Hardening - Artigo Revista Infra Magazine 1

Este artigo tem como objetivo demonstrar de forma prática algumas técnicas de blindagem de sistema também conhecidas como Hardening.

Infra Magazine 1

[Artigo disponível no **Leitor Digital DevMedia**. Clique aqui para acessá-lo]

> Clique aqui para ler todos os artigos da Infra Magazine 1

Atenção: esse artigo tem um vídeo complementar. Clique e assista!

Do que se trata o artigo:

Este artigo tem como objetivo demonstrar de forma prática algumas técnicas de blindagem de sistema também conhecidas como *Hardening*. Esta técnica consiste na implementação de diretivas de segurança que devem ser seguidas antes, durante e após a instalação e configuração de servidores GNU/Linux.

Para que serve:

A técnica de *Hardening* pode ser utilizada em qualquer sistema operacional. Com o grande aumento no número de ameaças existentes na Internet é fundamental que o sistema de um servidor esteja preparado para superar todas as tentativas de invasão. Esta técnica não deve ser implementada somente em servidores que ficam conectados diretamente a Internet, muitas vezes fornecendo serviços como, por exemplo servidores web, mas também em máquinas que provêm serviços internos de rede como servidores de arquivos e de impressão. Com a blindagem de sistemas é possível aumentar o desempenho do hardware, liberando recursos que estão sendo utilizados por aplicativos desnecessários, implementando configurações específicas em alguns serviços, além de gerar um ambiente mais seguro.

Em que situação o tema é útil:

Hardening pode ser utilizado para evitar que usuários mal intencionados aproveitem da ausência do administrador e implantem scripts maliciosos em servidores infectando toda a rede, bloquear que o usuário administrador faça login diretamente no terminal, efetuar logout por tempo de inatividade, remover pacotes que não são utilizados, remover permissões especiais de binários executáveis, dentre outras técnicas que serão apresentadas posteriormente.

Autores: Flávio Alexandre dos Reis, Marcos Fabiano Verbena e Eduardo Pagani Julio

Devido ao crescente número de ameaças existentes na Internet e dentro dos ambientes corporativos, se faz necessária

a utilização de técnicas capazes de proporcionar maior segurança, estabilidade e tranquilidade para os administradores de redes. Pensando nisso, este artigo tem como objetivo apresentar o conceito e técnicas disponíveis para a implementação de *hardening* em servidores GNU/Linux.

Hardening, ou blindagem de sistemas, consiste na utilização de técnicas para prover mais segurança a servidores que disponibilizam serviços externos, como servidores Web, ou até mesmo serviços internos, como servidores de banco de dados, de arquivos, entre outros.

Neste artigo são analisadas e discutidas técnicas para a aplicação de *hardening* desde a instalação do sistema operacional, o processo de particionamento de discos, análise de serviços desnecessários e inseguros, localização de senhas fracas, verificação de usuários inválidos, desconexão de usuários não autorizados, implementação de políticas de utilização de serviços de rede, gerenciamento de privilégios e aplicação de segurança no terminal.

Para a realização da parte prática do artigo, foram utilizados três computadores com as seguintes configurações: Cliente 1: Ubuntu 9.10; Cliente 2: Ubuntu 9.10; Servidor: Debian Lenny.

Para o servidor, o sistema operacional escolhido foi o GNU/Debian Lenny, por ser um Linux reconhecidamente pela comunidade como estável e robusto, voltado para servidores. As técnicas aqui demonstradas foram implementadas no servidor e as máquinas clientes foram utilizadas apenas para a realização de acessos e monitoramento, conforme a **Figura 1**.

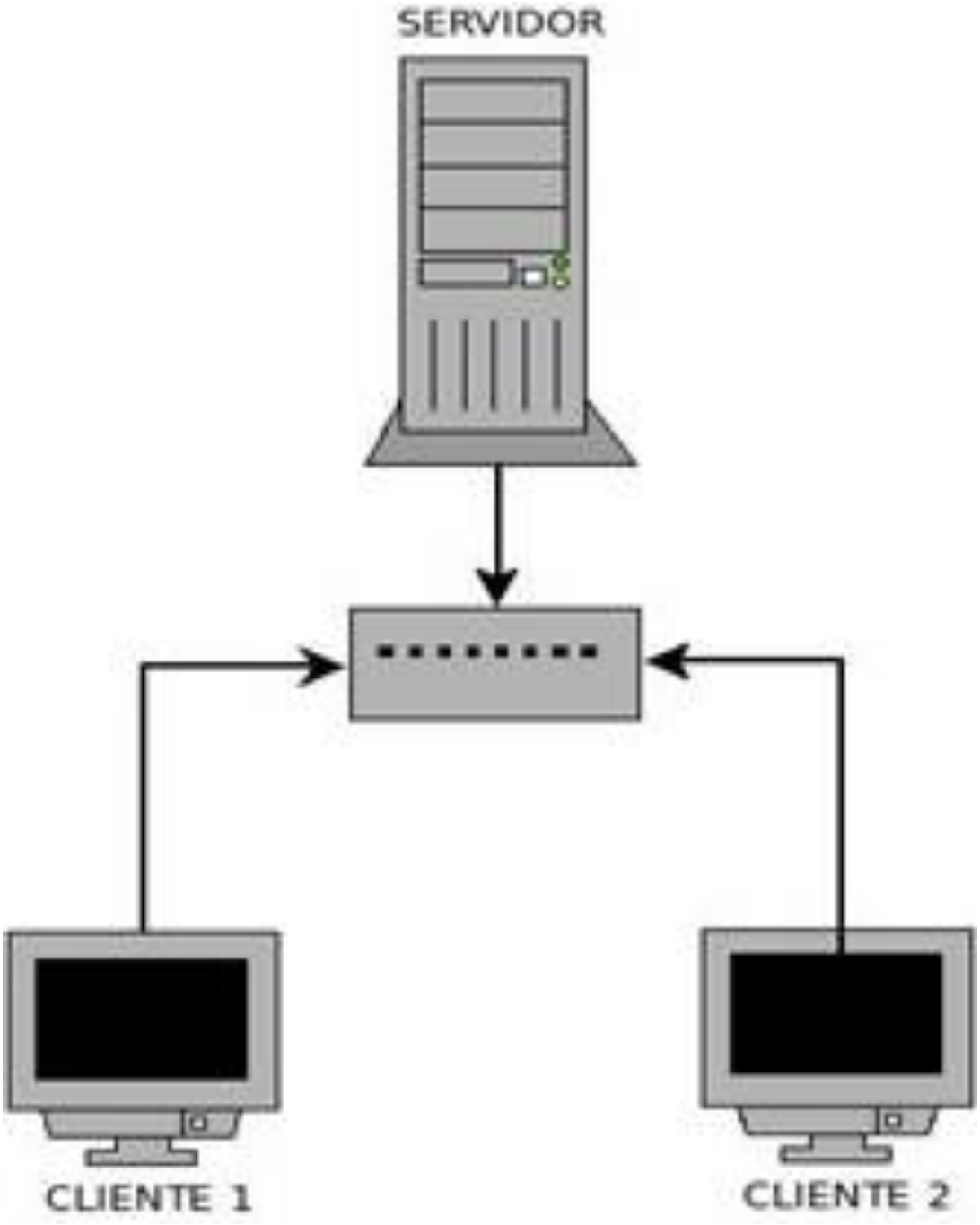

Figura 1. Estrutura utilizada para o artigo.

> **Dica DevMan**: O administrador deverá ter atenção ao implementar as técnicas. Caso não tenha experiência nos ex emplos aqui demonstrados, faça os testes em máquinas virtuais a fim de entender e evitar possíveis falhas na imple mentação.

Hardening

O *hardening* consiste na realização de alguns ajustes finos para o fortalecimento da segurança de um sistema. 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.

Em um sistema GNU/Linux é possível atingir um alto nível de segurança implementando configurações que permitam o aperfeiçoamento da segurança aplicada ao sistema. Quando se deseja aplicar a técnica de *hardening* há três grandezas que devem ser consideradas: segurança, risco e flexibilidade.

O administrador de redes deve analisar muito bem essas grandezas e encontrar um estado de harmonia entre elas, levando o sistema a uma alta produtividade e segurança, pois quanto maior a segurança menor o risco e também a flexibilidade.

É importante ressaltar que as técnicas aqui apresentadas podem não ser adequadas para todas as situações. Por isso, antes de implantar efetivamente as técnicas de *hardening*, é fundamental que haja um estudo completo do cenário e serviços em questão. Inicialmente recomenda-se sempre instalar versões atuais dos sistemas operacionais, que contenham correções e *patches* de segurança, pois pode ser problemático utilizar uma versão antiga sem atualizações, deixando o sistema temporariamente vulnerável no caso da existência de pacotes com falhas. Serviços críticos como web, email e DNS devem estar sempre nas versões mais atuais. Softwares desnecessários devem ser desinstalados e pacotes inseguros devem ser substituídos por alternativas mais confiáveis.

A seguir são apresentadas algumas utilizações de *hardening* muito úteis na configuração de sistemas operacionais.

Particionamento de discos

O particionamento de discos é um ponto importante quando se pensa em segurança. Ao particionar o disco, é inserida no sistema uma maior segurança, pois cada partição tem sua tabela de alocação de arquivos separada. Um exemplo de como pode ser configurada a tabela de partições é apresentado na **Tabela 1.**

Ponto de Montagem	nosuid	noexec	noatime
/boot	X	-	-
/	-	-	-
/home	X	x	-
/usr	X	-	-
/tmp	X	x	-
/var	X	x	-
/var/log	X	x	x

Tabela 1. Opções da tabela de partições.

> **Dica DevMan: nosuid** é o parâmetro usado para inibir a execução de binários com permissão de suid bit; **noexec** é o parâmetro usado para inibir a execução de um binário na partição; e **noatime** é o parâmetro responsável por el iminar a necessidade de escrita no disco para arquivos que precisam ser somente lidos.

O comando *mount* (comando UNIX usado para montar partições) permite utilizar algumas opções para aumentar a segurança nas partições. Crackers podem aproveitar do diretório */tmp*, onde por padrão, qualquer usuário pode gravar dados no sistema, para introduzir um *backdoor* ou qualquer outro programa malicioso para ter um acesso completo ao sistema.

> **Backdoor:** são programas que instalam um ambiente de serviço em um computador, tornando-o acessível à distân cia, permitindo o controle remoto da máquina sem que o usuário saiba, como uma porta dos fundos não autorizada.

Serviços desnecessários e inseguros

Depois do sistema instalado, deve ser realizada uma verificação minuciosa de todos os programas instalados e se são realmente necessários, mesmo sendo uma instalação básica. Um servidor nunca deve conter programas “clientes”. Serviços como telnet, rshd, rlogind, rwhod, ftpd, sendmail, identd, wget, dentre outros, deverão ser removidos. Estes

serviços podem ser desinstalados usando o gerenciador de pacotes do sistema operacional, ou desativando-os em todos os níveis de inicialização. Além disso, podem-se remover entradas específicas dos programas no *boot* do sistema operacional.

Neste exemplo são apresentados os passos necessários para a remoção de pacotes que não são utilizados no sistema. A remoção de pacotes obsoletos deverá ser executada, evitando assim que vulnerabilidades sejam exploradas. Para seguir o exemplo, crie um diretório em */root* chamado auditoria, onde será gerado um arquivo com todos os pacotes que estão instalados, podendo assim analisar quais serão removidos.

O comando *dpkg -l*, utilizado em distribuições GNU/Linux Debian e derivados, 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 terceira colunas, e o comando *sed*, nesse exemplo, retira as cinco primeiras linhas. Esse resultado é gravado em um arquivo texto chamado *pacotes.txt*, como pode ser visto na **Listagem 1**.

Listagem 1. Listando pacotes instalados no Debian.

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

Em distribuições Red Hat e derivados utiliza-se o comando *rpm –qa*, fazendo assim uma pesquisa em todos os pacotes instalados, como pode ser visto na **Listagem 2.**

Listagem 2. 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 alvo execute *downloads* de arquivos, através de um servidor Web forjado, por exemplo. Assim, o *cracker* pode jogar qualquer *script* que possa danificar o sistema ou até mesmo abrir uma porta para novas invasões.

O comando *aptitude* remove o aplicativo *wget*, e a opção *purge* faz com que os arquivos de configuração do aplicativo sejam removidos, caso existam, não deixando vestígios de sua instalação, como pode ser visto na **Listagem 3**.

Listagem 3. Removendo aplicativos.

```
# aptitude purge wget
```

Outro exemplo é o pacote *wireless-tools*. Caso o servidor em questão não tenha nenhuma placa de rede sem fio, não há necessidade de tal aplicativo.

Procura de senhas fracas

A senha deve ser única, intransferível e de propriedade de um único usuário. O administrador não deve saber essas senhas. Um procedimento muito comum para isso consiste na alteração da senha no momento do primeiro login do usuário. Dessa forma o administrador não terá acesso à senha escolhida pelo usuário. Mas, na maioria das vezes, as senhas escolhidas são fáceis de serem descobertas, pois são usadas sequências simples, como 123456, ou informações pessoais como datas de aniversário, nome próprio, entre outras.

Nestes casos, é possível utilizar ferramentas capazes de avaliar se a senha do usuário é fraca ou não. O utilitário *John the Ripper* pode ser utilizado com a finalidade de encontrar senhas fracas escolhidas por usuários de um sistema. Encontrando senhas fracas, o administrador poderá solicitar que o usuário efetue a substituição da senha por outra que atenda a política de segurança, tornando assim o sistema mais robusto e tolerante à exploração de vulnerabilidades.

Se uma auditoria está sendo executada em um servidor GNU/Linux de uma empresa, será necessário descobrir se os usuários estão usando senhas fracas. No que diz respeito às senhas, a norma ISO 27002 diz nos itens 11.2.3 e 11.3.1 que devem ser controladas por meio de um processo de gerenciamento formal e que os usuários sejam solicitados a seguir boas práticas de segurança da informação na seleção e uso de senhas.

No exemplo, o aplicativo *John the Ripper* é utilizado para teste de senhas fracas. Ele é uma ferramenta de *Brute Force* (força bruta, que testa combinações de senha para encontrar a resposta), que pode ajudar a descobrir senhas fracas de usuários, checando diretamente o arquivo */etc/shadow* dos servidores. O aplicativo pode ser instalado com o *aptitude*, como pode ser visto na **Listagem** 4.

Listagem 4. Instalando o *John the Ripper*.

```
#aptitude install john
```

O *John the Ripper* é executado passando como parâmetro o arquivo */etc/shadow*. O comando *time* foi acrescentado para que possa contar o tempo que o aplicativo levou para descobrir a senha dos usuários e também do *root*, conforme mostra a **Figura 2.**

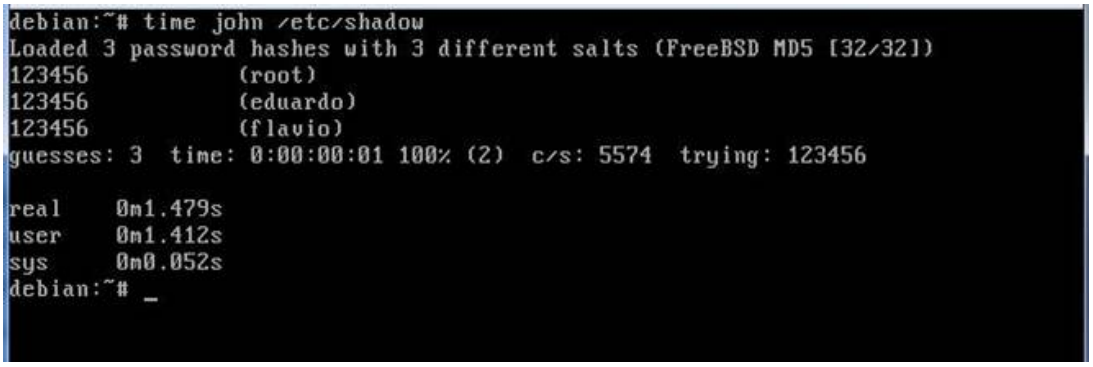

Figura 2. Execução do *John the Ripper* procurando por senhas fracas.

Regras para definir senhas fortes devem ser levadas em consideração como, por exemplo, estipular um número mínimo de caracteres (por exemplo, 10 caracteres), utilizar letras minúsculas, maiúsculas, números e caracteres especiais. Faça com que as senhas sejam alteradas em intervalos de tempo curtos, que podem ser definidos pelo administrador. Um valor que pode ser adotado é a cada 20 dias, por exemplo. Defina a quantidade de senhas já utilizadas que não poderão ser reaproveitadas. Deve-se também educar os usuários para que não divulguem suas senhas para terceiros.

Tomando essas providências simples, pode-se impedir que algum atacante consiga quebrar a senha utilizando força bruta. Mesmo que a senha seja quebrada em algum momento, com a troca de senhas periódica, esta poderá já não ser mais válida.

O PAM (*Pluggable Authentication Module*) é um conjunto de bibliotecas compartilhadas que permitem ao administrador do sistema local definir como determinadas aplicações autenticam os usuários, sem a necessidade de modificar e recompilar programas. O PAM é um recurso que auxilia muito quando se pensa em segurança. Existem vários módulos que podem ser implementados, aumentando assim o controle de criação e troca de senhas. A **Figura 3** mostra que o usuário não conseguiu mudar a senha na primeira tentativa, recebendo a mensagem “Escolha uma senha mais longa”. Isso ocorre devido a uma alteração no módulo do PAM. A opção min=10 foi adicionada, limitando assim um tamanho mínimo para a senha escolhida, como pode ser visto na **Listagem 5**.

Figura 3. Alterando as senha dos usuários.

Listagem 5. Limitando o tamanho mínimo de uma senha.

```
password required pam_unix.so nullok obscure min=10 md5
```

Usuários Inválidos

Nos Sistemas GNU/Linux há três tipos de usuários, usuário *root*, que é o administrador do sistema, usuários comuns, os quais possuem uma senha para logar no sistema e acesso a um diretório *home* onde 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 um *shell* válido, não é possível digitar comandos e interagir com o sistema. O usuário de sistema *www-data*, responsável por receber requisições do servidor Web, que esteja com um *shell* válido, poderá introduzir vulnerabilidades ao seu sistema.

Desconexão de usuários não autorizados

Inicialmente é importante ter o conhecimento de que usuários não autorizados podem estar dentro ou fora da empresa.

O acesso não autorizado por sistemas externos deve ser cancelado com extrema urgência, em especial se o usuário estiver ocultando sua identidade. Para os casos de acesso por usuários internos não autorizados, podem ser necessárias ações disciplinares dependendo da natureza do acesso.

É importante salientar que, quando ocorre uma invasão de sistema, é fundamental que as evidências relacionadas ao acesso indevido sejam registradas antes da desativação da conta não autorizada, tendo cuidado para não destruir provas relacionadas ao crime.

Colocar senha criptografada no GRUB

Muitos administradores não estão preparados para lidar com estruturas críticas. O simples acesso físico de um usuário a sala de servidores pode representar uma violação de segurança grave, pois este poderá conseguir acesso de *root* se reiniciar o servidor e alterar a senha do *root* através do gerenciador de *boot* (*grub*). Esse processo poderá ser evitado se uma senha criptografada for adicionada ao gerenciador, não permitindo que um usuário qualquer inicie o sistema no modo de segurança, por exemplo.

Política de utilização de serviços de Rede

TCP wrappers oferecem controle de acesso a vários serviços. A maioria dos serviços de rede modernos, como SSH, Telnet e FTP, utilizam os *TCP wrappers* que ficam monitorando a entrada de um pedido e o serviço requisitado.

O uso do *TCP wrappers* é uma boa prática na implementação de segurança em redes, limitando o uso dos serviços de rede. Liberar acesso somente a IPs desejados, configurar as restrições do *ssh* não permitindo *login* como *root* e configurar os módulos do *pam* para restringir acesso ao servidor em determinado horário, são boas práticas que devem ser adotadas.

Gerenciamento de privilégios

O usuário *root* é o mais visado por *crackers* ou usuários mal intencionados. Seu foco é conseguir a senha *root* e obter acesso total ao sistema.

Para dificultar a ação destas ameaças, desativar o *login* como usuário *root* nos terminais modo texto torna-se fundamental. Dessa forma, o administrador deverá efetuar o *login* como usuário comum e quando for necessário executar uma tarefa administrativa tornar-se *root* com o comando *su*. Determinar a data de validade para a senha dos usuários e, com auxílio do comando *usermod*, remover *shells* válidos de usuários que não estão em uso também são ações importantes a serem tomadas para garantir a robustez do sistema.

Segurança no Terminal

Quando é citado o assunto segurança, logo se imagina a exploração de uma vulnerabilidade por uma ameaça remota e, na maioria das vezes, as ameaças internas são esquecidas ou até mesmo subestimadas. Usuários internos mal intencionados podem causar grandes prejuízos e, se o usuário em questão tiver acesso físico aos servidores, a situação fica ainda mais grave.

Um usuário mal intencionado com acesso físico a sala de servidores pode usufruir de uma estação logada como usuário *root* e assim danificar o sistema. A variável TMOUT tem a função de executar um *logout* automático após determinado tempo de inatividade do terminal. Seu valor pode ser configurado no arquivo */etc/profile*. O valor a ser adicionado deverá ser analisado com cuidado, evitando assim acessos indevidos. Valores muito altos podem dar espaço para que o usuário tenha acesso à estação logado como root, valores muito baixos podem interferir em tarefas onde é necessário uma pesquisa, por exemplo. O arquivo */etc/profile* será editado e adicionado a variável com valor de 60 segundos, como pode ser visto na **Listagem 6**.

Listagem 6. Inserindo valores no /etc/profile.

```
#vim /etc/profile

TMOUT=60
export PATH TMOUT
```

O arquivo */etc/profile* só é lido durante o boot do sistema, dessa forma utiliza-se o comando *source* para que o arquivo possa ser lido novamente atribuindo ao sistema as alterações aplicadas, como pode ser visto na **Listagem 7**.

Listagem 7. Comando source.

```
#source /etc/profile
```

Após 60 segundos de inatividade, o *shell* fará *logout* automático, conforme mostra a **Figura 4.**

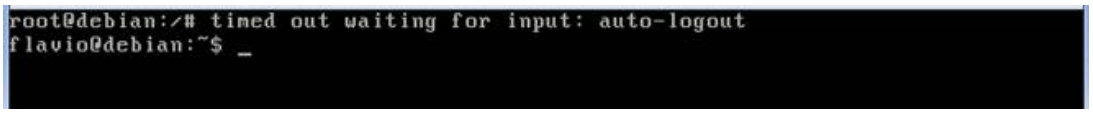

Figura 4. Terminal efetuando *logout* automático.

Outro ponto importante a ser levado em consideração quando se pensa em controle de acesso em uma organização, é avaliar quem tem acesso aos servidores. Usuários mal intencionados podem simplesmente usar o CTRL+ALT+DEL para reiniciar o servidor, parando assim todos os serviços disponíveis em uma rede. Isso pode ocorrer em 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. No entanto, independentemente desta norma, pode-se inibir a função CTRL+ALT+DEL editando o arquivo */etc/inittab*, conforme a **Listagem 8.**

Listagem 8. Editando o arquivo /etc/ssh/inittab.

```
#vim /etc/inittab

# Antes
# What to do when CTRL-ALT-DEL is pressed.
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

# Depois
# What to do when CTRL-ALT-DEL is pressed.
#ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

#ou

# Depois
# What to do when CTRL-ALT-DEL is pressed.
ca:12345:ctrlaltdel:/bin/echo “Opção desativada !”
```

SSH

O SSH (*Secure Shell*) é um programa usado para acessar remotamente outro computador usando uma rede, executar comandos em uma máquina remota e copiar arquivos de um computador para outro. Fornece autenticação forte e comunicação segura sobre canais inseguros (como a Internet, por exemplo). É muito usado para logar em um sistema GNU/Linux através de uma máquina Windows, Mac ou mesmo outro GNU/Linux, na qual os tradicionais *telnet* e *rlogin* não podem fornecer criptografia da senha e da sessão. Detalhes importantes precisam ser levados em consideração quando um servidor *ssh* é configurado. Para isso, algumas configurações importantes devem ser feitas no arquivo de configuração, conforme apresenta a **Listagem 9**.

Listagem 9. Editando o arquivo /etc/ssh/sshd_config.

```
#vim /etc/ssh/sshd_config

# Altere a porta padrão
Port 42129
# Protocolo 2 (anteriores possuem falhas de segurança)
Protocol 2
# Tempo ativado para digitar a senha
LoginGraceTime 45
# Não aceitar login como root
PermitRootLogin no
# Não aceitar login sem senha
PermitEmptyPasswords no
# Usar o modulo do pam para se autenticar
UsePAM yes
# Definir usuários que tem permissão de fazer login
AllowUsers “flavio marcos eduardo”
```

Após efetuar as alterações, o serviço *ssh* precisa ser reiniciado (ver **Listagem 10**).

Listagem 10. Reiniciando o serviço ssh

```
$invoke-rc.d ssh restart
```

Portas Abertas

Quando o sistema novo é instalado, alguns aplicativos (serviços) podem abrir portas introduzindo assim vulnerabilidades no sistema. Com o aplicativo *nmap*, pode-se fazer uma busca por todas as portas abertas no sistema e, em seguida, podem ser criadas regras no *firewall* para bloquear as que não devem estar disponíveis. Uma listagem de portas feita utilizando o aplicativo *nmap* pode ser visto na **Listagem 11.**

Listagem 11. Listagem de portas utilizando *nmap*.

```
#nmap -A -p 1-65535 localhost
PORT     STATE SERVICE VERSION

22/tcp   open  ssh     protocol 2.0
25/tcp   open  smtp    Postfix smtpd
53/tcp   open  domain  dnsmasq 2.47
80/tcp   open  http    httpd 2.2.11
631/tcp  open  ipp     CUPS 1.3.9
3306/tcp open  mysql   MySQL 5.0.75-0
```

nmap: *Network Mapper*, é um aplicativo livre e de código aberto, sobre licença GPL, utilizado para explorar uma r ede afim de efetuar uma auditoria de segurança.

Analisando o resultado do *nmap* pode-se observar que a porta 80 está aberta, normalmente usada por um servidor Web. Se esse serviço não for utilizado, a porta deverá ser fechada, evitando que vulnerabilidades sejam exploradas. Uma regra do *firewall iptables* poderá ser utilizada para bloquear tentativas de acesso a essa porta, como exibe a

Listagem 12.

Listagem 12. Bloqueando a porta 80 com *iptables*.

```
#iptables –A INPUT –i eth0 –p tcp –dport 80 –j DROP
```

Após adicionar a regra no *firewall*, faça novamente a pesquisa com o *nmap* e analise o resultado. Observe que a porta 80 está sendo filtrada, conforme a **Listagem 13**.

Listagem 13. Conferindo portas com nmap.

```
#nmap -A -p 80 localhost
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     protocol 2.0
25/tcp   open  smtp    Postfix smtpd
53/tcp   open  domain  dnsmasq 2.47
80/tcp   filtered   http
631/tcp  open  ipp     CUPS 1.3.9
3306/tcp open  mysql   MySQL 5.0.75-0
```

SUID BIT

O SUID BIT é uma das permissões especiais disponíveis no GNU/Linux. Quando está atribuída a um binário, é possível que um usuário execute o mesmo com os privilégios de seu dono. Se o dono do binário for o usuário *root*, o usuário vai executar o binário como *root*.

Podemos encontrar problemas de segurança ao ter binários com essa permissão especial configurada. Segundo a norma ISO 27002, no item 11.6.1, o acesso à informação e às funções dos sistemas de aplicações por usuário e pessoal do suporte devem ser restritas, de acordo com o definido na política de controle de acesso. Esse ponto deve estar muito bem determinado e esclarecido para todos os funcionários da organização. Pois caso algo seja violado, existe um documento para comprovar que aquilo não é certo.

Através de um *script* é possível localizar e alterar essa permissão, como apresenta a **Listagem 14**.

Listagem 14. Script sugerido para remoção de permissões SUID-BIT.

```
#vim /root/auditoria/localiza_suid.sh
#!/bin/bash

# Envia uma mensagem na saída padrão de vídeo
echo "Verificando arquivos com permissão de SUID BIT.."

# faz uma busca em todo sistema por arquivos que contenham a permissão de SUID BIT e salva em um arquivo texto
find / -perm -4000 > /root/auditoria/list.suid

# Envia mensagem na tela orientando o que deve ser feito
echo -n "Deseja remover o SUID BIT dos arquivos?(S/N):"

# Recebe a opção escolhida
read acao

# Executa a ação
case $acao in
  S|s)
    chmod -Rv -s /
    echo " Permissões de SUID BIT Removidas!"
    sleep 3
    exit
  ;;
  N|n)
    exit
  ;;
  *)
    echo "Opção Inválida!!"
    sleep 3
```

```
    exit
  ;;
esac
```

Depois de criado o script, é necessário alterar suas permissões, deixando que apenas o usuário *root* possa executá-lo, e retirando qualquer permissão dos demais usuários e grupos. Para essa função é usado o comando *chmod* (**Listagem 15**).

Listagem 15. Alterando as permissões do script.

```
#chmod 700 /root/auditoria/localiza_suid.sh
```

O primeiro passo é executar o script, e como parâmetro deverá ser utilizado “n” ou “N”. Nesse modo ele apenas irá gerar uma lista de binários que possuem a permissão de SUID-BIT no sistema, de acordo com a **Listagem 16.**

Listagem 16. Executando o script com o parâmetro n ou N.

```
#./localiza_suid.sh n
```

Observe no diretório */root/auditoria* que o arquivo *list.suid* foi criado (**Listagem 17**).

Listagem 17. Listando o diretório auditoria.

```
# ls /root/auditoria/list.suid
```

Como se trata de um sistema recém-instalado, o administrador não deve encontrar nenhum problema. Mas é necessária a atenção do mesmo, pois se estiver analisando um servidor de uma empresa, onde este se encontra em produção, devem-se remover as permissões *suid*.

Agora execute o *script* utilizando o parâmetro “s” ou “S”, removendo assim a permissão SUID BIT de todos os binários, como visto na **Listagem 18**. Deve-se observar quais binários serão necessários ficar com a permissão ativa. Considerando que o *firewall* está sendo desenvolvido, o comando *su* será necessário para que o usuário comum possa virar *root* e assim executar as políticas implementadas no *firewall*. Utilizando o comando *chmod*, atribui-se a permissão *suid* somente ao comando *su*, como visto na **Listagem 19**.

Listagem 18. Executando o script com o parâmetro (s ou S).

```
#./localiza_suid.sh s
```

Listagem 19. Adicionando permissão de SUID-BIT ao binário su.

```
# chmod +s /bin/su
```

Outro binário que precisamos deixar com a permissão SUID BIT ativa é o passwd, para que os usuários consigam trocar suas próprias senhas. Com o comando chmod atribui-se a permissão SUID BIT ao binário */usr/bin/passwd*, como visto na **Listagem 20**.

Listagem 20. Adicionando permissão de SUID-BIT ao *passwd*.

```
# chmod +s /usr/bin/passwd
```

Com essa modificação, um dos problemas será resolvido. Mas apenas retirar a permissão SUID BIT dos comandos não evitará problemas no sistema de arquivos. Segundo a norma ISO 27002 nos itens 10.4 e 10.4.1, deve-se proteger a integridade do software e da informação e ter um controle contra códigos maliciosos.

Uma *backdoor* é um código malicioso. Um *cracker* pode instalar uma *backdoor* no sistema utilizando uma técnica como, por exemplo, o *PHP Injection* (inserção de script malicioso através de páginas vulneráveis), a colocando dentro do */tmp* com permissões de SUID BIT e executá-la remotamente.

Para exemplificar um ataque ao diretório */tmp*, é utilizado o comando *adduser* para adicionar um usuário flavio, como visto na **Listagem 21**.

Listagem 21. Adicionando o usuário flavio.

```
#adduser flavio
```

Seguindo o exemplo, será utilizado o comando *cp* para copiar todos os *shells* do sistema para a partição */tmp* e, logo após, será atribuída a permissão de SUID BIT, como visto na **Listagem 22**.

Listagem 22. Teste com binários que tenham permissão SUID-BIT.

```
# cp /bin/*sh* /tmp
# chmod 4755 /tmp/*sh*
```

A seguir é apresentada uma sequência de comandos, onde o usuário *flavio* está logado no terminal *tty1*, o que pode ser verificado com o comando *w*. Em seguida é realizado o acesso ao diretório */tmp*, onde é possível executar o *script sh* de acordo com a **Listagem 23**.

Listagem 23. Usuário comum com permissão de root.

```
$ w
22:31:21 up  3:46,  1 users,  load average: 0,07, 0,07, 0,08
USER     TTY      FROM          LOGIN@   IDLE   JCPU   PCPU WHAT
flavio   tty1   -             22:20    0.00s  0.26s  0.01s w
$ cd /tmp
$ ./sh
```

Pode-se observar com o comando *id* que o usuário *flavio* ganhou acesso de *root "euid=0(root)"*. Sendo o usuário *root* o dono desse *shell* com a permissão de SUID BIT ativa, o usuário consegue executar o *shell* sh como *root*, conforme mostra a **Figura 5**.

Figura 5. Usuário comum recebendo direitos de *root*.

Para evitar esse tipo de problema, alguns parâmetros do comando *mount* podem ser utilizados. O parâmetro apresentado aqui é o *nosuid*, que inibe a execução de binários na partição indicada. Logado como usuário *root*, aplica-se essa opção à partição montada em */tmp*, como visto na **Listagem 24**.

Listagem 24. Remontando o diretório /tmp com a opção nosuid.

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

É importante lembrar que o diretório */tmp* necessariamente deverá estar em uma partição separada. Isso segue as boas práticas de particionamento.

Após ter aplicado a correção ao diretório */tmp*, faça novamente o *login* com o usuário flavio e tente executar a *shell sh* novamente. Observe que a diretiva *"euid=0(root)"* não mais é listada. Dessa forma o usuário já não tem mais permissões de *root*, conforme mostra a **Figura 6**.

Figura 6. Usuário comum sem direitos de root.

NOEXEC

Um *script* malicioso pode ser inserido em uma partição do sistema causando danos ao servidor. O comando *mount* tem um parâmetro chamado *noexec*, seu uso pode evitar que um *script* seja executado dentro dessa partição. Assim, para solucionar este problema, pode-se aplicar a opção *noexec* e remontar a partição.

Utilizando o comando *mount* com o parâmetro *remount* a partição */tmp* será remontada com a opção de leitura e escrita e acrescentando o *noexec*, de acordo com a **Listagem 25**.

Listagem 25. Remontando o diretório /tmp com a opção noexec.

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

Para conferir se a alteração foi efetivada, utilizamos o comando *mount* novamente (ver **Figura 7**).

n

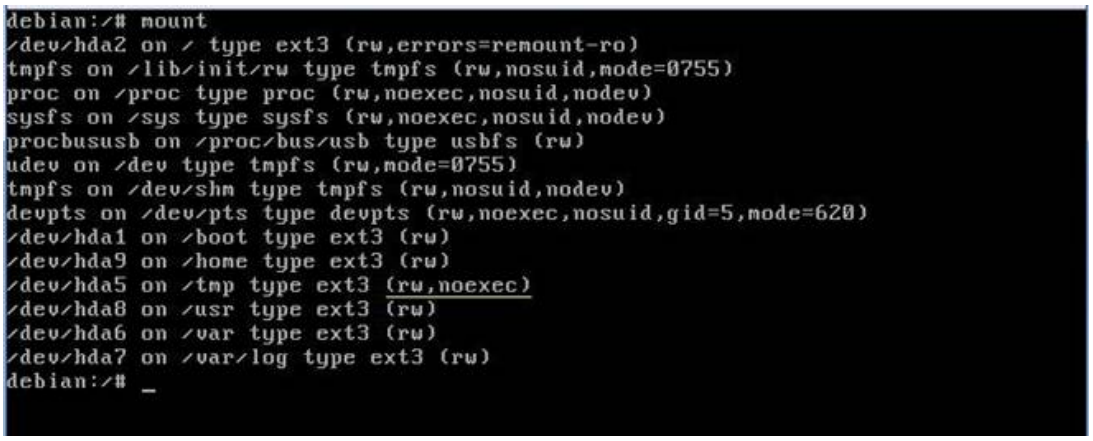

Figura 7. Comando *mount*, em destaque a partição */tmp* com opção *noexec*.

Após remontar a partição, foi realizada mais uma tentativa de executar o binário *sh*. No entanto, agora sua execução está impedida, independente do usuário logado. No exemplo foi executado como *root*, como mostra a **Figura 8**.

Figura 8. Tentando executar um binário em uma partição com a opção *noexec* aplicada.

Realizando a montagem das partições manualmente, os parâmetros não serão aplicados de forma fixa às partições, ou seja, ao reiniciar o sistema, essas configurações serão perdidas. Para que as configurações sejam mantidas mesmo após um restart, basta editar o arquivo */etc/fstab* (responsável por armazenar a tabela de partições do GNU/LINUX, que é lido durante o boot do sistema). Dessa forma, quando a opção *noexec* é inserida diretamente no arquivo, já será implementado o bloqueio ao iniciar a montagem da tabela de partições do sistema. Um exemplo desse arquivo pode ser visto na **Figura 9**.

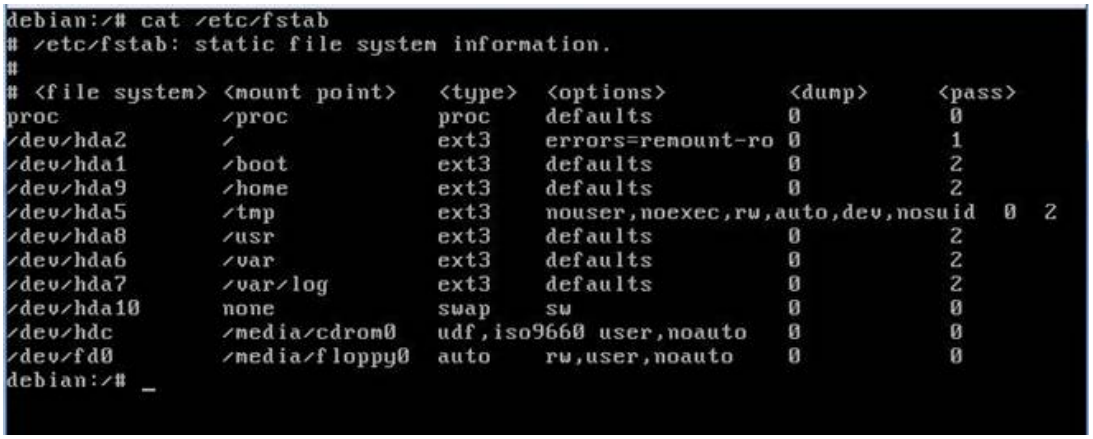

Figura 9. Arquivo */etc/fstab* alterado com as opções *noexec*.

O administrador poderá encontrar um problema ao executar o aplicativo *aptitude* se as partições */var* e */tmp* estiverem com o *noexec* e *nosuid* ativos, pois esse aplicativo precisa executar *scripts* dentro de tais partições. O *script* da **Listagem 26** pode ser utilizado para contornar tal problema. Dessa forma pode-se remover a proteção ao executar o aplicativo e, em seguida, reativá-las.

Listagem 26. Script para alterar opção noexec e nosuid em partições desejadas.

```
#!/bin/bash
case $1 in
  start)
    # monta as partições listadas
    mount -o remount,rw,noexec /var
    mount -o remount,rw,noexec /tmp
    mount
    echo "Partições SEM permissão de execução"
  ;;
  stop)
    # monsta as partições listadas
    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
```

Conclusão

Neste artigo foram tratados os conceitos de *hardening*, além de demonstrar algumas técnicas para endurecer o acesso ao sistema. Contudo, o assunto é muito extenso e importante, devendo ser tratado como fator fundamental em um projeto de implementação de um servidor, antes mesmo de entrar em produção, desde a instalação até a disponibilização de serviços na rede.

Sistemas como o *AppArmor* e o *SELinux* estão sendo utilizados para aumentar ainda mais a segurança em sistemas Linux, criando uma camada a mais de segurança, e devem ser considerados para também endurecer o sistema.

É importante salientar que existem técnicas de *hardening* mais específicas, como o *hardening* em *kernel*, em serviços como *Apache* e *MySql*, entre outros, que serão temas de novos artigos.

Links

Site do nmap
http://nmap.org/

Hardening
http://www.ufpa.br/dicas/vir/inv-back.htm

Linux Hardening
http://www.csirt.pop-mg.rnp.br/docs/hardening/linux.html> Acesso em: 13 de janeiro de 2010.

César Domingos
BS7799 Da Tática a Prática em Servidores Linux