ASPECTOS DE SEGURANÇA DO LINUX 2.2

Paulo André S. Perez
IC - UNICAMP
Campinas - SP
perez@dcc.unicamp.br

Paulo Lício de Geus
IC - UNICAMP
Campinas - SP
paulo@dcc.unicamp.br

Resumo

Tendo em vista a crescente utilização e importância que o Sistema Operacional Linux vem tendo, neste trabalho apresentamos e discutimos os principais aspectos relativos à segurança do Linux, enfatizando as particularidades de seu kernel e das aplicações servidoras para ele desenvolvidas. Para cada componente da segurança do sistema abordamos sua importância, alguns dos ataques aos quais eles podem ser submetidos, e enumeramos alguns parâmetros de configuração para incrementar sua segurança.

Abstract

Having in mind the growing use and importance of the Linux Operating System, in this work we present and discuss the main issues regarding Linux security, emphasizing specific topics of its kernel and of the main service applications developed for this platform. For each of the security components of the system, we show its importance as part of the overall security concern, some of the attacks these components can be subjected to and list some configuration parameters to increase the system's security.

1 Introdução

Dentre os sistemas operacionais da atualidade, um dos que tem mais se destacado é o Linux. Nascido na Finlândia, no início dos anos 90, o Linux foi fruto de uma empreitada audaciosa por parte de um estudante de computação, Linus Torvalds. Linus procurava desenvolver um sistema operacional completo mas que fosse robusto o suficiente para rodar nos microcomputadores da época. Sozinho não conseguiria tal proeza, desta forma, assim que esboçou a primeira versão de teste do sistema, colocou o código fonte na Internet para que todos pudessem copiá-lo, usá-lo e até propor modificações no mesmo. Rapidamente ele passou a contar com a ajuda de milhares de programadores de todo o mundo, que passavam horas depurando e melhorando o Linux. Desde então o Linux tem sido um sistema operacional de código fonte aberto e custo zero. Estes fatores, associados ao fato do sistema ser robusto e confiável, fez o Linux ser um dos mais populares sistemas da atualidade, para ambos os meios, acadêmico e empresarial.

O Sistema Operacional é ponto chave na segurança dos sistemas de computação, e um dos mais discutidos no momento é o Linux. Então, neste trabalho apresentamos e discutimos os aspectos de segurança que concernem ao Linux, com especial atenção à versão 2.2.x.

Na Seção 2 apresentamos o "Estado da Arte" em que se encontra o assunto em questão, para nas Seções 3 a 10 discutirmos os principais tópicos de segurança do Linux. Na Seção 11 apresentamos nossas conclusões e na Seção 12 apresentamos as referências bibliográficas utilizadas nesta análise.

2 Estado da Arte

Muito tem sido discutido sobre segurança, em especial no que tange à Internet e ao comércio eletrônico, mas deste espectro pouco diz respeito à segurança do sistema operacional Linux. Na literatura, o número de artigos que trata do assunto em questão é limitado; geralmente o que mais pode ser encontrado são HOWTOs [2] e FAQs, ou ainda mensagens em listas de discussão e algumas páginas web que apresentam alguns dos conhecidos *bugs* de segurança.

Data de 1996 a segunda edição do livro de Garfinkel [1], "Practical Unix & Internet Security", que trata de forma geral o assunto e é tido por muitos como sendo a melhor referência sobre segurança no âmbito de sistemas operacionais. Como pode ser observado, na época da edição, pouco se falava do Sistema Operacional Linux e portanto pouca atenção, ou quase nenhuma, foi dada ao mesmo no livro.

A mais recente versão do kernel do Linux, versão 2.2, apresenta novos recursos diretamente relacionados à segurança do sistema e que portanto merecem ser abordados.

Recentemente foi divulgada uma distribuição do Linux, de nome Bastille, cujo enfoque está centrado na segurança do sistema. Além desta, há pelo menos dois outros esforços semelhantes: khaOS e Secure Linux. Isto demonstra a importância que o assunto tem, assim como as facilidades que o código aberto e a disseminação ampla trazem à plataforma.

3 Componentes da Segurança

Podemos ver a segurança de um sistema como sendo um modelo em camadas, embora não no sentido estrito, como mostrado na Figura 1.

Por este modelo podemos ver que a segurança de um sistema começa pela segurança física do mesmo; em seguida passa pela segurança do kernel do sistema operacional; depois temos os protocolos, sistemas de arquivos e sistemas de autenticação; em um nível superior estão os serviços oferecidos por

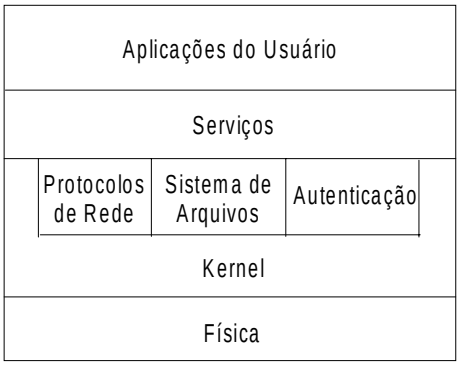

Figura 1 - Camadas de Segurança

este sistema, como DNS, NIS e outros; finalmente as aplicações do usuário. Cada componente deste modelo representa um potencial ponto de vulnerabilidade, e como tal deve ser tratado.

Neste trabalho, ao apresentarmos os aspectos mais relevantes à segurança do Sistema Operacional Linux, estaremos enfocando cada uma destas camadas e suas relações com as demais.

4 Segurança Física

Como não poderia deixar de ser, o primeiro aspecto de segurança que deve ser considerado é a segurança física do sistema, pois em muitos casos, por mais seguro que esteja um sistema, um acesso indevido ao console, ou mesmo ao hardware, pode comprometer todo o sistema.

Nos primórdios da computação, e ainda hoje em algumas instâncias, os equipamentos de computação de uma empresa ficam num local de acesso restrito a poucas pessoas devidamente credenciadas, desta forma limitando os potenciais ataques. Contudo, cada vez mais temos computadores espalhados por todos os lugares de uma organização, incluindo servidores departamentais e outros. Estas máquinas podem estar fisicamente vulneráveis ao ataque de terceiros, que maliciosamente podem simplesmente ter acesso, danificar ou roubar informações sigilosas.

Um exemplo simples porém significativo pode estar dentro do próprio ambiente universitário. Imagine uma estação de trabalho Linux num dos inúmeros laboratórios de ensino. Normalmente, num ambiente como este, existem servidores de arquivo que exportam os *homedirs* dos usuários para toda a rede, incluindo as estações de uso dos alunos. Neste contexto, o administrador de tal rede deve tomar todo cuidado possível para evitar que um aluno possa tornar-se root de alguma destas máquinas, pois, uma vez como root, ele poderá navegar por todo o sistema de arquivos da rede, incluindo os *homedirs* dos professores, da administração do sistema e o que mais existir. Apesar disto parecer um cuidado óbvio, mostraremos a seguir que existem certos cuidados importantes a serem tomados quanto ao sistema operacional de tais máquinas, para evitar potenciais acessos de root.

4.1 Acesso ao Hardware

Uma máquina Linux que proveja algum serviço crítico, como por exemplo servidor de arquivos, com certeza deve estar num local de acesso restrito. Caso isto não seja possível, alguns cuidados especiais devem ser tomados:

- Evitar que os usuários possam fazer uso local da estação, seja removendo o teclado/monitor da mesma (e desabilitando-os na BIOS), ou criando o arquivo */etc/nologin*, o qual impede que usuários comuns (exceção ao root) consigam efetuar o *login*.
- Procurar adquirir equipamentos que possuam cadeados/chaves de proteção que impeçam que o gabinete seja aberto, pois num caso extremo, o invasor poderia apagar a BIOS e tentar mudar suas configurações, ou mesmo roubar o disco rígido. Um disco Linux, que usa o sistema de arquivos *ext2*, uma vez roubado, pode ser acessado em qualquer outra máquina Linux, ou seja, os dados gravados neste disco estão completamente vulneráveis a este tipo de ataque (porém, veja Seção 7.4).
- Sempre fazer *backups* periódicos e os manter num **local seguro** e **longe** de onde se encontram seus servidores. Desta forma, mesmo que todo o seu equipamento seja roubado ou danificado, ainda assim poder-se-á recuperar o que foi perdido.

4.2 Segurança da BIOS

Muitos equipamentos do mercado já vêm com alguns recursos de segurança na BIOS e que podem ajudar no tocante ao acesso ao hardware. Como dito anteriormente, um sistema de arquivos *ext2*, em um disco removido do sistema original, pode ser acessado por qualquer ambiente Linux. O único recurso neste caso, com relação a dados, é o uso de criptografia nos sistemas de arquivos. O outro lado do problema de acesso físico aos discos é, obviamente, a possibilidade de iniciar um outro sistema operacional, Linux ou não. Desta forma, a primeira providência a ser tomada é configurar a BIOS para fazer o *boot* (carga do sistema operacional) apenas pelo disco rígido. Além disto, deve ser colocada senha no *setup*, de forma a evitar que as configurações da BIOS sejam modificadas.

A menos que seja extremamente necessário, não é aconselhável colocar uma senha de acesso geral, pois no caso de uma queda de energia, quando esta voltar, a máquina poderá ficar parada aguardando que alguém digite a senha para que a mesma continue o processo de *boot*.

Não deve ser esquecido que se o gabinete do computador não for "trancado", o mesmo poderá ser aberto e o conteúdo da BIOS poderá ser apagado.

Outro fato relevante é que algumas BIOS vêm de fábrica com senhas padrão do fabricante e conhecidas entre os *hackers*. Nesta caso a solução é alterar a própria BIOS.

4.3 Segurança do LILO

O Linux Loader, também conhecido como LILO, é um pequeno programa cuja função é fazer a carga do Linux, ou seja, é ele que carrega o sistema operacional para a memória RAM e inicia sua execução.

Apesar de parecer inofensivo, o LILO é um dos mais sérios riscos à segurança do Linux, isto porque no momento em que o LILO entra em execução pode-se ter acesso a um *prompt* no qual podem ser fornecidos parâmetros para a carga do Linux. Estes parâmetros podem ir desde a definição de parâmetros para algum dispositivo, até o conhecido "linux single".

Quando no *prompt* do LILO o usuário digita "linux single" ele está dizendo que o Linux deve ser carregado no modo *single use*r (mono-usuário). Neste modo, que foi originalmente concebido para manutenção ou recuperação do sistema, o Linux entra no *runlevel* S, onde somente as funções básicas do sistema são iniciadas, para logo em seguida entrar em execução uma *shell* de root[1], ou seja, sem nenhum procedimento de *login* o usuário no console recebe um *shell* de root. É apropriado comentar que nas distribuições onde isto ocorre, acrescentando-se ao arquivo */etc/inittab* a linha "~~:S:wait:/sbin/sulogin", é possível contornar tal problema.

Similarmente, o usuário pode digitar no *prompt* "linux init=/usr/bin", fazendo com que o *kernel* execute diretamente o *shell*, ao invés de primeiro executar o programa *init*, que é o pai de todos os processos. Como é o *init* que coloca em execução os *scripts* de inicialização, estes serão ignorados e qualquer mecanismo de segurança que eles implementem não terá o menor efeito.

Para fazer com que as opções do LILO sejam acessíveis apenas ao root, deve-se acrescentar no arquivo de configuração do LILO (*/etc/lilo.conf*) as diretivas "restricted" e "password = xxxxx". Estas fazem com que toda vez que o usuário forneça parâmetros para o LILO, estes somente serão processados mediante uma senha que deve coincidir com o que foi especificado na diretiva *password*.

Este tipo de cuidado impede o acesso indevido, porém é bom não esquecer de fixar os atributos do arquivo *lilo.conf* em 600, de forma que nenhum outro usuário, que não o root, possa saber qual é a senha a ser usada.

4.4 Bloqueando Terminais

Em muitas situações é comum que algum usuário se ausente de seu computador por algum tempo deixando sua sessão ainda "logada". Neste contexto, é sempre aconselhável que antes de se ausentar o usuário "trave" (lock) sua estação, o que corresponde a bloquear o acesso da mesma (via console) a outras pessoas. Isto normalmente é feito através do utilitário *vlock* (modo texto) ou do utilitário *xlock* (que roda sob X-Windows). Ambos os utilitários bloqueiam o acesso ao console (ou terminal virtual) até que o usuário forneça sua senha.

Muitos wms (Window Managers) permitem que o *xlock* seja executado automaticamente após um determinado tempo sem atividade por parte do usuário. Tal tipo de procedimento é aconselhável, uma vez que muitos usuários simplesmente esquecem de efetuar o *logout*, ou bloquear seus terminais, quando se ausentam de seu ambiente de trabalho.

Um potencial problema de segurança do *xlock*, que deve ser levado em consideração, está no fato de que o XFree86[2], e todas as aplicações que rodam sobre ele (incluindo o *xlock*), pode ser abortado por qualquer usuário que tenha acesso ao console e que pressione simultaneamente as teclas CTRL+ALT+BACKSPACE. Com isto, se o X-Windows foi executado a partir de um *shell*, após o mesmo ser abortado o invasor terá acesso a este *shell*, independentemente de saber ou não a senha do usuário que tinha bloqueado o terminal. Para evitar que isto possa ocorrer, deve-se acrescentar a opção *DontZap* na seção *ServerFlags* do arquivo de configuração do XFree86 (*/etc/X11/XF86Config* ou equivalente).

5 Segurança do Kernel do Linux

A personalidade, por assim dizer, de um sistema operacional está em seu kernel, pois é este que responde pelas principais funções do sistema. A versão 2.2 do kernel do Linux apresenta uma série de melhoramentos; destas destacam-se o suporte ao multiprocessamento simétrico (SMP), novos dispositivos, sistemas de arquivos e protocolos de rede.

5.1 Dispositivos

Os dispositivos do Linux estão representados dentro do diretório */dev*. No que tange à segurança, dois dispositivos especiais devem ser levados em consideração, que são o */dev/random* e o */dev/urandom*.

Como sabemos, grande parte dos algoritmos de criptografia estão baseados na utilização de números aleatórios, e desta forma são muito sensíveis à entropia destes números. Se um programa "malicioso" conseguisse interferir nos números aleatórios que o sistema operacional gera para as aplicações, este poderia também estar forçando os resultados, que poderiam ser chaves, que os programas de criptografia viessem a produzir. Um caso também crítico seria se este programa "malicioso" conseguisse prever quais seriam os números aleatórios que o sistema operacional fosse gerar para as aplicações, pois da mesma forma, o programa poderia prever quais seriam os resultados destas aplicações.

[1] Em algumas distribuições do Linux (como exemplo o Slackware) isto não ocorre.

[2] O XFree86 é a implementação de X-Windows mais popular entre os Unixes abertos que rodam em plataformas Intel.

No Linux, os dispositivos *random* e *urandom* são os elos entre o gerador de números aleatórios do sistema operacional e as bibliotecas/aplicações que fazem uso destes números. O *random* baseia-se em eventos externos, como por exemplo interrupções de teclado, para produzir os números aleatórios, desta forma buscando assegurar uma maior entropia dos números gerados. O *urandom* utiliza-se de um algoritmo convencional de números aleatórios baseados numa semente e numa tabela de geração, por conseguinte não tendo uma entropia tão grande quanto a do *random*. A menos que o usuário faça uso explícito do *random*, os números aleatórios são fornecidos através do *urandom*.

Um sério ataque que ocorreu ao Linux, baseava-se justamente em, utilizando-se de um número muito grande (cerca de 50000) de números aleatórios gerados pelo *urandom*, estimar quais seriam os próximos números gerados pelo mesmo. Desta forma, uma vez conhecidas quais as aplicações de criptografia que estavam em execução (por exemplo o pgp), seria possível estimar quais seriam as chaves por elas produzidas.

O kernel 2.2 oferece uma geração segura de números aleatórios (dentro das possibilidades de um sistema determinístico), porém aconselha-se sempre preferir o uso de */dev/random*, de forma a assegurar que os números utilizados realmente sejam aleatórios e desconhecidos de outras aplicações.

5.2 Módulos

Em versões antigas do kernel do Linux, era possível que um módulo do usuário fosse dinamicamente carregado e incorporado ao kernel do sistema. Ao ser executado, estaria no que chamamos "modo kernel" de privilégios, que correspondem aos privilégios de mais alto nível. É claro que isto representa um seríssimo risco à segurança do sistema, pois o usuário poderia escrever um módulo mal intencionado, que atuaria sem nenhuma limitação do sistema.

Felizmente, este tipo de risco foi rapidamente percebido e removido. Atualmente, apenas root pode inserir módulos no kernel do Linux, e estes devem estar situados no diretório */lib/modules/versãodokernel*.

5.3 Opções do Kernel

O kernel do Linux incorpora uma série de recursos de segurança, principalmente no que diz respeito ao protocolo TCP/IP, os quais são tratados na Seção 6. Mas um significativo melhoramento, introduzido no kernel 2.2, está no fato de que a maioria das opções do kernel pode ser dinamicamente modificadas e não apenas em tempo de compilação do kernel, como anteriormente. Estas opções estão situadas em */proc/sys* e seus subdiretórios. Um exemplo é o arquivo */proc/sys/net/ipv4/ip_forward*, cujo valor define se o kernel irá (valor 1) ou não (valor 0) fazer o *forwarding* de pacotes IP. Assim, o comando "echo 1 > /proc/sys/net/ipv4/ip_forward" habilita o *forwarding*. É interessante notar que, por questões de segurança, estes arquivos podem ser alterados apenas por root.

6 Protocolos de Rede

Um grande número de ataques vem da exploração de fragilidades presentes nas implementações dos protocolos de rede. No Linux, o protocolo nativo é o TCP/IP, e várias facilidades de segurança para o TCP/IP têm sido incorporadas ao sistema. A seguir apresentamos as principais opções do kernel, que dizem respeito à segurança. Também indicamos a utilização das mesmas de acordo com o perfil da máquina Linux, que pode ser: H - para uma estação da rede sem nenhuma função especial; S - para máquinas que atuam como servidores; F - para máquinas que servem como *firewalls*; e "-" quando não for particular a um certo uso.

- *Packet socket* (F) - Esta opção habilita o uso do protocolo de pacotes, que permite que uma aplicação comunique-se diretamente com o dispositivo de rede, sem passar pela camada de rede do Linux. Este recurso é normalmente utilizado por programas que fazem captura de pacotes na rede (*eavesdropping*) como é o caso do *tcpdump*.
- *Kernel/User netlink socket* (F) - Permite a comunicação bidirecional entre certas partes do kernel e os processos dos usuários, de forma que estes processos sejam capazes de saber informações sobre a rede. Programas como o *arpd* utilizam-se deste recurso.
- *Network firewalls* (F) - Para que o Linux atue como um *firewall*, esta opção deve estar habilitada.
- *Socket filtering* (F) - Permite que programas no espaço do usuário possam filtrar qualquer *socket* do sistema, e desta forma permitir ao kernel a possibilidade de filtrar certos tipos de dados que passam pelo *socket*.
- *IP: forwarding/gatewaying* (F) - Habilita ou não o *forwarding* de pacotes IP.
- *IP: syn cookies* (H,S,F) - Um tipo comum de ataque é o conhecido "SYN flooding", que causa negação de serviços por parte da máquina atacada. Esta opção permite que conexões legítimas possam ser mantidas mesmo quando sob ataques deste tipo, sem esgotamento de recursos da máquina.
- *IP: verbose route monitoring* (F) - Faz com o que o kernel emita mensagens sobre o roteamento, incluindo alertas sobre pacotes "estranhos" e outros.
- *IP: firewalling* (F) - Habilita a capacidade de *firewalling* de pacotes IP.
- *IP: always defragment* (F) - Faz com que qualquer pacote IP, antes de ser roteado, seja primeiro desfragmentado, para que em seguida o mesmo seja submetido aos filtros de pacotes

IP. Algumas formas de ataque se aproveitavam da fragmentação de pacotes IP para burlar os filtros.

- *IP: transparent proxy support* (F) - Permite ao Linux (atuando como *firewall*) redirecionar o tráfego de rede originado na rede local para uma rede externa, de forma transparente.
- *IP: aliasing support* (S) - Esta opção faz com que seja possível que uma única interface de rede tenha diferentes endereços IP.
- *IP: ARP daemon support* (F) - Normalmente a resolução de endereços IP em endereços físicos é feita pelo próprio kernel, mas através desta opção é possível utilizar aplicações para fazer a resolução, como por exemplo o *arpd*.
- *IP: drop source routed frames* (H,S,F) - Esta opção é especialmente interessante, pois faz com que sejam descartados os pacotes IP que sejam *source routed*, ou seja, que tragam em si a rota a ser seguida pelos pacotes. Este tipo de pacote pode ser usado para ataques do tipo "IP spoofing".

Se a máquina Linux for utilizada como *firewall* existe o programa *ipfwadm* (versões do kernel até 2.0.x), que permite a gerência das regras de filtragem. Se a versão do kernel do Linux for 2.2.x, o programa a ser utilizado é o *ipchains*, que basicamente difere do anterior por fazer uso da nova interface que o kernel oferece para a definição das regras de filtragem de pacotes.

O kernel 2.2 inclui o protocolo IPv6, porém sem a implementação do IPSec. Os administradores que desejarem implementar uma VPN (Virtual Private Network) devem fazer uso do IPSec para IPv4 (disponível para o Linux 2.2) ou do CIPE (Cryptographic IP Encapsulation), os quais permitem o tunelamento de pacotes IP criptografados.

7 Segurança de Arquivos e Sistemas de Arquivos

É inútil dotar o sistema de ferramentas de criptografia e autenticação, se o mesmo grava seus dados de forma não segura, em arquivos que possam ser indevidamente acessados. Neste contexto, o Sistema de Arquivos tem especial importância, pois é ele que responde pelo acesso e os direitos de acesso que os usuários têm sobre os arquivos.

7.1 Permissões de Arquivos

Além dos atributos básicos que um arquivo pode ter (*read*, *write* e *execute*), existem dois outros atributos de extrema relevância para a segurança, que são:

- **SUID**: Este atributo, quando definido em um arquivo, indica que este, caso seja um executável ou um *script*, quando do momento de sua execução, terá o seu *userid* atribuído de acordo com o *userid* de quem é o proprietário do arquivo, ao invés de quem o executa. Este tipo de privilégio é especialmente perigoso pois um arquivo cujo dono seja o root, com **SUID** setado e que possa ser executado por qualquer usuário, independente de quem o execute, terá os privilégios de um processo do próprio root.
- **SGID**: Este atributo, quando setado em um arquivo, similarmente ao **SUID**, fará com que o processo herde os privilégios do grupo do proprietário do arquivo.

Apesar de muitas distribuições do Linux, por padrão, ajustarem as permissões dos arquivos especiais do sistema, é sempre aconselhável tomar certos cuidados:

- Em todos os arquivos de *log* do sistema, como por exemplo o arquivo */var/log/messages* e */var/log/secure*, ajustar suas permissões para 600 (-rw------), o que significa que apenas root poderá ler e gravar nestes arquivos. Isto porque nestes arquivos podem aparecer informações sobre possíveis erros no sistema, que podem vir a ser explorados pelo atacante.
- Existe um atributo pouco conhecido, que na verdade diz respeito apenas ao sistema de arquivos do Linux (ext2), que permite que um arquivo nunca possa ser modificado/apagado. Este atributo pode ser setado e removido apenas pelo root e pode ser uma forma interessante de reforçar a segurança de arquivos críticos. Isto pode ser feito através da linha de comando "chattr +i file".
- Como root, verificar a PATH e certificar-se de que realmente esteja executando o programa que deseja, no local que este deveria estar. Há, na maioria das listas de *bugs* de segurança, relatos sobre incidentes onde root inadvertidamente executa programas "maliciosos" do usuário, pensando serem programas do sistema.
- Procurar por arquivos que possuam os **SUID** ou **SGID** setados; eles podem ser uma potencial fonte de problemas.
- Arquivos e diretórios que podem ser gravados por qualquer usuário (atributo *w* setado para outros) são um considerável ponto de vulnerabilidade.

7.2 Integridade de Arquivos

Um dilema ao qual vivem submetidos os administradores de sistema é o de saber se os arquivos do sistema não foram indevidamente alterados. Uma forma de tentar detectar a possível ação de um intruso é através da utilização de ferramentas como o Tripwire [3]. Este pacote lê um arquivo de configuração que indica quais arquivos e diretórios devem ser monitorados, e então acompanha as modificações sobre estes.

O Tripwire, para cada arquivo/diretório especificado, calcula um *checksum* deste e com isto verifica se o arquivo sofreu qualquer tipo de alteração, reportando os resultados. Há a possibilidade de se escolher entre diferentes algoritmos de assinatura digital, alguns bastante robustos criptograficamente.

O Tripwire cria uma base de dados contendo os *checksums* dos arquivos monitorados, e é extremamente aconselhável que este arquivo fique numa mídia que não possa ser corrompida por um invasor. Desta forma, aconselha-se que a base de dados ou seja armazenada numa mídia removível, que deve ser removido e guardado em local seguro, ou num meio do tipo *write-once*.

É aconselhável que o Tripwire seja periodicamente executado, e para tanto pode-se colocar o comando para execução na *crontab* do sistema.

7.3 Cavalos de Tróia

O nome Cavalo de Tróia vem da história que justamente narra como soldados, escondidos dentro de um cavalo de madeira que foi supostamente dado de presente, conseguiram entrar na cidade de Tróia para invadi-la. Da mesma forma, muitas vezes quando um usuário faz o *download* de um programa qualquer na Internet pode estar, inadvertidamente, trazendo junto um sério risco para a segurança do sistema.

Muitos invasores divulgam na Internet programas que trazem dentro de si comandos que deliberadamente podem danificar o sistema, apagando arquivos ou alterando configurações. Podem ainda enviar para alguém@algumlugar o arquivo de senhas ou outras informações sigilosas. Este tipo de programa é que normalmente chamamos de "Cavalo de Tróia".

Para evitar este tipo de risco, sempre vale seguir algumas regras:

- Quando for fazer o *download* de algum software crítico, principalmente aqueles que serão utilizados por root, procurar utilizar um *site* confiável e conhecido publicamente. Sempre evitar utilizar *caches*, ou mesmo copiar arquivos de áreas de usuários comuns.
- Muitos *sites*, como o caso da Red Hat, junto de seus arquivos também fornecem *checksums* MD5 ou assinaturas PGP, que podem ser verificadas para comprovar a autenticidade/integridade dos arquivos.
- Apenas executar como root as tarefas mais importantes, e que somente podem ser executadas por root. Um erro comum é usar root para tarefas comuns, como editar um trabalho ou mesmo entrar num *chat*. Inclusive vale citar que muitas versões de clientes IRC eram "Cavalos de Tróia", de modo que alguém como root, que inadvertidamente usasse um destes programas, estaria expondo todo o sistema a um potencial ataque.

7.4 Sistemas de Arquivo Criptografados

Num caso extremo de segurança, onde o próprio acesso ao disco rígido não é garantido, pode ser interessante criptografar os dados contidos no sistema de arquivos. Para tanto existe, para o Linux, o Cryptographic File System (CFS) [4], que usa um servidor NFS rodando na máquina local para armazenar os arquivos criptografados.

Uma outra versão mais sofisticado do CFS é o TCFS (Transparent Cryptographic File System) [5], que faz o mesmo que o CFS mas de forma transparente, sendo integrado ao próprio sistema de arquivos do Linux.

8 Autenticação de Usuários

"Como é que o Linux sabe que um usuário é quem ele diz ser ?". A resposta está no mecanismo de autenticação de usuários, que é tratado nesta seção.

8.1 Usuários e Senhas

Tradicionalmente os sistemas UNIX validam um usuário através de sua senha, que fica armazenada no arquivo */etc/passwd*. Como é de se imaginar, a senha gravada no *passwd* não está em texto claro; na verdade, ela é cifrada através de uma função unidirecional (que não pode ser invertida) variante do DES. Como não apenas as senhas dos usuários, mas também seus dados básicos (*userid, homedir*, etc) estão armazenados no arquivo *passwd*, e estes são corriqueiramente utilizados pelos aplicativos, o arquivo não possui e não deve possuir nenhum atributo que impeça os usuários de lerem seu conteúdo. Desta forma, também ficam expostas as senhas criptografadas de todos os usuários.

Até alguns anos atrás, face à impossibilidade de inverter a função de ciframento das senhas e principalmente devido ao limitado desempenho dos computadores, o que inviabilizava buscas exaustivas de senhas, este mecanismo de criptografia e armazenamento de senhas era seguro.

Atualmente, o grande desempenho dos novos microprocessadores e as redes de computadores, permitem que vários computadores possam estar interagindo na busca de senhas. Assim, o sistema que apenas se utiliza do mecanismo convencional de senhas tornou-se vulnerável, sendo um sério risco à segurança do sistema.

Os dois mais conhecidos programas de domínio público, que têm por objetivo quebrar senhas, são o "Crack" e o "John the Ripper". Ambos atuam de forma similar, exaustivamente buscando por palavras de dicionário, ou ainda cadeias de dígitos/letras que, cifrados, coincidam com alguma das senhas armazenadas no arquivo *passwd*. Quando isto ocorre, significa que a senha de algum usuário foi encontrada. Este tipo de busca normalmente primeiro encontra as senhas tidas como triviais (palavras, datas etc) para em seguida, após alguns dias de processamento, também encontrar senhas algo mais complexas.

8.2 Shadow Passwords

No intuito de incrementar a segurança do mecanismo de senhas do UNIX, e do Linux em específico, Haugh [6] concebeu o mecanismo conhecido

como Shadow Passwords, para a plataforma Linux. Este é uma variante do modelo convencional, onde as senhas dos usuários são removidas do arquivo */etc/passwd* e gravadas no arquivo */etc/shadow*, o qual tem permissões 600. Desta forma, apenas root tem acesso às senhas cifradas.

Além de desmembrar o arquivo de senhas, também foram incorporados novos recursos, como:

- *Shadow Groups*: Com isto, os grupos podem também ter senhas e suas senhas ficam armazenadas no arquivo */etc/gshadow*, podendo serem definidos administradores de grupos e a possibilidade de um usuário mudar seu *groupid* (gid), mediante um tipo de login no grupo.
- *Double Length Passwords*: Tradicionalmente as senhas em Unixes em geral possuem 8 caracteres de comprimento, sendo os excedentes ignorados. Porém, agora é possível ter-se senhas de 16 caracteres, o que significativamente dificulta a busca exaustiva por senhas.
- *Password Aging*: Com esta opção é possível a root definir o tempo de vida das senhas dos usuários, desta forma forçando-os a regularmente mudar suas senhas.
- *Account Expiration and Locking*: Permite definir quando uma conta irá expirar, ou ainda bloquear a mesma.
- *Dial-up Passwords*: Torna disponível a definição de senhas para acesso através de rede discada.

É expressamente recomendado que os administradores Linux utilizem o mecanismo de *shadow* ao invés do mecanismo tradicional de senhas (os utilitários *pwconv* e *grpconv* fazem a migração para o *shadow*). Apesar dele não resolver todos os problemas, com certeza dificulta a ação de atacantes contra a autenticação básica de usuários.

8.3 Cracklib - ProActive Password Sanity Library

Como foi dito anteriormente, os programas que buscam quebrar as senhas procuram por senhas tidas como triviais, portanto estas senhas devem ser evitadas. Neste sentido, existe a biblioteca Cracklib, desenvolvida por Muffett [7], que basicamente é utilizada pelo Linux para verificar se a senha fornecida pelo usuário (quando vai cadastrar/alterar sua senha) é vulnerável a ataques ou não. Esta filtragem inicial proporciona um significativo incremento na segurança das senhas.

É importante notar que a Cracklib não substitui por si só o programa *passwd*; desta forma, para que a mesma seja funcional, esta deve ser invocada através dos programas do sistema. As distribuições mais recentes do Linux utilizam o PAM, para gerenciar senhas, o qual utiliza a Cracklib.

8.4 Kerberos

Desenvolvido pelo Projeto Athena do MIT [15], o Kerberos é um robusto sistema de autenticação de usuários, que dentre outras vantagens, evita o tráfego pela rede de senhas em texto claro.

O Kerberos encontra-se em sua versão 5, também disponível para o Linux.

8.5 PAM - Pluggable Authentication Modules

Com o objetivo de prover um mecanismo unificado de autenticação, foi concebido o PAM [8]. Através do PAM é possível tornar transparente às aplicações os métodos utilizados para autenticação e as configurações dos mesmos. Isto é possível através da utilização de módulos que literalmente podem ser "plugados" ao mecanismo de autenticação.

Dentre os mais recentes recursos incorporados ao PAM destacam-se:

- Ciframento das senhas através do algoritmo MD5, ou o tradicional DES. Com isto, encontrar-se senhas através de busca exaustiva tornou-se algo computacionalmente inviável.
- Definição de limites para os recursos utilizados pelos usuários, tais como tempo de CPU, quantidade de memória e outros. Este recurso possibilita impor restrições de forma a evitar ataques por negação de serviço (Denial of Service Attack - DoS).
- Utilização de Shadow Passwords. Isto dificulta o acesso às senhas dos usuários.
- Validação das senhas através da Cracklib. Com isto, é possível evitar que os usuários utilizem senhas que possam ser facilmente "quebradas".
- É possível limitar o local (*host*) e horário dos *logins* dos usuários.
- A configuração do mecanismo de autenticação a ser utilizado pode ser feita para cada aplicativo individualmente. Inclusive é possível utilizar o Kerberos como mecanismo de autenticação.

Como exemplo, a distribuição do Linux Red Hat 6.0 vem com o PAM, e seus arquivos de configuração ficam situados no diretório */etc/pam.d*. Abaixo apresentamos o arquivo de configuração do *login*.

```
auth        required    /lib/security/pam_securetty.so
auth        required    /lib/security/pam_pwdb.so shadow nullok
auth        required    /lib/security/pam_nologin.so
account     required    /lib/security/pam_pwdb.so
password    required    /lib/security/pam_cracklib.so
password    required    /lib/security/pam_pwdb.so use_authtok md5 shadow
session     required    /lib/security/pam_pwdb.so
session     optional    /lib/security/pam_console.so
```

Nele podemos ver nas linhas *password* como é que o PAM irá tratar a manipulação de senhas do usuário, que se constitui de dois passos:

- Primeiramente a senha em questão será validada através do módulo *cracklib*, desta forma impedindo que senhas triviais sejam utilizadas.
- Segundo, a senha é tratada pelo módulo *pwdb*, que é o responsável pela manipulação da base de dados de senhas. Este módulo somente será

ativado se a senha tiver sido validada pelo módulo anterior (*use_authok*) e a mesma deverá ser cifrada com o algoritmo MD5 e armazenada através do mecanismo de Shadow Passwords.

Por este pequeno exemplo podemos ver a flexibilidade e os novos recursos de segurança oferecidos pelo PAM.

9 Segurança dos Serviços de Rede

A maior parte da literatura trata justamente deste assunto, porque talvez seja o mais susceptível a ação de terceiros "mal intencionados". Assim, nesta seção estão contemplados alguns dos serviços que tipicamente são alvo de ataque, como o DNS, WWW, NFS, Sendmail e outros.

9.1 DNS

A mais importante recomendação para o DNS é mantê-lo atualizado, procurando não omitir *hosts* ou endereços, e também não fornecendo informações que possam ser utilizadas pelos invasores. Este é o caso do campo INFO, onde muitos colocam detalhes da máquina e sistema operacional utilizado.

O DNS pode ser útil para aumentar a segurança do Linux, pois muitos serviços podem ser configurados de forma a rejeitar *hosts* que não tenham entradas de DNS válidas, o que tipicamente ocorre com os *hosts* não autorizados que tentam conectar-se à rede.

Apesar de ter sido especificado um padrão para serviço de autenticação através do DNS, o mesmo ainda não possui versões estáveis para o Linux e que sejam amplamente utilizadas.

9.2 NIS e NIS+

O NIS, Network Information Service, tem por objetivo prover informações de configuração aos *hosts* de uma rede. Estas informações normalmente dizem respeito aos *hosts* (nomes, *aliases*, endereços etc), redes (nomes, endereços etc), serviços, usuários (*login names*, *uid*, *gid*, *password*, *homedir* etc) e outros.

Através do NIS, vários *hosts* de uma rede podem verificar o *login* dos usuários em uma base de dados de administração centralizada. Quando um usuário tenta efetuar seu *login* em qualquer máquina da rede, esta busca junto a um servidor de NIS as informações sobre aquele usuário, incluindo sua senha cifrada (o NIS utiliza o mecanismo tradicional de senhas do UNIX), para então validar o *login* em questão. Apesar das senhas que trafegam pela rede estarem cifradas, isto não representa muita dificuldade para os invasores: através do NIS podem literalmente ter uma cópia do arquivo *passwd* (ou mesmo do *shadow*) contendo todas as senhas cifradas de todos os usuários da rede NIS. Em seguida, podem utilizar programas para "quebra" de senhas, "off-line", e eventualmente conseguir acesso a contas com senhas triviais.

Uma versão mais elaborada do NIS, conhecida como NIS+, além de incrementar a base de dados da rede com novas informações, acrescentou um interessante recurso de segurança. Permite que toda a conversação entre o cliente NIS+ e o servidor seja cifrada, usando o algoritmo DES. Isto dificulta a ação de eventuais invasores realizando "eavesdropping" (escuta na rede). Contudo, para o Linux apenas existe a versão cliente do NIS+, até o momento.

9.3 NFS

O NFS (Network File System) provê o compartilhamento de arquivos numa rede. Este serviço é o mecanismo básico para que os usuários de uma rede Linux possam utilizar seus *homedirs* independentemente da estação em que estão logados.

O NFS do Linux já oferece algumas características bastante desejáveis, que outros Unixes comerciais nem oferecem. O exemplo mais importante é sem dúvida a delimitação de acesso a hierarquias exportadas, onde o esquema do Linux é simples e completo. Outros Unixes comerciais somente atingem a granularidade necessária com o auxílio de outros sub-sistemas da rede, tais como o NIS/NIS+.

As versões anteriores do NFS dependiam de um *daemon* (*nfsd*), que era o processo servidor de NFS. A implementação era limitada (em beta-teste) e representava um sério gargalo para o desempenho do sistema. Objetivando melhorar o desempenho, o serviço NFS foi incorporado ao espaço de kernel no Linux 2.2. Apesar do benefício trazido, isto pode vir a ser um sério risco à segurança do sistema, pois eventuais *bugs* (de segurança ou não) no NFS podem comprometer todo o sistema operacional.

9.4 WWW

Face a popularização da Web, um dos serviços mais utilizados tem sido o WWW, e com isto também tem sido vítima de muitos ataques, principalmente aqueles do tipo "buffer overflow". Neste caso, valem as seguintes recomendações:

- Nunca deixar o servidor de www (*httpd*) rodar com privilégios de root. Deve-se criar um usuário de acesso limitado e no arquivo *httpd.conf* definir qual é este usuário na cláusula *User*. O processo pai dos servidores *httpd* roda como root para poder se atrelar à porta 80 do protocolo TCP, porém os filhos, que efetivamente atendem o serviços das conexões dos clientes, fazem "su" para um usuário menos privilegiado antes de iniciarem o serviço.
- As mais recentes versões do *httpd* permitem especificar o nível de detalhamento do *log* a ser gerado pelo servidor; caso exista alguma suspeita de tentativa de ataque, pode-se definir na cláusula *LogLevel* o nível *debug* ou *info.*

- Por *default*, não se deve permitir que usuários executem CGIs ou *scripts* em suas páginas. CGIs podem conter programas maliciosos ou mesmo erros, que podem vir a comprometer a segurança do sistema (com permissões do processo *httpd*).
- Periodicamente deve-se verificar os arquivos de *log*, em especial o *error_log*, e procurar por *hosts* que causam muitos erros, pois eles podem representar eventuais invasores.
- A Apache [9], que produz uma das versões mais utilizadas de *httpd* para Linux, reportou que seu servidor tem sido constantemente vítima de ataque de negação de serviço do tipo *tcp_fin_wait2*. O kernel Linux de versões 2.0.x e acima não está vulnerável a este tipo de ataque, contudo deve-se acompanha de perto eventos gerados por tal serviço, por via das dúvidas.

No caso de se desejar conexões seguras através da Web, pode-se optar por instalar o módulo *mod_ssl* [10], que é a interface Apache para o SSLeay.

9.5 SSH

Nos acessos remotos, normalmente é utilizado o programa *telnet*, porém este apresenta um sério risco à segurança do sistema. Todas as mensagens entre o cliente e o servidor trafegam em texto claro (não cifrado), incluindo a senha de *login* utilizada pelo usuário. Para resolver este problema, permitindo que acessos remotos possam ser feitos de forma segura, existe o SSH (Secure Shell) [11], que estabelece conexões cifradas e autenticadas entre cliente e servidor.

O SSH é um conjunto de programas que substitui, por versões seguras, o *rlogin, rsh* e o *rcp*. Para atingir seus objetivos, o SSH utiliza um mecanismo de chaves públicas para autenticar usuários (através do algoritmo RSA), além de cifrar a comunicação entre os dois *hosts* da conexão (algoritmos Blowfish, Arcfour, IDEA, DES e 3DES). O SSH também faz compressão de mensagens e permite a comunicação segura entre aplicações e um servidor X11 remoto (permite também o redirecionamento de outras conexões).

Existem versões de clientes para o SSH que rodam em plataformas Windows e MacOS, que substituem as funcionalidades oferecidas pelos programas *telnet* e *ftp* em ambas. Isto evita o tráfego de senhas em claro para usuários Unix remotos.

Nunca é aconselhável permitir que os usuários de um sistema Linux tenham acesso remoto (via Internet), pois geralmente é através deste meio que os invasores entram no sistema. Porém, se isto é necessário, com certeza deve-se instalar e utilizar o serviço SSH, de forma a evitar que eventuais senhas ou dados possam ser interceptados.

9.6 Sendmail, Imap e Pop

Um dos primeiros serviços a surgir, antes mesmo da própria Internet, foi o e-mail. Durante todo este tempo, ele sempre foi um dos mais utilizados; e sua implementação mais conhecida, *sendmail*, talvez seja a que possua a mais longa lista de erros e falhas de segurança. Por isso, é sempre aconselhável definir um número reduzido de máquinas que serão servidores de e-mail, e que portanto rodarão o *daemon sendmail*. Nas demais estações não há risco na utilização de clientes Sendmail.

Devido ao histórico de falhas de segurança do Sendmail, sempre é aconselhável mantê-lo atualizado, sendo que as mais recentes versões podem ser obtidas no site oficial do Sendmail[12].

As mesmas considerações feitas para o Sendmail também valem para o Imap e para o Pop, ambos serviços utilizados pelas máquinas cliente para leitura de e-mails. Aconselha-se um cuidado especial na utilização destes, pois as senhas de *logon* dos usuários nestes serviços trafegam em texto claro pela rede. Canais seguros de comunicação, baseados em SSL ou SSH, podem ser utilizados para evitar tal vulnerabilidade.

10 Aplicações do Usuário

O Linux, sendo um sistema operacional multiusuário, implementa proteção de memória para as aplicações do usuário. A princípio, podemos assumir que os programas executados pelos usuários comuns (exceção a root) não representam risco de segurança para o sistema.

Todavia, uma forma de ataque de negação de serviço é buscar exaurir os recursos do sistema, sejam eles espaço em disco ou mesmo memória. Como *a priori* o Linux não limita o espaço em disco utilizado pelos usuários e suas aplicações, ele está vulnerável a programas que tentem ocupar todo o espaço em disco. Para contornar este problema, é possível habilitar no kernel e utilizar o mecanismo de quotas, através do qual pode-se limitar o espaço em disco disponível para cada usuário ou grupo de usuários.

A alocação de memória é limitada para os processos dos usuários de forma a impedir a total exaustão da mesma. Porém, recentemente foi descoberto um *bug* no mecanismo de IPC (Inter Process Communication) do Linux, que permitia que processos do usuário alocassem (através da função *shmget*) toda a memória disponível. Como conseqüência disto, os demais processos do sistema começavam a ser abortados por falta de memória.

11 Conclusão

Com base em tudo que foi apresentado neste trabalho podemos concluir:

- A crescente utilização do Linux faz com que o mesmo seja periodicamente atualizado, com novas características disponibilizadas em trit-

mo bastante alto, mas também faz com que ele se torne um dos principais alvos de ataque.

- A segurança física do sistema nunca deve ser esquecida, e com certeza o administrador deve configurar o LILO de forma a impedir acessos indevidos a root.
- O Sistema de Arquivos do Linux, assim como qualquer outro, apresenta vulnerabilidades que podem ser contornadas através da utilização de ferramentas como o CFS e o TCFS.
- As facilidades oferecidas pelo Shadow Passwords e principalmente pelo PAM devem ser utilizadas, de forma a consideravelmente incrementar a segurança do sistema.
- A versão 2.2 do kernel do Linux está relativamente estável, porém é aconselhável que o administrador periodicamente procure por atualizações mais estáveis, ou mesmo se informe de *bugs* de versões anteriores.
- O histórico de evolução do Linux revela uma curva ascendente bastante inclinada com relação ao oferecimento de novos serviços e características. Isto não desmerece, no sentido de "maior complexidade, portanto mais número de falhas, portanto maior número de falhas de segurança"; ao invés disso, muitas das novas características, estão simplificando a configuração da máquina, "normalizando" mecanismos de configuração tradicionalmente não coesos em Unixes comerciais, e até corrigindo falhas estruturais tradicionais do Unix.
- A camada TCP/IP do Linux mostra-se como uma das mais seguras dentre os Unixes, porém os serviços devem ser alvo de cuidado constante. Para tanto, é aconselhável que o administrador do sistema inscreva-se em listas de notificação de *bugs*, como é o caso da Bugtraq [13] e da Bugzilla [14], para manter-se informado.

Todas estas características estão tornando a plataforma Linux extremamente atrativa para o contexto de segurança. O acesso livre e completo ao código fonte leva esta perspectiva a níveis mais altos ainda, já que em último caso o administrador tem a possibilidade de efetuar correções de emergência, independentemente da agilidade da distribuidora de sua versão em particular.

12 Referências Bibliográficas

[1] GARFINKEL, S., e SPAFFORD, G. *Practical Unix & Internet Security*. O'Reilly & Associates, 1996.

[2] FENZI, K., e WRESKI, D. *Linux Security HOWTO.* http://sunsite.unc.edu/pub/Linux/docs/HOWTO

[3] KIM, G., e SPAFFORD, G. *Tripwire.* ftp://coast.cs.purdue.edi/pub/COAST/Tripwire

[4] CFS - Cryptographic File System. http://www.replay.com/redhat

[5] TCFS - Transparent Cryptographic File System. http://edu-gw.dia.unisa.it/tcfs/

[6] HAUGH, J. *Shadow Password Suit for Linux.* ftp://sunsite.unc.edu/pub/Linux/system/Admin

[7] MUFFETT, A. *A Proactive Password Sanity Library.* ftp://black.ox.ac.uk/~ftp/src/security/

[8] SAMAR, V., e SCHEMERS, R. *Unified Login with Pluggable Authentication Modules.* http://www.pilgrim.umass.edu/pub/osf_dce/RFC/

[9] Apache Server Project. http://www.apache.org

[10] ENGELSCHALL, R. S. *Module mod_ssl - Apache Interface to SSLeay.* http://www.engelschall.com/sw/mod_ssl/

[11] SSH COMMUNICATIONS SECURITY LTD. *Secure Shell.* http://www.cs.hut.fi/ssh

[12] Sendmail Organization. http://www.sendmail.org

[13] Bugtraq List. http://www.bugtraq.net

[14] Red Hat. *Red Hat Bug Tracking Database- Bugzilla.* http://developer.redhat.com/bugzilla/

[15] STEIN, J. G., NEUMAN, C. e SHILLER, J. L. *Kerberos: An Authentication Service for Open Network Systems*. USENIX Conference Proceedings, Dallas, Texas, 1998.