

FUNDAÇÃO UNIVERSIDADE ESTADUAL DE MARINGÁ

DIN – DEPARTAMENTO DE INFORMÁTICA

## **SEGURANÇA EM SERVIDORES LINUX**

Ricardo Bittencourt Socreppa

Professor Orientador  
Prof. Msc.César Fernando Moro

MARINGÁ  
2005

RICARDO BITTENCOURT SOCREPPA

## **SEGURANÇA EM SERVIDORES LINUX**

Monografia apresentada ao curso de Especialização da Universidade Estadual de Maringá como parte dos requisitos para a obtenção do título de Especialista em Desenvolvimento de Sistemas para WEB.

Orientador: Prof. Msc. César Fernando Moro

MARINGÁ  
2005

## **DEDICO**

A minha Família principalmente aos meus Pais e a minha namorada Gislaine Cristine Bortolossi que muito contribuíram para a conclusão desta monografia.

## RESUMO

Este trabalho mostra as diferentes formas que um *hacker*, pode conseguir acesso ao servidor Linux, explicando como o invasor pode explorar falhas tanto no sistema, quanto nos serviços que o servidor disponibiliza, para chegar ao ponto principal que é o acesso ao *root*, e conseqüentemente assim ter o controle da máquina.

Juntamente com os modos que o *hacker* utiliza para invadir o servidor, também são apresentadas algumas técnicas que podem ser utilizadas pelo administrador para aumentar a segurança de um servidor Linux.

Palavra Chave: segurança, hacker, servidor, invasão, Linux.

## SUMÁRIO

<b>LISTA DE FIGURAS</b>	7
<b>INTRODUÇÃO</b>	8
<b>1 MÉTODOS DE INVASÃO</b>	11
1.1 ENGENHARIA SOCIAL	11
1.1.1 Falsa Autoridade	12
1.1.2 Personificação	12
1.1.3 Simpatia	13
1.1.4 Interesse Pessoal	14
1.1.5 Inflar Egos	14
1.1.6 Ocupação Furtiva	15
1.1.7 Recompensa	15
1.2 VÍRUS	16
1.3 BACKDOORS	17
1.4 CAVALO DE TRÓIA (TROJAN HORSE)	17
1.5 NEGAÇÃO DE SERVIÇO (DENIAL OF SERVICE)	18
<b>2 INSTALAÇÃO DE UMA POLÍTICA DE SEGURANÇA</b>	20
2.1 POLÍTICA DE SEGURANÇA	20
<b>3 SEGURANÇA BÁSICA</b>	23
3.1 PROGRAMAS SERVIDORES DESNECESSÁRIOS	23
3.2 USUÁRIOS	26
3.3 CONTROLE DE ACESSOS SOBRE USUÁRIOS	28
3.4 GERENCIAMENTO DE CONTAS INATIVAS	30
3.5 MANTER O SISTEMA ATUALIZADO	31
<b>4 FIREWALLS NA SEGURANÇA DO SERVIDOR</b>	33
4.1 FIREWALLS	33
4.2 FERRAMENTAS DE IMPLEMENTAÇÃO DE FIREWALL NO LINUX	35
4.3 CONFIGURAÇÃO DE UM FIREWALL COM IPTABLES	37
4.3.1 Ajustar uma política padrão de firewall	38
4.3.2 Criação de regras de firewall	38
4.3.3 Abertura e fechamento de portas específicas	39
4.3.4 Usar endereço IP como fonte de destino	39
4.3.5 Filtrando pela interface	40
4.4 TABELA NAT	40
<b>5 CRIPTOGRAFIA NO LINUX</b>	43
5.1 CRIPTOGRAFIA	43
<b>6 SOFTWARE DE DETECÇÃO DE INTRUSOS</b>	46
6.1 DETECÇÃO DE INTRUSOS (IDS)	46
6.2 TIPOS DE IDS	48
6.2.1 Baseado em Rede	48
6.2.2 Baseado em Host	49
6.3 MÉTODOS DE DETECÇÃO DOS IDS	49
6.3.1 Baseado em Assinaturas	49
6.3.2 Baseado em Anomalia	50
6.4 SNORT	51
<b>7 CONCLUSÃO</b>	53
<b>BIBLIOGRAFIA</b>	55
<b>ANEXOS</b>	56

## LISTA DE FIGURAS

Figura 3.1 – Gerenciamento de Pacotes do Kurumin 4.1	24
Figura 3.2 – Demonstração do comando ps ax	25
Figura 3.3 – Relação de entradas no sistema de arquivo e suas permissões de arquivos e diretórios	28
Figura 4.1 – Representação de uma rede protegida por um Firewall	34
Figura 4.2 - Funcionamento do Firewall	36
Figura 4.3 – Funcionamento de um Roteador NAT	41
Figura 5.1 Arquivo /etc/shadow onde é armazenado as senhas criptografadas dos usuários	44
Figura 6.1 – Representação da Tabela TOP 10 – Medidas de Segurança 2004	47

## INTRODUÇÃO

No final da década de 60, surgia a internet. Inicialmente ela foi criada e desenvolvida para ser utilizada pelo exército americano, a fim de não centralizar todas as informações registradas em computadores em um único local do país.

Mas com o passar dos anos houve um crescimento e a popularização da internet, englobando não só computadores militares americanos, mas muitos outros pelo mundo. Com a transmissão de vários tipos de informações, e acessados por vários perfis de pessoas, surge à necessidade de assegurar a segurança das informações trafegadas na rede. Transações bancárias e *e-commerce* são exemplos de tecnologias que necessitam dessa segurança, pois trafegam informações de usuários de suma importância e sigilo, tais como números de cartão de crédito e senhas. Estas informações devem ser protegidas tanto para a transmissão quanto no armazenamento e acesso posterior.

Nunca se preocupou tanto com a segurança como agora, principalmente devido ao surgimento dos famosos *hackers*, que exploram algum tipo de vulnerabilidade dos sistemas para obter informações ou apenas danificar o sistema. Assim as empresas precisam aumentar a proteção de suas máquinas e principalmente seus servidores.

O termo servidor seguro é freqüentemente usado com o significado de anular ou minimizar as chances de um invasor conseguir obter acesso a ele. A segurança está relacionada à necessidade de proteção de suas informações confidenciais, de elementos não autorizados, ou seja, está relacionada à proteção contra o acesso ou manipulação, intencional ou não, de informações por elementos não autorizados, ou seja, invasores.

A necessidade de proteção deve ser definida em termos das possíveis ameaças e riscos e dos objetivos de uma organização. Jamais o administrador do servidor pode

achar que sua máquina não possui dados valiosos, conseqüentemente não precisa de métodos e programas para a sua segurança.

*“Um engano de muitos usuários do Linux é o de considerar que sua máquina não é suficiente importante para ser alvo de hackers. Eles pensam ‘Não tenho nada importante em minha máquina; quem poderia querer violá-la’” (James Lee, Brian Hatch e George Kurtz, 2001, p. 4).*

Se os *hackers* conseguirem acesso ao servidor desprotegido, poderão invadir outros servidores mais importantes e seguros usando os recursos da máquina invadida, podendo até incriminar o administrador do servidor usado na invasão.

O melhor método contra a invasão é a prevenção, portanto, o objetivo desta monografia é apresentar algumas técnicas usadas pelos *hackers* para invadir um servidor como também os programas e configurações, que o administrador deve utilizar e fazer para tornar um servidor Linux seguro e confiável.

Nesse caso, para o administrador do servidor conseguir aumentar ou colocar um nível de segurança mais alto é preciso entender e utilizar os seguintes elementos:

- Entender quais os métodos e técnicas utilizadas pelos *hackers* para burlar a segurança e conseguir o acesso total da máquina. Estes métodos e técnicas podem ser usados em diversos sistemas operacionais e serão abordados no capítulo 1;
- Como criar uma política de segurança em uma empresa e como ela é importante na segurança do servidor. A criação desta política será apresentada no capítulo 2;
- Utilizar técnicas para segurança básica como desabilitar programas servidores desnecessários, controle de acesso ao usuário do servidor, desabilitar contas de usuários inativas e manter o sistema atualizado, serão os tópicos apresentados no capítulo 3;



- Um *firewall* instalado e configurado de forma correta, onde os filtros serão melhores implementados, será detalhado no capítulo 4;
- Criptografia nos dados cruciais, já que dados criptografados são difíceis de serem quebrados, será mostrada no capítulo 5;
- O que são os detectores de intrusos, como eles funcionam, seus tipos e como instalar e configurar no servidor Linux. Os IDS serão apresentados no capítulo 6;
- Finalmente no capítulo 7 será apresentada a conclusão deste trabalho;

## CAPÍTULO 1 – MÉTODOS DE INVASÃO

### INTRODUÇÃO

Os *hackers* possuem vários métodos e técnicas para burlar a segurança das máquinas e dos servidores Linux, sendo assim conseguir acesso total. No caso dos servidores Linux o invasor provavelmente tentará conseguir o acesso ao usuário *root* (Cap. 2). O objetivo deste capítulo é mostrar para o administrador, os diversos métodos e técnicas que o *hacker* utiliza para conseguir realizar a invasão com sucesso e concretizar seus objetivos.

#### 1.1 ENGENHARIA SOCIAL

Engenharia Social é o termo utilizado para a obtenção de informações importantes de uma máquina ou servidor Linux, por exemplo, através de seus usuários e colaboradores. Essas informações podem ser obtidas pela ingenuidade ou confiança. Os ataques desta natureza podem ser realizados através de telefonemas, envio de mensagens por correio eletrônico, salas de bate-papo, até mesmo pessoalmente.

*“[...] É a tentativa do hacker de conseguir que alguém o ajude em uma invasão, enganando ou confundindo essa pessoa. Geralmente isso é feito sem que as pessoas sequer saibam que estão danificando sua própria segurança.” (James Lee, Brian Hatch e George Kurtz, 2001, p. 116)*

Segundo a (NBSO, 2003) a Engenharia Social é o ataque onde o invasor abusa da ingenuidade ou confiança do usuário, para obter informações que podem ser utilizadas para ter acesso não autorizado a computadores.

A Engenharia Social, o método mais simples e infelizmente, um dos mais eficientes de se descobrir uma senha do usuário, qual o tipo do sistema, no caso do servidor

Linux, qual distribuição esta sendo utilizada, como é organizada a rede da empresa e até mesmo os tipos dos computadores usados. Basta o *hacker* ter uma boa lábia.

Importante observar que o sucesso da engenharia social depende da compreensão do comportamento do ser humano, pois este é o elemento mais vulnerável de qualquer sistema de segurança.

Dentro da engenharia social, existem várias categorias como a falsa autoridade, personificação, simpatia, interesse pessoal, inflar egos, ocupação furtiva e recompensa. O *hacker* pode usar uma dessas categorias como também combiná-las para melhor atingir seus objetivos.

#### 1.1.1 Falsa Autoridade

O *hacker* pode obter informações simplesmente convencendo o administrador do servidor ou um usuário comum de que eles são pessoas importantes e precisam deles para realizar determinada tarefa.

Fingindo ser alguém de um cargo importante da empresa, por exemplo, o *hacker* passa a se identificar com um diretor da empresa em uma viagem a serviço, com isso é capaz de conseguir qualquer informação apenas perguntando. Neste caso o *hacker* não precisa fingir ser uma pessoa real, apenas ser uma pessoa que possui autoridade para solicitar as informações ou acesso que necessita.

#### 1.1.2 Personificação

A personificação é semelhante à falsa autoridade quanto ao fato de que o *hacker* tenta convencer o administrador ou usuário comum de que têm o direito de acompanhar suas ações. Ela é uma versão da falsa autoridade em que o *hacker* assume a identidade de uma pessoa real.

Para o *hacker* personificar uma pessoa que o membro da empresa conheça é muito difícil e fácil de ser descoberto, mas é muito mais fácil para ele personificar uma pessoa importante pelo telefone, *e-mail*, ou em uma sala de bate-papo.

Depois de convencer o membro da empresa, de que o *hacker* é uma pessoa importante, ele solicita informações ou acessos que não são suspeitos vindos daquela pessoa, como por exemplo:

*From: [carlosalberto@znet.com.br](mailto:carlosalberto@znet.com.br)  
To: Departamento de Informática*

*Oi, aqui é o Carlos Alberto da Gerencia. Como você sabe, estou de férias essa semana, mas preciso receber meus e-mails hoje. O firewall não está permitindo que eu acesse, porque estou me comunicando de Manaus e não de casa. Você poderia, por favor, abrir o acesso para meu endereço IP, por ai? O Ip é 192.168.30.15. Uma vez que eu consiga entrar, enviarei mensagens do meu endereço e-mail interno real, mas para que você saiba que sou eu, meu cadastro na empresa é SK215887-7.*

*Desde já agradeço.*

Através deste tipo de e-mail o *hacker* pode conseguir o acesso aos *e-mails* e também ao servidor.

### 1.1.3 Simpatia

A simpatia é um dos métodos mais confiáveis usados pelo *hacker*, onde ele finge precisar de algo muito importante, sendo assim, faz com que as pessoas da empresa sintam pena dele e queiram ajudá-lo.

Um exemplo de simpatia seria, quando o *hacker* chegue, fingindo ser do departamento de *marketing*, fala que precisa reiniciar sua senha ou que não consiga mudar o anuncio em tempo hábil e com isso seu chefe vai matá-lo. Dependendo de como a *hacker* falar, o administrador vai fazer as reivindicações dele sem mesmo verificar se esta pessoa pertence à empresa.

O recurso da simpatia é usado por milhões de pessoas em seu cotidiano, e os *hackers* fazem uso dela da mesma forma para conseguir o acesso a informações úteis.

#### 1.1.4 Interesse Pessoal

O *hacker* utiliza o interesse pessoal, para conseguir uma maior cooperação do administrador ou usuário comum, para isso, ele cria cenário que pode afetar o cargo da pessoa que ele esta manipulando. Por exemplo:

Na folha de pagamento. Fingindo ser um funcionário do departamento de contabilidade, um consultor de segurança disfarçado se aproxima do administrador do servidor, dizendo de que ele não podia acessar o sistema. Uma vez tendo explicado que precisa executar alguns processos cruciais ou a folha de pagamento atrasaria, o administrador do servidor fornece ao *hacker* mais acesso que o necessário para estar absolutamente certo de que não haverá nenhuma barreira para a emissão da folha e do pagamento do administrador.

O Interesse pessoal combinado com a simpatia pode ser muito eficaz para o *hacker* na obtenção de informações para uma possível invasão.

#### 1.1.5 Inflar Egos

O *hacker* utiliza o inflar egos para fazer com que o administrador do servidor ou usuário se sinta bem consigo mesmo, com isso esta pessoa se torna mais fácil de manipular.

Quando as pessoas são elogiadas, querem continuar sendo para isso se defenderão menos para continuar os elogios. Por exemplo:

O *hacker* finge ser uma pessoa a um nível superior ao do administrador do servidor e começa a elogiar o trabalho que esta sendo feito por ele no NPD e fala que o presidente da companhia também está muito satisfeito com o seu trabalho. Através deste tipo de manipulação o *hacker* pode conseguir várias informações sigilosas.

#### 1.1.6 Ocupação Furtiva

O *hacker* consegue penetrar em áreas que normalmente não podem ser acessadas, usando a ocupação furtiva, fingido ser da companhia de telefone, por exemplo. As pessoas que exercem esta profissão parecem ter algum tipo de campo de invisibilidade ao seu redor, ou seja, elas não são notadas a menos que seja absolutamente necessário.

Com isso estes profissionais se tornam perfeitos para o *hacker* personificar, uma vez que eles podem andar por todos os lugares da empresa. Personificando este profissional o *hacker* pode procura de informações que possa ajudar em uma futura invasão ou até mesmo ficar cara a cara com o servidor para tentar invadi-lo.

#### 1.1.7 Recompensa

O *hacker* pode usar a recompensa para enganar o administrador ou usuários comuns, a fim de que forneçam informações. Por exemplo:

O *hacker* liga para o administrador da empresa dizendo ser um gerente e que precisa de tal informação para realizar um negócio, que se der certo, o dono da empresa irá dar um fim de semana para todos os funcionários em um hotel fazenda, tudo por conta da empresa. Com isso o funcionário acaba passando a informação em troca de um “prêmio”.

## 1.2 VÍRUS

O vírus de computador é um programa de computador, utilizado maliciosamente ou não, que se reproduz embutindo-se em outros programas. Quando estes programas são executados, o vírus é ativado e pode se espalhar ainda mais, geralmente danificando sistemas e arquivos do computador onde ele se encontra.

Após infectar o computador, eles passam a atacar outros arquivos. Se um destes arquivos infectados for transferido para outro computador, este também vai passar a ter um vírus alojado, esperando o momento para infectá-lo, ou seja, quando for também executado. Por isso são chamados de vírus, devido à sua capacidade de auto-replicação, parecida com a de um ser vivo.

A (NBSO, 2003) afirma que o vírus é um programa capaz de infectar outros programas e arquivos de um computador. Para realizar a infecção o vírus embute uma cópia de si mesmo em um programa ou arquivo, que quando executado também executa o vírus dando continuidade ao processo de infecção. A diversas formas do vírus se propagar como, por exemplo: abrir arquivos anexados em *e-mails*, abrir arquivos de texto e planilhas, abrir arquivos armazenados em outros computadores, instalar programas duvidosos ou de procedência desconhecida.

Alguns vírus podem permanecer ocultos, onde continuam infectando arquivos e executando uma série de atividades sem que o usuário perceba. Ainda existe um tipo de vírus que ficam encubados e só são ativados em determinada data.

Os *hackers* podem utilizar o vírus para destruir, por exemplo, toda a base de dados de uma empresa, sem mesmo precisar invadir o sistema. Ele só precisa que um usuário da empresa execute o programa malicioso, disfarçado de um programa inofensivo.

### 1.3 BACKDOORS

Os *backdoors* são portas abertas por programas que às vezes os administradores de sistema não desativam com isso o sistema fica aberto para que o *hacker* possa invadir a máquina.

Os *backdoors* são abertos devido a defeitos de fabricação ou falhas no projeto dos programas, isto pode acontecer tanto acidentalmente ou ser introduzido ao programa propositalmente.

Conforme (NBSO, 2003), alguns fabricantes incluem *backdoors* em seus produtos, alegando necessidades administrativas, conseqüentemente constitui uma grande ameaça na segurança do computador ou do servidor uma vez que o produto tenha sido instalado na máquina.

Os *hackers* podem utilizar os *backdoors* que os programas abrem para invadir as máquinas ou os servidores.

### 1.4 CAVALO DE TRÓIA (TROJAN HORSE)

A denominação “Cavalo de Tróia” (*Trojan Horse*) foi atribuída aos programas que permitem a invasão de um computador alheio com espantosa facilidade. Nesse caso, o termo é análogo ao famoso artefato militar fabricado pelos gregos espartanos. Um “amigo” virtual presenteia o outro com um “presente de grego”, que seria um aplicativo qualquer.

Quando o usuário ou administrador executa um programa que contém um *Trojan*, o programa pode abrir um *backdoor*, sem que seja percebido, sendo assim o *hacker* pode abrir uma porta para uma futura invasão.



O *trojan*, geralmente vem anexado em um *e-mail* ou está disponível em algum programa, que parece inofensivo disponível para *download* na internet.

(NBSO, 2003) afirma que o cavalo de tróia é um programa que além de executar funções para as quais foi projetado, também executa outras funções normalmente maliciosas e sem conhecimento do usuário. Uma das principais funções do *Trojan* é a inclusão de *backdoors*, permitindo que o invasor possa tomar acesso total da máquina.

Importante ressaltar que o *hackers*, caso não achem um *backdoor* na máquina, podem criar esta porta através do *trojan*, usando a ingenuidade do usuário para instalá-lo na máquina.

### 1.5 NEGAÇÃO DE SERVIÇO (*DENIAL OF SERVICE*)

O *Denial of Service* é um ataque que consiste em sobrecarregar um servidor com uma quantidade excessiva de solicitações de serviços, tentando travar a máquina em questão ou indisponibilizando este serviço.

Conforme diz (NBSO, 2003) o atacante utiliza o *Denial of Service*, para tirar de operação um serviço ou um computador conectado à internet. Alguns tipos de ataques de negação de serviço seriam:

- Gerar uma grande sobrecarga no processamento de dados de um computador, de modo que qualquer usuário não consiga utilizá-lo.
- Gerar um tráfego de dados para uma rede, ocupando toda a banda disponível de modo que qualquer computador desta rede fique indisponível.
- Tirar serviços importantes de um provedor do ar, por exemplo, impossibilitando o acesso dos usuários às suas caixas de correios no servidor de *e-mail* ou no servidor *web*.

Os ataques de negação de serviços podem ser usados pelos *hackers* no intuito de prejudicar as empresas nos seus negócios, pois indisponibilizando determinado serviço, fica impossível a empresa trabalhar.

## **RESUMO DO CAPÍTULO**

Neste capítulo foram apresentados diversos métodos que os *hackers* utilizam para conseguir acesso à máquina, explorando tanto, falhas no sistema com a falha humana que geralmente é usada com maior frequência pelos invasores. Importante ressaltar que estes métodos independem do sistema operacional.

No próximo capítulo serão abordadas algumas técnicas para criar uma política de segurança, que quando respeitadas pelos funcionários da empresa, trarão uma confiabilidade e segurança dos seus atos.

## CAPÍTULO 2 – INSTALAÇÃO DE UMA POLÍTICA DE SEGURANÇA

### INTRODUÇÃO

A importância de uma política de segurança em uma empresa pode ser um dos primeiros passos que o administrador deve executar para aumentar a segurança de suas máquinas e servidores. O Objetivo deste capítulo é mostrar para o administrador como se deve proceder para criar na sua empresa uma política de segurança.

#### 2.1 POLÍTICA DE SEGURANÇA

O uso de uma política de segurança é essencial para auxiliar o administrador da rede a manter toda estrutura da empresa protegida.

É na política de segurança que se estabelecem às regras de utilização dos recursos pelos usuários e o que será considerado uma intrusão ou tentativa de ataque. Que ações realizar quando alguém tenta fazer o *login* sem sucesso por três vezes ou queira acessar uma página restrita, de quanto em quanto tempo atualizar o antivírus e quais horários são permitidos acessar a Internet, são alguns pontos definidos na política de segurança da empresa.

Para definir uma política de segurança, estas três perguntas podem auxiliar no processo de criação da política de segurança (JUNIOR, 1999):

- O que se quer proteger? Depende de cada empresa, porque cada uma possui sua própria estrutura de hardware e software e é diferente uma da outra em relação a suas prioridades e recursos. Um banco terá uma política bem mais rígida que uma empresa pequena, por exemplo;
- Contra quem o sistema deve ser protegido? Com o crescente aumento da utilização da informática nas mais diversas áreas e a concorrência entre as

empresas, cresceu também o número de ataques internos e externos. Quanto maior as empresas, maiores são as ameaças;

- Que nível de segurança é necessário? O custo do nível de segurança que deve ser empregado deve levar em conta quanto valem as informações que a empresa possui. A implantação dos recursos de segurança deve suprir as necessidades da empresa na prevenção de violações contra seu sistema de informação.

Quando se pensa em segurança, espera-se na realidade que as informações estejam disponíveis, corretas e fora do alcance de pessoas não autorizadas. Em (DIAS, 2000) define-se os seguintes objetivos de segurança:

- Confidencialidade ou privacidade: proteger as informações contra acesso de qualquer pessoa não autorizada;
- Integridade de dados: assegurar que os dados não foram modificados sem previa autorização;
- Disponibilidade: garantir que o usuário conseguirá usar o sistema na hora que for necessário;
- Consistência: certificar que o sistema atua de acordo com as expectativas dos usuários autorizados;
- Isolamento ou uso legítimo: regular o uso do sistema para poder verificar quem acessou e como;
- Auditoria: identificar os autores e suas ações, o que poderá auxiliar na recuperação do sistema, caso houver algum problema;
- Confiabilidade: garantir que o sistema atuará conforme o esperado pelo usuário e o administrador da rede.

Importante salientar que o grau de prioridade destes objetivos pode mudar de uma empresa para outra.

A política deve ser escrita e ter o aval dos diretores da empresa, tornando-a assim um documento que deve ser do conhecimento de todos e que deve ser seguido. O não cumprimento de alguma política pode abrir uma falha na segurança que poderá trazer prejuízos para a empresa, que vai desde perda de dados devido ao ataque de algum vírus até roubo de informações confidenciais.

Na política de segurança deve ficar bem especificado quem são os responsáveis e pelo que são responsáveis (é importante ter uma hierarquia definida). Os aspectos legais e éticos também são importantes e devem ser claros e detalhados o suficiente para que sejam compreendidos por todos os funcionários da empresa.

Um dos pontos chaves na implantação das políticas de segurança é a conscientização do usuário, todos devem estar cientes dos riscos que o descumprimento de uma política pode causar para a empresa.

## **RESUMO DO CAPÍTULO**

Neste capítulo foram apresentadas algumas técnicas que podem ser usadas pelos administradores a fim de criarem uma política de segurança na empresa, conseqüentemente, os usuários estarão mais seguros em fazer suas operações.

No próximo capítulo serão abordados conceitos de segurança básica nos servidores Linux, como por exemplo, desativar os programas servidores desnecessários, criar os tipos de usuários, grupos e como pode ser feito o controle de acesso sobre estes usuários, desativar contas de usuários inativas e atualizar constantemente o sistema.

## CAPÍTULO 3 – SEGURANÇA BÁSICA

### INTRODUÇÃO

Antes mesmo de definir alguns métodos e programas de segurança mais avançados para o servidor Linux, é preciso entender e aplicar métodos de segurança básica. Este capítulo tem como objetivo mostrar como o administrador deve proceder para realizar a segurança básica, como por exemplo, fechar os programas servidores que não estão sendo utilizados, proteger o usuário *root* e criar permissões de usuários mais restritas, desativar contas de usuários que não estão sendo usadas e manter o sistema atualizado.

#### 3.1 PROGRAMAS SERVIDORES DESNECESSÁRIOS

Conforme (RODERICK W, 2001) a maioria dos programas servidores tem uma história de *bugs*, que permite que estranhos consigam acesso à máquina, e mesmo aqueles que não possuem *bugs*, também podem sofrer algum tipo de invasão.

Como os programas servidores oferecem geralmente acesso a um computador, eles aumentam o risco de uma possível invasão. O invasor pode conseguir acesso através de um *bug* do programa servidor, uma má configuração ou até mesmo uma senha comprometida. Um mecanismo para proteger o sistema, seria fechando os programas servidores desnecessários, pois uma vez fechado o *hacker* não poderá utilizar aquele servidor para praticar a invasão.

(RODERICK W, 2001) cita que a tarefa de localizar programas servidores desnecessários pode ser dividida em duas sub-tarefas:

- Indicar os programas servidores que estão sendo executados no servidor;
- Determinar qual desses programas servidores é desnecessário para o funcionamento normal do servidor;

Infelizmente não existe nenhum registro centralizado de executar programas servidores no sistema operacional Linux, conseqüentemente, é necessário reunir diversas informações de muitas fontes diferentes, só assim o administrador poderá definir quais programas servidores não estão sendo utilizados, afirma (RODERICK W, 2001).

Para conseguir realizar a primeira tarefa, que é de indicar os programas servidores que estão sendo executados, o administrador poderá: usar o pacote de gerenciamento de sistemas, examinar os arquivos de iniciação do programa servidor, examinar os processos sendo executados através do comando *ps* e até mesmo, *scanners* externos.

O pacote de gerenciamento de sistema é uma ferramenta útil na localização de programas servidores, pois ele possui um banco de dados que contém uma listagem de pacotes instalados na máquina e uma descrição sobre cada pacote conforme mostra a figura 3.1. Através destas informações o administrador irá conhecer quais programas estão instalados e sua função na máquina.

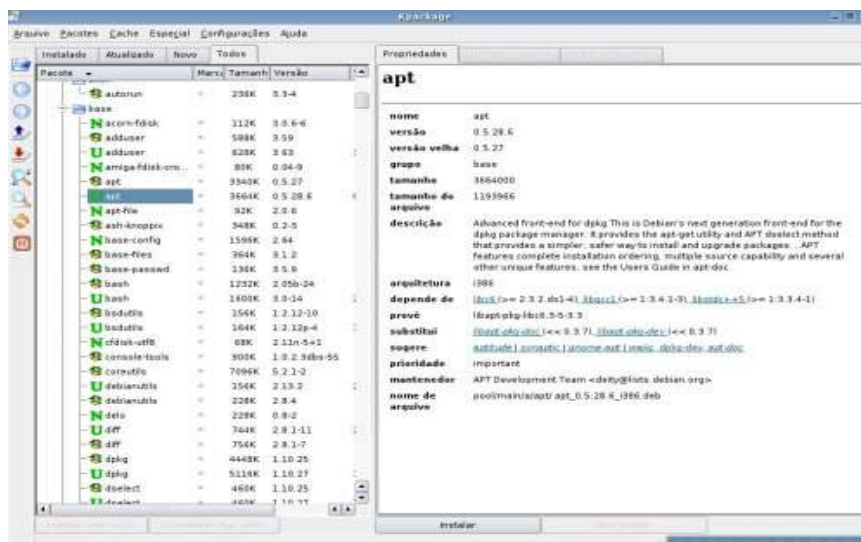


Figura 3.1 - Gerenciamento de Pacotes do Kurumin 4.1

Uma outra maneira de buscar programas servidores instalados na máquina é examinar os arquivos comuns de iniciação. Segundo (RODERICK W, 2001) pode-se fazer isso de três modos:

- **Configurações de superservidor:** Verificar os arquivos `/etc/inetd.conf` e `/etc/xinetd.conf` e os arquivos dentro do diretório `/etc/xinetd.d`. Esses contêm referências a todos os programas servidores iniciados através do superservidor.
- **Scripts de iniciação SysV:** Verificar as localizações dos scripts de iniciação do Sysv (normalmente, `/etc/rc.d/rc?.d`, onde "?" é o número do nível de execução) de programas servidores iniciados através do *script Sysv*.
- **Scripts locais de inicialização:** Muitas distribuições usam *scripts* chamados `rc.local` ou `boot.local`, para executar programas locais, ou seja, aqueles que estão instalados de uma maneira única a um computador específico ao invés de uma maneira padrão da distribuição em questão.

Uma outra ferramenta para localizar programas servidores, seria o `ps`. Este comando retorna informações sobre processos que estão sendo executados na máquina. Este comando possui vários parâmetros para que o administrador possa usar para fazer a busca de programas servidores, como por exemplo, `ps ax` conforme mostra a figura 3.2.

```

kurumin@socoreppa:~$ ps ax
  PID TTY          STAT       TIME COMMAND
    1 ?        S           0:00 init [5]
    2 ?        SN          0:00 [ksoftirqd/0]
    3 ?        S<          0:00 [events/0]
    4 ?        S<          0:00 [khelper]
    5 ?        S<          0:00 [kacpid]
   20 ?        S<          0:00 [kblockd/0]
   32 ?        S           0:00 [pdflush]
   33 ?        S           0:00 [pdflush]
   35 ?        S<          0:00 [aio/0]
   34 ?        S           0:00 [kswapd0]
  142 ?        S<          0:00 [ata/0]
  145 ?        S           0:00 [kseriod]
  150 ?        S<          0:00 [kcryptd/0]
  151 ?        S<          0:00 [kmirrord/0]
  154 ?        S           0:00 [kjournald]
  663 ?        Ss          0:00 /usr/sbin/acpid -c /etc/acpi/events -s /var/run/acpid.
  690 ?        Ss          0:00 /sbin/portmap
  820 ?        Ss          0:00 /usr/bin/kdm
  830 ?        S           0:41 /etc/X11/X -dpi 75 -nolisten tcp vt7 -auth /var/lib/kd

```

Figura 3.2 - Demonstração do comando `ps ax`



(RODERICK W, 2001) diz que uma das ferramentas mais poderosas para localizar programas servidores seriam os *scanners* externos como, por exemplo, Nessus (<http://www.nessus.org>) ou Nmap (<http://www.insecure.org/nmap/>). Estes programas executam em um computador diferente daquele que o administrador quer analisar. Dependendo do objetivo do administrador estes *scanners* podem reportar informações como o sistema operacional da máquina e até mesmo se um programa servidor possui alguma vulnerabilidade conhecida. Um exemplo bem básico desta ferramenta seria digitar o seguinte comando usando o nmap: `nmap sub100.com.br`

Após a descoberta dos programas servidores instalados no servidor, é necessário analisar quais não serão usados no sistema, isso pode ser feito com a informação do programas ou no caso do administrador não saber quais programas não vão ser usados, pode-se desabilita-lo e esperar alguma reação de um serviço disponível na máquina caso inutilize aquele serviço será necessário habilitar o programa servidor novamente.

### 3.2 USUÁRIOS

O Linux é um sistema operacional multiusuário, ou seja, pode ter vários usuários acessando informações ao mesmo tempo ou não e o mesmo usuário pode estar conectado varias vezes no mesmo servidor.

Existem três tipos de usuários no sistema operacional Linux, são eles:

- *root*, pode acessar qualquer diretório ou informação contida no servidor Linux como também alterá-la, conseqüentemente, o administrador deve tomar muito cuidado, pois se o *hacker* conseguir acesso a este usuário terá controle total sobre o servidor.

*“O nível de usuário mais alto em uma máquina Linux é chamado de root [...]. O usuário root tem controle total sobre todos os aspectos da*

*máquina – não é possível ocultar nada do root e ele pode fazer o que desejar.” (James Lee, Brian Hatch e George Kurtz, 2001, p. 4).*

- Usuário comum são usuários criados pelo *root*, onde podem se conectar ao servidor, criar e manipular arquivos em seu diretório base (geralmente */home/nome do usuário/*). Normalmente estes usuários têm acesso restrito a arquivos e diretórios no servidor. São essas contas de usuários que as pessoas utilizam para fazer seus trabalhos.
- Usuário do sistema são contas de usuários usadas para propósitos específicos do sistema e não de um usuário comum, elas não se conectam ao servidor. Um exemplo é usuário de sistema é o *lp*, onde este usuário normalmente manipula solicitações de impressão.

Os usuários comuns e o *root* possuem senhas para obter acesso ao servidor Linux, portanto para uma melhor segurança de acesso ao sistema, a senha não pode ser divulgada nem de usuários comuns, e muito menos do *root*, para a própria proteção do sistema, outro ponto importante sobre a senha é que muitos administradores indicam para seus usuários não colocarem senhas pequenas ou senhas que podem ser fáceis de adivinhar como, por exemplo, a data de aniversário do usuário como senha.

Conforme (RODERICK W, 2001), diz que para criar uma boa senha, ela deve conter uma coleção de letras maiúsculas e minúsculas aleatórias, números e quaisquer outros caracteres que o computador permitir, pois tais senhas provavelmente não deverão conter em um dicionário de dados de um *hacker*, por exemplo. Infelizmente este tipo de senha é muito difícil de ser memorizada, por isso é importante que o administrador também informe aos usuários que irão criar as senhas, modos de memorização que ajudarão na descoberta da senha, por exemplo, na frase "Eu casei com Gislaine à mais de 10 anos em Maringá" a senha poderia ser "EcG+10aM".

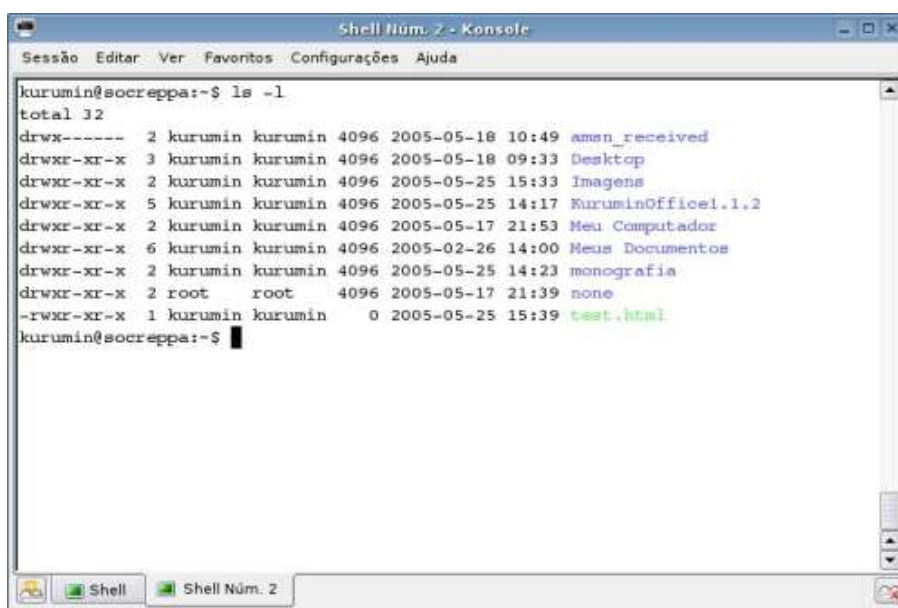
Os usuários do Linux também podem pertencer a grupos de usuários formando uma “família”, onde podem acessar arquivos em comum.

### 3.3 CONTROLE DE ACESSOS SOBRE USUÁRIOS

O controle de acesso de usuários no servidor é muito importante para a segurança, onde podem restringir o que o usuário pode fazer no servidor. Existem vários tipos de controle, que podem ser usados, entre eles, permissão de arquivos e diretórios, cotas de sistema e limites de recurso do sistema.

As permissões de arquivos e diretórios no Linux é um mecanismo de segurança para que só determinado usuário possa ter acesso ao diretório ou arquivo. Para os arquivos um usuário pode especificar quem pode ler, escrever e executar (no caso de programas executáveis). Já para os diretórios um usuário pode especificar quem pode ler o diretório (listar o conteúdo), escrever (adicionar ou remover arquivos no diretório) ou executar programas no diretório.

Para identificarmos como está a permissão de um arquivo ou diretório, é só digitar no terminal `ls -l`. como no exemplo abaixo.



```
Shell Núm. 2 - Konsole
Sessão  Editor  Ver  Favoritos  Configurações  Ajuda

kurumin@socreppa:~$ ls -l
total 32
drwx----- 2 kurumin kurumin 4096 2005-05-18 10:49 amen_received
drwxr-xr-x  3 kurumin kurumin 4096 2005-05-18 09:33 Desktop
drwxr-xr-x  2 kurumin kurumin 4096 2005-05-25 15:33 Imagens
drwxr-xr-x  5 kurumin kurumin 4096 2005-05-25 14:17 KuruminOffice1.1.2
drwxr-xr-x  2 kurumin kurumin 4096 2005-05-17 21:53 Meu Computador
drwxr-xr-x  6 kurumin kurumin 4096 2005-02-26 14:00 Meus Documentos
drwxr-xr-x  2 kurumin kurumin 4096 2005-05-25 14:23 monografia
drwxr-xr-x  2 root   root     4096 2005-05-17 21:39 none
-rwxr-xr-x  1 kurumin kurumin  0 2005-05-25 15:39 test.html
kurumin@socreppa:~$
```

Figura 3.3 – Relação de entradas no sistema de arquivo e suas permissões de arquivos e diretórios

As permissões tanto do arquivo quanto do diretório podem ser identificadas na figura 3.3, primeira coluna do lado esquerdo como, por exemplo, *drwxr-xr-x*, do diretório Desktop.

As informações sobre a permissão do arquivo ou diretório são dividida em quatro partes, o primeiro caractere da esquerda indica o tipo de arquivo (- arquivo normal, *d* diretório, *l* link e *s* soquete), do segundo ao quarto indica sobre a permissão do proprietário, do quinto ao sétimo indica a permissão do grupo do usuário e do oitavo ao décimo a permissão de outros usuários.

As letras *rw*x nas permissões de arquivos e diretórios significam: (r) para leitura, (w) para escrita e (x) para execução.

Por exemplo, no arquivo *test.html*, da figura 2.1, a permissão deste arquivo é *-rwxr-xr-x*, ou seja :

-	<b>rwx</b>	<b>r-x</b>	<b>r-x</b>
Tipo do Arquivo	Permissão do Proprietário	Permissão do Grupo	Permissão dos Outros Usuários

No caso de uma boa regra de permissões, podem aumentar em muito a segurança do servidor Linux contra possíveis ataques de *hackers*.

Usar quotas no servidor Linux, pode evitar que o HD da máquina não se esgote facilmente, com arquivos inúteis que os usuários comuns possam colocar ou que um *hacker*, que consiga o acesso ao sistema através de um usuário comum trave o servidor chegando no limite do HD.

*“As quotas de disco são restrições na quantidade de blocos de espaço em disco e de nós (arquivos, diretórios...) que um usuário possa ter.” (James Lee, Brian Hatch e George Kurtz, 2001, p. 15)*

Ou seja, usar quotas pode diminuir as chances de que o disco não seja exaurido por uma pequena ou grande quantidade de usuários.

Um outro controle para o administrador usar na segurança de seu servidor, seria usar limites para seus usuários, onde o ele pode colocar alguns limites no tamanho do arquivo básico, do segmento de dados, período máximo que aquele usuário possa usar a CPU e a quantidade máxima de arquivos abertos de um usuário.

Usar os limites de usuários pode impedir que um usuário monopolize o servidor e impeça que os demais usuários utilizem o servidor.

### 3.4 GERENCIAMENTO DE CONTAS INATIVAS

Conforme informa (RODERICK W, 2001) as contas de usuários raramente são permanentes, por exemplo, em uma universidade quando o aluno se forma aquela conta criada não será mais utilizada, por isso ela deve ser removida sendo assim diminuirá o risco de ser usada em uma invasão. Mas nem sempre o administrador do sistema conseguirá receber a informação que aquela conta será mais usada, por isso ele pode criar uma senha que expire em determinada data, usando o seguinte comando.

```
[root@socreppa /] usermod -e 2007-12-31 ricardo
```

Este comando diz ao sistema que a conta ricardo se tornará inativa em 31 de dezembro de 2007. Mas existe uma abordagem menos dramática, a fim de não finalizar uma conta em determinada data seria usar o seguinte comando.

```
[root@socreppa /] change -M 30 -W 5 ricardo
```

Já este comando diz ao sistema que ricardo deve mudar a sua senha a cada 30 dias e para avisar em um prazo de impedimento de cinco dias antes de inativar a senha.

Estes dois processos automatizados podem ajudar a reduzir o número de contas inativas no sistema.

### 3.5 MANTER O SISTEMA ATUALIZADO

Muitos sistemas comprometidos devem as suas posições inglórias à falta de manutenção adequada. Alguns minutos gastos com verificações, e instalando *updates* de programas servidores regularmente podem poupar incontáveis horas de trabalho depois, pois, com frequência, os programas incluem correções de *bugs* de segurança, afirma (RODERICK W, 2001).

Os *bugs* de segurança dos programas servidores podem tomar muitas formas e ter muitos tipos diferentes de efeitos, podendo corromper dados, quebrar o programa afetado ou fazer com que se comporte de maneira estranha. Estes *bugs* podem, por exemplo, permitir que uma pessoa escreva arquivos arbitrários em lugares arbitrários, permitindo que se altere a configuração de um programa servidor ou até mesmo dar ao invasor a habilidade de executar programas sob algum outro nome de usuário e até mesmo a *root*.

Por causa dos *bugs* de segurança é necessário que o administrador sempre atualize os programas servidores. Para que isso aconteça é necessário monitorar sempre quando um determinado programa foi modificado por causa de uma falha de segurança.

(RODERICK W, 2001) afirma que é possível fazer o monitoramento de atualizações de programas servidores de várias maneiras:

- **Pacote de software de Web sites e mailing lists:** As maiorias de pacotes de software incluindo os programas servidores possuem Web sites oficiais e mailing

*lists*, onde é possível conseguir uma informação ou atualização de *bug* de segurança;

- **O Web site da distribuição do Linux:** Todas as distribuições possuem uma pagina na *Web*, e onde é possível conseguir uma informação ou atualização de *bug* de segurança de um determinado programa servidor;
- **Fontes de Informações de segurança genérica:** Sites e livros voltados a segurança podem ser uma fonte de informação para atualizar os programas servidores.

Geralmente usar estas três abordagens citadas a cima podem ajudar o administrador a deixar seu sistema atualizado.

## RESUMO DO CAPÍTULO

Neste capítulo foram apresentados como desativar os programas servidores desnecessários, tipos de usuários que o servidor Linux possui e como pode ser feito o controle se acesso dos mesmos na máquina, uma vez que, este tipo de controle pode assegurar que o servidor não possa ser travado, acabar com seu espaço em disco e permitir que um usuário não acesse os arquivos e diretórios dos outros, como desativar contas necessárias e por fim salientar a importância de deixar um sistema sempre atualizado.

No próximo capítulo será abordado como pode ser feita a segurança do servidor Linux, através de uma implementação de *firewall* e como ele pode ser classificado de acordo com base na forma de bloqueio de pacotes.

## CAPÍTULO 4 – FIREWALLS NA SEGURANÇA DO SERVIDOR

### INTRODUÇÃO

O *firewall* tem como objetivo aumentar a segurança do servidor Linux ou da rede local, que possui conexão com redes inseguras, como por exemplo, a internet. O objetivo deste capítulo é apresentar os conceitos relacionados ao *firewall*, como funcionam, como podem ser configurados e como podem ajudar na segurança.

#### 4.1 FIREWALLS

O firewall é um dispositivo com a função de servir como ponto único de entrada em uma rede interna ou servidor, onde quando configurado corretamente, deve aceitar somente conexões requisitadas por *hosts* autorizados, as demais serão descartadas (figura 4.1). Este dispositivo pode ser um computador, um roteador ou uma solução proprietária (*Hardware + Software*).

(RODERICK W, 2001) afirma que tradicionalmente um *firewall* é um computador que fica entre duas redes e controla o acesso entre elas. Essa função é semelhante à de um roteador, onde a única diferença é que o *firewall* pode bloquear computadores em uma rede de acessar servidores especiais em outra rede.

*“Antigamente, parede de tijolos eram construídas entre construções em complexos de apartamentos de forma que se ocorresse um incêndio ele não poderia se espalhar de uma construção para outra. De uma forma completamente natural, as paredes foram chamadas de firewall” (SIYAN, Karanjit, 1995, p. 11)*

*“A solução de firewall é usada para criar uma barreira entre a uma rede insegura e a rede local da empresa. Esta barreira não permite a passagem de tráfego indesejado de informações e faz com que certos serviços ou dados sejam acessados somente a partir da rede privada e/ou por pessoas autorizadas” (Magrin, Maria Heloiza, 2004, p. 120)*



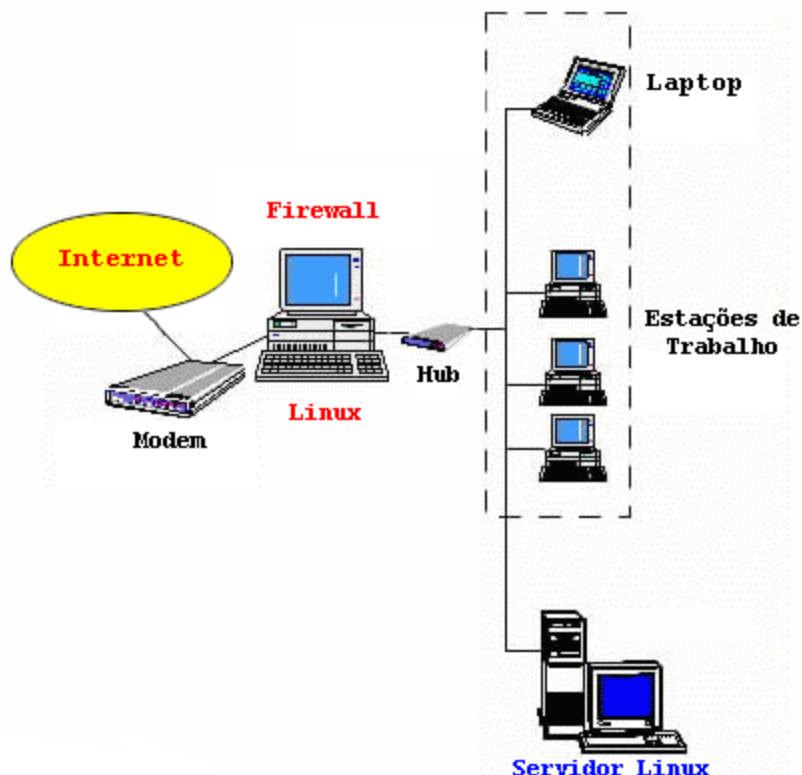


Figura 4.1 – Representação de uma rede protegida por um Firewall

O *firewall* pode ser classificado em dois grandes tipos: baseados em filtragem de pacotes e baseados em aplicações, onde esta classificação é obtida através de como bloqueiam o tráfego entre redes diferentes.

- **Firewalls baseados em filtragem de pacotes:** Utiliza endereços IP e portas de acesso, através de um conjunto de regras estabelecidas pelo administrador, bloqueiam ou permitem o tráfego entre duas redes, geralmente a Internet.
- **Firewalls baseados em aplicações:** Os *firewalls* baseados em aplicações trabalham como se fosse um intermediador nas comunicações entre duas redes. Verifica as requisições provenientes de usuários remotos e bloqueiam ou não a sua utilização. O cliente e o servidor não conversam diretamente, o servidor *proxy* intermédia a conexão e analisa de acordo com as regras definidas, a autorização para a conexão, permitindo ou bloqueando.

O ponto mais importante na implementação de um *firewall* é a construção de um conjunto de regras aplicadas nos pacotes (*chain*), que reflitam de maneira real e segura o modo como usuários devem acessar a rede externa (Internet) e a rede interna (Intranet).

## 4.2 FERRAMENTAS DE IMPLEMENTAÇÃO DE FIREWALL NO LINUX

O sistema operacional Linux por ser um software livre, onde uma comunidade de programadores contribui para seu desenvolvimento e melhorias. Quando se fala em desenvolvimento e melhorias do *Kernel* se têm uma preocupação com *firewalls* e segurança.

Ao longo do tempo o *Kernel*, vem sofrendo várias modificações, sendo assim o aparecimento de várias versões se tornou freqüente. Junto com essas versões apareceram várias ferramentas de *firewalls* novas para a melhoria da segurança. São elas:

- ***Ipfwadm*** - O *IP Firewall Administration*, ou simplesmente *ipfwadmin* foi a ferramenta padrão para construção de regras de *firewall*, para o *Kernel* anterior à versão 2.2..
- ***Ipchains*** - O *ipchains* foi a atualização, feita para o kernel 2.2 do *ipfwadm*. A idéia do *ipchains* foi ter o poder do *ipfwadm*, mas com uma simplicidade e facilidade no que diz respeito à criação de regras.
- ***Iptables*** - A nova geração de ferramentas de *firewall* para o *Kernel* 2.4 ou 2.6 do Linux. Além de possuir a facilidade do *ipchains*, implementa a facilidade do NAT e filtragem de pacotes mais flexíveis que o *ipchains*.

Como as versões do *Kernel*, estão nas 2.4 e 2.6, a grande maioria dos servidores utiliza o *iptables* para configurar seus *firewalls*.

O Funcionamento do *firewall* é explicado por (RODERICK W, 2001) da seguinte forma representada pela figura 4.2: Quando o pacote é processado na rede inicialmente é tomada uma decisão de roteamento, ou o pacote vai um computador local ou para um outro computador fora da rede. Dependendo da resposta o pacote é passado por duas *chain*, *INPUT* ou a *FORWARD*, cada uma delas pode processar ou modificar os dados entrando de várias maneiras, mas o padrão é não modificar os dados. Caso o destino do pacote seja, os processos locais (Ex. *Netscape*, *telnet*, *Apache*...) ele passa pela *chain*, *INPUT*, se não vai ser roteado pela *chain* *FORWARD*, e por fim, qualquer uma das *chains* que o pacote passe vai ser encaminhado para a *chain* *OUTPUT* e depois sairá da rede.

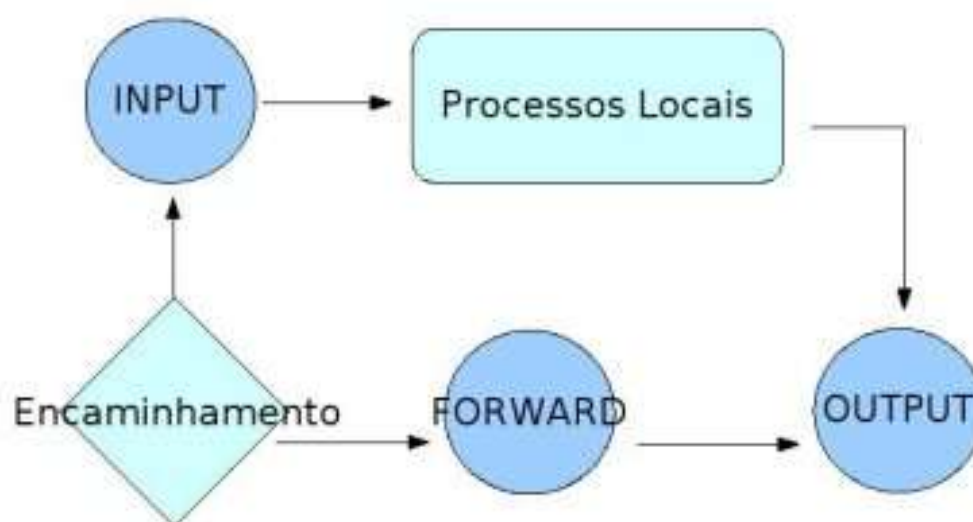


Figura 4.2 - Funcionamento do Firewall

Cada *chain* é uma coleção de regras, que é combinada, por sua vez, com a entrada do pacote. Caso a regra seja combinada com o pacote, é aplicada uma ação chamada de *target*.

As *chain* mostradas na figura 4.2, segundo (RODERICK W, 2001) têm as seguintes funções:

- **INPUT**: tem a função de proteger os processos locais;
- **OUTPUT**: tem a função de bloquear acessos de saída indesejados;
- **FORWARD**: tem a função de intervir nos encaminhamentos dos pacotes.

As *targets*, mais usadas são o ACCEPT (aceita o pacote para processamento), DROP (ignora o pacote), QUEUE (passa o pacote para um programa de espaço de usuário) e RETURN (para o processamento da *chain* e retorna a *chain* que chamou a *chain* atual). Alguns *targets*, adicionais podem ser ativados no *kernel* para também serem usadas nas regras, como por exemplo a REJECT (rejeita o pacote e diz ao remetente que ele foi rejeitado), MASQUERADE (usado em NAT) e LOG (usado para informações de registro sobre filtragem de pacote).

As *chains* são organizadas em tabelas. As três *chains* mostradas na figura 4.2 formam a tabela *filter*, onde gerencia a maior parte do tráfego padrão. As outras duas tabelas padrões, a nat (usado em NAT) e a mangle (que é usada para alterações especializadas de pacote). Importante ressaltar que é possível colocar novas *chains* dentro de uma tabela e chamá-las a partir de *chains* existentes, ou seja, é possível que o administrador crie padrões de processamento especializados e complexos, para filtrar ou alterar dados do pacote.

As tabelas e as *chains* são recursos do *kernel* do Linux e o *iptables* é a ferramenta que o administrador vai usar para manipula-la, ou seja, ele poderá acrescentar regras em qualquer uma das *chains* mostradas na figura 4.2 ou em outras. Por exemplo, o administrador poderia acrescentar uma regra na *chain* INPUT, para bloquear todos os pacotes direcionados a uma determinada porta da rede ou acrescentar na *chain* OUTPUT uma regra que interrompa pacotes direcionados a sistemas que usuários da rede não possam se comunicar.

### 4.3 CONFIGURAÇÃO DE UM FIREWALL COM IPTABLES

Antes mesmo de começar a configuração de um *firewall*, é preciso verificar se o *iptables* já está instalado na máquina, para isso o administrador pode digitar o comando `# iptables -L -t filter`. Este comando mostra as *chains* da tabela *filter*, mas não as regras contidas em cada *chain*.

#### 4.3.1 Ajustar uma política padrão de *firewall*

A primeira etapa na configuração de um *firewall*, seria ajustar a política padrão. Essa é uma declaração do que o *firewall* tem que fazer com pacotes que não combinam com as regras. No *iptables* esta política pode ser ajustada através da opção `-P`, por exemplo:

```
[root@socreppa /] # iptables -P INPUT DROP
[root@socreppa /] # iptables -P OUTPUT DROP
[root@socreppa /] # iptables -P FORWARD DROP
```

O exemplo mostrado a cima ajusta a política padrão separadamente em cada uma das três *chains* da tabela *filter*.

As configurações mais seguras de um *firewall* usam DROP ou REJECT, como política padrão, pois significa que qualquer tipo de pacote que não é permitido em etapas subsequentes, é bloqueado.

#### 4.3.2 Criação de regras de *firewall*

Para criar regras de *firewall* no *iptables* é necessário usar o comando `--append` (`-A` opção resumida), em qual *chain* vai ser adicionado à regra, depois o critério de avaliação do pacote e por fim o comando `--jump` (`-j` opção resumida) para dizer qual ação (*target*) vai ser usada no pacote.

```
[root@socreppa /] # iptables --append CHAIN {Avaliação} --jump TARGET
[root@socreppa /] # iptables -A CHAIN {Avaliação} -j TARGET
```

Mais comandos e alguns parâmetros serão mostrados nos anexos deste trabalho.

#### 4.3.3 Abertura e fechamento de portas específicas

Uma maneira de filtrar pacotes seria usar portas específicas de fonte e destino, por exemplo, o administrador pode configurar o servidor de e-mail para passar pacotes enviados à porta 25, através dos seguintes comandos:

```
[root@socreppa /] # iptables -A INPUT -p tcp -dport 25 -j ACCEPT  
[root@socreppa /] # iptables -A OUTPUT -p tcp -dport 25 -j ACCEPT
```

Nos comandos mostrados acima, o *kernel*, aceita os pacotes direcionados à porta 25, para as suas *chains* locais, INPUT e OUTPUT. Mesmo que a política padrão estiver ajustada para bloquear pacotes, o servidor aceitará correspondências e também será capaz de enviar.

#### 4.3.4 Usar endereço *IP* como fonte de destino

É possível combinar os pacotes de endereço *IP* de fonte e destino ou blocos de rede, por exemplo, se o administrador souber que o bloco de rede 192.168.0.0/24 é usado por pessoas que não têm permissão para utilizar o sistema, ele poderia bloquear todo o acesso a partir daquele bloco de rede através dos seguintes comandos:

```
[root@socreppa /] # iptables -A INPUT -s 192.168.0.0/24 -j DROP  
[root@socreppa /] # iptables -A OUTPUT -s 192.168.0.0/24 -j DROP
```

Essa opção também pode ser usada junto com conexões baseadas em portas, onde poderá permitir que apenas computadores específicos acessem determinadas portas do servidor, especialmente em configuração padrão DROP.

```
[root@socreppa /] # iptables -A FORWARD -s 192.168.0.0/24 -p tcp \ --destination
-port 22 -j ALLOW
[root@socreppa /] # iptables -A FORWARD -s 192.168.0.0/24 -p tcp \ --source -
port 22 -j ALLOW
```

#### 4.3.5 Filtrando pela interface

Uma outra opção de filtragem seria usar a interface de rede (Ex.ppp0, eth1...), como identificador. Esta opção é mais útil em um roteador ou um servidor com múltiplas interfaces. Filtrando a interface pode ajudar o administrador a proteger o sistema contra uma possível invasão, onde um sistema remoto tenta contatar um roteador usando um endereço da rede local, na esperança de que o ele tenha uma configuração mais vaga para computadores locais.

Este tipo de filtragem pode ser feito através do seguinte comando:

```
[root@socreppa /] # iptables -A INPUT -s 192.168.0.0/24 -i eth0 -j DROP
[root@socreppa /] # iptables -A FORWARD -s 192.168.0.0/24 -i eth0 -j DROP
[root@socreppa /] # iptables -A INPUT -s !192.168.0.0/24 -i eth1 -j DROP
[root@socreppa /] # iptables -A FORWARD -s !192.168.0.0/24 -i eth1 -j DROP
```

Os dois primeiros comandos protegem pacotes direcionados ao roteador e sistemas internos a partir de sistemas externos com interface eth0, que reivindicam ser sistemas internos. Já os dois últimos comandos evitam os sistemas internos com interface eth1 de reivindicar endereços *IP*, fora da rede local.

#### 4.4 TABELA NAT

Segundo (RODERICK W, 2001) a tabela *NAT* permite que um roteador mude o conteúdo de pacotes TCP/IP, ou seja, tem a capacidade de mudar os endereços de fonte e de destino do pacote. Neste caso para a usar a tabela *NAT* e necessário ter um roteador com duas interfaces de rede , sendo uma para a rede interna e outra para a rede externa.

Para entender o funcionamento da tabela *NAT*, é preciso analisar a seguinte transação: Caso o usuário tente se conectar com um site externo (200.250.8.4 – [www.wnet.com.br](http://www.wnet.com.br)) . O *browser* gera um pacote de solicitação *HTTP*, endereçando de seu endereço IP (192.168.99.3). O cliente envia a solicitação ao roteador *NAT*. Quando o roteador receber este pacote, irá fazer uma análise e depois mudar o endereço *IP* fonte para um endereço externo próprio do roteador (10.9.18.7). e envia os pacotes para o endereço solicitado pelo usuário. Caso o endereço retorne uma informação o roteador vai fazer o processo inverso e finalmente chega à resposta ao usuário, esta representação pode ser vista na figura 4.3 .

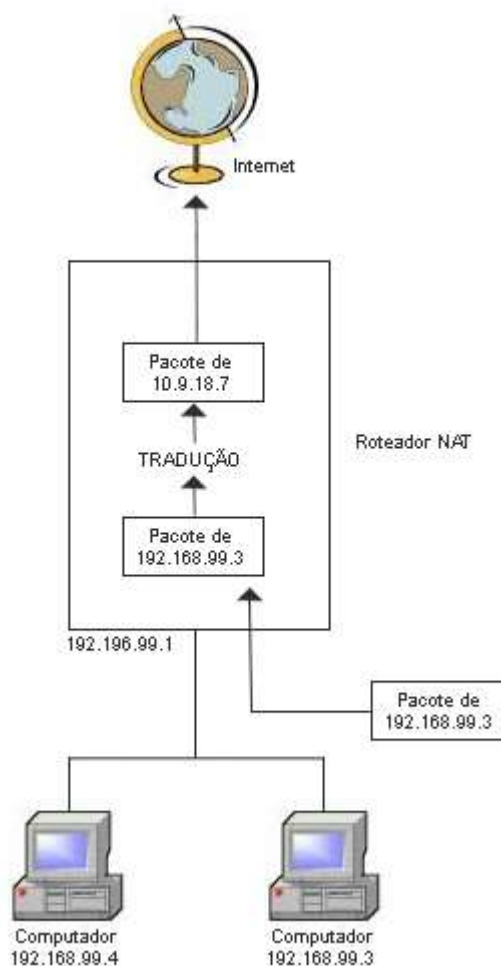


Figura 4.3 – Funcionamento de um Roteador NAT



Conforme (RODERICK W, 2001), algumas formas de *NAT*, em especial mascaramento de *IP* (figura 4.3), oferece um benefício extra: proteção automática como *firewall* de rede privada. Para fazer com que o *NAT* faça este mascaramento é necessário o administrador digitar o seguinte comando:

```
[root@socreppa /] # iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

No comando acima a interface *eth1* é aquela que se conecta na rede externa.

Nos anexos será mostrado como instalar o *iptables*, e como definir algumas regras de *firewalls*.

## RESUMO DO CAPÍTULO

Neste capítulo foi apresentado como os *firewalls* funcionam, como pode ser configurado, sua classificação de acordo com e feito o bloqueio dos pacotes e algumas ferramentas que podem ser utilizadas para a implementação do mesmo.

No próximo capítulo será abordado como o sistema operacional Linux utiliza a criptografia para sua segurança, tipos de criptografia e os algoritmos usados pelo Linux.

## CAPÍTULO 5 – CRIPTOGRAFIA NO LINUX

### INTRODUÇÃO

A criptografia é um método de segurança muito importante no sistema Linux, onde as senhas dos usuários são criptografadas, impedindo que usuários comuns ou *hackers* a descubram. O objetivo deste capítulo é mostrar para o administrador como a criptografia funciona e como ela é usada no sistema operacional Linux.

#### 5.1 CRIPTOGRAFIA

Criptografia é um método que modifica o texto original da mensagem a ser transmitida (texto normal), gerando texto criptografado na origem, através de um processo de codificação definido por um método de criptografia, ou seja, é escrever texto através de códigos conhecidos apenas pelos interessados.

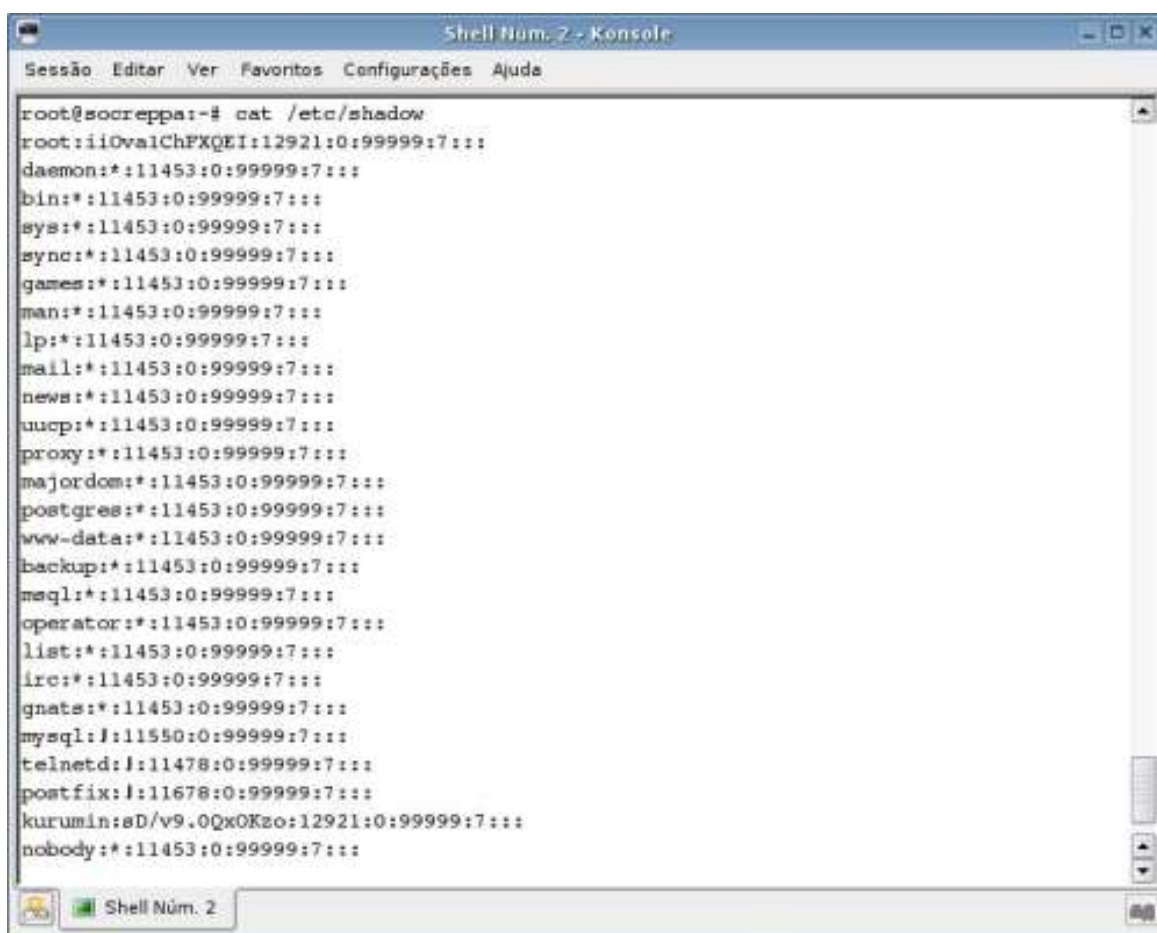
*“A criptografia envolve a conversão de uma string de texto, com base em um algoritmo repetível, em uma forma que é muito deferente da string original.” (James Lee, Brian Hatch e George Kurtz, 2001, p. 250)*

A criptografia pode ser dividida em duas classes, uma denominada Simétrica e outra chamada Assimétrica.

- **Chave Simétrica ou Chave Privada:** É a técnica de criptografia, onde utilizam a mesma chave para criptografar e decriptografar os dados. Sendo assim, a manutenção da chave em segredo é fundamental para a eficiência do processo. Ex. IDEA (*International Data Encryption Algorithm*), DES (*Data Encryption Standard*).
- **Chave Assimétrica ou Chave Pública:** É a técnica de criptografia, onde as chaves utilizadas para criptografar e decriptografar são diferentes, sendo, no

entanto relacionadas. A chave utilizada para criptografar os dados é formada por duas partes, sendo uma pública e outra privada, da mesma forma que a chave utilizada para decryptografar. Ex. RSA, ElGamal, Diffie-Hellman.

A segurança para as senhas dos usuários Linux é uma das medidas mais importantes para o sistema operacional. Portanto o Linux utiliza a criptografia para proteger as senhas dos usuários que são armazenadas no arquivo `/etc/shadow` (figura 5.1), sendo que essa criptografia é gerada por um algoritmo criptográfico unidirecional, ou seja, pode criptografar uma senha, mas não pode gerar uma senha de um valor criptográfico.



```

root@socreppa:~# cat /etc/shadow
root:$iOvalChFXQEI:12921:0:99999:7:::
daemon:*:11453:0:99999:7:::
bin:*:11453:0:99999:7:::
sys:*:11453:0:99999:7:::
sync:*:11453:0:99999:7:::
games:*:11453:0:99999:7:::
man:*:11453:0:99999:7:::
lp:*:11453:0:99999:7:::
mail:*:11453:0:99999:7:::
news:*:11453:0:99999:7:::
uucp:*:11453:0:99999:7:::
proxy:*:11453:0:99999:7:::
majordom:*:11453:0:99999:7:::
postgres:*:11453:0:99999:7:::
www-data:*:11453:0:99999:7:::
backup:*:11453:0:99999:7:::
mysql:*:11453:0:99999:7:::
operator:*:11453:0:99999:7:::
list:*:11453:0:99999:7:::
irc:*:11453:0:99999:7:::
gnats:*:11453:0:99999:7:::
mysql:!:11550:0:99999:7:::
telnetd:!:11478:0:99999:7:::
postfix:!:11678:0:99999:7:::
kurumin:sD/v9.0QxOKzo:12921:0:99999:7:::
nobody:*:11453:0:99999:7:::

```

Figura 5.1 Arquivo `/etc/shadow` onde é armazenado as senhas criptografadas dos usuários

Um algoritmo de criptografia é uma fórmula que pode ser repetida para converter em uma forma muito diferente da original. Existem muitos algoritmos de criptografia diferentes, dos mais simples e fáceis de decifrar aos mais complicados e virtualmente impossíveis de se decifrar.

O Linux utiliza dois algoritmos para criptografar senhas: DES e MD5.

- A **Data Encryption Standard (DES)** é um algoritmo simétrico. Foi criado pela IBM em 1977 e, apesar de permitir cerca de 72 quadrilhões de combinações (256), seu tamanho de chave (56 bits) é considerado pequeno, pois ele tem um tamanho igual a 13 caracteres.
- O **MD5** é assimétrico, sendo considerado algoritmo de *hashing* e não de criptografia. Entretanto como o DES, converte a senha em uma forma que não pode ser decifrada. Foi inicialmente proposto em 1991 por Ron Rivest, do MIT, que também trabalha para a *RSA Data Security*. A sigla MD significa *Message Digest*. Este algoritmo produz um valor hash de 128 bits e seu tamanho é bem maior que 13 caracteres.

## RESUMO DO CAPÍTULO

Neste capítulo foi apresentado como a criptografia funciona, suas categorias e como ela é usada para proteger as senhas dos usuários no servidor Linux.

No próximo capítulo será abordado como os softwares de detecção de intrusos podem ajudar na segurança do servidor, seu funcionamento e tipos.

## CAPÍTULO 6 – SOFTWARE DE DETECÇÃO DE INTRUSOS

### INTRODUÇÃO

A metodologia de detecção de intrusão tem como objetivo detectar atividades que violem a segurança do servidor. O objetivo deste capítulo é mostrar para o administrador, o que são softwares de detecção de intrusos (*IDS*), como funcionam e suas categorias.

#### 6.1 DETECÇÃO DE INTRUSOS (*Intrusion Detection System - IDS*)

Detecção de intrusão é um processo de coleta de informações que procura identificar sinais de que um ataque está iniciando ou ocorrendo.

*“(...) a detecção de intrusão da rede permite identificar e reagir a ameaças contra o seu ambiente (...)” (NORTHCUTT, 2002, p. 156).*

O Sistema de Detecção de Intrusão (*IDS - Intrusion Detection Systems*) é uma ferramenta de gerenciamento de segurança que auxilia e automatiza o processo de monitorar os eventos ocorridos em uma rede.

O IDS pode detectar ataques e violações de segurança, código de vírus antes de infectar toda a rede (ex: worm NIMDA), armazenar os eventos em arquivos para posterior análise permitindo estatísticas dos eventos ocorridos na rede.

O IDS pode trabalhar basicamente de duas maneiras, analisando o tráfego da rede (baseado em rede) ou analisando uma determinada máquina (baseado em host) a procura de códigos maliciosos para identificar sinais de que um ataque está sendo iniciado. Estes dois modelos serão comentados posteriormente. Com esta análise é possível detectar sinais de ataques internos e externos.

Em uma pesquisa realizada pela Módulo Security Solutions (MODULO, 2003), indica que em 2004, a utilização de IDS nas empresas ocupa o nono lugar no **TOP 10 MEDIDAS DE SEGURANÇA 2004**, conforme a tabela abaixo.

MEDIDAS	PORCENTAGEM (%)
ANTIVÍRUS	76
CAPACITAÇÃO TÉCNICA	75
SISTEMAS DE BACKUP	72
POLÍTICA DE SEGURANÇA	71
PROCEDIMENTOS FORMALIZADOS	71
IMPLEMENTAÇÃO DE FIREWALL	71
ANALISE DE RISCOS	66
CRIPTOGRAFIA	64
<b>SISTEMAS DE DETECÇÃO DE INTRUSOS</b>	<b>63</b>
SOFTWARE DE CONTROLE DE ACESSOS	58

Figura 6.1 – Representação da Tabela TOP 10 – Medidas de Segurança 2004

O IDS vem ocupando um lugar de destaque na proteção do servidor, auxiliando a monitorar e tomar medidas de contra ataques realizados, permitindo ao administrador do servidor maior rapidez no combate às invasões.

Encontrando algum indício de ataque, o IDS é capaz de tomar algumas ações de contra-ataque. As ações de contra-ataque podem ser (@BSOLUTA, 1999):

- Ativação de alertas nas estações de gerência via SNMP;
- Reconfiguração de elementos de rede como *firewall*;
- Encerramento da conexão através do envio de pacotes de *reset* para a máquina atacante e para a máquina atacada, com o objetivo de encerrar a conexão entre eles.

Um IDS não pode ser usado com a única fonte de segurança de um servidor, nem em substituição a um *firewall*, mas sim em conjunto com outros métodos para aumentar a segurança do servidor.

Um IDS é composto basicamente por dois dispositivos principais, o Console de Comando e o Sensor. O console de comando, ou simplesmente console, tem como função permitir o controle do IDS, monitorar o estado do sensor e processar os alertas enviados pelo sensor. O sensor é o dispositivo responsável pela coleta de informação para análise de descoberta de uma invasão.

## 6.2 TIPOS DE IDS

Existem dois tipos de IDS: o baseado em Rede e o baseado em Host. Qual modelo escolher depende da estrutura de cada empresa, podendo até ser instalado os dois modelos trabalhando no mesmo servidor.

### 6.2.1 Baseado em Rede

É composto geralmente por sensores, que são responsáveis por escutar o tráfego que passa pelo segmento de rede e a estação de console que recebe os alertas.

*“O sensor de IDS da rede fareja o tráfego e o analisa, procurando várias assinaturas que poderiam indicar um scan ou uma sonda, atividade de reconhecimento ou tentativa de explorar uma vulnerabilidade. Tradicionalmente, os sistemas de detecção de intrusão não interferem com o tráfego de alguma forma. Ao contrário de um firewall ou filtro de pacotes, que toma decisões sobre qual tráfego permitir, um sensor de detecção é, na realidade, um farejador de pacotes que também realiza análise. Contudo alguns sensores estão começando a dar respostas ativas para o tráfego suspeito, como ao terminar conexões TCP suspeitas.” (NORTHCUTT, 2002, p 156).*

O sensor trabalha em modo promíscuo, isso quer dizer que todos os pacotes que circulam pelo segmento de rede são capturados, independentes de qual era o destino do pacote. É possível instalar um ou mais sensores por segmento de rede, como vantagem se tem a tolerância à falhas, mas muitas vezes o custo impede esta opção.

### 6.2.2 Baseado em Host

Monitora o tráfego de máquinas individuais, permitindo uma melhor precisão na análise e gerando menos falso positivo.

Podem trabalhar de três formas:

- Verificando a integridade do sistema de arquivos: procuram por mudanças não autorizadas no sistema de arquivo, a partir de uma base criada do sistema quando considerado confiável. É configurável, sendo possível indicar quais arquivos ou diretórios podem sofrer alterações, diminuindo assim os alertas;
- Verificando a conexão da rede: verifica as conexões do *host* a procura de ataques ou atividades maliciosas. Tem menos problemas com a sobrecarga de tráfego, pois monitora, somente o tráfego destinado a um determinado *host*;
- Verificando os arquivos de *log*: observam o conteúdo dos *logs* e avisam quando algo suspeito é detectado. Possui uma vantagem, se vários *hosts* salvarem seus *logs* em um único ponto este sistema pode monitorar mais que um *host*.

## 6.3 MÉTODOS DE DETECÇÃO DOS IDS

Os alertas são gerados através de um destes dois métodos de detecção: Baseado em Assinaturas e Baseado em Anomalias.

### 6.3.1 Baseado em Assinaturas



Este método trabalha procurando regras pré-estabelecidas no tráfego da rede. Quando é encontrado algum código na rede que esteja descrito em alguma regra, é gerado um alerta ou evento que permita uma ação defensiva.

As assinaturas são fornecidas pelos desenvolvedores do IDS, mas podem ser desenvolvidas pelo próprio administrador do servidor ou encontradas em sites especializados em segurança, neste caso, pode ser necessário adaptar a regra para o IDS específico.

É possível comparar este método com um antivírus, que trabalha analisando cada pacote e comparando com as regras existentes, assim para criar uma regra se deve analisar as características do pacote que contém o ataque.

É necessário ter um certo cuidado na criação da regra, uma falha pode comprometer a segurança porque não vai detectar o ataque corretamente. É interessante testar a regra depois de criada.

### 6.3.2 Baseado em Anomalia

Possui uma base de dados do comportamento da rede, a partir desta base é que o sistema verifica o que é ou não permitido e quando encontra algo fora do padrão gera o alerta. São mais complicados de configurar, pois é mais difícil estabelecer o que é padrão em uma rede com muitos usuários.

Uma vantagem deste método é que ele pode detectar ataques desconhecidos, já que não trabalha analisando uma regra específica.

Ex.: O sistema tem em sua base de dados que um determinado usuário sempre só acessou a Internet pela parte da manhã, então se for detectado para este usuário um acesso à tarde o sistema irá gerar um alerta.

## 6.4 SNORT

No sistema operacional Linux, existem vários softwares de detecção de intrusos que podem ser utilizados como, por exemplo: *RealSecure*, *eTrust*, *PortSentry*, *Prelude*, *SNORT* e etc. Esta monografia irá mostrar as características do *SNORT*, bem como a instalação e configuração que será mostrada nos anexos.

O *SNORT* é um sistema de detecção de intrusão baseado em rede amplamente utilizado. Foi desenvolvido por Marty Roesch em 1998. Possui uma arquitetura simples baseada em *plugins*, executando basicamente as funções de captura de pacotes na rede, análise dos pacotes e geração de alertas. É um sistema leve, capaz de trabalhar em grandes redes e detectar uma grande variedade de ataques em tempo real, sendo o seu sistema de detecção baseado em assinaturas.

Estouro de *buffers*, varredura de portas (*portscan*) e ataques CGI (*Common Gateway Interface*), são alguns exemplos de ataques que o *SNORT* pode detectar.

O *SNORT* pode ser configurado para trabalhar de três modos:

- *Sniffer*: simplesmente lê os pacotes da rede e mostra o resultado na console do programa gerenciador;
- Gerador de log de pacotes: trabalha de maneira semelhante ao modo *sniffer*, porém armazena todos os pacotes em arquivo para uma análise futura;
- Detector de intrusão: é o método mais flexível e completo para analisar o tráfego da rede. Pode-se definir novas regras de detecção além das já disponíveis. Neste modo, somente são gerados alertas ou armazenados no log os pacotes definidos nas regras.

## **RESUMO DO CAPÍTULO**

Neste capítulo foi apresentado como os detectores de intrusos funcionam, seus tipos e como instalar e configurar um IDS para aumentar a segurança no servidor Linux.

## CAPÍTULO 7 – CONCLUSÃO

Neste trabalho foram apresentados diversos métodos de ataque que um *hacker* pode utilizar para conseguir o acesso ao servidor ou uma máquina comum, como também diversas técnicas usadas para proteger o servidor Linux de ataques.

Todo esforço feito para se manter um elevado nível de segurança corporativo é importante. Os *hackers* estão sempre buscando e testando novas ferramentas e vulnerabilidades que possam ser usadas e exploradas contra qualquer sistema computacional, esteja ele despreparado e vulnerável ou não. As técnicas usadas para burlar os sistemas de segurança estão cada vez mais sofisticadas, permitindo que qualquer pessoa mal intencionada possa iniciar um ataque, seja por vingança, brincadeira ou espionagem industrial.

Um fato percebido é que segurança total não existe, para que haja segurança é necessário um conjunto de práticas que sejam executadas periodicamente buscando sempre atualizar e corrigir falhas dos sistemas existentes. Por exemplo, o administrador sempre terá que atualizar seus sistemas todos os dias, pois milhares de *hackers* estão em busca de falhas no sistema para conseguir o acesso.

Pode-se perceber que a segurança de um servidor ou das máquinas comuns não depende somente de administrador e sim de todos os funcionários da empresa, uma vez que todos utilizam o sistema e dependem dele para trabalhar. Por isso umas das primeiras medidas de segurança que o administrador deve tomar é criar uma política de segurança, através dela ele poderá conscientizar os demais funcionários que se seguirem esta política, diminuirá em muito os riscos de invasão.

Uma vez criada a política de segurança, o administrador pode partir para a segurança básica que consiste em delimitar através de permissões, o que o usuário pode fazer. Este tipo de segurança pode, por exemplo, bloquear os com que os funcionários tentem obter informações sigilosas a fim de vender para um concorrente.

Este trabalho trouxe também a importância de um *firewall* em uma rede interna onde ele deve estar bem configurado uma vez que vai proteger a rede interna e o servidor de ameaças da rede externa (*internet*).

O *IDS* não deverá substituir o *firewall* na tarefa de proteger as redes de dados, devendo agir em conjunto com ele e outras ferramentas de segurança de maneira a garantir a execução da política de segurança corporativa. Outro dado importante é que o sistema de detecção de intrusos, apenas mostra quem esta tentando invadir o sistema ou quem já invadiu, não previne os ataques, daí a importância deste recurso ser usado de forma complementar aos outros recursos de segurança apresentados neste trabalho.

## BIBLIOGRAFIA

Weber, Raul Fernando. **Segurança na Internet**. Anais da XIX JAI - Jornada De Atualização em Informática. PUCPR campus Curitiba PR, 17 a 21 de julho de 2000.

HATCH, BRIAN; LEE, JAMES; KURTZ, GEORGE. **Hackers Expostos – Linux**, São Paulo, Makron Books, 2002.

SIYAN, Karanjit, **Internet Firewalls and Network Security**, New Riders Publishing, Indianapolis, 1995.

MAGRIN, Maria Heloiza, **Guia do Profissional Linux**, Digerati books, São Paulo, 2004.

NORTHCUTT, Stephen. **Como detectar invasão em rede**. Rio de Janeiro: Ciência Moderna, 2000.

NORTHCUTT, Stephen; ZELTSER, Lenny et al. **Desvendando segurança em redes**. Rio de Janeiro: Editora Campus, 2002.

@BSOLUTA, Verdade. **Sistema de detecção de intrusão e aspectos legais**, 1999 Disponível em: [www.absoluta.org/seguranca/seg\\_ids.htm](http://www.absoluta.org/seguranca/seg_ids.htm) Acesso em: 18 março. 2005.

MODULO Security. **Nona pesquisa nacional de segurança da informação**, Setembro de 2003. Disponível em: [www.modulo.com.br/index.jsp](http://www.modulo.com.br/index.jsp). Acesso em: 20 março. 2005.

DIAS, Claudia. **Segurança e auditoria da tecnologia da informação**. Rio de Janeiro: Axcel Books, 2000.

JUNIOR, José Teixeira Helvécio et al. **Redes de computadores**. São Paulo: Makron Books, 1999.

NIC BR Security Office, **Cartilha de Segurança para Internet**, Março de 2003. Disponível em: <http://www.nbso.nic.br/docs/cartilha/>. Acesso em: 23 de maio. 2005.

W. SMITH, Roderick. **Redes Linux avançadas**. Rio de Janeiro: Editora Ciência Moderna

## ANEXOS

### IPTABLES

Para instalar o *iptables* independente da distribuição de Linux usada devemos digitar o seguinte comando:

```
[root@socreppa /] # rpm -Uvh iptables.rpm
```

Após a instalação do pacote *iptables*, ele já está pronto para as configurações iniciais, agora esta monografia irá mostrar alguns comandos de configurações para proteger seu servidor de alguns ataques *hackers*, específicos, são eles:

- **Restrição de acesso ao servidor por IPs**

```
[root@socreppa /] # iptables -A OUTPUT -p icmp -d ! 192.168.1.0/24 -j ACCEPT
```

Neste caso a regra acima faz com que todos os pacotes vindos da classe ip 192.168.1.1 à 192.168.1.255 sejam aceitos.

- **Ping da Morte**

```
[root@socreppa /] # iptables -A FORWARD -p icmp --icmp-type echo-request -m limit --limit 1/s -j ACCEPT
```

- **Backdoors**

**(Porta FTP)**

```
[root@socreppa /] # iptables -A INPUT -p tcp --dport 21 -j LOG --log-prefix
```

**(Porta Wincrash)**

```
[root@socreppa /] # iptables -A INPUT -p tcp --dport 5042 -j LOG --log-prefix
```

**(Portas BackOrifice)**

```
[root@socreppa /] # iptables -A INPUT -p tcp --dport 12345 -j LOG --log-prefix
```

```
[root@socreppa /] # iptables -A INPUT -p tcp --dport 123456 -j LOG --log-prefix
```

- **IP Spoofing**

```
iptables -A INPUT -s 10.0.0.0/8 -i Interface da eth0 -j DROP
```

```
[root@socreppa /] iptables -A INPUT -s 172.16.0.0/16 -i Interface da eth0 -j DROP
```

```
[root@socreppa /] iptables -A INPUT -s 192.168.0.0/24 -i Interface da eth0 -j DROP
```

Na tabela abaixo serão mostrados alguns comandos do *iptables*.

Comandos	Descrição
-A	Este comando acrescenta uma regra às existentes no sistema, ou seja, permite atualizar regras já existentes na estrutura do <i>firewall</i> .
-I	Este comando insere uma nova regra dentro das existentes no <i>firewall</i> .
-D	Este comando exclui uma regra específica no <i>firewall</i> .
-P	Este comando define a regra padrão do <i>firewall</i> .
-L	Este comando lista as regras existentes no <i>firewall</i> .
-F	Este comando ZERA todas as regras criadas no <i>firewall</i> (o chamado <i>flush</i> ).
-h	Este comando mostrará o <i>help</i> , ajuda de comando.
-R	Este comando substitui um regra no <i>firewall</i> .
-C	Este comando basicamente checa as regras.
-Z	Este comando zera uma regra específica.
-N	Este comando cria uma nova regra com um nome.
-X	Este comando exclui uma regra específica por seu nome.

Os parâmetros usados do *iptables* serão mostrados na tabela abaixo



Parâmetro	Descrição
-p! (protocolo)	Define qual o protocolo TCP/IP deverá ser tratado. São eles: TCP, UDP e ICMP
-s! (origem) -d! (destino)	Define qual o endereço de origem (-S) e de destino que a regra atuará. Este comando possui dois argumentos: endereço/máscara e porta. Ex.: -S 10.0.0.1/24 80.
-i! (interface)	Define o nome da interface de rede onde trafegará os pacotes de entrada e saída do firewall. Muito utilizado em mascaramento e técnicas de NAT. Exemplo: -W eth1.
-j! (ir para)	Serve para redirecionar uma ação desde que as regras sejam similares.
-f!(fragmento)	Trata datagrama fragmentados.

## Instalação e Configuração do **SNORT**

Para instalar o **SNORT** execute o comando:

```
[root@socreppa /]rpm -Uvh snort.rpm.
```

O **SNORT**, já possui algumas regras para a detecção de intrusos (*/etc/snort.conf*) mas, para obter uma maior segurança no servidor, o administrador deve adicionar algumas regras de acordo com a necessidade, um exemplo, quando passar pacotes duvidosos entre duas redes (192.168.10.0/24 e 192.168.11.0/24) gerar uma alerta:

```
alert tcp any any <> [192.168.10.0/24,192.168.11.0/24]
alert udp any any <> [192.168.10.0/24,192.168.11.0/24]
alert ip any any <> [192.168.10.0/24,192.168.11.0/24]
```

O administrador também pode liberar somente algumas portas para os usuários acessarem do servidor, controlando assim o que o usuário pode fazer e aumentando a segurança, através das seguintes linhas de configuração:

```
pass tcp 192.168.10.0/24 any -> 192.168.11.1 25
pass tcp 192.168.10.0/24 any -> 192.168.11.1 80
pass tcp 192.168.10.0/24 any -> 192.168.11.1 110
pass tcp 192.168.10.0/24 any -> 192.168.11.1 143
pass tcp 192.168.10.0/24 any -> 192.168.11.1 21
pass tcp 192.168.10.0/24 any -> 192.168.11.1 137:139
```

Agora para rodar o *Snort*, basta o administrador digitar o seguinte comando:

```
[root@socreppa init.d] # ./ snort start
```