levando-se em conta somente serviços voltados a isso. Seria interessante para um trabalho futuro, explorar também serviços voltados a servidores de Internet, buscando segurança em servidores Apache e uma atenção maior a *Firewall*.

Como o trabalho desenvolvido é voltado explicitamente ao sistema operacional Linux, pode-se realizar basicamente os mesmos passos para sistemas operacionais da Microsoft. Examinar ferramentas e técnicas de segurança para servidores *Windows* e até realizar comparações com as ferramentas e técnicas adotadas neste trabalho.

6 REFERÊNCIAS BIBLIOGRÁFICAS

[ABSOLUTA 2000a] ABSOLUTA. *CHECKLIST* de segurança para sistemas UNIX [*online*]. 12 mar. 1998. Disponível em: http://www.absoluta.org [acessado em 13 mar. 2003].

[ABSOLUTA 2000b] ABSOLUTA. *VIRTUAL Private Networks* (VPNs) [*online*]. 12 mar. 1998. Disponível em: http://www.absoluta.org/seguranca/seg_vpn.htm [acessado em 13 mar. 2003].

[ANDRADE 1996] ANDRADE, Lenimar N. Os modos de Permissão no Unix [*online*]. Jul. 1996. Disponível em: http://www.dicas-L.unicamp.br/FAG.html [acessado em 10 abr. 2003].

[ANÔNIMO 2000] ANÔNIMO. Segurança Máxima para Linux. Campus, 2000.

[BAUER 2002] BAUER, Michael D.. *Building Secure Servers with Linux*. Paperback, 2002.

[CISNEIROS 1998] CISNEIROS, Hugo. *The Linux Manual* [*online*]. 1998. Disponível em http://www.netdados.com.br/tlm/ [acessado em 21 abr. 2003].

[DANESH 2000] DANESH, Arman. Dominando o Linux Red Hat 6.0 "A Bíblia". Makron Books, 2000.

[GERLACH 1999] GERLACH, Cristiano. Técnicas adotadas por *Crackers* para entrar em redes corporativas e redes privadas [*online*]. 1999. Disponível em: http://www.rnp.br/ [acessado em 8 mai. 2003].

[SIDDIQUI 2002] SIDDIQUI, Shadab. *Linux Security.* Paperback, 2002.

[SLOHA 1999] SLOHA, Liliana Esther Velásquez Alegre. Os *logs* como ferramenta de detecção de intrusão [*online*]. 14 mai. 1999. Disponível em: http://www.rnp.br/ [acessado em 8 mai. 2003].

[UNICAMP 2000] UNICAMP, RESUMO das funcionalidades de alguns *softwares* [*online*]. 2000. UNICAMP – Equipe de segurança em Sistemas e Redes. Disponível em: http://www.security.unicamp.br/docs/software/ferramentas/index.html [acessado em 8 mai. 2003].

[WRESKI 2000a] WRESKI, Davis. *Linux Security Quick Reference Guide* [*online*]. 2000. Disponível em: http://www.Linuxsecurity.com/docs/QuickRefCard.pdf [acessado em 9 mai. 2003].

[WRESKI 2000b] WRESKI, Davis. *Linux Security Administrator´s Guide* [*online*]. 2000. Disponível em:

http://www.Linuxsecurity.com/docs/SecurityAdminGuide/SecurityAdminGuide.html [acessado em 9 mai. 2003].

[WRESKI 2000c] WRESKI, Davis. *Linux Security HOWTO* [*online*]. 2000. Disponível em:

http://www.Linuxsecurity.com/docs/LDP/Security-HOWTO.html [acessado em 9 mai. 2003].

7 ANEXO A - CONFIGURAÇÃO DE UM SERVIDOR LINUX PARA INTRANETS

Desabilitando todos os serviços não utilizados

Inicialmente deve-se desabilitar todos os serviços não utilizados. Basta editar o arquivo "/etc/inetd.conf" e desabilitar o que não é necessário comentando os serviços (adicionando o caractere # no início da linha referente ao mesmo), e após, enviando um sinal SIGHUP ao inetd para atualizá-lo com o arquivo "inetd.conf" alterado.

Primeiro Passo

Alterar as permissões do arquivo "/etc/inetd.conf" para 600, assim apenas o usuário root terá permissões de leitura ou escrita no mesmo.

```
[root@localhost]# chmod 600 /etc/inetd.conf
```

Segundo Passo

Certificar que o dono do arquivo "/etc/inetd.conf" seja o root.

```
[root@localhost]# ls -l /etc/inetd.conf
```

Terceiro Passo

Editar o arquivo inetd.conf (vi /etc/inetd.conf) e desabilitar serviços como: shell, login, exec, talk, ntalk, imap, pop-2, pop-3, finger, auth, ao menos que eles sejam utilizá-los. Se esses serviços forem desligados significa menos risco ao servidor.

Isso é feito colocando o caracter “#” no início da linha que indica o serviço assim:

```
#pop3     stream  tcp     nowait   root    /usr/sbin/tcpd  /usr/sbin/popa3d
#imap2   stream  tcp     nowait   root    /usr/sbin/tcpd  imapd
#finger    stream  tcp     nowait   nobody  /usr/sbin/tcpd  in.fingerd -u
```

Após feita as modificações, basta reinicializar o inetd da seguinte forma:

```
[root@localhost]# inetd restart
```

Quarto Passo

Enviar um sinal HUP para o processo inetd.

```
[root@localhost]# killall -HUP inetd
```

Quinto Passo

Configurar o arquivo "/etc/inetd.conf" como imutável, utilizando o comando *chattr* para que ninguém possa modificar esse arquivo.

Para configurar esse arquivo como imutável basta executar o seguinte comando:

```
[root@localhost]# chattr +i /etc/inetd.conf
```

Isso irá prevenir quaisquer alterações (acidentais ou de qualquer forma) ao arquivo "inetd.conf". O única usuário que poderá configurar ou limpar esse atributo é o superusuário root. Para modificar o arquivo inetd.conf é necessário retirar o atributo de imutável do arquivo:

Para retirar o atributo de imutável, basta executar o seguinte comando:

```
[root@localhost]# chattr -i /etc/inetd.conf
```

Corrigindo as permissões no diretório "/etc/rc.d/init.d"

Corrigir as permissões dos arquivos de script que são responsáveis por iniciar e parar todos os processos normais que necessitam ser executados durante o boot. Para fazer isso:

```
[root@localhost]# chmod -R 700 /etc/rc.d/init.d/*
```

Isso significa que apenas o root tem permissão para Ler (read), Escrever (write) e Executar (execute) arquivos script nesse diretório.

Controle de acesso do Sendmail.

Levando em consideração que o software Sendmail esteja devidamente instalado (com instalações default do Sistema Operacional Linux), basta fazer o controle de acesso da seguinte forma:

É feito a edição do arquivo /etc/mail/access e adicionada à ele algumas considerações como:

```
[root@localhost]# cat /etc/mail/access
200.150.59 RELAY
hackers.lab 550 Nao permetimos Hackers
gamk.com.br REJECT
diego.gamk.com.br OK
```

Nesta configuração foi permitido o RELAY da rede 200.150.59.0/24, foi também rejeitado e-mails de gamk.com.br menos os vindo da máquina diego.gamk.com.br.

Foi também rejeitado (550 = REJECT) e-mails vindo de hackers.lab enviando a mensagem de erro como "Não é permitido Hackers".

Depois de fazer a configuração basta digitar:

```
[root@localhost]# makemap hash access
```

Assim, o sendmail está configurado.

Desabilitanto expn e vrfy

Testando a utilidade dos comandos:

```
[root@localhost]# telnet 127.0.0.1 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 www.virtualnet.net ESMTP Sendmail 8.11.2/8.11.2; Sat, 10 Mar 2001
20:23:04 -0300
expn gamk
250 2.1.5 GAMK
vrfy diegolinke
250 2.1.5 Diego Linke
```

Pode-se perceber que o Sendmail acabou de entregar através dos comandos "expn" e "vrfy" dois usuário válidos na máquina.

É necessário desabilitar esta função para que isto não aconteça. Para fazer isso basta editar o arquivo sendmail.cf e procurar a linha:

```
O PrivacyOptions=authwarnings
```

Substitua por esta:

```
O PrivacyOptions=authwarnings,noexpn,novrfy
```

OBS: Caso não se saiba qual a localização desse arquivo (sendmail.cf), basta localizá-lo com o comando find assim:

```
[root@localhost]# find / -name sendmail.cf
```

```
O PrivacyOptions=authwarnings
```

Substituir por esta:

```
O PrivacyOptions=authwarnings,noexpn,novrfy
```

Basta reiniciar o sendmail e as modificações terão efeito.

```
[root@localhost]# sendmail -bd -q15m
```

Testando a nova configuração:

```
[root@localhost]# telnet 127.0.0.1 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 www.virtualnet.net ESMTP Sendmail 8.11.2/8.11.2; Sat, 10 Mar 2001
20:23:04 -0300
expn gamk
502 5.7.0 Sorry, we do not allow this operation
vrfy diegolinke
252 2.5.2 Cannot VRFY user; try RCPT to attempt delivery (or try finger)
```

É necessário remover o número da versão do sendmail. Para isso, basta editar o arquivo sendmail.cf e procurar pela linha:

O SmtpGreetingMessage=$j Sendmail $v/$Z; $b

Substituindo por:

O SmtpGreetingMessage= GAMK Mail Server [smtp.gamk.com.br]

Analisando os resultados:

```
[root@localhost]# telnet 127.0.0.1 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 GAMK ESMTP Mail Server [smtp.gamk.com.br]
```

Deixando o servidor Web Seguro

Essas configurações levam em consideração a instalação default do Apache em distribuições do Linux.

Como as configurações estão sendo feitas no Apache, deve-se inicialmente trabalhar no arquivo de configuração, o /etc/apache/httpd.conf.

Editando o arquivo e procurando pelas linhas:

```
<Directory "/usr/local/apache/htdocs">
    Options Indexes FollowSymLinks MultiViews Includes
    AllowOverride Limit AuthConfig
    Order allow,deny
    Allow from all
</Directory>
```

OBS: Elas variam de acordo com a configuração atual.

Como o próprio nome diz a diretiva "<Directory>" é responsável pelas configurações de um diretório no web server.

"Options" - configura os recursos oferecidos no diretório.
"AllowOverride" - configura o que o usuário poderá alterar em suas configurações.
"Order" - configura a ordem de interpretação das regras.
"Allow" - permite o acesso (temos também o "Deny" que nega o acesso).

Para "Options" tem-se:

None - Nenhum.
All - Todas as opções.
Indexes - Permite a visualização dos arquivos, caso não exista um index.
Includes - Permite a utilização de SSI (Server Side Include)
IncludesNoExec - Permite a utilização de SSI exceto o comando "exec" do SSI.

```
FollowSymLinks    - Permite a utilização de links simbólicos no apache.
ExecCGI           - Permite a execução de CGI's no diretório.

Para "AllowOverride" tem-se:

None              - Nenhuma alteração.
All               - Permite todas alterações.
AuthConfig        - Permite configurações de autenticação.
FileInfo          - Permite a inclusão de MIME para a árvore de diretórios.
Limit             - Permite o controle de acesso por diretório.
Options           - Permite alterar as diretivas do Options (veja acima).
```

A diretiva "AllowOverride" serve para liberar o que o cliente poderá alterar.

Na diretiva "AccessFileName" configura através de qual arquivo isso será configurado pelo cliente (padrão .htaccess):

```
AccessFileName .htaccess
```

Feito isso qualquer cliente criando um arquivo .htaccess (vale a pena lembrar que este arquivo é um arquivo oculto (por começar com um ".")) poderá configurar o que foi liberado pela diretiva "AllowOverride".

Supondo que na configuração do AllowOverride foi colocado:

```
AllowOverride Limit AuthConfig
```

Com esta configuração o cliente poderá fazer o controle de acesso de seus arquivos, além de poder usar o sistema de autenticação na página Web.

A Diretiva "<Limit>" controla os tipos de cabeçalhos de solicitação que podem ser utilizados no diretório.

A Diretiva "Order" trabalha dentro da diretiva "<Limit>" e nela configura-se qual a diretiva "Allow from" ou "Deny from" será interpretada primeira.

Exemplo 1:

```
Order deny,allow
Deny from all
Allow from 192.168.5 12.192.168.6
```

Isso irá negar todas as conexões, menos as requisições de 192.168.5.X ou 192.168.6.X.

Exemplo 2:

```
Order allow,deny
Allow from all
Deny from 192.168.5 12.192.168.6
```

Neste caso fará o contrário da primeira, aceitará todas as conexões menos as requisições de 192.168.5.X ou 192.168.6.X.

Exemplo de arquivo .htaccess:

Esse arquivo é criado pelo usuário no seu diretório “home”. Por exemplo:

/home/usuário/.htaccess

```
<Limit GET POST>
Order deny,allow
Deny from all
Allow from 192.168.0 .unixsecurity.com.br
</Limit>
```

Neste exemplo ninguém conseguirá ver a página exceto quem tiver dentro da rede 192.168.0.X ou tiver abaixo do domínio unixsecurity.com.br.

Autenticação do Apache

Pode-se configurar a autenticação no apache tanto no httpd.conf quanto no .htaccess.

As diretivas para autenticação são:

```
AuthType          - Tipo da autenticação (no nosso caso "Basic").
AuthName          - Nome da área restrita (para ser passado para o clien-
te).
AuthUserFile      - Arquivo de usuários/senhas.
AuthGroupFile     - Arquivo onde contém os usuários/grupos.
```

Para usar o sistema de autenticação do Apache usa-se a diretiva "require" (sempre entre a diretiva "<Limit>"), com ela tem-se alguns parâmetros, são eles:

```
require user [grupo1,grupo2,...]
require group [grupo1,grupo2,...]
require valid-user
```

Com o "require valid-user" qualquer cliente que se autenticar com um usuário e senha terá acesso a área restrita.

Com o "require group [grupo1,grupo2,...]" apenas os usuários que se autenticarem, e que pertencem ao grupo listado na diretiva, que terão acesso a área restrita.

Com o "require user [grupo1,grupo2,...]" apenas os usuários que se autenticarem, e que forem listados na diretiva terão acesso a área restrita.

Alguns exemplos:

```
AuthName        "Area restrita"
AuthType        Basic
AuthUserFile    /usr/local/apache/sites/www.gamk.com.br/restrita/.htpasswd
<Limit GET POST>
require valid-user
</Limit>
```

Nesta configuração qualquer usuário que se autenticar conseguirá acesso a área restrita.

O arquivo onde contém os usuários/senha é ".htpasswd" (setado na diretiva AuthUserFile) ele é gerado através do comando "htpasswd" (que vem junto com o apache), sua sintaxe é a seguinte:

```
htpasswd [-c] <authuserfile> <user>
```

Exemplo:

```
[root@localhost]# htpasswd -c .htpasswd gamk
New password:
Re-type new password:
Adding password for user gamk
```

O parâmetro "-c" é usado apenas na primeira vez, pois ele indica "create" (para criar o arquivo), das próximas vezes o parâmetro deve ser omitido.

Exemplo:

```
[root@localhost]# htpasswd .htpasswd diego
New password:
Re-type new password:
Adding password for user diego
```

OBS: o mesmo exemplo acima serve para alteração de senhas.

A autenticação "Basic" usa a função crypt() para encriptar a senha no arquivo de senhas, portanto as senhas são limitadas a 8 caracteres. Ou seja, uma senha como "diegolinke", basta apenas "diegolin" para ser considerada válida.

```
AuthName        "Area restrita"
AuthType        Basic
AuthUserFile    /usr/local/apache/sites/www.gamk.com.br/restrita/.htpasswd
<Limit GET POST>
require user diego, gamk, mauricio
</Limit>
```

Neste exemplo apenas os users (diego, gamk e mauricio) terão acesso a este diretório, depois que se autenticarem, é claro. Mesmo que exista um usuário chamado "ricardo" no arquivo de senhas e ele se autentique com a sua senha, ele não conseguirá ter acesso a este diretório.

Pode-se trabalhar com grupos (detalhe para a diretiva "AuthGroupFile"):

```
AuthName        "Area restrita"
AuthType        Basic
AuthUserFile    /usr/local/apache/sites/www.gamk.com.br/restrita/.htpasswd
AuthGroupFile   /usr/local/apache/sites/www.gamk.com.br/restrita/.htgroup
<Limit GET POST>
require group rh, clientes
</Limit>
```

Apenas usuários pertencentes aos grupos “rh” e “clientes” terão acesso a este diretório (depois que se autenticarem).

Sintaxe de arquivo de grupos:

nome_do_grupo: [user1 user2 user3 ...]

Exemplo de arquivo de grupos:

```
rh: joão wagner ricardo
clientes: sandro marcos rosimar jean
```

Basta reiniciar o serviço do Apache e todas as configurações terão efeito:

Para testar as suas configurações do httpd.conf digite:

```
[root@localhost]# apachectl configtest
Syntax OK

[root@localhost]# apachectl start
apachectl start: httpd started
```

Configurando um Servidor de FTP

Geralmente distribuições do Linux não possuem o pacote proftpd-1.2.x, então é necessário fazer a instalação do mesmo:

```
[root@localhost]# tar zxvf proftpd-1.2.6.tar.gz
```

Após entra-se na pasta criada

```
[root@localhost] cd proftpd-1.2.6
```

Fazendo a configuração

```
[root@localhost] ./configure --sysconfdir=/etc --prefix=/usr/local/
```

A opção “sysconfdir” será onde ficará o arquivo proftpd.conf, que será feita toda a configuração do proftpd, a opção “prefix” será onde ficará instalado o executável do proftpd.

Agora será feita a compilação do proftpd

```
[root@localhost] make
```

E finalmente é feita a instalação do proftpd

```
[root@localhost] make install
```

Após a instalação serão feitas algumas configurações para ele poder funcionar corretamente.

Editando o arquivo proftpd.conf

```
[root@localhost] vi /etc/proftpd.conf

ServerName                      "FTP LinuxIT"
ServerType                      standalone
DefaultServer                   on
```

ServerName: Indica o nome do servidor.

ServerType: Não é necessário fazer alterações.

DefaultServer: Sempre com a opção on.

A partir da linha 34 começam as configurações de ftp anônimo, caso deseja-se ter um ftp anônimo, não se faz modificações, caso contrário, basta colocar o caractere “#” no início das linhas 34 até 53.

É necessário criar o grupo “nogroup” para o proftpd poder rodar:

```
[root@localhost] groupadd nogroup
```

Para iniciar o proftpd:

```
[root@localhost] proftpd
```

Para ver se o proftpd está rodando:

```
[root@localhost] ps ax | grep proftpd
```

A resposta a esse comando será a seguinte:

```
23246?        S      0:00 proftpd (accepting connections)
```

Para matar qualquer processo do proftpd:

```
[root@localhost] killall proftpd
```

Para reiniciar o Proftpd:

```
[root@localhost] killall –HUP proftpd
```

É interessante colocar o proftpd para inicializar toda vez no boot, isso é feito no arquivo /etc/rc.d/rc.local:

```
[root@localhost] vi /etc/rc.d/rc.local
```

No final do arquivo basta adicionar essas linhas:

```
#Inicio
echo “Inicializando o Proftpd”
/usr/local/sbin/proftpd
# Fim
```

O caminho do executável do proftpd depende do que foi colocou na instalação do mesmo.

Algumas configurações adicionais no proftpd.conf

A linha “**Serverident on**” faz com que não apareça a versão do ftp para quem o acessar, aparecerá o que está escrito entre aspas.

```
ServerIdent on "FTP Server"
```

Pode-se restringir alguns grupos de acessar o ftp, no exemplo abaixo é liberado somente conexões ftp de usuários que pertencem aos grupos ftpusers e webmaster, e negando qualquer outro grupo.

```
<Limit LOGIN>
DenyGroup !ftpusers,!webmaster
</Limit>
```

Fazendo algumas modificações no arquivo /etc/ftpaccess

O ftpaccess é arquivo de configuração de núcleo do ftpd. Por meio das diretivas nesse arquivo, pode-se controlar o funcionamento do ftpd.

A seguir um exemplo do ftpaccess.

```
$ more /etc/ftpaccess
class all   real,guest,anonymous          *
email root@localhost
loginfails 5
readme            README*     login
readme            README*     cwd=*
message     /home/ftp/welcome.msg   login
message     /home/ftp/.message      cwd=*
compress          yes               all
tar               yes               all
chmod             no                guest,anonymous
delete                  no                guest,anonymous
overwrite         no                guest,anonymous
rename                  no                guest,anonymous
log transfers anonymous,real  inbound,outbound
shutdown /etc/shutmsg
passwd-check      rfc822 warn
```

Com essa configuração, nenhum usuário terá permissão de executar comandos como “chmod”, “delete”, “rename” pois eles estão com a opção “no”. Caso esses comandos estivessem com “yes” colocaria o servidor em risco, pois, os usuários poderiam fazer alterações indesejáveis.