

AÑO ----- 0
NÚMERO ----- 8
FECHA: 2013-06-24

#8

"HORSE"

HD

Hackers & DEVELOPERS

Magazine digital de distribución
mensual sobre Software Libre, Hacking y Programación
para profesionales del sector de Tecnologías de la Información

Staff

Eugenia Bahit	GLAMP Hacker & eXtreme Programmer
Indira Burga	Ingeniera de Sistemas
María José Montes Díaz	Técnica en Informática de Gestión
Milagros Infante Montero	Est. Ingeniería de Sistemas
Sergio Infante Montero	Ingeniero de Software



Hackers & Developers Magazine se distribuye bajo una licencia **Creative Commons Atribución NoComercial CompartirIgual 3.0 Unported**. Eres libre de copiar, distribuir y compartir este material.
FREE AS IN FREEDOM!

Hackers & Developers Magazine, es una **iniciativa sin fines de lucro** destinada al fomento y difusión de las tecnologías libres presentes o futuras, bajo una clara óptica docente y altruista, que resulte de interés técnico y/o científico a profesionales del sector de Tecnologías de la Información. Hackers & Developers Magazine **se sostiene económicamente con el apoyo de la comunidad**, no recibiendo subvención alguna de ninguna empresa, organización u organismo de Gobierno. **Necesitamos de tu apoyo para poder mantener este proyecto.**

“Hacker es alguien que disfruta jugando con la inteligencia”

Richard Stallman
Free Software, Free Society
(Pág. 97), GNU Press 2010-2012

En esta edición:

JackTheStripper: Instala, configura y asegura tu Ubuntu Server 12.04..	4
Archlinux: Fortificando nuestro servidor.....	10
Examinar y manipular contenido de archivos por línea de comandos. .	16
Primeros pasos para Javascript Avanzado.....	20
Creando dibujos con Two.js.....	26
Administración de usuarios y permisos en MySQL®.....	31
El primer paso al éxito.....	37

Créditos

Hackers & Developers Magazine es posible gracias al compromiso de:

Responsable de Proyecto
Eugenia Bahit

Responsables de Comunicación

Indira Burga (Atención al Lector) - Milagros Infante (Difusión)

Staff

Eugenia Bahit
Arquitecta GLAMP & Agile Coach
www.eugeniabahit.com

Indira Burga
Ingeniera de Sistemas
about.me/indirabm

Milagros Infante Montero
Estudiante de Ingeniería en Sistemas
www.milale.net

María José Montes Díaz
Técnica en Informática de Gestión
archninfo.blogspot.com.es

Sergio Infante Montero
Ingeniero de Software
neosergio.net

Difusión

Hackers & Developers Magazine agradece a los portales que nos ayudan con la difusión del proyecto:



www.debianhackers.net



www.desarrolloweb.com



www.desdelinux.net

E-mail de Contacto:

contacto@hdmagazine.org

Web Oficial: www.hdmagazine.org
Cuenta Twitter Oficial: [@HackDevMagazine](https://twitter.com/HackDevMagazine)

JackTheStripper: Instala, configura y asegura tu Ubuntu Server 12.04

El pasado miércoles 19 de junio estuve en el programa [#linuxIO](#)¹ junto al genial [Pablo Bernardo](#)² (a.k.a. [@voylinux](#) en Twitter) y al querido y paciente [Borja Sánchez](#)³ (a.k.a. [@borjitaweb](#)) mostrando en vivo y en directo como montar, configurar y asegurar un servidor Web basado en Ubuntu Server 12.04 LTS. Y fue allí que utilicé mi nueva herramienta “JackTheStripper”: el “*deployer*” del que les quiero hablar en este artículo.

Escrito por: **Eugenia Bahit** (GLAMP Hacker & eXtreme Programmer)



Eugenia es **Arquitecta de Software**, docente e instructora de tecnologías **GLAMP** (GNU/Linux, Apache, MySQL, Python y PHP) y **Agile coach** (UTN) especializada en Scrum y **eXtreme Programming**. Miembro de la **Free Software Foundation**, **The Linux Foundation** y **Debian Hackers**. Creadora de [python-printr](#), [Europio Engine](#) y colaboradora de [Vim](#).

Webs:

Cursos de programación: www.cursosdeprogramacionadistancia.com

Web personal: www.eugeniabahit.com

Redes sociales:

Twitter / Identi.ca: [@eugeniabahit](https://twitter.com/eugeniabahit)

Grandes misterios se mueven alrededor del mundo de la seguridad informática y sobre todo, de la instalación, configuración y fortalecimiento de servidores Web. Misterios, mitos, mentiras convertidas en aparentes verdades y tantas otras frases y conceptos nada más alejados de la realidad, vienen inundando un terreno que hasta no hace mucho, se encontraba árido.

Y así, es como **un concepto puramente comercial**, fruto de las grandes mentes que se dedican al estudio profesional del marketing, terminó -sin pretenderlo- confundiendo la mente de programadores, diseñadores y desarrolladores de aplicaciones Web. **Web Hosting: el concepto comercial que desinforma.**

1 <http://www.desarrolloweb.com/en-directo/configuracion-servidor-web-apache-linuxio-8232.html>

2 <http://elkarmadelteclado.com/>

3 <https://twitter.com/borjitaweb>

El Web Hosting, no es más que un concepto comercial que engloba las características técnicas y administrativas que una empresa ofrece en forma conjunta a servicios de valor agregado.

Transferencia mensual, “soporte para” y “soporte de”, “panel de control” y compañía, no son más que características comerciales de los servicios que las empresas ofrecen a sus consumidores para que sus sitios Web sean accedidos en la red, pero nada tienen que ver con cuestiones técnicas.

Cuando comenzamos a desarrollar sitios Web, contratamos algún famoso “plan de Web hosting”. Luego, el marketing comienza a invadirnos y las características comerciales del servicio de nuestro proveedor, se transforman en una obsesión que nos lleva a querer emprender nosotros mismos el rol de proveedores y es así, como nos pasamos a “planes para revendedores” o peor aún, adquirimos “supuestos” VPS o dedicados que administramos a través de... ¡El navegador Web! Pero ¡Oh, por Dios! ¡Momento! ¿De verdad crees que administras un servidor a través del navegador Web y que eso es bueno?

Tanto marketing, tanta publicidad comercial, nos ha llenado de “miedo” y ni hablar, si venimos de sistemas operativos privativos donde parte de nuestra existencia se dedica a actualizar la base de datos del antivirus, reactivar números de series, renovar licencias vencidas que podrían generar el fin del mundo para nuestro ordenador (?) y vaya uno a saber cuántas otras amenazas más con las que posiblemente, si nos descuidamos, nuestro ordenador termine sirviendo de diminuto escaparate en el mismísimo e-inexistente infierno.

Es hora de perderle el miedo a los servidores, a GNU/Linux, la línea de comandos y darse la oportunidad de probar que es posible, vivir libre de preocupaciones tecnológicas cuando hablamos de servidores.

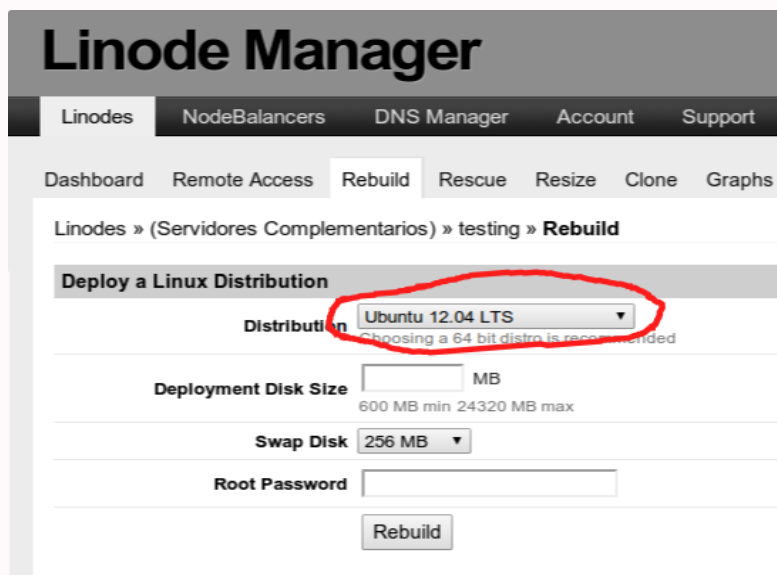
Tu primer servidor propio

Hoy en día es posible **contratar un servidor propio** o VPS, **por menos de USD 20** al mes. Es el caso de los servidores de [Linode](#)⁴, empresa que nos provee el **VPS que utilizamos en Hackers & Developers Magazine** por tan solo **USD 18** al mes por pago anual.

Al contratar un servidor en [Linode](#), así como sucede en la mayoría de las empresas que

4 <http://bit.ly/promo-linode>

proveen servidores *self-managed*, en el mismo momento del contrato el servidor se activa quedando disponible para su instalación. Ese es el momento, en el cuál solo basta con **seleccionar de una lista desplegable, la distribución que se desea instalar como base del servidor.**



Así se elige e instala una distro en [Linode](#)

Si te preguntas porqué elegir **Ubuntu 12.04 LTS** y no otra *distro*, la respuesta es simple:

1. Es una de las más estables y que con mayor soporte cuenta;
2. Es la más simple de utilizar, para usuarios poco familiarizados con la línea de comandos o que vienen de sistemas operativos privativos;

La parte más complicada es: **¿cómo instalar, configurar y asegurar el servidor para poder servir aplicaciones Web que necesitan de una plataforma GLAMP?** Y para ello, utilizaremos [JackTheStripper](#)⁵, la herramienta que paso a paso, “desnuda” los secretos de la instalación, configuración y aseguramiento de servidores.

¡Manos a la obra!

Con [Ubuntu Server 12.04 LTS](#) recién instalado en tu servidor [Linode](#), solo necesitarás seguir los pasos detallados a continuación.

1. Conéctate a tu servidor por SSH

Si no sabes como hacerlo, sigue estas instrucciones:

Si en tu ordenador **utilizas un sistema operativo privativo**, te recomiendo instalar en un ordenador de prueba o máquina virtual, la versión de escritorio de *Ubuntu 12.04 LTS*⁶ y así de paso, contar con un entorno

⁵ <http://www.eugeniahahit.com/proyectos/jackthestripper>

⁶ Para descargar Ubuntu 12.04 LTS versión de escritorio, por favor, visita <http://www.ubuntu.com/download/desktop/thank-you?release=lts&bits=32&distro=desktop&status=zeroc>. La descarga podría comenzar automáticamente. Si necesitas instrucciones

de desarrollo profesional, idéntico al de un servidor real. Para descargar Ubuntu 12.04 LTS versión de escritorio, por favor, visita:

<http://www.ubuntu.com/download/desktop/thank-you?release=lts&bits=32&distro=desktop&status=zeroc>.

La descarga podría comenzar automáticamente. Si necesitas instrucciones de instalación, no dudes en leer el paso a paso de Ubuntu ingresando en <http://www.ubuntu.com/download/desktop/install-desktop-long-term-support>.

Desde GNU/Linux, abre un terminal y ejecuta el siguiente comando: `ssh root@123.456.78.90`

Reemplaza 123.456.78.90 por la IP de tu nuevo servidor. Si no sabes cuál es la IP de tu servidor, ingresa al [panel de cliente de tu cuenta de Linode](#), pulsa sobre tu servidor y luego en la pestaña "Remote Access". Allí verás tu IP.

2. Descarga el tarball de JackTheStripper

```
wget http://www.eugeniabahit.com/code/jack-the-stripper/current.tar.gz
```

3. Descomprime el paquete

```
tar -xzf current.tar.gz
```

4. Ejecuta JackTheStripper

```
cd deploymyserver  
./dms.sh
```

JackTheStripper te irá indicando paso a paso qué es lo que está haciendo, sin que debas preocuparte siquiera en el futuro por el mantenimiento de tu servidor. Cuando JackTheStripper finalice su servidor, se habrán ejecutado las siguientes tareas:

1. Configuración de un hostname;
2. Reconfiguración de la zona horaria del sistema;
3. Actualización del sistema operativo por completo, asegurando contar con las últimas versiones disponibles de cada uno de los paquetes del sistema;
4. Creación de un nuevo usuario con privilegios de administrador, para manejar tu sistema de forma segura, sin depender de conexiones remotas mediante el usuario root;
5. Generación de llaves públicas RSA necesarias, para que el acceso SSH a tu servidor pueda efectuarse solo y únicamente desde tu ordenador;
6. Configuración, optimización y aseguramiento del servicio SSH, haciéndolo inmune frente intentos de acceso vandálicos o no autorizados;
7. Configuración del Firewall, generando reglas de seguridad estrictas en iptables que liberen de riesgos de accesos no autorizados a tu servidor;
8. Creación de un script de automatización para el Firewall, a fin de que las reglas implementadas no se pierdan tras reinicios

de instalación, no dudes en leer el paso a paso de Ubuntu ingresando en <http://www.ubuntu.com/download/desktop/install-desktop-long-term-support>

- del servidor;
9. Reforzamiento de la seguridad contra ataques por fuerza bruta, mediante la instalación, configuración y optimización de fail2ban;
 10. Instalación, configuración y optimización de MySQL para un rendimiento óptimo y un funcionamiento seguro;
 11. Instalación, configuración y optimización de PHP 5, para que la ejecución de tus aplicaciones sea tan segura como eficaz en el rendimiento. Entre los paquetes PHP que se instalan, se encuentran: php5, php5-cli, php-pear, php5-suhosin, php5-mysql y PHPUnit;
 12. Reforzamiento de la seguridad de Apache, mediante la instalación, configuración y optimización de módulos como ModSecurity y ModEvasive;
 13. Instalación de paquetes adicionales que pueden ser elementales tanto para la gestión del sistema como para el *deploy* de tus propias aplicaciones. Entre los paquetes adicionales instalados por JackTheStripper se encuentran: Bazaar (como SCV), tree, el módulo MySQLdb para Python, python-pip, el módulo WSGI de Apache para correr aplicaciones Web hechas con Python puro o en compañía de frameworks como Django;
 14. Configuración de Nano, Vim y la terminal, para que puedas trabajar cómodamente y con mayor facilidad a través de línea de comandos;
 15. Agregado de tareas de actualización y mantenimiento periódicas al Cron, para evitar que debas preocuparte por ellas;
 16. Incorporación del comando personalizado `blockip` para que puedas bloquear IPs de forma permanente, sin que las reglas de bloqueo se pierdan tras reinicios;

Los secretos que desnuda JackTheStripper

Ejecutar **JackTheStripper** y seguir creyendo que todo es un gran misterio sin resolver, no tiene sentido. Conocer los secretos que **JackTheStripper** desnuda, no es nada difícil, máxime teniendo en cuenta que **es Software Libre**.

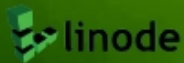
Para **conocer el paso a paso de JackTheStripper**, tienes varias opciones:

1. Si te interesa extender el proyecto y/o adaptar el **código fuente** a tus necesidades, puedes explorar el código descargando el mismo paquete que ejecutas: <http://www.eugeniahit.com/code/jack-the-stripper/current.tar.gz>. JackTheStripper se encuentra escrito en **Bash** e implementa un breve *script* en **Python**.
2. Si quieres ir directamente "a los hechos", puedes ver un paso a paso de los **comandos ejecutados** por **JackTheStripper**, ingresando en <http://pastebin.com/n4wLDDZi>
3. Si solo deseas ver cómo **JackTheStripper** modifica los **archivos de configuración** de las herramientas que instala, puedes visitar mi perfil en [pastebin.com](http://pastebin.com/u/eugeniahit) ingresando en <http://pastebin.com/u/eugeniahit> y encontrar allí, todos los archivos de configuración clasificados.

Si aún te quedan dudas sobre cómo instalar, configurar y asegurar tu propio servidor, te invito a **ver el deploy en video**, visitando el siguiente enlace: <https://www.youtube.com/watch?v=pPPm7lMhj4M>

Los servidores elegidos por
Hackers & Developers

obtén ahora
**TU PROPIO
SERVIDOR**



VPS

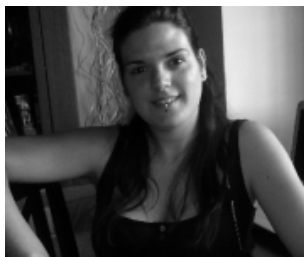
\$
USD **20**
/mes

¿El enlace de la imagen no funciona? Copia y pega esta URL: <http://bit.ly/promo-linode>

Archlinux: Fortificando nuestro servidor

En artículos anteriores vimos cómo utilizar la técnica de enjaulado para proteger nuestro servidor. Sin embargo, no podemos basar nuestra seguridad sólo en eso. En esta entrega veremos algunas herramientas para añadir seguridad a nuestro servidor.

Escrito por: **María José Montes Díaz** (Archera & Programadora)



Estudiante de Grado Ingeniería en Tecnología de la información. Técnico en informática de gestión. Monitora FPO. Docente de programación Python y Scratch para niños de 6-12 años. Activista del software libre y cultura libre.

Webs:

Blog: <http://archninfo.blogspot.com.es/>

Redes sociales:

Twitter: [@MMontesDiaz](https://twitter.com/MMontesDiaz)

Cuando ofrecemos un servicio mediante un servidor, podemos sufrir diferentes tipos de ataques. Enjaular los servicios es útil cuando el servicio ya ha sido comprometido. Veamos cómo protegernos de diferentes ataques que podemos sufrir, para intentar evitar que el servidor sea vulnerable.

Al disponer de un servidor, en la mayor parte de los casos, necesitaremos tener disponibles una serie de servicios adicionales para su administración. Por ejemplo, es muy posible que necesitemos un servicio FTP para poder subir archivos. También será necesario disponer de una conexión SSH para poder acceder remotamente. Estos servicios son susceptibles de recibir ataques de fuerza bruta.

Otros ataques que podemos sufrir son ataques DDOS. Para mitigar este tipo de ataques, también será necesario tomar medidas adicionales.

Limitando conexiones

Los ataques de denegación de servicio, consisten en realizar múltiples peticiones al

servicio para que éste se sature y no pueda atender a las peticiones legítimas. Para mitigar este tipo de ataques, en NGiNX podemos utilizar unas directivas que evitarán una saturación por exceso de peticiones o conexiones.

Lo primero que vamos a hacer es aumentar el número máximo de ficheros que se pueden tener abiertos. Creamos el archivo `/etc/sysctl.d/max_file.conf` con el siguiente contenido:

```
fs.file-max = 250000
```

En el archivo `/etc/security/limits.conf`, añadimos:

```
@http          soft  nofile          100000
@http          hard  nofile          250000
```

Ahora vamos a editar el archivo de configuración de NGiNX. Añadimos lo siguiente:

```
#El número de núcleos o procesadores del sistema:
#cat /proc/cpuinfo | grep processor | wc -l

worker_processes 2;

#El valor establecido en fs.file-max
worker_rlimit_nofile 250000;

...
events {
    # worker_rlimit / worker_processes
    worker_connections 125000;
}
...
http {
    ...
}
```

En la sección **http** definimos las diferentes zonas que necesitamos con sus correspondientes limitaciones. Para ello utilizaremos la directiva `limit_req_zone`, donde se indica el nombre de la zona, el tamaño máximo de memoria compartida y la ratio de peticiones por IP permitidas:

```
http {
    ...
    limit_req_zone $binary_remote_addr zone=req_one:1m rate=30r/s;
    ...
    server {
        ...
    }
}
```

Analizando la directiva anterior podemos observar que hemos definido una zona llamada **req_one**, con un tamaño de **1 M** y un máximo de **30** peticiones por segundo.

La memoria compartida almacena los estados de las diferentes claves. Cada IP será una clave. Con un megabyte podemos mantener aproximadamente 16.000 estados de 64 bytes. Si se agota el espacio para una zona, el servidor devuelve un error 503.

Para limitar el número de conexiones simultáneas por sesión, disponemos de la directiva **limit_req_conn**:

```
http {
    ...
    limit_conn_zone $binary_remote_addr zone=conn_one:1m;
    ...
    server {
        ...
    }
}
```

Para habilitar las zonas, utilizaremos las directivas **limit_req** y **limit_conn**. Éstas pueden utilizarse en los contextos *http*, *server* y *location*.

```
server {
    ...
    limit_req zone=req_one burst=20 ;
    limit_conn conn_one 5;
}
```

Con estas directivas limitamos el número de conexiones simultáneas por sesión a cinco como máximo (**limit_conn**) y con **limit_req**, limitamos a un máximo de 30 peticiones por segundo de media, en ráfagas de 20. En el caso de superarse la ratio de peticiones, se almacenan en la memoria compartida para ser tratadas después. Si estas peticiones almacenadas superan el número definido en **burst**, recibirán un error, 503 por defecto.

En el caso de que no queramos la ralentización de las peticiones que excedieron la ratio y preferimos que no sean atendidas, se puede especificar el parámetro **nodelay**:

```
limit_req zone=req_one burst=20 nodelay;
```

Tenemos la posibilidad de utilizar una herramienta, [ddos-deflate](http://deflate.medialayer.com/)⁷. Este script nos permite bloquear IP's que excedan un número de conexiones por un tiempo limitado. Puede ser usado bien con iptables o con APF (Advanced Policy Firewall). Además, nos brinda la posibilidad de ser informados por correo electrónico de las IP's bloqueadas.

Para instalarlo, sólo tendremos que ejecutar como root:

7 <http://deflate.medialayer.com/>

```
wget http://www.inetbase.com/scripts/ddos/install.sh
chmod 0700 install.sh
./install.sh
```

La configuración de esta utilidad se encuentra en */usr/local/ddos/ddos.conf*:

```
PROGDIR="/usr/local/ddos"
PROG="/usr/local/ddos/ddos.sh"
IGNORE_IP_LIST="/usr/local/ddos/ignore.ip.list"
CRON="/etc/cron.d/ddos.cron"
APF="/etc/apf/apf"
IPT="/sbin/iptables"

#Frecuencia en minutos para la ejecución de script
FREQ=1

#Número máximo de conexiones que vamos a permitir por IP
NO_OF_CONNECTIONS=100

#Si no estamos usando APF, poner 0
APF_BAN=0

#Si sólo queremos probar el funcionamiento, sin bloquear, poner 0
KILL=1

#A qué dirección queremos que envíe las notificaciones
EMAIL_TO="usuario@ejemplo.org"

#Tiempo máximo que estarán bloqueadas.
BAN_PERIOD=600
```

Disponemos de una lista blanca para aquellas IP's que no queremos bloquear aunque superen el límite. Esta lista la definimos en el archivo */usr/local/ddos/ignore.ip.list*, introduciendo una IP por línea.

Ataques de fuerza bruta, cómo protegernos

Para la protección de este tipo de ataques, disponemos de varias herramientas. Dos muy populares son **Fail2Ban** y **SSHGuard**. Estas herramientas bloquean IP's tras un número de intentos de acceso fallidos.

Fail2Ban es una herramienta altamente configurable y extensible. En el caso de SSHGuard, no es tan configurable y, tanto los servicios que protege, como los cortafuegos que es capaz de utilizar, están definidos en el código fuente. Sin embargo, esta segunda herramienta consume muy poco, con lo que es una alternativa interesante para servidores con escasos recursos.

Ambas herramientas no soportan todavía **journald**, así que es necesario la instalación de **syslog-ng** para que funcionen correctamente:

```
pacman -S syslog-ng
systemctl enable syslog-ng
systemctl start syslog-ng
```

Para instalar las herramientas:

```
pacman -S fail2ban
pacman -S sshguard
```

Para utilizar **Fail2Ban**, deberemos editar el archivo `/etc/fail2ban/jail.conf` y buscar el servicio preconfigurado para el cortafuegos que estemos utilizando. Por ejemplo, para habilitar la protección al servicio SSH, vía iptables, busquemos la sección `[ssh-iptables]` y la dejemos de forma similar a:

```
[ssh-iptables]
enabled = true
filter  = sshd
action  = iptables[name=SSH, port=ssh, protocol=tcp]
         sendmail-whois[name=SSH, dest=usuario@dominio.com,
         sender=fail2ban@dominio.com]
logpath = /var/log/auth.log
maxretry = 5
```

Cuando el servicio es iniciado, nos avisará con correo electrónico a la dirección indicada en `dest`. En cada bloqueo de IP, también recibiremos un correo.

En el caso de **SSHGuard**, tan sólo es necesario habilitar iptables para que la herramienta pueda añadir sus reglas. La herramienta comprueba automáticamente qué cortafuegos estamos utilizando y se encarga de revisar los logs para todos aquellos servicios que es capaz de controlar y estén siendo utilizados en nuestro equipo. Empecemos ejecutando como root:

```
iptables -N sshguard
iptables -A INPUT -j sshguard
iptables-save > /etc/iptables/iptables.rules

# Para ip6:
ip6tables -N sshguard
ip6tables -A INPUT -j sshguard
ip6tables-save > /etc/iptables/ip6tables.rules
```

Antes de iniciar el servicio, deberemos hacer una modificación en la unidad del mismo. El problema es que viene configurado para utilizar **journald**, pero aún no funciona correctamente. Creamos el archivo `/usr/local/lib/systemd/system/sshguard-syslog.service` con el contenido:

```
[Unit]
Description=Block hacking attempts
After=iptables.service ip6tables.service network.target
Wants=iptables.service ip6tables.service

[Service]
ExecStart=/usr/sbin/sshguard -l /var/log/auth.log -b /var/db/sshguard/blacklist.db

[Install]
WantedBy=multi-user.target
```

Ahora que tenemos iptables configurado y la unidad correspondiente, procedemos a levantar el servicio:

```
#Iniciar el servicio
systemctl start sshguard-syslog

#Habilitar en cada inicio del sistema
systemctl enable sshguard-syslog
```

En el caso de SSHGuard, disponemos de cuatro intentos para acceder al sistema. Si fallamos, se nos bloquea durante siete minutos. Si volvemos a equivocarnos, se duplica el último tiempo que estuvimos bloqueados.

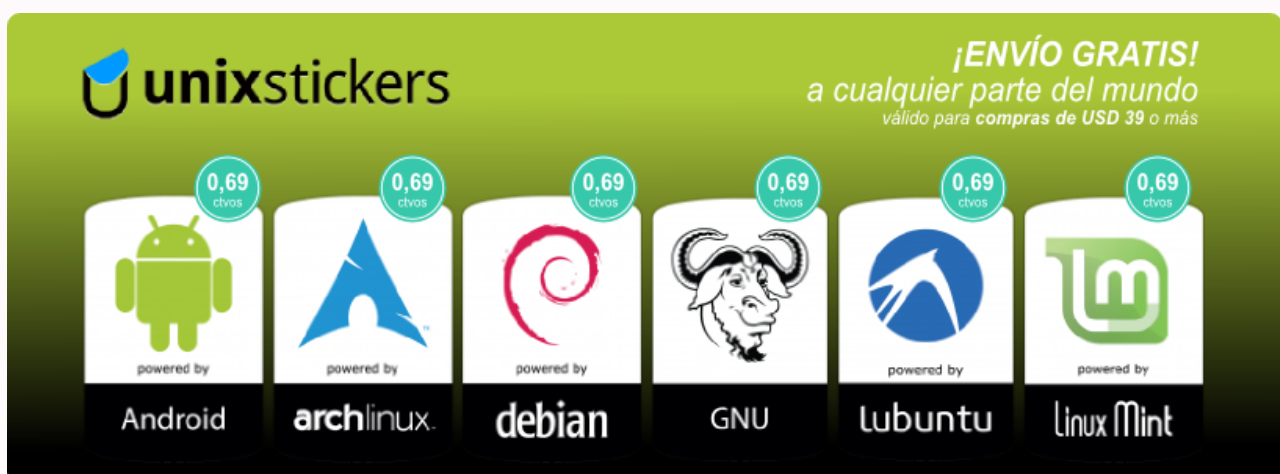
Enlaces de interés:

<https://wiki.archlinux.org/index.php/Sshguard>

<https://wiki.archlinux.org/index.php/Fail2ban>

<http://deflate.medialayer.com/>

<http://www.sshguard.net/>



The advertisement features a green background with the 'unixstickers' logo on the left. On the right, it says '¡ENVÍO GRATIS! a cualquier parte del mundo' and 'válido para compras de USD 39 o más'. Below this, there are six white stickers, each with a circular badge indicating a price of '0,69 ctvos'. The stickers are for: Android (with the Android robot logo), archlinux (with the Arch Linux logo), debian (with the Debian logo), GNU (with the GNU Tux logo), lubuntu (with the Lubuntu logo), and Linux Mint (with the Linux Mint logo). Each sticker also has the text 'powered by' above the distribution name.

¿El enlace de la imagen no funciona? Copia y pega esta URL: <http://bit.ly/promo-stickers>

GNU/Linux para programadores:

Examinar y manipular contenido de archivos por línea de comandos

En ediciones anteriores comentábamos lo importante que es que un programador tenga la capacidad de moverse libremente por línea de comandos y aprendimos a movernos a través del sistema de archivos. Ahora, nos enfocaremos en las diversas formas que tenemos para examinar internamente, documentos de texto de cualquier tipo y manipular su contenido.

Escrito por: **Eugenia Bahit** (GLAMP Hacker & eXtreme Programmer)



Eugenia es **Arquitecta de Software**, docente e instructora de tecnologías **GLAMP** (GNU/Linux, Apache, MySQL, Python y PHP) y **Agile coach** (UTN) especializada en Scrum y **eXtreme Programming**. Miembro de la **Free Software Foundation**, **The Linux Foundation** y **Debian Hackers**. Creadora de **python-printr**, **Europio Engine** y colaboradora de **Vim**.

Webs:

Cursos de programación: www.cursosdeprogramaciondistancia.com
Web personal: www.eugeniabahit.com

Redes sociales:

Twitter / Identi.ca: [@eugeniabahit](https://twitter.com/eugeniabahit)

Al igual que la vez que hablamos sobre el manejo y manipulación de archivos y directorios, iremos viendo desde las opciones más comunes a las más complejas, comenzando por aquellas que quizás te resulten más familiares. Es el caso de `cat`: el comando que utilizamos para “leer” un archivo.

A decir verdad, `cat`, concatena dos o más archivos mostrando el resultado en pantalla:

```
eugenia@cococha-gnucita:~/borrador/uno$ echo "soy el archivo a" > a
eugenia@cococha-gnucita:~/borrador/uno$ echo "soy el archivo hola" > hola
```



```
eugenia@cococha-gnucita:~/borrador/uno$ cat a hola
soy el archivo a
soy el archivo hola
eugenia@cococha-gnucita:~/borrador/uno$
```

Sin embargo, el mismo comando también se utiliza solo para **ver en pantalla el contenido de un archivo**:

```
eugenia@cococha-gnucita:~/borrador/uno$ cat .htaccess
RewriteEngine On
RewriteRule !(^static|favicon) app_engine.php
AddType image/x-icon .ico
RewriteRule ^favicon favicon.ico [NC,L]
eugenia@cococha-gnucita:~/borrador/uno$
```

El comando `cat` es especialmente útil para visualizar *scripts* o cualquier otro archivo de texto plano, exceptuando aquellos dinámicos y/o de gran tamaño como archivos de *logs*.

Para visualizar archivos de gran tamaño, disponemos de tres opciones: a) **visualizar las últimas líneas de un archivo** lo cual es especialmente útil para visualizar el último error o datos de acceso en, por ejemplo, los *logs* de Apache:

```
eugenia@mynewdream:~/ruta/a/logs$ tail -n 2 access.log
189.188.26.118 - - [09/Apr/2013:22:06:13 -0300] "GET /static/pdf/material-sin-
personalizar-python.pdf HTTP/1.1" 206 33104
"http://www.cursosdeprogramacionadistancia.com/static/pdf/material-sin-
personalizar-python.pdf" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.31 (KHTML,
like Gecko) Chrome/26.0.1410.43 Safari/537.31"
189.188.26.118 - - [09/Apr/2013:22:06:16 -0300] "GET /static/pdf/material-sin-
personalizar-python.pdf HTTP/1.1" 206 914949
"http://www.cursosdeprogramacionadistancia.com/static/pdf/material-sin-
personalizar-python.pdf" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.31 (KHTML,
like Gecko) Chrome/26.0.1410.43 Safari/537.31"
eugenia@mynewdream:~/ruta/a/logs$
```

En la muestra anterior, `-n 2` indica que se desean ver 2 líneas. Se puede indicar la cantidad deseada.

La opción b) es **visualizar las últimas líneas de un archivo en tiempo real**, lo que es especialmente útil para auditar logs y saber todo lo que sucede en tiempo real. Esto se logra pasando `-F` como argumento en vez de `-n`:

```
tail -F logs/error.log
```

Es importante aclarar que `-F` es sinónimo de `-f` (en minúsculas).

Y luego, la tercera opción u opción c), es **leer las primeras líneas de un archivo** lo cual es útil en líneas generales para visualizar, por ejemplo, archivos binarios:

```
eugenia@mynewdream:~$ head -n3 php.pdf
%PDF-1.5
%000
3 0 obj
eugenia@mynewdream:~$ head -n3 agile.pdf
%PDF-1.4
%äüöß
2 0 obj
```

o también, cualquier archivo de texto convencional:

```
eugenia@cococha-gnucita:~/borrador/ramtesting/fibo$ head -n2 foop
#!/usr/bin/php
<?php
eugenia@cococha-gnucita:~/borrador/ramtesting/fibo$ head -n2 foopl
#!/usr/bin/perl

eugenia@cococha-gnucita:~/borrador/ramtesting/fibo$
```

Cuando un archivo es de gran tamaño y sin embargo necesitamos verlo completo, podemos utilizar `less` o `more`: `cat ARCHIVO | less` (luego avanzamos con Av Pág o flecha abajo) `cat ARCHIVO | more` (avanzamos con la tecla intro)

Muchas veces, **buscar texto dentro de un archivo** nos será de mucha ayuda.

```
eugenia@cococha-gnucita:~/borrador$ grep -e "key" template.py
def render_regex(self, stack, key):
    dicc = dict(key=key)
    regex = re.compile('<!--%(key)s-->(.\|\\n){1,}<!--%(key)s-->' % dicc)
    render = render.replace('<!--%s-->' % key, '')
eugenia@cococha-gnucita:~/borrador$
```

Para lo cual podremos utilizar la siguiente sintaxis:

```
grep -e "EXPRESION" DONDE

Busca todos los archivos que comiencen por <?php cuya extensión sea .php
eugenia@cococha-gnucita:~/borrador$ grep -e "^<?php" *.php
ejercicios_sql_290113.php:<?php
ejercicios_SQL.php:<?php
lala.php:<?php
settings.php:<?php
eugenia@cococha-gnucita:~/borrador$
```

El comando `grep` también es sumamente útil cuando se desea auditar *logs* en tiempo real. Pero en este caso la sintaxis será diferente:

```
COMANDO | grep -e "EXPRESION"
```

Dónde `COMANDO`, podría ser cualquiera (no solo `tail`). Pero veamos un ejemplo con `tail` para auditoría en tiempo real:

```
eugenia@mynewdream:~/ruta/a/logs$ tail -F access.log | grep -e "GET\ \/ analisis"
190.194.58.46 - - [09/Apr/2013:22:38:37 -0300] "GET / analisis HTTP/1.1" 200 58535
"http://www.cursosdeprogramacionadistancia.com/ analisis" "Mozilla/5.0 (X11; Linux
i686) AppleWebKit/537.31 (KHTML, like Gecko) Chrome/26.0.1410.43 Safari/537.31"
190.194.58.46 - - [09/Apr/2013:22:38:58 -0300] "GET / analisis HTTP/1.1" 200 58535
"http://www.cursosdeprogramacionadistancia.com/curso-poo" "Mozilla/5.0 (X11; Linux
i686) AppleWebKit/537.31 (KHTML, like Gecko) Chrome/26.0.1410.43 Safari/537.31"
^C
eugenia@mynewdream:~/ruta/a/logs$
```

Dentro de la manipulación interna de un archivo, la forma más básica es **escribir texto sin utilizar un editor (>):**

```
eugenia@cocochoa-gnucita:~/borrador$ echo "lalala" > documento
eugenia@cocochoa-gnucita:~/borrador$ cat documento
lalala
eugenia@cocochoa-gnucita:~/borrador$ echo "Hola Mundo" > documento
eugenia@cocochoa-gnucita:~/borrador$ cat documento
Hola Mundo
eugenia@cocochoa-gnucita:~/borrador$
```

Como se puede ver, este comando sobre-escibe el contenido del archivo. Si no se desea sobre-escibir el contenido, es necesario **agregar texto al final del archivo (>>):**

```
eugenia@cocochoa-gnucita:~/borrador$ echo "Ahora no lo sobre-escribo" >> documento
eugenia@cocochoa-gnucita:~/borrador$ cat documento
Hola Mundo
Ahora no lo sobre-escribo
eugenia@cocochoa-gnucita:~/borrador$
```

Primeros pasos para Javascript Avanzado

Al parecer el futuro de JavaScript seguirá siendo muy prometedor; hay una cantidad enorme de herramientas construidas con este interesante lenguaje de Programación y en este artículo encontrarás el fundamento para entender estas variables.

Escrito por: **Sergio Infante Montero** (Ingeniero de Software)



Ingeniero Informático con estudios de **Master de Dirección Estratégica en TI**. Senior Software Developer en Belatrix Software Factory, activista, contribuidor y consultor de proyectos **FLOSS** y escritor de artículos y libros técnicos de programación.

Perfiles:

<http://about.me/neosergio>

Twitter: [@neosergio](https://twitter.com/neosergio)

Muchos sabemos que el uso de Javascript actualmente se ha incrementado de manera exponencial, con la aparición de tantos frameworks, librerías, complementos, módulos, incluyendo formas de usarlo para desarrollo de interfaces de escritorio, desarrollo de aplicaciones móviles y demás cosas asombrosas todas construidas con este poderoso lenguaje.

Por lo tanto enfocarse en términos avanzados de su uso se esta volviendo cada vez más necesario, para poder modificar a gusto ciertos comportamientos de las herramientas o incluso crearse algunas propias.

Objetos y Variables

La mayoría de programadores que ya han visto código en Javascript, saben que se usa la palabra reservada **var** para declarar variables, por ejemplo si deseo crear variables autor y título haciendo referencia a un artículo podría hacer lo siguiente:

```
var autor = 'sergio', título = 'inception deck en el agilismo';
```

Sin embargo muy pocos saben o recuerdan que esto se puede declarar también de esta manera:

```
var articulo = {autor: 'sergio', titulo: 'inception deck en el agilismo'};
```

Es una sola variable llamada artículo, que contiene un diccionario, entonces nos damos cuenta que ahora es un objeto que contiene autor y título.

Es importante tener presente esta forma de declaración ya que podemos usarlo de esta manera:

```
console.log(articulo.nombre) //imprime: sergio
console.log(articulo.titulo) //imprime: inception deck en el agilismo
```

Interesante ¿eh?, pasemos ahora a las funciones.

Funciones

Pues para declarar una función como recordarán es bastante simple:

```
var resumen = function() {
  //codigo
};
```

Si también con **var**, al igual que el objeto artículo, pero ¿que hay entonces con esta forma de escribir?, ¿es lo mismo?

```
function resumir() {
  //codigo
}
```

Es importante diferenciar cuando usar y no usar una función anónima

Pues aunque parezca que lo son, pues esto no es cierto, al usar la palabra reservada **var** se esta definiendo la función en tiempo de ejecución a diferencia de usar sólo **function** que lo hace accesible sin importar el orden, por ejemplo:

```
resumir(); // Esta dara error, comentala o ubicala al final
resumirNuevamente(); // Se mostrara el alert

var resumir = function() {
  alert('Esta es la funcion resumir');
```

```
}  
  
function resumirNuevamente() {  
    alert('Esta es la funcion resumir nuevamente');  
}
```

Si deseas probar su ejecución por tu cuenta ten cuidado con la primera línea por que tendrás este error:

```
Uncaught TypeError: Property 'resumir' of object [object Object] is not a function
```

Puedes comentarla o ponerla al final. Lo que sucede exactamente en el ejemplo anterior es que **resuminNuevamente** es una función que tiene ese nombre (**resumirNuevamente**) a diferencia de resumir que es un objeto que tiene asignada una función sin nombre o también conocida como *función anónima*.

Las funciones anónimas tienen un gran potencial para crear métodos privados entre otras cosas propias de la orientación a objetos

En caso de que se necesiten argumentos se podría tener lo siguiente:

```
var resumir = function (nombre, titulo){  
    return titulo + " fue escrito por: " + nombre;  
}
```

o bien podría ir de esta manera:

```
var resumir = function (articulo) {  
    return articulo.titulo + " fue escrito por: " + articulo.nombre;  
}
```

Función que podría llamarse como

```
resumir(articulo);
```

Sin embargo, cómo debemos pensar seriamente que una función es un objeto, esto cambiaría a:

```
articulo.resumen = resumir;  
articulo.resumen(articulo);
```

"No basta con que el código funcione." - Robert C. Martin |

Tendríamos el mismo resultado, pero hay un problema, el código está empezando a quedar feo, al mismo artículo si le volvemos a pasar artículo no luce bien, esto quedaría mejor:

```
var resumir = function(){  
    return this.titulo + " fue escrito por: " + this.nombre;  
}
```

Sin embargo ya no podríamos usar la función de esta manera:

```
alert(resumir());
```

Tendríamos este error:

```
undefined fue escrito por: undefined
```

Esto sucede porque la función está tomando el alcance al objeto **Window**, que no tiene esas propiedades (titulo y nombre). Por lo tanto para invocarla tiene que ser de la siguiente manera:

```
articulo.resumen = resumir;  
alert(articulo.resumen());
```

Ahora sí funciona porque el alcance está configurado al entorno artículo, y por lo tanto usará sus propiedades, ahí tendrías un método.

También podríamos definirla de esta manera:

```
articulo.resumen = function(){  
    return this.titulo + " fue escrito por: " + this.nombre;  
}
```

Ahora como se darán cuenta es algo mas simple de comprender, vamos entonces ahora a usar **prototipos** para hacer mas potentes a los objetos.

Objetos

Veamos cómo sería un constructor:

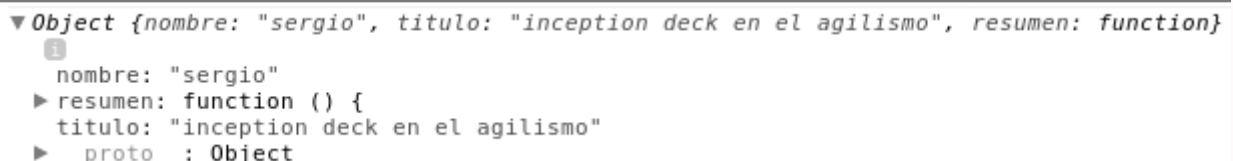
```
var Funcion = function(a, b){
  this.a = a;
  this.b = b;
}
var nuevaFuncion = new Funcion('hola', 'mundo');
```

La palabra clave **new** le dice al interprete de Javascript que se va a usar la **Funcion** como un constructor, exactamente lo que brinda es una copia del objeto **Funcion** después de ser ejecutado y es asignado a **nuevaFuncion**, por lo tanto como es copia se puede usar para hacer mas objetos de ese tipo.

Veamos entonces lo de los prototipos, podemos tener lo siguiente:

```
var articulo = {autor: 'sergio', titulo: 'inception deck en el agilismo'};
articulo.resumen = function() {
  return this.titulo + " fue escrito por: " + this.nombre;
}
console.log(articulo)
```

Veremos la siguiente imagen:



```
▼ Object {nombre: "sergio", titulo: "inception deck en el agilismo", resumen: function}
  1
  nombre: "sergio"
  ▶ resumen: function () {
    titulo: "inception deck en el agilismo"
    ▶ __proto__: Object
```

Podemos darnos cuenta que tiene una propiedad llamada **__proto__**, este es otro objeto, se podría decir que es el objeto original desde el cual fue creado, y es muy importante tenerlo en cuenta porque esto significa que todos los objetos tienen todas las propiedades y métodos de su prototipo (padre).

Este prototipo funciona de manera similar que otros lenguajes, cuando un método no se encuentra se revisa su superclase y luego la superclase de su superclase y así sucesivamente, hasta llegar al último objeto que será **null**.

Para poder llegar a **__proto__** podemos usar **prototype**, y de esa manera especificar el comportamiento:

```
Funcion.prototype
```


Es por ello que podemos tener esta porción de código:

```
var Saludo = function(nombre){
  this.nombre = nombre;
}
Saludo.prototype.despedida = function(){
  return "Adios " + this.nombre;
}
var saludo1 = new Saludo('sergio');
alert(saludo1.despedida());
```

Como se puede ver **saludo1.despedida()** devuelve una función la cual se puede ejecutar cuando se desee, este concepto se llama closure.

Estos son los conceptos básicos que hay que tener en cuenta para introducirse al JavaScript Avanzado.

Happy Hacking :)

The advertisement illustrates a process flow. On the left, it says "cobrar con PayPal™" with a blue arrow pointing to a central icon of a house with a dollar sign and a check, representing a "Cuenta Bancaria Virtual en USA". A blue arrow points from this icon to the right, where a Payoneer MasterCard is shown. Text in the center states "comisión: 0%" and "costo de apertura: registrándote mediante esta publicidad y haciendo un depósito inicial: USD 100.- \$0". On the right, it says "extraer y comprar con PaYoneer en todo el mundo!" and "comisión: 1%". A button at the bottom left says "Clic aquí para registrarte".

¿El enlace no funciona? Prueba ingresando la URL manualmente: <http://bit.ly/promo-payoneer>

Creando dibujos con Two.js

Uno de los elementos importantes en la web en estos días son los gráficos avanzados, existen muchas maneras de dibujarlos y Two.js es una genial opción para lograrlo.

Escrito por: **Milagros Alessandra Infante Montero** (Est. Ing. Informática)



Estudiante de Ingeniería Informática. Miembro de la comunidad de software libre [Lumenhack](#). Miembro del equipo de traducción al español de [GNOME](#). Apasionada por el desarrollo de software, tecnología y gadgets. Defensora de tecnologías basadas en software libre y de código abierto.

Webs:

Blog: www.milale.net

Redes sociales:

Twitter / Identi.ca: [@milale](#)

Cada vez que deseamos colocar gráficos avanzados en nuestra web nos preguntamos que herramienta nos ayudaría, por ejemplo podríamos usar Canvas, SVG, WebGL y más. Existe una librería de dibujo casi nueva que provee una API para lograr resultados similares, esta es two.js y veremos como poder utilizarla.

Dentro de las características que presenta Two.js⁸, encontramos que se centra en formas vectoriales ya que está inspirado en gráficos en movimiento plano y el objetivo es hacer la creación y animación de estos de forma más sencilla, pero no soporta texto ni imágenes. Se basa en un escenario gráfico, lo que implica que cuando se crea un objeto (Two.Vector, Two.Group y más), two lo almacena y lo recuerda; luego de creado el objeto se puede aplicar a este un número u operación (polygon.scale, group.opacity, vector.distanceTo(v), etc).

Two.js tiene un bucle de animación, puede ser autónomo o combinado con otra librería de animación; cuenta con un intérprete SVG, los elementos SVG pueden ser creados con cualquier aplicación y se puede traer luego al escenario de two.js.

8 <https://github.com/jonobr1/two.js#custom-build>

Empezamos

Para empezar a probar esta genial librería, debemos descargar e incluir two.js en nuestra página <html>

```
<script src="./path-to-two/two.js"></script>
```

Al escribir en la consola two, debe devolver una función y ya está listo para empezar a probarlo.

two.js

Debemos crear una instancia two y ponerla en nuestra página, ya que el constructor two toma un objeto con un número de parámetros.

```
var two = new Two ({  
  //fullscreen hará que el área de dibujo tome toda la ventana del navegador  
  //claro que si se desea colocar un número de pixel determinado también es posible  
  fullscreen: true  
});
```

Nosotros podemos decidir que renderizador usar, por ejemplo:

```
Two.Types.canvas  
Two.Types.svg  
Two.Types.webgl
```

Two.js usará de manera predeterminada SVG, **no** realizará automáticamente ningún tipo de detección de características para ver que soportará el navegador, esto deberá hacerse de manera manual.

Luego de tener la instancia Two, usaremos el método appendTo, lo que hace es tomar un elemento de HTML como un parámetro.

```
<div id="base"></div>  
<script src="./two.min.js"></script>  
<script src="./base.js"></script>
```

Y en el archivo base.js, ponemos lo siguiente para poder empezar a dibujar formas.

```
var elemento = document.getElementById("base"),  
two = new Two({
```

```
fullscreen: true
}};

two.appendTo(elemento);
```

¿Cómo dibujar?

Guiándonos de la documentación oficial que explica cada parámetro⁹ empezaremos a dibujar objetos para ver todo lo que podemos hacer.

Las formas más simples de hacer son las líneas, se dibuja entre dos coordenadas en todo el espacio y devuelve un objeto `Two.Polygon`, los dos primeros parámetros son x e y para el final de la línea y las otras coordenadas para el otro final, para dibujarla hacemos lo siguiente:

```
var linea = two.makeLine(14, 14, 116, 240);
linea.linewidth = 14;
linea.stroke = "rgba(255, 0, 0, 0.5)";
```

Dibujar un rectángulo también es posible, aquí se toman 4 parámetros, los dos primeros serán x e y para el centro del rectángulo, el tercer parámetro es el ancho y el cuarto la altura, como en el caso anterior devuelve un objeto `Two.Polygon`.

```
var rectangulo = two.makeRectangle(115, 90, 150, 100);
rectangulo.fill = "orange";
rectangulo.opacity = 0.25; //acepta valores decimales entre 0 y 1
rectangulo.noStroke(); //remueve el borde del rectángulo

two.update(); //two le dice todo al renderizador en la pantalla
```

Todas las funciones `two.make...` devuelven un objeto `Two.Polygon`

En el caso de los círculos, la función `makeCircle` toma tres parámetros, los dos primeros son las coordenadas x e y para el centro del círculo y el tercer parámetro es el radio.

```
var circulo = two.makeCircle(114, 114, 94);
circulo.fill = '#FF00FF';
two.update();
```

⁹ <http://jonobr1.github.io/two.js/#documentation>

Respecto a la propiedad fill, puede aceptar cualquier color válido de CSS.

Crear elipses también es algo sencillo, los dos primeros parámetros indican el centro de la elipse y luego se coloca el ancho y la altura.

```
Var elipse = two.makeEllipse(120, 48, 92, 36);
elipse.stroke = '#FF0099';
elipse.linewidth = 5; //indicar el ancho de línea
elipse.noFill(); //remueve el color de relleno y predeterminadamente será blanco
```

Y podemos de esta manera seguir creando muchas formas más con la ayuda de la documentación.

“El diseño es un lenguaje y lo principal es cómo usas ese lenguaje”. - Tibor Kalman

Grupos y animación

Las formas no solo pueden dibujarse individualmente, también se pueden agrupar e interactuar con ellas como una sola. El método makeGroup es lo que se usará para lograrlo con el método add.

```
var grupo = two.makeGroup(),
    circulo = two.makeCircle(114, 114, 94),
    rectangulo = two.makeRectangle(0, 0, 140, 140);

rectangulo.fill = "green";
circulo.fill = "red";

grupo.add(circulo);
grupo.add(rectangulo);

two.update();
```

Luego, también podemos hacer transformaciones de manera individual con las formas, usando la propiedad translation, esta nos permite mover la forma a la izquierda, derecha, arriba o abajo. Con el uso del constructor Two.Vector, se toma dos parámetros x e y, que serán las coordenadas para el centro de la forma.

```
grupo.translation.x = 140;
grupo.translation.y = 140;
```

```
two.update();
```

Los grupos así como las formas se ordenan de atrás para adelante, de acuerdo a como los creamos

```
var GrupoAriiba = two.makeGroup(),
    GrupoFondo = two.makeGropu(),
    circulo = two.makeCircle(114, 114, 94),
    rectangulo = two.makeRectangle(0, 0, 140, 140);

rectangulo.fill = "green";
circulo.fill = "red";

GrupoAriiba.add(circulo);
GrupoAriiba.add(rectangulo);

GrupoFondo.add(circulo); //con esto, el circulo estará sobre el cuadrado

two.update();
```

Two.js permite poner animación a sus formas¹⁰, reproduce las formas creadas al poner `two.update()`, hacer esto repetitivamente se logra con `two.play()`, ya que es como un evento de actualización, de esta manera podremos lograr producir animación, al ejecutarlo irá de marco en marco. El otro método para animación es `two.bind()`; el cual toma una cadena como su primer parámetro indicando a que evento escuchar y una función como su segundo argumento que indicará que hacer luego de terminar lo primero, indicando esta línea:

```
two.bind('update', referenceToFunction);
```

Estas son algunas de las cosas que podemos hacer con Two.js y lograr de manera creativa formas animadas muy geniales.

10 <http://jonobr1.github.io/two.js/examples/morph.html>

Administración de usuarios y permisos en MySQL®

La administración de usuarios y permisos en MySQL® no puede dejarse librada solo a DBAs ni mucho menos al simple azar. Por ello, en este artículo, aprenderemos a gestionar usuarios y permisos en MySQL®, de forma simple y comenzado desde lo más básico.

Escrito por: **Eugenia Bahit** (GLAMP Hacker & eXtreme Programmer)



Eugenia es **Arquitecta de Software**, docente e instructora de tecnologías **GLAMP** (GNU/Linux, Apache, MySQL, Python y PHP) y **Agile coach** (UTN) especializada en Scrum y **eXtreme Programming**. Miembro de la **Free Software Foundation**, **The Linux Foundation** y **Debian Hackers**. Creadora de **python-printr**, **Europio Engine** y colaboradora de **Vim**.

Webs:

Cursos de programación: www.cursosdeprogramacionadistancia.com

Web personal: www.eugeniabahit.com

Redes sociales:

Twitter / Identi.ca: [@eugeniabahit](https://twitter.com/eugeniabahit)

Cualquier informático que por diversos motivos suela trabajar con MySQL® conectándose como root, debe tener la capacidad de administración suficiente y necesaria para tomar medidas con respecto a la seguridad de las bases de datos, que hasta incluso podrían “sacar las papas del fuego” en situaciones de suma emergencia.

Al contrario de lo que generalmente se cree, gestionar usuarios y permisos en MySQL®, es una tarea sumamente sencilla que está al alcance de cualquier programador o administrador de sistemas. Lo más complejo, es tomar las medidas indicadas realizando el análisis propio de un DBA.

Pero en este artículo, nos centraremos en el cómo se gestiona y respecto a lo demás, nos limitaremos a decir que:

La medida más apropiada es aquella que define una política clara sobre las necesidades reales de los usuarios del sistema.

Dicho esto, concentrémonos en cómo llevar a cabo las políticas elegidas.

Creación de usuarios

Para agregar un nuevo usuario , es tan simple como ejecutar la siguiente sentencia:

```
CREATE USER nombre_de_usuario
IDENTIFIED BY 'clave en texto plano';
```

Por ejemplo, para agregar al usuario juanperez con la clave 123456, se utilizaría:

```
CREATE USER juanperez
IDENTIFIED BY '123456';
```

Una muy buena práctica es limitar la conexión de los usuarios por *host*, para **prevenir conexiones desde hosts no deseados**. Para ello, se utiliza:

```
'nombre_de_usuario'@'host'
```

Por ejemplo, para crear al usuario beatrix y solo permitir su conexión local, la sentencia sería la siguiente:

```
CREATE USER 'beatrix'@'localhost'
IDENTIFIED BY '123456';
```

Si luego se deseara **cambiar (o asignar en caso de inexistencia) una contraseña a un determinado usuario**, se utilizará SET PASSWORD como se muestra a continuación:

```
SET PASSWORD
FOR usuario = PASSWORD('nueva clave en texto plano');

# o también:
SET PASSWORD
FOR 'usuario'@'host' = PASSWORD('nueva clave en texto plano');
```

Por ejemplo, para agregar p1r0m4n14c0 como contraseña del usuario root, la sentencia sería:


```
SET PASSWORD
FOR 'root'@'localhost' = PASSWORD('p1r0m4n14c0');
```

Otorgar permisos

Para otorgar permisos en MySQL®, se deben considerar:

Permiso: El tipo de consultas que se permitirá efectuar al usuario (SELECT, INSERT, DELETE, UPDATE);

Database: Las bases de datos y/o tablas sobre las cuáles aplicarán dichos permisos;

Usuario: El o los usuarios a los cuáles serán otorgados los permisos aplicados.

La configuración de permisos se realizará con la siguiente sentencia:

```
GRANT permiso
ON database
TO usuario;
```

Por ejemplo, para otorgar permisos de selección sobre la tabla `categoria` de la base de datos `weblibros` al usuario `juanperez`, se ejecutará:

```
GRANT SELECT
ON weblibros.categoria
TO juanperez;
```

Suponiendo que la base de datos `weblibros` contenga dos tablas:

```
mysql> show tables from weblibros;
+-----+
| Tables_in_weblibros |
+-----+
| categoria           |
| libro               |
+-----+
2 rows in set (0.00 sec)
```

Si en vez de `root` la consulta fuese realizada por el usuario `juanperez`, obtendría lo siguiente:

```
mysql> show databases;
+-----+
```

```

| Database          |
+-----+
| information_schema |
| weblibros         |
+-----+
2 rows in set (0.00 sec)

```

```

mysql> show tables from weblibros;
+-----+
| Tables_in_weblibros |
+-----+
| categoria           |
+-----+
1 row in set (0.00 sec)

```

Y si el usuario juanperez intentara acceder a una tabla no permitida, el acceso le sería negado:

```

mysql> select * from weblibros.libro;
ERROR 1142 (42000): SELECT command denied to user 'juanperez'@'localhost' for table 'libro'

```

Al momento de escribir las sentencias para otorgar permisos, disponemos de diferentes opciones.

Opciones para indicar el tipo de permisos:

```

GRANT SELECT          # un permiso específico
GRANT SELECT, INSERT, UPDATE # varios permisos
GRANT ALL             # todos los permisos

```

Opciones para indicar a qué bases de datos/tablas aplicarán los permisos:

```

ON database.table      # a una tabla
ON database.table1,
database.table2,
database2.table
ON database.*         # a todas las tablas de la misma DB

```

Opciones para indicar a quién/quienes aplican los permisos:

```

TO usuario           # a un usuario
TO usuario1, usuario2 # a varios usuarios

```

Un ejemplo completo

Suponiendo que en sistema existen las siguientes bases de datos y tablas:

DATABASE	TABLAS
europioengine	categoria, materiaprima, producto
webcursos	categoria, libro

Crearemos primero 3 usuarios, a los que solo se les permita conectarse de forma local:

```
CREATE USER 'vanina'@'localhost'  
IDENTIFIED BY 'v4n1n41974';  
  
CREATE USER 'roxana'@'localhost'  
IDENTIFIED BY 'r0x4n4-1275';  
  
CREATE USER 'silvana'@'localhost'  
IDENTIFIED BY '33885409sil';
```

Y ahora, otorgaremos permisos de selección y actualización, a todos los usuarios recién creados, para todas las tablas de la base de datos europioengine:

```
GRANT SELECT, UPDATE  
ON europioengine.*  
TO vanina, roxana, silvana;
```

Conceder permiso de conceder permisos

Cuando otorgamos permisos a un determinado usuario, es posible además, conceder un permiso extra que le permita asignar sus mismos privilegios a otros usuarios. Una situación de ejemplo podría ser la siguiente:

Al usuario Ana le asignamos permisos de selección sobre la base de datos ventas. Entonces, el usuario Ana, puede asignar permisos de selección sobre la base de datos ventas a cualquier otro usuario.

De eso se trata la cláusula WITH GRANT OPTION que se especifica de la siguiente manera:

```
GRANT permisos  
ON tablas  
TO usuario  
WITH GRANT OPTION
```

Si seguimos con el ejemplo anterior, ahora, al usuario vanina, le otorgaremos todos los

permisos para la tabla categoria de la base de datos weplibro, permitiéndole conceder dichos permisos a cualquier otro usuario existente:

```
GRANT    ALL
ON       weplibros
TO       vanina
WITH     GRANT OPTION;
```

Revocando permisos

Revocar permisos es algo similar a concederlos. Solo cambia un poco la sintaxis:

```
GRANT => REVOKE
TO    => FROM
```

La sintaxis básica, se vería como la siguiente:

```
REVOKE  privilegio
ON      tabla
FROM    usuario
```

Por ejemplo:

```
REVOKE  SELECT, INSERT
ON      midatabase.tabla_productos
FROM    anitamayer
```

Puedo además, revocar privilegios en cascada, para los casos en los cuáles, el usuario en cuestión, haya otorgado privilegios a terceras personas:

```
REVOKE  SELECT, INSERT
ON      midatabase.tabla_productos
FROM    anitamayer
CASCADE
```

O puedo optar por la alternativa contraria de recibir un error en caso en el cuál, pretender revocar privilegios a un usuario, afecte a otros usuarios:

```
REVOKE  SELECT, INSERT
ON      midatabase.tabla_productos
FROM    anitamayer
RESTRICT
```

El primer paso al éxito

Si pensamos en usabilidad podemos estar cuestionando las formas clásicas de estructurar nuestras interfaces, ¿es realmente bueno lo que hago?

Escrito por: **Fabio Durán Verdugo** (Ingeniero de Software)



Miembro [GNOME Foundation](#) y [GNOME Chile](#). Colabora en el [BugSquad](#) team de GNOME. Amante y propulsor del Software Libre y código abierto. Actualmente también esta dedicado a la docencia universitaria. Python lover.

Perfiles:

<http://live.gnome.org/fabioduran>

Twitter: [@fabioduran](#)

LinkedIn: <http://www.linkedin.com/pub/fabio-duran-verdugo/26/469/a73>

A la gran mayoría de los que nos gusta desarrollar software nos sucede el mismo problema; al momento de estar programando no nos interesa nada más que desarrollar y resolver nuestro problema a veces sin importar quien es el usuario final.

A raíz de esto muchas veces nos toca recibir la cruda noticia de un gruñón que grita: “¡Esta cosa no sirve para nada!”, “¡No hace lo que yo necesito!”, “¡Eres un pésimo desarrollador!”, “¡Haz hecho la cosa más complicada, para un detalle muy simple!”, pero la verdad, ¿es tan así?, ¿o el problema fue, que el novel usuario no supo como utilizar nuestra creación?

Positiva o lamentablemente, según sea el punto de vista, siempre que estemos desarrollando un software independientemente de su objetivo, y lo publicamos por algún medio, estaremos siendo evaluados por los usuarios, que a veces sin saberlo catalogan dentro de sus parámetros y siguiendo una tendencia a la cotidianidad, reproducirán y concluirán si el producto es aceptable o no para su uso.

La usabilidad; una palabra tan fácil de pronunciar y buscar, pero tan complicada de implementar, tanto así que es un concepto sin un significado absoluto que siempre se define y se reduce a una forma ambigua de determinar si algo es de fácil uso, de fácil de aprendizaje y que entrega satisfacción.

Realmente no se si podemos definir estos conceptos de forma más precisa, teniendo en cuenta la diversidad de culturas a la cual estamos expuestos diariamente.

Problema

Actualmente, el desarrollo de software en general y en especial el desarrollo de software libre no incluye etapas para la creación de interfaces, sólo contemplan recomendaciones de un equipo que se dedica a realizar pruebas, pero las que pueden o no interpretadas de diferentes formas y en muchos casos no llegando a ningún consenso entre los mismos miembros del equipo de pruebas o de usabilidad.

Peor aún, autores como Jakob Nielsen, que poseen un gran reconocimiento en el estudio de la usabilidad, siempre la mencionan como un patrón crítico al momento de evaluar las factibilidades de un proyecto, más aún, proponen que la usabilidad sea considerada una ciencia que se proyecte y se constituya en una buena práctica para mejorar la calidad, seguir una moda y buscar un estándar para quienes dirigimos nuestras obras.

La usabilidad entonces debería ser considerada en todo momento, desde la génesis del proceso o ciclo de vida de desarrollo de un producto, hasta las últimas fases antes de entrar en producción.

¡Cuidado!

Un punto importante a considerar es que la usabilidad no sólo tiene que ver con la interfaz gráfica del usuario, también debemos saber que la usabilidad en el software está ligada a la interacción del mismo y que ésta interacción, no está definida en la interfaz gráfica, sino que se manifiesta en el código que implementa las funcionalidades del sistema. La interfaz gráfica de usuario es la parte visible de tal interacción, pero un sistema con un diseño de interacción pobre, no puede mejorar su nivel de usabilidad tan solo cambiando la interfaz gráfica. Por lo tanto, la interacción debe diseñarse en conjunto a la lógica del problema, para asegurarnos de que la lógica del sistema sea realmente usable.

Algunas recomendaciones para mejorar nuestro software

Es recomendable siempre tener una idea acerca de las características de los usuarios y de los aspectos de mayor relevancia y sus necesidades dentro del software.

Si construyes un software para una plataforma ya estructurada bajo un esquema ya probado, como lo es GNOME, lo ideal es que siga las métricas de construcción definidas por la plataforma, o sea, considerar colores, iconos, posición y disposición de los widgets, márgenes, etc., con esto se lograremos que tu software no sea un ente extraño dentro de un planeta viviente.

Es bueno estudiar de las guías de diseño de interfaces. Ejemplo, la guía de desarrollo de interfaz humana de GNOME (<https://developer.gnome.org/hig-book/>), que entrega recomendaciones de como podríamos construir nuestras interfaces de modo que sean más atractivas, fácil de aprender y utilizar.

Podemos considerar a pesar de estar discontinuado, visitar además el proyecto de la empresa Novell y parte de "the OpenSuse Project", llamado "betterdesktop"

(<http://www.betterdesktop.org>) , en donde encontraremos variados datos de forma cuantitativa y cualitativa acerca del comportamiento del usuario ante diversas actividades básicas. La idea era que los desarrolladores y comunidades de software libre pudieran crear información a partir de estos datos y mejorar sus interfaces. Algo relevante de este proyecto es que esta documentado en video, lo que te permite ver el experimento y recoger impresiones propias.

Otras recomendaciones que se pueden encontrar en variados sitios dedicados al tema son:

- Usar widgets u objetos estándares de la plataforma en que desarrollas. O sea si desarrollas en GNOME, usa el stocks de widgets de GTK+, si lo haces en KDE, usa el stocks de QT.
- Evitar la inconsistencia entre interfaces, por ejemplo si defines un diseño y posición de los botones en una ventana. No la cambies en la siguiente, esto puede confundir a los usuarios.

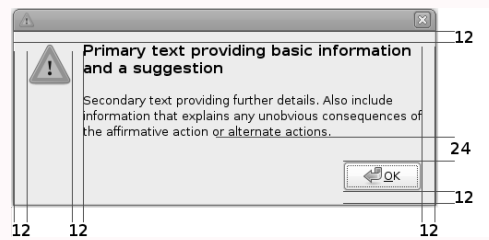
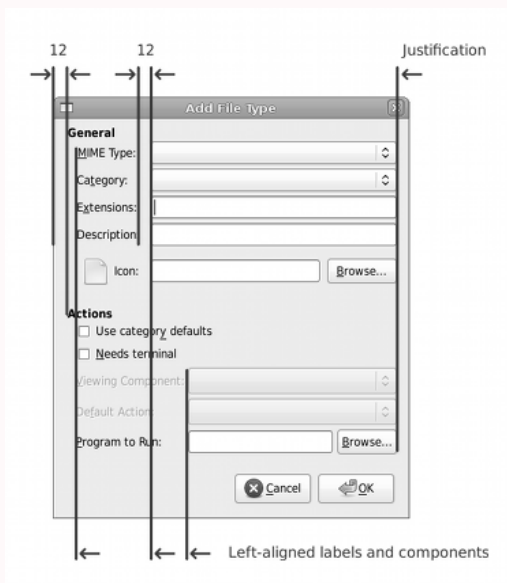
Un ejemplo básico que puedes encontrar en productos del mercado.



Fíjate en los botones, la imagen izquierda muestra un orden aceptable definido en un cuadro de dialogo con un widget de calendario, en la imagen de la derecha estos fueron cambiados, a esto se le llama inconsistencia.

Generalmente la inconsistencia entre las interfaces pasa por una falta de comunicación entre el equipo de desarrollo o diseño.

- Siempre indica como el usuario debe ingresar los datos, usa nombre de etiquetas simples.
- Entrega los mensajes de error o advertencia de manera clara y precisa, no solo usando un “¡fallo!”, además es recomendable usar iconos de stocks de error o advertencia para que el usuario visualice de forma directa el sentido del mensaje.
- Define alineaciones y márgenes para distanciar los widgets unos de otros, observa las siguientes imágenes extraídas de la HIG de GNOME.



- Entrega siempre información al usuario de lo que está realizando, evita que diga “¡el programa se colgó!”, cuando realmente esta guardando o sincronizando algo.
- No limitar la base de los usuarios, intenta integrar accesibilidad o que sea compatible tu software con las plataformas que ya la tienen.

Teniendo en cuenta estas pocas y básicas recomendaciones nos podríamos ahorrar tiempo, más de algún mal rato y porque no dinero, dado que siempre para el usuario posterior a la implementación del software requiere de un esfuerzo adicional en aprender nuevos aspectos o nuevas interfaces.

Considera siempre que...

Todos cometemos errores, puede pasar tanto si se está explorando o aprendiendo a usar un sistema o sea un experto que simplemente se equivocó. El software debe permitir a los usuarios deshacer los resultados de sus acciones.

Piensa siempre en simplicidad, diseña para que sólo se muestre información y elementos relevantes sin sobrecargar con iconos u objetos innecesarios para permitir al usuario concentrarse en la acción que está ejecutando.

Debemos aprender que al presentar la información de manera estéticamente agradable considerando la alineación de los elementos, colores y gráficos, mejorará la disposición y aceptación del usuario a nuestro desarrollo.

Una pequeña anécdota...

Hace un tiempo en el lugar donde trabajo requeríamos de un software tipo *payroll*, para poder llevar de mejor manera los contratos y remuneraciones de los trabajadores. Contactamos a una empresa y coordinamos una demo. La primera pantalla expuesta fue

el formulario de fichas para trabajadores y mientras el vendedor intentaba explicar como funcionaba, yo no paraba de escuchar “¡el software el buenísimo!”, “¡tiene todo lo que necesitamos!”, “¡es excelente!”. Ante mis dudas pedí que hiciéramos el ejercicio de ingresar un par de trabajadores para ver que tan fácil y bueno era el software. Todo resulto ser un fiasco, los usuarios estaban frustrados y las expresiones de aprobación desaparecieron. Definitivamente existían, muchos campos innecesarios, validaciones absurdas, y una interfaz que solo al verla cansaba.

La moraleja de esta historia es: primero veamos el software, realicemos pruebas y posteriormente evaluemos los resultados para generar una opinión. En muchos casos no necesitamos ser expertos en usabilidad, solo aplicando un poco de sentido común podemos decidir si es usable o no una aplicación.

Happy Hacking. :-)

U! Tu zona exclusiva

HD

Para publicar tu mensaje en la Zona U!, envíanos un e-mail contacto@hdmagazine.org, indicando ZonaU! En el asunto del mensaje.

ASCII Art "Horse"
por Chris.com

```

`T",` ,
`8, `
`" " oooob."T,
` ".)0;8:doob.`-
,`- -dP(d808Yo8;
-o8b- , ,)do008:'o;`Y8.`-
,`bo , , ,)000888o' :o0. " `
,"`"d...88000008088o :08o; . ;;;,b
,d00000" " " " " " "088888o :0880o.; :o888d
""8880b...-- :o88088o. :o' " " " " "Y80P
d8888 . . . . . :o800888 : :
"" .d008bo`'` , , ;0880:08o : :
,d08" . , -)do808o:" " ; :
,d08 ( . T)8P:8o : : : :
-" " ; ;0"Kd0o : :
,K, " .doo : : :
.doo : : " " : : . 'o:
;0000000.o:" " " " " : : ; : . 'o:
`" , . . . . . d;o:" " " " : : o ; : o : : ;
d, , , . . . . . ooo : : : : : : : o ; : : o " ' : o
,d' : : : : : :0000080o:" " : : : : : o80oo; ; o:
'P" ; ; ; ; ;0Pd888808 : : : . 'o0oo; ; ; ; : : o880oo;' 0"
,8: o : : o0` 8888800o : : : : : o8080o : : : ; ; ; ; : : o88800o;
,YP , , ; ; ; :0: 888888o : : : : : 88880oo : : : : : o8888888o;
'd : ; ; ; :0: :888888 : ; : 88888880ooooooooo888888880o;
dPY: :o80 Y088880:0; ; 088888888880000888" Y8o:088o;
' 0: 'ob` "88888880o; ;o888888888888" " `800:.`00b`
Y: ,:o: `808888800o" " " " " " " " " " " `00ob`Y8b`
:: ; :o: `80880:o0oP `80o `Y0.
` : 0o: `8880:o0P 880 :0Y
:o; 8oP :888o:P do: 80:
,oo0:80' ,d8888o:0' d0o ;:
;08odo' 888880:o' do: :oo:
d"`)80' "Y0880o' "80: o8b'
'_ ' " d:08oK -hrr- d00o' :o":
0:8o:b. :88o: `8:,
`80:;7b,, `8' Y:
`Y0;`8b'
`0o;`8:
`OP"8.`
: Y8P
`o`
Y8bod.
` " " " "
    
```



¡GRACIAS POR LEERNOS!