

# Seguridad en Servidores Debian

René Mayorga  
rmayorga@debian.org.sv

Comunidad de usuarios Debian de El Salvador

24 de abril de 2010

# Esquema

- 1 Introducción
- 2 Planeación
- 3 Selección de software
- 4 Asegurando los usuarios
  - Software para controlar amenazas locales
- 5 SSH
- 6 Alimentando la paranoia
- 7 Comentarios, sugerencias, experiencias

# Seguridad

¿Cuál es la mejor técnica de seguridad disponible?

# Seguridad

¿Cuál es la mejor técnica de seguridad disponible?

¿Ideas?

# Seguridad

¿Cuál es la mejor técnica de seguridad disponible?

¿Ideas?

La mejor técnica de seguridad:

Consiste simplemente en desconectar el cable de energía, y ni aún así podemos estar 100 por ciento seguros, pues todavía existen datos en nuestro disco que podrían ser comprometidos por cualquier técnica.

# Seguridad

¿Cuál es la mejor técnica de seguridad disponible?

¿Ideas?

La mejor técnica de seguridad:

Consiste simplemente en desconectar el cable de energía, y ni aún así podemos estar 100 por ciento seguros, pues todavía existen datos en nuestro disco que podrían ser comprometidos por cualquier técnica.

Pero a pesar de todo, todavía resulta una de las mejores técnicas para mantener seguro mi equipo.

Trataremos de ver un par de ideas que pueden hacer que duerma más tranquilo.

# Pensando antes que todo

¿Cuál será la función principal del servidor?

# Pensando antes que todo

¿Cuál será la función principal del servidor?

Es necesario saber que **\*que\*** quiero hacer

Esto influye directamente en las aplicaciones y la manera como voy a instalar el servidor. Ejemplo: Para que necesito gnome o KDE en un servidor de correo ?

## Pensando antes que todo

¿Cuál será la función principal del servidor?

Es necesario saber que **\*que\*** quiero hacer

Esto influye directamente en las aplicaciones y la manera como voy a instalar el servidor. Ejemplo: Para que necesito gnome o KDE en un servidor de correo ?

Recordemos, en el caso específico de Debian, existe un sistema base, que contiene lo necesario, luego de eso, es tarea del administrador instalar lo que requiere para su trabajo.

## Pensando antes que todo

¿Cuál será la función principal del servidor?

Es necesario saber que **\*que\*** quiero hacer

Esto influye directamente en las aplicaciones y la manera como voy a instalar el servidor. Ejemplo: Para que necesito gnome o KDE en un servidor de correo ?

Recordemos, en el caso específico de Debian, existe un sistema base, que contiene lo necesario, luego de eso, es tarea del administrador instalar lo que requiere para su trabajo.

En otras palabras, **\*NO\*** usemos `taskel`

# planeando mis Particiones

¿Para qué hago diferentes particiones en mi OS ?

# planeando mis Particiones

¿Para qué hago diferentes particiones en mi OS ?

¿Ideas?

# planeando mis Particiones

¿Para qué hago diferentes particiones en mi OS ?

¿Ideas?

Con las particiones puedo manejar permisos, y hacer mi sistema mucho mas seguro es decir, asegurarme que después de mi instalación no se pueda instalar nuevo software. A menos que el administrador lo requiera.

## ¿Cómo está eso de la seguridad en las particiones?

a, ver, si ejecutamos `man mount` Podemos ver algunas cosas interesantes, como :

```
rw      Mount the file system read-write.
[.....]
ro      Mount the file system read-only.
[.....]
exec    Permit execution of binaries.
[.....]
noexec  Do not allow direct execution
of any binaries.
```

## ¿Cómo está eso de la seguridad en las particiones?

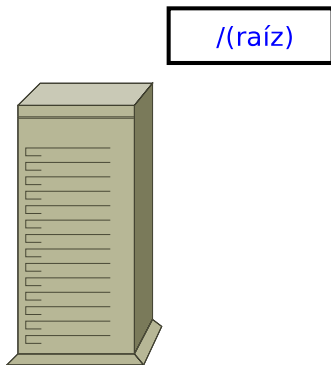
a, ver, si ejecutamos `man mount` Podemos ver algunas cosas interesantes, como :

```
rw      Mount the file system read-write.
[.....]
ro      Mount the file system read-only.
[.....]
exec    Permit execution of binaries.
[.....]
noexec  Do not allow direct execution
of any binaries.
```

Pero, pero!, yo ya conocía desde antes esas opciones!.

# Veamos lo de las particiones

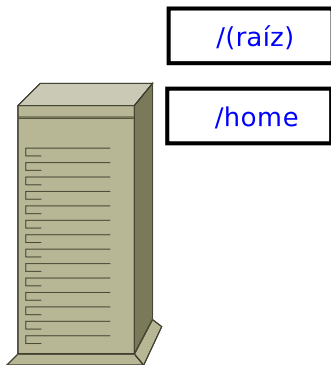
Pensemos en un esquema de particiones básicas:



Mi super-ultra-complicado-y-maravilloso-servidor

Figura: siempre /

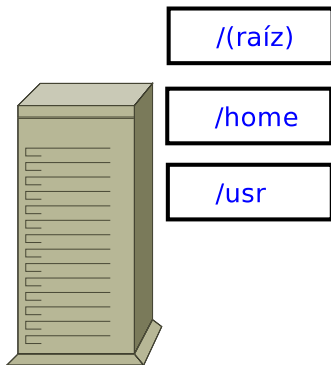
# Veamos lo de las particiones



Mi super-ultra-complicado-y-maravilloso-servidor

Figura: que particiones

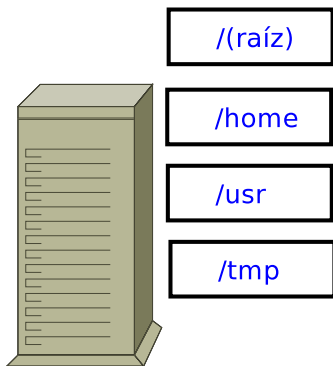
# Veamos lo de las particiones



Mi super-ultra-complicado-y-maravilloso-servidor

Figura: que particiones

# Veamos lo de las particiones



Mi super-ultra-complicado-y-maravilloso-servidor

Figura: que particiones

# y la seguridad

Pero que tiene que ver todo esto con la seguridad ?

# y la seguridad

Pero que tiene que ver todo esto con la seguridad ?

## Datos de los usuarios

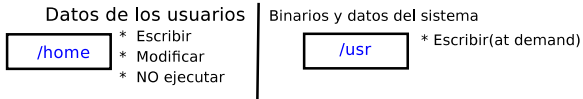


/home

- \* Escribir
- \* Modificar
- \* NO ejecutar

Figura: que hago con esa partición

# y la seguridad



# y la seguridad

## Datos de los usuarios

`/home`

- \* Escribir
- \* Modificar
- \* NO ejecutar

## Binarios y datos del sistema

`/usr`

- \* Escribir(at demand)

Es decir, después de instalado el sistema Operativo no existe necesidad de escribir nuevamente en `/usr`

# y la seguridad

## Datos de los usuarios

`/home`

- \* Escribir
- \* Modificar
- \* NO ejecutar

## Binarios y datos del sistema

`/usr`

- \* Escribir(at demand)

Es decir, después de instalado el sistema Operativo no existe necesidad de escribir nuevamente en /usr

## Datos temporales

`/tmp`

- \* /tmp es el home de los crackers
- \* no es necesario ejecutar \*NADA\*
- \* permisos de escritura

# y la seguridad

## Datos de los usuarios

`/home`

- \* Escribir
- \* Modificar
- \* NO ejecutar

## Binarios y datos del sistema

`/usr`

- \* Escribir(at demand)

Es decir, después de instalado el sistema Operativo no existe necesidad de escribir nuevamente en `/usr`

## Datos temporales

`/tmp`

- \* `/tmp` es el home de los crackers
- \* no es necesario ejecutar \*NADA\*
- \* permisos de escritura

`/var`

- \* Similar a `/tmp`
- \* los daemons necesitan escribir logs, pero \*NO\* es necesario ejecutar nada.

# Particionando para hacer mi servidor seguro

Entonces, puedo tener una configuración que evite:

# Particionando para hacer mi servidor seguro

Entonces, puedo tener una configuración que evite:

- A un usuario común y silvestre descargar un código (exploit) y luego de compilarlo, ejecutarlo en su home o /tmp

# Particionando para hacer mi servidor seguro

Entonces, puedo tener una configuración que evite:

- A un usuario común y silvestre descargar un código (exploit) y luego de compilarlo, ejecutarlo en su home o /tmp
- un programa instalado no podría descargar ni modificar bibliotecas del sistema, pues puedo mantener /usr/lib y /lib montados con **ro**

# Particionando para hacer mi servidor seguro

Entonces, puedo tener una configuración que evite:

- A un usuario común y silvestre descargar un código (exploit) y luego de compilarlo, ejecutarlo en su home o /tmp
- un programa instalado no podría descargar ni modificar bibliotecas del sistema, pues puedo mantener /usr/lib y /lib montados con **ro**
- no se podrían agregar programas nuevos o modificar programas en /bin, /sbin, /usr/bin, /usr/sbin, solo el administrador

## Particionando para hacer mi servidor seguro

Entonces, puedo tener una configuración que evite:

- A un usuario común y silvestre descargar un código (exploit) y luego de compilarlo, ejecutarlo en su home o /tmp
- un programa instalado no podría descargar ni modificar bibliotecas del sistema, pues puedo mantener /usr/lib y /lib montados con **ro**
- no se podrían agregar programas nuevos o modificar programas en /bin, /sbin, /usr/bin, /usr/sbin, solo el administrador

La idea es simple, solo **root** usando mount -o remount puede alterar los flags para ejecutar o tomar alguna acción, luego volver al estado anterior para que el usuario no pueda tomar ventaja de algún exploit para escalabilidad de permisos.

## Seleccionando mi software

Primero recordando **\*NO\*** usar taskel en la instalación

luego, una vez instalado el OS podemos decirle a APT que no instale **Recommends** por defecto, de esta manera nos aseguramos que el software instalado en el Servidor sea solo lo necesario y las dependencias correctas.

Para esto ejecuto:

```
apt-config dump | grep Recom | sed 's/1/0/' > \  
/etc/apt/apt.conf.d/05userconf
```

# Software a instalar

La idea es mantener **\*solo\*** el software necesario. Por defecto en Debian muchos servicios se encuentran desactivados y las configuraciones por defecto son **básicas**, por lo que es bueno solo instalar software necesario.

# los usuarios.....

El mayor problema de seguridad, la mayor amenaza potencial, y el mayor riesgo de botar todo mi esquema de seguridad, son sin duda alguno los propios usuarios.

## los usuarios.....

El mayor problema de seguridad, la mayor amenaza potencial, y el mayor riesgo de botar todo mi esquema de seguridad, son sin duda alguno los propios usuarios.

por tanto existen varias opciones a tomar en cuenta, primero, preguntarme:

## los usuarios.....

El mayor problema de seguridad, la mayor amenaza potencial, y el mayor riesgo de botar todo mi esquema de seguridad, son sin duda alguno los propios usuarios.

por tanto existen varias opciones a tomar en cuenta, primero, preguntarme:

- Necesita un shell en el sistema?

## los usuarios.....

El mayor problema de seguridad, la mayor amenaza potencial, y el mayor riesgo de botar todo mi esquema de seguridad, son sin duda alguno los propios usuarios.

por tanto existen varias opciones a tomar en cuenta, primero, preguntarme:

- Necesita un shell en el sistema?
- que servicios podrán estar disponibles a todos los usuarios

## los usuarios.....

El mayor problema de seguridad, la mayor amenaza potencial, y el mayor riesgo de botar todo mi esquema de seguridad, son sin duda alguno los propios usuarios.

por tanto existen varias opciones a tomar en cuenta, primero, preguntarme:

- Necesita un shell en el sistema?
- que servicios podrán estar disponibles a todos los usuarios
- voy a necesitar comportarme como un BOFH ?

## ahora después de decidir

pues el usuario podrá entrar, ahora a pensar como hago para que no me arruine la vida

# Pensemos que software vamos a usar

## cracklib

Cracklib2 es una librería en C, su función principal es prevenir al usuario usar passwords sencillos.

Existe un módulo para PAM, llamado **libpam-cracklib**

# cracklib

Una vez instalado **libpam-cracklib** simplemente es necesario editar mi configuración de PAM.

```
$EDITOR /etc/pam.d/common-password  
password    required    pam_cracklib.so minlen=6 lcredit=0 \  
ucredit=-1 dcredit=-1 ocredit=0  retry=3
```

# cracklib

La documentación la pueden encontrar con `man 8 pam cracklib` pero, resumiendo PAM-cracklib acepta algunas de estas opciones

# cracklib

La documentación la pueden encontrar con `man 8 pam cracklib` pero, resumiendo PAM-cracklib acepta algunas de estas opciones

# cracklib

La documentación la pueden encontrar con `man 8 pam cracklib` pero, resumiendo PAM-cracklib acepta algunas de estas opciones

- `retry`, errores del usuario

# cracklib

La documentación la pueden encontrar con `man 8 pam cracklib` pero, resumiendo PAM-cracklib acepta algunas de estas opciones

- `retry`, errores del usuario
- `minlen=N`, número de caracteres **minimos** para los passwords

# cracklib

La documentación la pueden encontrar con `man 8 pam cracklib` pero, resumiendo PAM-cracklib acepta algunas de estas opciones

- `retry`, errores del usuario
- `minlen=N`, número de caracteres **minimos** para los passwords
- `dcredit=N`, número de caracteres numéricos requeridos

# cracklib

La documentación la pueden encontrar con `man 8 pam cracklib` pero, resumiendo PAM-cracklib acepta algunas de estas opciones

- `retry`, errores del usuario
- `minlen=N`, número de caracteres **minimos** para los passwords
- `dcredit=N`, número de caracteres numéricos requeridos
- `ucredit=N`, letras Mayúsculas

# cracklib

La documentación la pueden encontrar con `man 8 pam cracklib` pero, resumiendo PAM-cracklib acepta algunas de estas opciones

- `retry`, errores del usuario
- `minlen=N`, número de caracteres **minimos** para los passwords
- `dcredit=N`, número de caracteres numéricos requeridos
- `ucredit=N`, letras Mayúsculas
- `lcredit=N`, letras minúsculas

# cracklib

La documentación la pueden encontrar con `man 8 pam cracklib` pero, resumiendo PAM-cracklib acepta algunas de estas opciones

- `retry`, errores del usuario
- `minlen=N`, número de caracteres **minimos** para los passwords
- `dcredit=N`, número de caracteres numéricos requeridos
- `ucredit=N`, letras Mayúsculas
- `lcredit=N`, letras minúsculas
- `ocredit=N`, Otros caracteres

## /etc/shadow

En `/etc/shadow` se guarda **mucha** información referente a los usuarios y sus passwords, por ejemplo, la última fecha de cambio de password el tiempo necesario para que el password expire, etc.

# /etc/shadow

En /etc/shadow se guarda **mucha** información referente a los usuarios y sus passwords, por ejemplo, la última fecha de cambio de password el tiempo necesario para que el password expire, etc.

Esto es un taller ¿verdad?

## /etc/shadow

En `/etc/shadow` se guarda **mucha** información referente a los usuarios y sus passwords, por ejemplo, la última fecha de cambio de password el tiempo necesario para que el password expire, etc.

Esto es un taller ¿verdad?

Entonces, todo a leer `man 5 shadow`

# Grupo Wheel

Wheel es un grupo común en algunos UNIXs como BSD pero en GNU/Linux no se encuentra por defecto este famoso grupo.

# Grupo Wheel

Wheel es un grupo común en algunos UNIXs como BSD pero en GNU/Linux no se encuentra por defecto este famoso grupo.

¿Alguien sabe que hace Wheel?

# Grupo Wheel

Wheel es un grupo común en algunos UNIXs como BSD pero en GNU/Linux no se encuentra por defecto este famoso grupo.

¿Alguien sabe que hace Wheel?

Básicamente en un UNIX, se permite a **cualquier** usuario que se encuentre dentro del grupo Wheel ejecutar **su**. Si el usuario no esta en Wheel aunque tenga conocimiento del password de otro usuario(incluido root) no podra hacer **su**

# Por que Wheel no existe en GNU/Linux y si en BSD

## Por que Wheel no existe en GNU/Linux y si en BSD

”

A veces, algunos listillos intentan hacerse con el poder total sobre el resto de usuarios. Por ejemplo, en 1984, un grupo de usuarios del laboratorio de Inteligencia Artificial del MIT decidieron tomar el poder cambiando el password de operador del sistema Twenex y manteniendolo secreto para el resto de usuarios. (De todas maneras, hubiera sido posible desbaratar la situacin y devolver el control a los usuario legitimos parcheando el kernel, pero no sabra como realizar esta operacin en un sistema Unix.) ”

– Richard M. Stallman, Manual de GNU su (man 1 su)

# No me interesan las experiencias juveniles de RMS, yo quiero Wheel!

Fácil, simplemente editamos `/etc/pam.d/su` y quitamos el comentario

# No me interesan las experiencias juveniles de RMS, yo quiero Wheel!

Fácil, simplemente editamos `/etc/pam.d/su` y quitamos el comentario

```
auth        required    pam_wheel.so
```

# No me interesan las experiencias juveniles de RMS, yo quiero Wheel!

Fácil, simplemente editamos `/etc/pam.d/su` y quitamos el comentario

```
auth        required    pam_wheel.so
```

Después de esto, nada mas es necesario asegurarnos de agregar a los usuarios al grupo `wheel` (si el grupo no existe, no llore, créelo!)

# ROOT no se debe de usar

Puedo restringir los dispositivos tty en los cuales root puede hacer login es decir, puedo dejar que root **\*solo\*** se pueda logear en /dev/tty5. Esto se puede hacer usando el archivo **/etc/securetty**

# ROOT no se debe de usar

Puedo restringir los dispositivos tty en los cuales root puede hacer login es decir, puedo dejar que root **\*solo\*** se pueda logear en `/dev/tty5`. Esto se puede hacer usando el archivo `/etc/securetty`

También debo evitar vivir validado en el sistema como root, en Debian, existe el grupo **adm** si yo soy el administrador del sistema puedo agregar mi usuario al grupo **adm** de esta forma podre revisar los logs del sistema, sin necesidad de estar como root.

# shell solo a quien lo necesita

ejecutemos `cat /etc/passwd — grep '/bin/sh'`

# shell solo a quien lo necesita

ejecutemos `cat /etc/passwd — grep '/bin/sh'`

¿Todos esos usuarios necesitan shell?

# shell solo a quien lo necesita

ejecutemos `cat /etc/passwd — grep '/bin/sh'`

¿Todos esos usuarios necesitan shell?

**\*NO\***, por tanto puedo cambiarlos a `/bin/false` y evitarme un punto extra de falla.

# necesito ssh, pero quiero dormir tranquilo

ssh puede ser nuestro mejor amigo, o nuestro peor enemigo.

# necesito ssh, pero quiero dormir tranquilo

ssh puede ser nuestro mejor amigo, o nuestro peor enemigo.

Por ejemplo, tener ssh abierto a **todo** mundo es un problema, pues soy un destino para script kiddies y ataques de fuerza bruta.

# TIPS para SSH

Un par de cosas útiles:

- Deshabilitar el login con root

# TIPS para SSH

Un par de cosas útiles:

- Deshabilitar el login con root
- No esconder el puerto, eso no funciona ni ayuda

# TIPS para SSH

Un par de cosas útiles:

- Deshabilitar el login con root
- No esconder el puerto, eso no funciona ni ayuda
- usar timeout

# TIPS para SSH

Un par de cosas útiles:

- Deshabilitar el login con root
- No esconder el puerto, eso no funciona ni ayuda
- usar timeout
- **\*SOLO\*** permitir ssh v2

# Defendiendome de script kiddies que usan fuerza bruta

Algún que otro software útil:

# Defendiendome de script kiddies que usan fuerza bruta

Algún que otro software útil:

- fail2ban

# Defendiendome de script kiddies que usan fuerza bruta

Algún que otro software útil:

- fail2ban
- sshguard

# Defendiendome de script kiddies que usan fuerza bruta

Algún que otro software útil:

- fail2ban
- sshguard
- auth2db

# Defendiendome de script kiddies que usan fuerza bruta

Algún que otro software útil:

- fail2ban
- sshguard
- auth2db
- scponly

# Auditando

Existe una gran cantidad de aplicaciones disponibles, que me pueden ayudar a monitorear mis logs, esto da sin duda una mejor perspectiva de que pasa en el sistema.

- multital
- acct
- fail2ban
- syslog
- samhain

## últimos detalles

Recordemos, siempre vamos a estar en la lucha entre seguridad versus usabilidad por lo que no vamos a lograr nunca una ni otra, lo importante es estar en un balance optimo.

Ahora, haciendo una pequeña reseña de que quedo y falto:

- El Sistema de archivos

## últimos detalles

Recordemos, siempre vamos a estar en la lucha entre seguridad versus usabilidad por lo que no vamos a lograr nunca una ni otra, lo importante es estar en un balance optimo.

Ahora, haciendo una pequeña reseña de que quedo y falto:

- El Sistema de archivos
- Firewalls

## últimos detalles

Recordemos, siempre vamos a estar en la lucha entre seguridad versus usabilidad por lo que no vamos a lograr nunca una ni otra, lo importante es estar en un balance optimo.

Ahora, haciendo una pequeña reseña de que quedo y falto:

- El Sistema de archivos
- Firewalls
- tcpwrappers

## últimos detalles

Recordemos, siempre vamos a estar en la lucha entre seguridad versus usabilidad por lo que no vamos a lograr nunca una ni otra, lo importante es estar en un balance optimo.

Ahora, haciendo una pequeña reseña de que quedo y falto:

- El Sistema de archivos
- Firewalls
- tcpwrappers
- archivos con permisos raros (SETUID, SETGID, etc)

## últimos detalles

Recordemos, siempre vamos a estar en la lucha entre seguridad versus usabilidad por lo que no vamos a lograr nunca una ni otra, lo importante es estar en un balance optimo.

Ahora, haciendo una pequeña reseña de que quedo y falto:

- El Sistema de archivos
- Firewalls
- tcpwrappers
- archivos con permisos raros (SETUID, SETGID, etc)
- etc, etc

# Gracias :)

Bueno, gracias a todos, esta presentación se encuentra bajo una licencia libre y pueden obtener una copia completa del código fuente en:  
<http://rmayorga.org/talks>.

De nuevo, mi correo [rmayorga@debian.org](mailto:rmayorga@debian.org), cualquier cosa en la que pueda ayudar, es un gusto