Revista Eletrônica da Faculdade Metodista Granbery
http://re.granbery.edu.br - ISSN 1981 0377
Curso de Sistemas de Informação - N. 9, JUL/DEZ 2010

Hardening em Sistemas Operacionais GNU/LINUX

Flavio Alexandre dos Reis [1], **Eduardo Pagani Julio** [1]

[1] Faculdade Metodista Granbery
CEP: 36010-532 – Juiz de Fora – MG – Brazil

reis.falexandre@gmail.com, epagani@gmail.com

Resumo

Esse artigo apresenta uma proposta de *Hardening* para blindagem de sistemas GNU/Linux, mostrando técnicas que podem ser utilizadas para que a segurança seja satisfatória, reduzindo assim risco ao sistema. São apresentadas técnicas de *hardening* para segurança em nível de *kernel*, em nível de usuários e em nível de sistema.

Palavras-Chave: Segurança. Servidores. Linux. *Kernel, Hardening*.

Abstract

This paper proposes a system for shielding Hardening Linux, showing techniques that can be used for security to be satisfactory, thus reducing risk to the system. Hardening techniques are presented for kernel-level security, the user level and system level.

***Key words**:: Security. Servers. Linux. Kernel, Hardening.*

1 INTRODUÇÃO

Na atualidade a segurança dos ativos de uma empresa é uma das atividades que mais chamam a atenção e que preocupam os administradores de rede. Com a evolução da Internet, as empresas estão cada vez mais expostas ao mundo virtual.

Uma operação em rede possibilita ganhos de produtividade pelo compartilhamento de recursos e propagação da informação, entre outras vantagens, inclusive com a finalidade de

divulgação de seus produtos. Contudo, esses benefícios trazem riscos. Conectar em rede significa possibilitar, mesmo sob condições específicas e com algum tipo de controle, o acesso externo aos recursos computacionais, inclusive às informações. Dessa forma, falhas na especificação das condições e controle de acesso poderão ser exploradas, obtendo assim acesso não autorizado a recursos da rede. Essas falhas podem causar impactos dos mais diferentes níveis, desde um simples constrangimento, passando pelo desgaste da imagem corporativa e chegando a perdas financeiras e de mercado.

A motivação fundamental para este enfoque deve-se à utilização desses padrões na rede mundial de computadores, a Internet, e à necessidade de obter uma segurança considerável em uma rede de computadores. A Internet é considerada um dos meios de comunicação mais importantes e revolucionários desenvolvidos na história da humanidade e vem contribuindo fortemente para o que se chama de globalização das informações.

Sistemas recém instalados podem conter vulnerabilidades que, se forem exploradas podem comprometer todo o projeto. *Hardening* é uma técnica de blindagem de sistemas que envolve um processo de mapeamento das ameaças, mitigação dos riscos e execução das atividades corretivas com foco na infra estrutura. Seu objetivo principal é tornar o sistema preparado para enfrentar tentativas de ataque.

O presente artigo apresenta algumas das principais técnicas de *hardening* que podem ser utilizadas para aumentar a segurança em um sistema operacional GNU/LINUX.

Além da Introdução, o artigo apresenta um outro capítulo contendo definições, vantagens e exemplificando técnicas de *hardening*.

2 HARDENING

Neste capítulo são apresentados conceitos, vantagens e procedimento pós-instalação. Muitos administradores sem experiência em segurança preparam seus servidores com uma instalação básica e, depois que suas aplicações estão disponíveis, nenhum procedimento é feito para manter a integridade do sistema (DOMINGOS, 2006).

Segundo Domingos (2008), *hardening* são ajustes finos efetuados no sistema após uma instalação.

Segundo Rodrigues (2008), o conceito de *hardening* caracteriza-se por medidas e ações que visam proteger um determinado sistema de invasores.

Segundo Hassell (2005), *hardening* é o processo de proteger um sistema contra ameaças desconhecidas. Os administradores de sistema devem endurecer uma instalação contra o que eles acham que poderia ser uma ameaça.

Segundo Melo (2009), uma instalação padrão de qualquer sistema que tenha a finalidade de servidor, o administrador tem por obrigação de melhorar a segurança ativando controles nativos ou implementando-os, esse processo é conhecido como *hardening*.

O GNU/Linux torna-se bastante seguro, uma vez que devidamente trabalhado. Por esse motivo, devem-se aperfeiçoar suas configurações. Quando uma técnica de *hardening* é aplicada, há três fatores que devem ser levados em consideração, são eles: Segurança, Risco e Flexibilidade. O administrador de redes deve dosar bem esses três fatores e levar o sistema a uma alta produtividade, garantindo segurança e tendo um nível de risco aceitável. Ter total segurança não é possível, mas quanto mais segura for sua rede, menos riscos ocorrerão. Um cenário pode ser ilustrado na Figura 1. Quanto mais seguro for o sistema, menor será o risco, porém a flexibilidade também será reduzida. Se o nível de segurança é reduzido, o risco aumenta e o sistema tem maior flexibilidade.

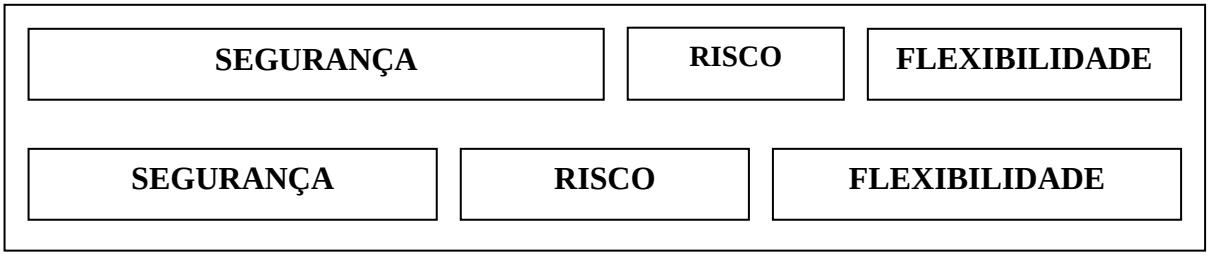

Figura 1 – Representação gráfica três fatores (Segurança, risco, Flexibilidade)

Não existe uma regra para a dosagem dos fatores apresentados; o administrador deve analisar cada situação. Por esse motivo, toda implementação deve ser bem avaliada antes de ser implementada.

2.1 Vantagens

A seguir são listadas algumas vantagens de se utilizar técnicas de *hardening*:

- em geral, utilizar técnicas de *hardening* é mapear ameaças e evitar que ocorram falhas e invasões em um sistema;

- evita que *scripts* maliciosos sejam executados;
- proteger a conta do usuário *root* de acessos indevidos;
- liberar apenas serviços que estejam realmente ativos no sistema;
- aplicar controles aos serviços disponíveis na rede;
- aplicar limite de acesso aos usuários.

Deve-se lembrar que as técnicas aqui apresentadas podem não se encaixar em todas as situações. É necessário avaliar qual o melhor procedimento para a aplicação em desenvolvimento. Recomenda-se instalar versões atuais dos sistemas operacionais, que contêm correções e *patches* de segurança. Instalar versões antigas e fazer o *upgrade* (atualizações) em seguida pode ser problemático, deixando o sistema temporariamente vulnerável (caso existam pacotes com falhas). Serviços críticos como o *apache (daemon do servidor web), imapd (daemon do servidor de e-mail), bind (daemon do servidor DNS),* devem estar sempre nas versões mais atuais. *Softwares* desnecessários devem ser desinstalados e pacotes inseguros devem ser substituídos por alternativas mais confiáveis

2.2 Segurança em sistema de arquivos

O particionamento de discos é um ponto importante quando se pensa em segurança. Particionar o disco proporciona maior segurança, pois cada partição tem sua tabela de alocação de arquivos separada.

O comando *mount* (comando UNIX usado para montar partições) permite utilizar algumas opções para aumentar a segurança nas partições. *Crackers* podem se aproveitar do diretório /*tmp*, onde, por padrão, qualquer usuário pode introduzir um *backdoor* ou qualquer outro programa malicioso para ter um acesso completo ao sistema. Na Tabela 1 é apresentado um exemplo de como pode ser configurada a tabela de partições (DOMINGOS 2006).

Tabela 1– Exemplo de particionamento

Ponto de montagem	nosuid	noexec	noatime
/boot	x	-	-
/	-	-	-
/home	x	x	-
/tmp	x	-	-

/var	x	x	-
/var/log	x	x	-
/opt	x	x	x

Onde:

- Ponto de Montagem: local onde uma partição é disponibilizada para leitura e gravação de dados;
- *nosuid*: parâmetro usado para inibir que binários com permissão de suid bit sejam executados em uma partição
- *noexec*: parâmetro usado para inibir que um binário seja executado em uma partição;
- *noatime*: elimina a necessidade de escrita no disco para arquivos que precisam ser só lidos.

2.3 Atualizações e *patches* de segurança

Patches são correções disponibilizadas a um sistema. Todo sistema operacional deve estar sempre atualizado, assim como seus componentes e pacotes instalados. É possível encontrar dicas enviadas por grupos de segurança, em que são lançadas as *Security Advisories*, alertando para possíveis falhas em softwares específicos.

Algumas distribuições como Debian e Ubuntu possuem um sistema automático para verificar se os pacotes estão desatualizados e passíveis de falhas, corrigindo ou emitindo um alerta ao administrador local. Maiores informações podem ser adquiridas acessando-se o portal do CSIRT PoP-MG, que possui uma documentação detalhando os processos de atualização para a maioria dos sistemas operacionais baseados em GNU/Linux (RODRIGUES, 2010).

2.4 Sudoers

O aplicativo *sudo* executa comandos como outro usuário, desde que a configuração do *sudo* seja efetuada corretamente para permitir tal tarefa. Isso não permite a execução de

aplicativos de forma segura, simplesmente permite ao administrador executar um programa como outro usuário (*suid*).

Este módulo não torna o código mais ou menos seguro, ele simplesmente usa um mecanismo diferente para ser executado como um usuário diferente. O *sudo* tem uma vulnerabilidade adicional: na medida em que uma senha é fornecida em uma variável, isso pode ser atacável. Em futuras versões do módulo, essa vulnerabilidade poderá ser tratada resolvendo assim essa questão (SUDO 2010).

Este módulo específico executa um único comando para obter uma saída que será repassada ao usuário. Na Tabela 2 é apresentado um exemplo de uso do aplicativo *sudo*.

Tabela 2 – Exemplo de uso do aplicativo sudo

```
$sudo invoke-rc.d networking restart
```

O arquivo */etc/sudoers* é composto de dois tipos de entradas, *aliases* e especificações do usuário. Nesse ponto serão especificadas as configurações de quem poderá utilizar o aplicativo sudo. Ao combinar múltiplas entradas para um usuário, são aplicadas em ordem.

O comando *visudo* é recomendado quando faz-se necessário uma alteração no *sudoers*. Esse programa faz uma cópia temporária do arquivo *sudoers*, verificando erros de configuração e sintaxe (GRATSOFT 2010).

Quando um sistema Ubuntu é instalado, o usuário criado durante a instalação poderá utilizar o aplicativo sudo, assim tomando poderes de root, porém os outros usuários não terão a mesma permissão. Na Figura 2 observa-se o usuário *flavio* utilizando do comando *sudo* para obter o *shell* como *root*.

Figura 2 – Exemplo de uso do aplicativo sudo

Na Tabela 3 é apresentado um exemplo de configuração do arquivo */etc/sudoer*, permitindo que o usuário possa executar o aplicativo *sudo*.

Tabela 3 – Exemplo de uso do aplicativo sudo

```
# /etc/sudoers
```

```
#
# This file MUST be edited with the 'visudo' command as root.
#
# See the man page for details on how to write a sudoers file.
#
Defaults    env_reset
# Host alias specification
# User alias specification
# Cmnd alias specification
# User privilege specification

root  ALL=(ALL) ALL

# Allow members of group sudo to execute any command after they have
# provided their password
# (Note that later entries override this, so you might need to move
# it further down)

%sudo ALL=(ALL) ALL

#includedir /etc/sudoers.d

# Members of the admin group may gain root privileges

%admin ALL=(ALL) ALL
```

Como se pode observar, assim como o *root* tem permissão total (*root ALL=(ALL) ALL*), o usuário do grupo *sudo (%sudo ALL=(ALL) ALL*) e *admin* (*%admin ALL=(ALL) ALL*) tem permissão total. Uma lista poderá ser gerada, mostrando de quais grupos o usuário *flavio* faz parte, como é mostrado na Figura 3.

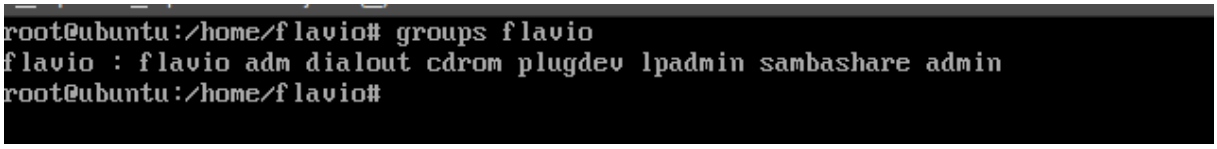

Figura 3 – Listando grupos do usuário flavio

Pode-se observar que o usuário *flavio* faz parte do grupo *admin*, logo, o mesmo poderá exercer poderes de administrador. Observe na Tabela 4 que um novo usuário é inserido.

Tabela 4– Inserindo novo usuário

```
$sudo adduser eduardo
```

Na Figura 4 pode-se observar os grupos a que o usuário recém criado pertence. Como padrão, todo novo usuário terá um grupo primário com seu próprio nome a fim de manter em segurança seus arquivos pessoais.

Figura 4 – Listando grupos do usuário eduardo

Na Figura 5 é apresentada uma tentativa do usuário *eduardo* de usar o aplicativo *sudo* para exercer funções de administrador. Observe que é enviada uma mensagem no *shell* avisando que o usuário não está configurado no arquivo *sudoers*, portando não poderá usufruir de tais direitos.

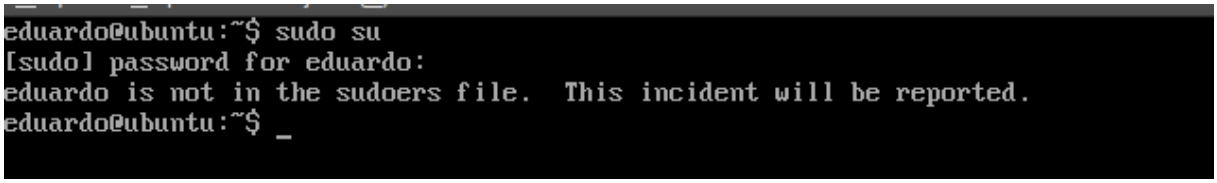

Figura 5 – Tentativa de uso do comando sudo por usuário recém criado

Caso venha existir necessidade do usuário *eduardo* exercer funções de administrador ou até mesmo parte dessas funções, pode-se criar um grupo para tal uso e inserir as configurações necessária no arquivo */etc/sudoers*.

2.5 Quota de disco

Quando um servidor é configurado, seja um servidor de arquivo ou um servidor de e-mail, por exemplo, cada usuário terá uma área disponível para gravação de dados. Com o sistema de quotas, é possível controlar a utilização de espaço no sistema de arquivos entre todos os usuários. Dessa forma pode-se impedir que um usuário exceda os limites físicos do sistema de arquivos, gravando arquivos de música, vídeo, imagens, ocupando todo o espaço disponível e comprometendo assim sua utilização por outros usuários. O sistema de quotas somente poderá ser aplicado em partições, e nunca em diretórios (DOMINGOS, 2006).

Partindo do princípio que o sistema está dividido em partições e que o diretório */home* está em uma dessas partições, o sistema de cotas poderá ser implementado. Na Figura 6 é apresentada uma lista com o comando *mount*, um filtro com o comando *grep home* é utilizado, apresentando assim que a partição /*home* está com quota de disco ativada.

Figura 6 – Quota de disco ativa no diretório /home

Na Figura 7 pode-se observar que o usuário *eduardo* teve seu limite de quotas excedido.

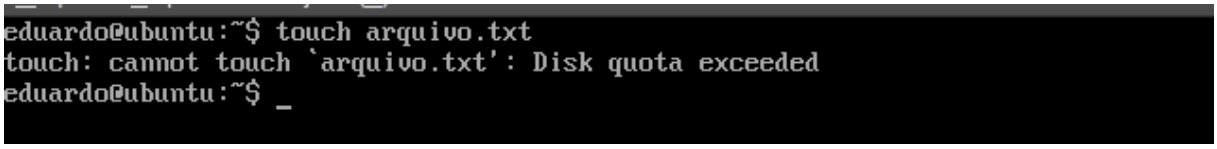

Figura 7 – Usuário Eduardo excedendo o limite de quotas

2.6 Serviços desnecessários e inseguros

Depois de instalado um sistema, deve-se procurar saber se todos os programas instalados são realmente necessários, mesmo sendo uma instalação básica. Em um servidor nunca deverão existir programas "clientes". Serviços como *telnet*, *rshd*, *rlogind*, *rwhod*, *ftpd*, *sendmail*, *identd*, *wget*, dentre outros, deverão ser removidos (DOMINGOS, 2006).

Estes serviços podem ser desinstalados usando-se o gerenciador de pacotes do sistema operacional, ou desativando-se todos os níveis de inicialização. Além disso, podem-se remover entradas específicas dos programas no *boot* do sistema operacional. Em distribuições baseadas em *Red-Hat*, pode-se usar o aplicativo *chconfig*, que, em suas versões mais novas, edita as entradas do *xinetd* automaticamente (RODRIGUES, 2010).

A remoção de pacotes obsoletos deverá ser executada, evitando assim que vulnerabilidades sejam exploradas. Na Tabela 5, a seguir, é apresentado o comando *dpkg*, utilizado em distribuições *GNU/Linux Debian* e derivados. Esse comando faz uma pesquisa no sistema e lista todos os pacotes instalados; um filtro com o comando *awk* é utilizado para formatar a saída do comando, mostrando assim somente a segunda e a terceira coluna; o comando *sed*, nesse exemplo, retira as 5 primeiras linhas. Esse resultado será gravado em um arquivo texto chamado *pacotes.txt*, dentro do diretório */root/auditoria*.

Tabela 5 – Listando pacotes instaladores no Debian.

```
# dpkg -l | awk '{print $2,$3}' | sed '1,5d' >/root/auditoria/pacotes.txt
```

Na Tabela 6, a seguir tem-se um exemplo de como se fazer a mesma pesquisa em distribuições *Red Hat* e derivados:

Tabela 6 – Listando pacotes instalados no Red Hat.

```
#rpm –qa > /root/auditoria/pacotes.txt
```

A análise desse arquivo pode ser um pouco demorada, ainda mais se o administrador estiver analisando a saída gerada por um servidor *Red Hat*, uma vez que essa distribuição traz um número maior de pacotes instalados que um *GNU/LINUX Debian*. Um pacote interessante para remover é o *wget*. Com esse comando, um *cracker* pode fazer com que o servidor vire um alvo, executando *download* de arquivos para dentro do mesmo através de um servidor *web* forjado. Assim o *cracker* pode jogar qualquer *script* que possa danificar o sistema ou até mesmo abrir uma porta para novas invasões.

O comando *aptitude*, nas Distribuições Debian e derivadas e com o comando *rpm* nas Distribuições Red-Hat e derivados (Tabela 7), remove o aplicativo *wget, e* a opção *purge* faz com que o arquivos de configuração do aplicativo sejam removidos, caso existam, não deixando vestígios de sua instalação.

Tabela 7 – Removendo aplicativos.

```
Debian
# aptitude purge wget

Red-Hat
#rpm –e wget
```

Existem outros pacotes que devem ser removidos, como por exemplo *telnet, ftpd, sendmail, identd*. Seu uso introduz uma vulnerabilidade ao sistema. Para esses serviços já existem novos pacotes com segurança já implementada. Outro exemplo seria o pacote *wireless-tools*; se o servidor em questão não tem nenhuma placa de rede sem fio, não há necessidade de tal aplicativo.

2.7 Desativando o uso do CTRL + ALT + DEL

Este é outro ponto importante a ser levado em consideração quando se pensa em controle de acesso. Em uma organização em que qualquer um tem acesso ao teclado do seu servidor, usuários mal intencionados podem simplesmente usar o CTRL+ALT+DEL para

reiniciar o servidor. Isso pode ocorrer com empresas que não têm uma política de acesso aos seus servidores. Segundo a norma ISO 27002, devem-se tratar as questões de acesso físico à sala de servidores. Essa função pode ser desativada no arquivo */etc/init/controlo-alt-delete.conf*, conforme a Tabela 8. Observe a parte em destaque, antes e depois da alteração.

Tabela 7 – Exemplo do arquivo /etc/init/control-alt-delete.conf.anterado

```
# control-alt-delete - emergency keypress handling
#
# This task is run whenever the Control-Alt-Delete key combination is
# pressed, and performs a safe reboot of the machine.

description "emergency keypress handling"
author      "Scott James Remnant <scott@netsplit.com>"

start on control-alt-delete

# Configuração original
task
exec shutdown -r now "Control-Alt-Delete pressed"

# Nova configuração
# comente as duas linhas anteriores (em destaque)
task
exec echo "Control-Alt-Delete pressed"
```

Esse arquivo é lido somente durante o *boot* do sistema. Para ativar o que foi alterado, será necessário executar o comando *init q*, apresentado na Tabela 8:

Tabela 8 – Comando init q

```
#init q
```

2.8 A Variável TMOUT

Um usuário mal intencionado com acesso físico à sala de servidores pode usufruir de uma estação logada como usuário *root*. A variável *TMOUT* tem a função de executar um *logout* automático após determinado tempo de inatividade e é configurada no arquivo */etc/profile*. Na Tabela 9 é apresentado um exemplo do arquivo */etc/profile,* no qual pode-se observar em destaque a variável com o valor de 60 segundos. O valor a ser adicionado deverá ser analisado com cuidado, evitando-se assim acessos indevidos.

Tabela 8 – Comando init q

```
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).

if [ -d /etc/profile.d ]; then
  for i in /etc/profile.d/*.sh; do
    if [ -r $i ]; then
      . $i
    fi
  done
  unset i
fi

if [ "$PS1" ]; then
  if [ "$BASH" ]; then
    PS1='\u@\h:\w\$ '
    if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
    fi
  else
    if [ "`id -u`" -eq 0 ]; then
      PS1='# '
    else
      PS1='$ '
    fi
  fi
fi
TMOUT=60
export PATH TMOUT
umask 022
```

Na Figura 8 pode-se observar que o sistema efetuou o *logoff* automaticamente, após 60 segundos de inatividade.

Figura 8 – Sistema efetuando logoff automaticamente

2.9 Usuários inválidos

Nos Sistemas GNU/Linux, há três tipos de usuários: usuário *root*, que é o administrador do sistema; usuários comuns, que possuem uma senha para logar no sistema e acesso a um diretório *home*, no qual os mesmos poderão ter privacidade com seus arquivos pessoais; e, por último, os usuários de sistema, responsáveis por controlar requisições de serviços.

O *shell* é a interface entre usuário e sistema. Sem o *shell* não é possível digitar comandos e interagir com o sistema. Analisando o arquivo */etc/passwd*, na Figura 9, observa-

se que o usuário de sistema *www-data,* responsável por receber requisições do servidor *web* está com um *shell* ativo. Dessa forma, o mesmo terá a possibilidade de efetuar *login*, deixando assim uma brecha de segurança.

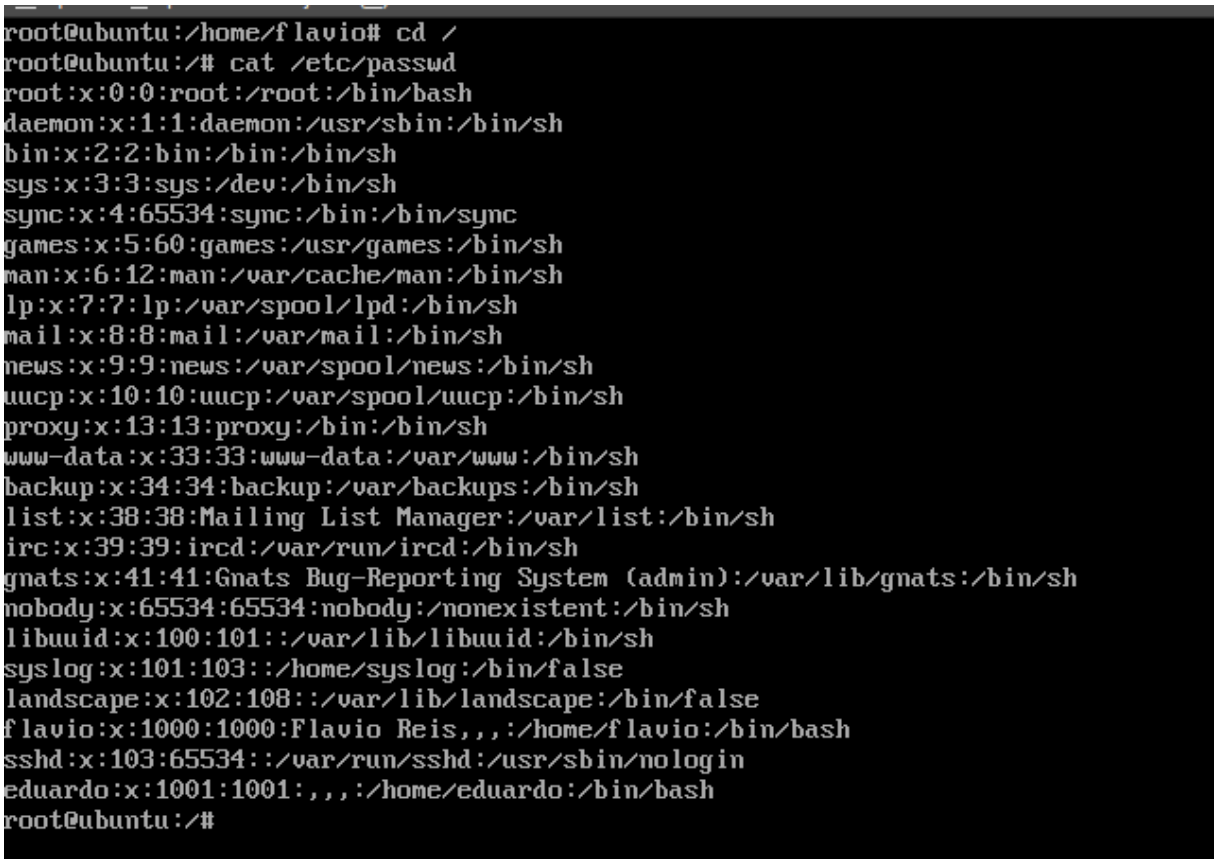

Figura 9 – arquivo */etc/passwd* padrão sem alterações, com *shell* ativo para todos usuários

Na Tabela 9 é apresentado um *script* para desativar o *shell* dos usuários de sistema, deixando-o ativo apenas para usuários especificados.

Tabela 9 – Script para remoção de *shell* dos usuários inválido.

```
# vim /root/auditoria/invalidos.sh

#!/bin/bash

for USER in $(cat /etc/passwd | cut -f 1 -d ‘‘:’’ | grep -v root | grep
-v flavio | grep -v eduardo)
do
  usermod -s /bin/false $USER
done
```

Na Tabela 10 são alteradas as permissões desse *script*, deixando que apenas o usuário *root* possa executar. O *script* é executado e em seguida o arquivo *passwd* é listado novamente, conforme pode ser observado na Figura 10, agora já com a *shell* setada para */bin/false*.

Tabela 10 – Alterando a permissão do script e executando o mesmo.

```
# chmod 700 invalidos.sh
# ./invalidos.sh
# cat /etc/passwd
```

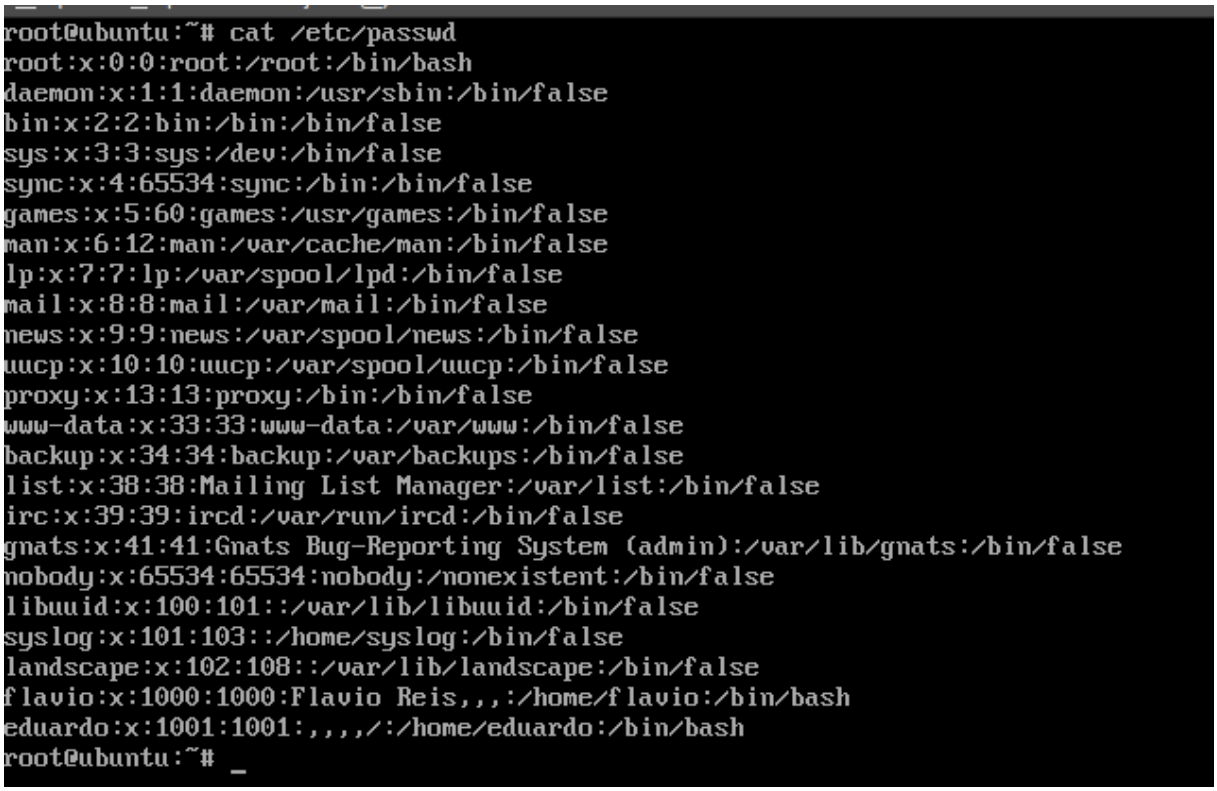

Figura 10 – Arquivo /etc/passwd após executar o script sugerido

2.10 NOEXEC

Um *script* malicioso pode ser inserido em uma partição do sistema, causando danos ao servidor. O comando *mount* tem um parâmetro chamado *noexec;* com seu uso pode-se evitar que com *script* seja executado dentro dessa partição.

Na Tabela 11, o primeiro comando remonta a partição */tmp* com a opção de leitura e escrita e acrescentando o *noexec;* o segundo comando é utilizado para conferir se a alteração foi efetivada, conforme pode ser observado na Figura 11.

Tabela 11 - Remontando o diretório /tmp com a opção *noexec*.

```
# mount -o remount,rw,noexec /tmp
# mount
```

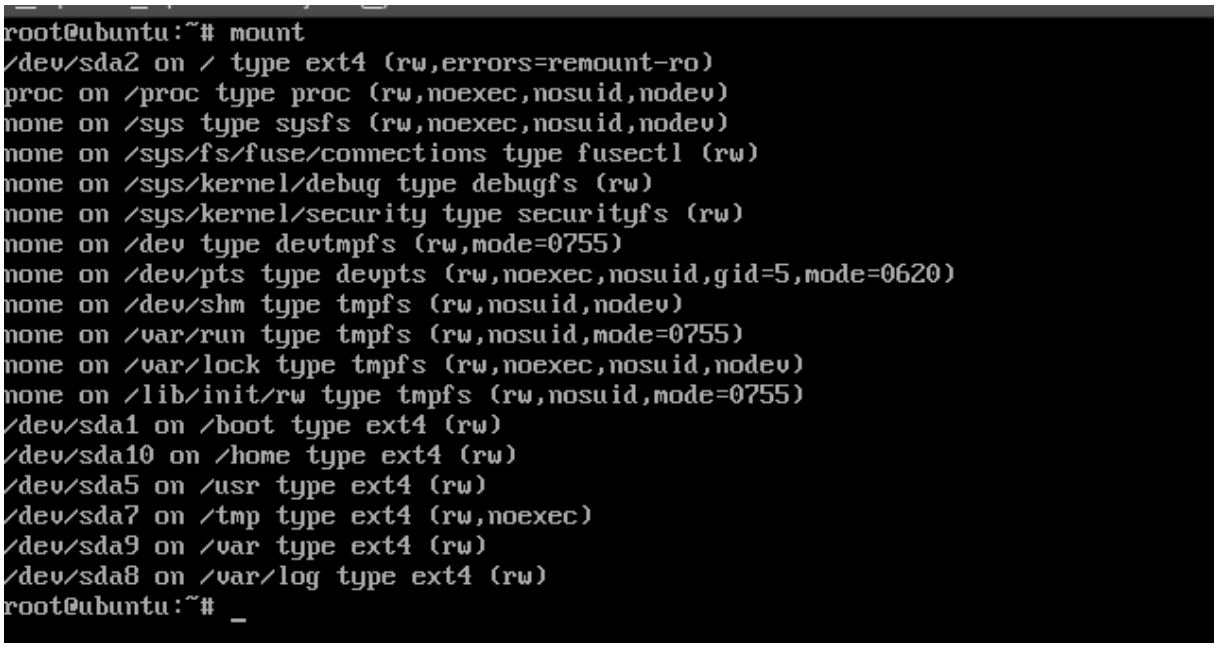

Figura 11 – Retorno comando mount, com noexec aplicado a partição /tmp

Após remontar a partição pode-se executar um *script* qualquer Observe que sua execução independe de usuário. No exemplo, foi executado como *root (*observe na Tabela 12, a exceção do *shell"teste.sh*", e a Figura 21, a mensagem de permissão negada).

Figura 12 – Executando um script no partição /tmp

O administrador poderá encontrar um problema ao executar o aplicativo *aptitude* se as partições /*var* e /*tmp* estiverem com o *noexec* e *nosuid* ativos. Esse aplicativo precisa executar *scripts* dentro de tais partições. Na Tabela 12 é apresentado um *script* que pode ser utilizado para contornar tal problema; dessa forma pode-se remover a proteção, executar o aplicativo e, em seguida, reativar as opções.

Tabela 12 – *Script* para alterar opção *noexec* e *nosuid* em partições desejadas.

```
#!/bin/bash
case $1 in
  start)
    mount -o remount,rw,noexec /var
    mount -o remount,rw,noexec /tmp
    mount
    echo "Partições SEM permissão de execução"
  ;;
  stop)
    mount -o remount,rw,exec /var
    mount -o remount,rw,exec /tmp
    mount
    echo " Partições COM permissão de execução "
  ;;
  *) echo "erro use $0 {start|stop}"
    exit 0
  ;;
esac
exit 1
```

Com esses cuidados o administrador poderá garantir uma segurança satisfatória para o sistema. Outras práticas de *hardening* podem ser utilizadas, esse artigo foram apresentadas apenas alguma das várias existentes. Outros exemplos podem ser encontrados na literatura.

3 CONCLUSÃO

Implementar um plano de segurança é um processo demorado, exigi uma analise detalhada do sistema. Enquanto isso, o sistema pode ser comprometido ou não estar operando adequadamente. Pode deixar escapar informações, ficar ocupado, infectar outro sistema na rede, ou mesmo fazer parte de ataques ordenados por outras máquinas *Bots*.

Independente do status de segurança, sistemas instáveis devido a problemas de hardware ou de energia podem se enfraquecer com seus esforços de segurança. Antes de proteger um sistema de produção atual, é preciso identificar se ainda é o seu sistema que precisa de proteção. É necessário se certificar de que esteja funcionando corretamente. Assim que implementada, é necessário determinar se as etapas seguidas estão realmente mantendo o sistema em segurança.

Testar o sistema identificando seu *status*, procure por evidências de invasões não autorizadas, presença de *malware*, *rookit* ou evidências de ataque. A limpeza e a recuperação de um sistema podem exigir ferramentas especializadas, siga instruções para remoção de

arquivos ou até mesmo a reconfiguração de todo o sistema. Em um sistema comprometido, devem-se analisar os custos de se recuperar o sistema ou reinstalá-lo. Um sistema em mal funcionamento poderá comprometer toda estrutura de uma rede.

Com esse artigo, pode-se verificar a importância de se criar um cronograma para análise e implementação de um plano de segurança. O tema *hardening* é bem extenso e permite que novos projetos sejam estudados e implementados. Detalhes em servidores *web* podem ser observados, como retirar *banners* com informações da versão do pacote que está sendo utilizado, alterar a senha padrão de um banco de dados, assim como algumas medidas devem ser tomadas por programadores, evitando ataque de *sql-injection*.

As ameaças são diversas, sejam elas externas ou até mesmo internas, *vírus*, *worns*, *trojan* e podem danificar não só um servidor, mas se espalhar por toda uma rede. Dessa forma o administrador deverá ter bons softwares de *antivírus* e *anti-spam* instalados, e sempre bem atualizados nas estações de trabalho, lembrando que contaminações podem surgir através de *pen-drives* infectados e arquivos maliciosos enviados por e-mail.

Um *firewall* bem configurado, trabalhando em conjunto com um *IDS*, poderá aumentar significamente a segurança em uma rede.

Como contribuição de trabalhos futuros poderá ser efetuada analisando e implementando o *SE-Linux*, que é uma ferramenta para aplicar *hardening* em nível de *Kernel*. O *AppArmor* utilizado pela distribuição Ubuntu também poderá ser analisado e implementado como um estudo de caso. Novos projetos serão apresentados, como Firewall, Filtro de Pacotes e Técnicas Avançadas de Roteamento, quando serão utilizadas ferramentas exemplificadas neste trabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

BENTO, Juliano. **Hardening em Linux**. Disponível em <ttp://www.slideshare.net/fgsl/palestra-hardening-linux-por-juliano-bento-v-fgsl-e-isgsl> Acesso em Abril de 2010.

DOMINGOS, César. **Curso de Segurança em Servidores GNU/Linux, Baseado na norma ISO 27002**. 4Linux – Free Software Solutions. 2010.

HASSELL, Jonathan. **Hardening Windows**. Editora Apress. 2ª Edição 2005. > Acesso em Abril de 2010.

ISO 27000. Disponível em <http://www.27000.org/ > Acesso em Abril de 2010.

RODRIGUES, B. M. Linux Hardening, **CSIRT PoP-MG.** Disponível em: <http://www.csirt.pop-mg.rnp.br/docs/hardening/linux.html> Acesso em: jan de 2010.

SUDO. **SUDOERS MANUAL**. Disponível em

<http://www.gratisoft.us/sudo/man/sudoers.html> Acesso em Jun 2010