NO. 9

REVISTA SoftwareLibre

Noviembre / Diciembre 2007

ENTREVISTA
ALVARO LÓPEZ

APLICACIONES PRÁCTICAS DE LOS HONEYPOTS
HONEYNET MÉXICO

BONDADES DE LA AUDITORÍA DE SEGURIDAD INFORMÁTICA
ENRIQUE A. SÁNCHEZ

AUTOMATIZACIÓN EN WEB CON WWW:: MECHANIZE
DAVID MORENO GARZA

ENLI 2007

www.revista-sl.org

RevistaSL

El software libre hecho revista

~# Editorial SL

Seguramente dos conceptos nunca pasarán de moda: las tecnologías orientadas al Web y la Seguridad Informática. Dentro del primer tema tenemos como ejemplo a los frameworks, herramientas que hoy día, sin duda agilizan la creación de sitios y sus componentes.

Tenemos como ejemplos dos frameworks: Django y Codeigniter, el primero escrito con Python el segundo con PHP, nuestros colaborados nos hablan de sus características y nos guían de manera fácil para comenzar a utilizarlos.

En la parte de seguridad un nuevo concepto comienza a tomar crecimiento: los Honeypots; en este número aprendermos los conceptos básicos de un Honeypot. Además, compartiremos con ustedes la experiencia al utilizar un túnel encriptado con un servidor SSH. Para finalizar con el tema de la seguridad, encontrarán los beneficios de la Auditoría en Seguridad Informática.

Sin dejar de mencionar la columna PL escrita por David Moreno Garza, colaborador experto en Perl, en esta ocasión nos ilustra sobre el uso de WWW::Mechanize.

También, compartimos la experiencia de ENLI 2007, realizado en la ciudad de Puebla, Puebla hablando un poco de lo que fue uno de los congresos que en poco tiempo se ha ido consolidando tomando la tendencia de ser uno de los más representativos del país. La personalidad que entrevistamos para este número fue Álvaro López Ortega, aka Alo, Technical Lead en Sun Microsystems habla para nuestros lectores sobre sus actividades y su proyecto más conocido: Cherokee; además fue uno de los invitados internacionales de este congreso.

Este número es el último del año, así que esperamos sea de su agrado, gracias a todos los lectores que han hecho posible que este proyecto siga en pie. Especialmente, agradecemos a los colaboradores, quienes han compartido un poco de su conocimiento y experiencia a lo largo de este año, haciendo que la revista poco a poco vaya evolucionando.

Queremos agradecer a todos los participantes en el concurso de la nueva imagen de la revista, a través de los votos que se recibieron en el sitio web, el ganador fue Edgar Guerra, de Colombia el trabajo que envió y ustedes eligieron ya pueden admirarlo en la portada de este número.

¡Nos leémos en el 2008!

Sonia Sánchez Díaz

www.revista-sl.org

contenido

Software Libre Hecho Revista

```
revistasl:~#nmap -A -O -p0-37 Revista SL

Starting Nmap 4.20 ( http://insecure.org ) at 2007-07-12 17:21 CDT
Interesting ports on Revista SL (index):
Not shown: 2 closed ports
PORT        STATE                          SERVICE
4/tcp       HoneyNet México          Aplicaciones practicas de los Honeypots
7/tcp       Andres Vargas            SSH Tunneling
11/udp      Santiago Bonet           Problemas detectados en la difusión del SL en las empresas
13/tcp      Enrique  Sánchez         Bondades de la auditoría de Seguridad Informática
15/tcp      Jesús Alvarez            Invasión de los clientes ligeros en las empresas
19/tcp      David Moreno             Automatización de web con WWW::Mechanize
23/tcp      Julio Acuña              Django Framework
27/udp      Marco Ocampo             CodeIgniter Framework
30/tcp      Flabio Sasso             GIMP Galaxy
34/tcp      Eventos                  ENLi 2007
36/tcp      En entrevista            Alvaro López
38/tcp      BuzonSL                  Los lectores nos cuentan

Device type: Revista de Software Libre
Running:  Revista SL
OS details: Software Libre Hecho Revista  V.: 9

revistasl:~#
```

Aplicaciones prácticas de los Honeypots en la protección y monitoreo de redes de información

Miguel José Hernández y López
miguel@honeynet.org.mx
Carlos Franciso Lerma Reséndez
cflerma@uat.edu.mx

Toda organización que considere las Tecnologías de Información (TI's) como la espina dorsal de su estructura operativa debe estar a la vanguardia en sus procesos de cambio, debido a que disponer de información confiable y a tiempo constituye una ventaja fundamental. La identificación de los riesgos a los cuales está sujeta la información corporativa es de vital importancia para asegurar la integridad, usabilidad y utilidad de la misma.

Tradicionalmente, la naturaleza del ramo de la Seguridad Informática ha sido puramente defensiva. Los muros de fuego, sistemas de detección de intrusos, y el cifrado son mecanismos que se usan defensivamente para proteger los recursos informáticos. Los dogmas estratégicos de la Seguridad Informática consisten en defender la infraestructura de información tan bien como sea posible, detectar posibles fallos en la estructura defensiva y reaccionar a esos fallos de manera pro activa. La naturaleza de la existencia y operación del enemigo informático es puramente defensiva, ya que este siempre está al ataque.

Los Honeypots han demostrado su valor como herramienta de investigación en el área de la seguridad de la información. Muchos investigadores y organizaciones, tanto públicas como privadas, que forman parte de la comunidad de la seguridad están utilizando actualmente redes “trampa” para aprender las tácticas, técnicas y los procedimientos que la comunidad “hacker” utiliza para irrumpir de manera no autorizada a bóvedas de información electrónica que podrían contener información potencialmente sensible. Este artículo analiza el funcionamiento de los honeypots y su tecnología, que se está
convirtiendo en el componente clave del sistema de capas de protección.

Honeypots – Que son y como funcionan

Los Honeypots son una tecnología nueva con enorme potencial para la comunidad informática. Los primeros conceptos fueron introducidos por primera vez por varios íconos en la seguridad informática, especialmente aquellos definidos por Cliff Stoll en el libro "The Cuckoo's Egg" y el trabajo de Bill Cheswick documentado en el libro "An Evening with Berferd".

Desde entonces, han estado en una continua evolución, desarrollándose de manera acelerada y convirtiéndose en una poderosa herramienta de seguridad hoy en día. Los Honeypots son en su forma más básica servidores de información falsos, posicionados estratégicamente en una red de prueba, los cuales son alimentados con información falsa que es disfrazada como archivos de naturaleza confidencial. A su vez, estos servidores son configurados inicialmente de manera que sea difícil mas no imposible el hecho de ser penetrados por un atacante informático, exponiéndolos de manera deliberada y haciéndolos altamente atractivos para un “hacker” en busca de un blanco. Por último, el servidor es habilitado con herramientas de monitoreo y rastreo de información, de manera que cada paso y rastro de actividad de un “hacker” pueda ser registrado en una bitácora que indique esos movimientos de manera detallada.

Las funciones principales de un Honeypot son:

• Desviar la atención del atacante de la red real del sistema, de manera que no se comprometan los recursos principales de información.

• Capturar nuevos virus o gusanos para su estudio posterior.

• Formar perfiles de atacantes y sus métodos de ataque preferidos, de manera similar a la usada por una corporación policíaca para construir el archivo de un criminal basado en su modus operandi.

• Conocer nuevas vulnerabilidades y riesgos de los distintos sistemas operativos, entornos y programas las cuales aún no se encuentren debidamente documentadas.

En un contexto más avanzado, un conjunto de Honeypots forma una Honeynet, proporcionando así una herramienta que abarca un

conjunto extendido de posibles amenazas y proporciona al administrador de sistemas mayor información para su estudio. Inclusive, hace más fascinante el ataque para intruso debido a que se incrementan las posibilidades, blancos y métodos de ataque.

Clasificación de los Honeypots

Los Honeypots se pueden clasificar de acuerdo a dos criterios: Según su Ambiente de Implementación y según su Nivel de Interacción. Estos criterios de clasificación hacen fácil entender su operación y utilización al momento de planear la implementación de uno de ellos dentro de una red de datos o infraestructura de TI's.

Honeypots según su Ambiente de Implementación.

Bajo esta categoría podemos definir dos tipos de Honeypots: Para la Producción y para la Investigación.

Honeypots para la Producción: Son aquellos que se utilizan para proteger a las organizaciones en ambientes reales de operación. Se implementan de manera colateral a las redes de datos o infraestructuras de TI's y están sujetas a ataques constantes las 24 horas del día, 7 días a la semana (24/7). Se les concede cada vez más importancia debido a las herramientas de detección que pueden brindar y por la forma cómo pueden complementar la protección en la red y en los hosts.

Honeypots para la Investigación: Estos Honeypots no son implementados con la finalidad de proteger redes, sino que constituyen recursos educativos de naturaleza demostrativa y de investigación cuyo objetivo se centra en estudiar patrones de ataque y amenazas de todo tipo. Gran parte de la atención actual se centra en los Honeypots para la investigación, que se utilizan para recolectar información sobre las acciones de los intrusos. El proyecto Honeynet, por ejemplo, es una organización para la investigación sobre seguridad voluntaria, sin ánimo de lucro que utiliza los Honeypots para recolectar información sobre las amenazas del ciberespacio.

Honeypots según su Nivel de Interacción

Dentro de este criterio de clasificación, el término “Nivel de Interacción” define el rango de posibilidades de ataque que un Honeypot le permite tener un potencial atacante. Estas categorías nos ayudan a entender no solo el tipo de Honeypot con el que se está trabajando, sino también ayudan a definir la gama de opciones en cuanto a las vulnerabilidades que se desea que un atacante explote. Estas son las características de mayor importancia al momento de empezar a construir el perfil de un atacante.

Honeypots de Baja Interacción: Normalmente, éstos Honeypots trabajan únicamente emulando servicios y sistemas operativos. La actividad del atacante se encuentra limitada al nivel de emulación del Honeypot. La ventaja de un Honeypot de Baja Interacción radica principalmente en su simplicidad, ya que estos tienden a ser fáciles de utilizar y mantener con un riesgo mínimo. Por ejemplo, un servicio FTP emulado, escuchando en el puerto 21, probablemente estará emulando un login FTP o probablemente suportará algunos comandos FTP adicionales, pero no representa un blanco de importancia crítica ya que probablemente no está ligado a un servidor FTP que contenga información sensible.

Por lo general, el proceso de implementación de un Honeypot de Baja Interacción consiste en instalar un software de emulación de sistema operativo (ej. VMWare Workstation o Server), elegir el sistema operativo
y el servicio a emular, establecer una estrategia de monitoreo y dejar que el programa opere por si solo de manera normal. Este proceso, de naturaleza similar al “plug and play”, hace que la utilización de este tipo de Honeypot sea extremadamente sencilla. Los servicios emulados mitigan el riesgo de penetración, conteniendo la actividad del intruso que nunca tiene acceso al sistema operativo real donde puede atacar o dañar otros sistemas.

La principal desventaja de los Honeypots de Baja Interacción radica en que registran únicamente información limitada, ya que están diseñados para capturar actividad predeterminada. Debido a que los servicios emulados solo pueden llegar hasta un cierto límite operacional, ésa característica limita la gama de opciones que se pueden anunciar hacia el potencial intruso. De igual manera, es relativamente sencillo para un atacante el detectar un Honeypot de Baja Interacción, ya que un intruso hábil puede detectar qué tan buena es la emulación con el debido tiempo.

Ejemplos de Honeypots de Baja Interacción son: Specter, Honeyd, y KFSensor.

Honeypots de Alta Interacción: Este tipo de Honeypots constituyen una solución compleja, ya que implica la utilización de sistemas operativos y aplicaciones reales montados en hardware real sin la utilización de software de emulación e involucrando aplicaciones reales que se ejecutan de manera normal, muchas veces en directa relación a servicios como bases de datos y directorios de archivos compartidos. Por ejemplo: Si se desea implementar un Honeypot sobre un servidor Linux que ejecute un servidor FTP, se tendrá que construir un verdadero sistema Linux y montar un verdadero servidor FTP.

Las ventajas de dicha solución son dos: Por un lado, se tiene la modus operandi de los atacantes debido a que los intrusos se encuentran interactuando frente a un sistema real. De esta manera, se está en posibilidad de estudiar la extensión completa de sus actividades: cualquier cosa desde nuevos rootkits, zero-days, hasta sesiones internacionales de IRC. Por otro lado, los Honeypots de Alta Interacción no asumen nada acerca del posible comportamiento que tendrá el
atacante, generando un entorno abierto que captura todas las actividades realizadas y que ofrece una amplia gama de servicios, aplicaciones y depósitos de información que pueden servir como blanco potencial para aquellos servicios que específicamente deseamos comprometer. Esto permite a las soluciones de alta interacción conocer comportamientos no esperados. Sin embargo, esta última capacidad también incrementa el riesgo de que los atacantes puedan utilizar estos sistemas operativos reales para lanzar ataques a sistemas internos que no forman parte de los Honeypots, convirtiendo una carnada en un arma. En consecuencia, se requiere la implementación de una tecnología adicional que prevenga al atacante el dañar otros sistemas que no son Honeypots o que prive al sistema comprometido de sus capacidades de convertirse en una plataforma de lanzamiento de ataques.

Hoy por hoy, el mejor ejemplo de un Honeypot de alta interacción está representado en las Honeynets.

Ventajas y desventajas

Los Honeypots son un concepto increiblemente simple, los cuales ofrecen una fortaleza muy poderosa. Podemos observar sus ventajas en los siguientes puntos:

• Nuevas Herramientas y Tácticas: Son diseñados para capturar cualquier cosa que interactúa con ellos, incluyendo herramientas o tácticas nunca vistas mejor conocidas como '0-days'.

• Mínimos Recursos: Esto significa que los recursos pueden ser mínimos y aún así se puede implementar una plataforma lo suficientemente potente para operar a gran escala. Ejemplo: Una computadora con un procesador Pentium con 128 Mb de RAM puede manejar fácilmente una red de clase B entera.

• Encriptación en IPv6: A diferencia de la mayoría de las tecnologías para la seguridad, también trabajan en entornos sobre IPv6. El Honeypot detectará un ataque sobre IPv6 de la misma forma que lo hace con un ataque sobre IPv4.

• Información: Pueden recopilar información de manera detallada a diferencia de otras herramientas de análisis de incidentes de seguridad.

• Simplicidad: Debido a su arquitectura, son conceptualmente simples. No existe razón por la cual se deba desarrollar o mantener nuevos algoritmos, tablas o firmas. Mientras mas simple sea la tecnología, habrá menos posibilidades de error. Como cualquier otra tecnología, los Honeypots también tienen debilidades inherentes a su diseño y funcionamiento. Esto se debe a que
éstos no reemplazan a las tecnologías actuales, sino que trabajan con las tecnologías existentes:

• Visión Limitada: Solo pueden rastrear y capturar actividad destinada a interactuar directamente con ellos. No capturan información relacionada a ataques destinados hacia sistemas vecinos, a menos que el atacante o la amenaza interactúe con el Honeypot al mismo tiempo.

• Riesgo: Inherentemente, el uso de todas las tecnologías de seguridad implican un riesgo potencial. Los Honeypots no son diferentes ya que también corren riesgos, especıficamente el de ser secuestrados y controlados por el intruso y ser utilizados como plataforma de lanzamiento de otros ataques.

Aplicaciones Prácticas

Cuando son utilizados con propósitos productivos, los Honeypots proveen protección a la organización mediante prevención, detección y respuesta a un ataque. Cuando son utilizados con propósitos de investigación, éstos recolectan información que depende del contexto bajo el cual hayan sido implementados. Algunas organizaciones estudian la tendencia de las actividades intrusivas, mientras otras están interesadas en la predicción y prevención anticipada.

Los Honeypots pueden ayudar a prevenir ataques en varias formas:

• Defensa contra ataques automatizados: Estos ataques son basados en herramientas que aleatoriamente rastrean redes enteras buscando sistemas vulnerables. Si un sistema vulnerable es encontrado, estas herramientas automatizadas atacaran y tomaran el sistema (con gusanos que se replican en la víctima). Uno de las métodos para proteger de tales ataques es bajando la velocidad de su rastreo para después detenerlos. Llamados “Sticky Honeypots”, estas soluciones monitorean el espacio IP no utilizado. Cuando los sistemas son analizados, estos Honeypots interactúan con el y disminuyen la velocidad del ataque. Esto se logra utilizando una variedad de trucos TCP, como poniendo el “Window Size” a cero o poniendo al atacante en un estado de espera continua. Esto es excelente para disminuir la velocidad o para prevenir la diseminación de gusanos que han penetrado en la red interna.

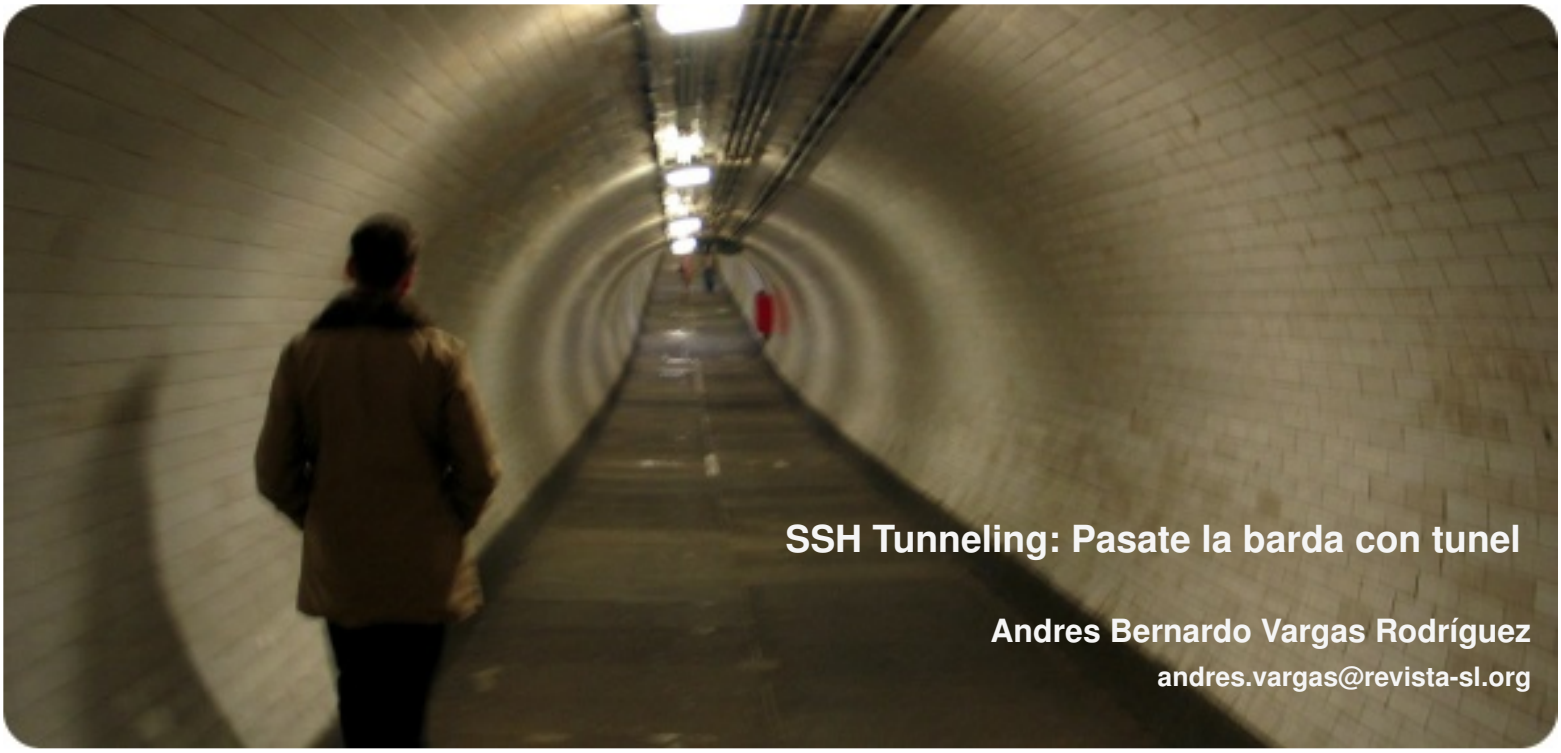
• Protección contra intrusos humanos: Este concepto se conoce como engaño o disuasión. La idea de esta contramedida es confundir al atacante y hacerle perder tiempo y recursos mientras interactúa con el Honeypot. Mientras ese proceso se lleva a cabo, se puede detectar la actividad del atacante y se tiene tiempo para reaccionar y detener el ataque.

• Métodos de Detección Precisa: Tradicionalmente, la detección ha sido una tarea extremadamente difícil de llevar a cabo. Las tecnologías como los Sistemas de Detección de Intrusos y sistemas de logueo han sido deficientes por diversas razones: Generan información en cantidades excesivas, grandes porcentajes de falsos positivos (o falsas alarmas), no cuentan con la habilidad de
detectar nuevos ataques y/o de trabajar en forma encriptada o en entornos IPv6. Los Honeypots son excelentes en el ramo de la detección, solventando muchos de los problemas de la detección clásica: Reducen los falsos positivos, capturan pequeñas cantidades de datos de gran importancia como ataques desconocidos y nuevos métodos de explotación de vulnerabilidades (zero-days) y trabajan en forma encriptada o en entornos Ipv6.

• Labor Ciber-Forense: Una vez que un administrador de red se da cuenta que uno(s) de sus servidores fue(ron) comprometido(s) ilegalmente, es necesario proceder inmediatamente a realizar un análisis forense en el sistema comprometido para realizar un control de daños causados por el atacante. Sin embargo, hay dos problemas que afectan a la respuesta al incidente: Frecuentemente, los sistemas comprometidos no pueden ser desconectados de la red para ser analizados y la cantidad de información que se genera es considerablemente extensa, de manera que es muy difícil determinar lo que hizo el atacante dentro del sistema. Los Honeypots ayudar a solventar ambos problemas, ya que son excelentes herramientas de análisis de incidencias que pueden rápida y fácilmente ser sacados de la red para un análisis forense completo, sin causar impacto en las operaciones empresariales diarias. La única actividad que guardan los Honeypots son las relacionadas con el atacante, ya que no son utilizadas por ningún otro usuario, excepto los atacantes. La importancia de los Honeypots, es la rápida entrega de la información, analizada en profundidad previamente, para responder rápida y eficientemente a un incidente

Información sobre este tema, así como resultados de investigaciones, manuales y documentación se puede encontrar en el sitio web del Proyecto HoneyNet México:

http://www.honeynet.org.mx



SSH Tunneling: Pasate la barda con tunel

Andres Bernardo Vargas Rodríguez
andres.vargas@revista-sl.org

Que tal compañeros de la revistaSL. Ya estamos en el número 9 de esta revista en el cual me he convertido ya en escritor de artículos sobre dispositivos móviles. Pero ahora no se me ocurre hablar sobre móviles, pero en realidad les voy a contar sobre un problema que se me presentó.

En mi escuela acaban de implementar un firewall, el cual bloquea todo el trafico por TODOS los puertos, excepto los puertos 80 (www), 443 (ssl), 21 (ftp), 22(ssh). Esto hace que no pueda conectarme al msn con mi cliente favorito, además que implementaron un squid, el cual bloquea ciertas paginas como youtube y el msn web. Y con esto además no me permite conectarme al irc, ni descargar mi correo pop ni enviar correo por smtp. :-(

En mi afán para brincarme el firewall, encontré la forma: con un túnel encriptado.

Requerimientos:

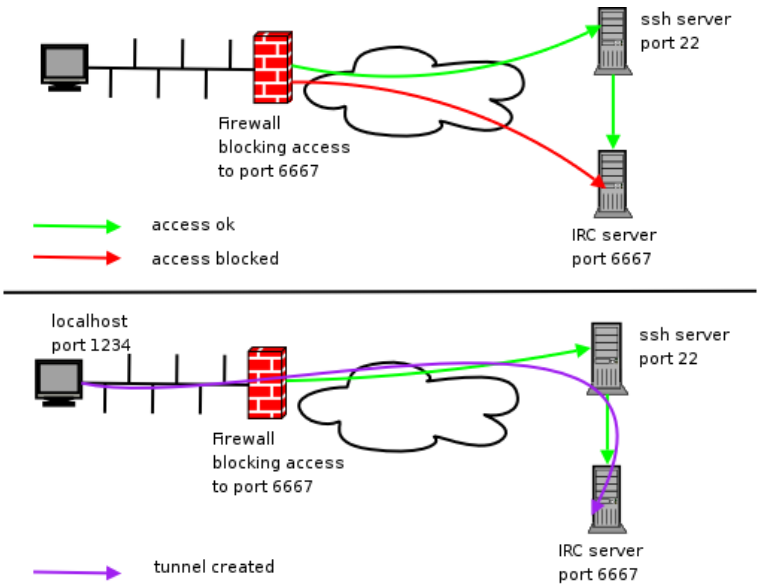
- Servidor ssh (un servidor remoto fuera del firewall)
- Cliente ssh

¿Cómo funciona esto?

Digamos que estamos en una pc que esta dentro de la red que tiene un firewall, éste no deja pasar el tráfico a través de algunos puertos. Pero si podemos llegar a otra computadora a través del puerto 22. Y como esa computadora se encuentra fuera del firewall puede llegar a conectarse al destino que deseamos llegar.

Entonces lo que se hace es crear un túnel entre la computadora _fuera_ del firewall y la computadora _dentro_ del firewall. Este túnel va a redireccionar todo el tráfico que nosotros le mandemos. Lo que implica que en la máquina dentro del firewall se crea un puerto local el cual enviará todo el tráfico hacia fuera del firewall hasta la computadora externa que redireccionara el tráfico al destino. A esto se le llama conexión proxy. Ya que usas una computadora de pasarela para salir al otro lado.

Hay otra opción para resolver esto, es hacer un portforward. Esto es



Port forwarfing

mas específico que una conexión proxy ya que contamos con un destino de tráfico específico, como un puerto, y una computadora. Lo que me refiero es que creamos un túnel a través de un puerto, todo el tráfico que le enviemos a ese puerto saldrá a una computadora específica, con un puerto específico, esto sirve para programas que no soportan conexiones con un proxy, como el IRSSI.

¿Cómo se hace esto?

Conexion proxy:

```
$ssh   usuario@host -D 2000 -N
```

Puedes usar cualquier puerto pero mejor házlo en un rango de 10025 y 10110 ya que son puertos que se pueden abrir sin ser root.

Esto hace que tengamos una conexión por ssh a un servidor pero si vemos:

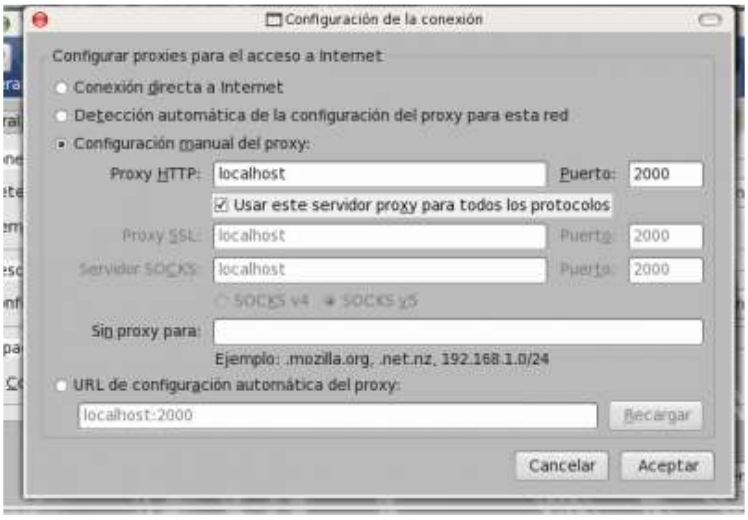
```
$ netstat -atop --numeric-port | grep ssh
tcp            0          0
127.0.0.1:2000
0.0.0.0:*                          LISTEN
24874/ssh                   off (0.00/0/0)
```

Tenemos el puerto 2000 abierto y a la escucha en nuestra pc local. Por lo que tenemos un puerto para hacer nuestra conexión proxy. Y ahora configurar algunos programas.

Ahora vamos a nuestro amsn. En cuenta> preferencias > Conexion > Me conecto a internet a través de un puerto proxy, uso socks5 y llenamos en host: 127.0.0.1 (localhost) donde esta el server ssh, y el puerto 2000. Y ¡guala! cerramos y ya podemos conectarnos al msn a través de nuestro túnel y además con tráfico encriptado.

Nuestro Correo Thunderbird (o Icedove).

Buscamos la parte de conexión en preferencias (dependiendo la versión cambia la ubicación) .



Hagamos algo aun mas interesante.

¿Y si configuramos todo gnome para que funcione así?

En Sistemas> Preferencias > Proxy de la Red

¿Y que pasa con las aplicaciones que no tienen soporte para conexión proxy?, por ejemplo algún cliente irc, o ftp. No podemos configurar para que use proxy ya que no tiene soporte. Aquí es donde entra el portforward. Con éste se crea un túnel con destinos específicos.

```
$ ssh user@host -L
<portlocal>:host_destino:<portdest> -N
```

Esto seria así:

```
$ ssh zodman@zod.com.mx -L
2001:irc.freenode.org:6667 -N
```

Con esto se conecta por ssh al server pero en el nestat -atop --numeric-port | grep ssh vemos otro túnel

```
tcp            0          0
127.0.0.1:2001
0.0.0.0:*                          LISTEN
25830/ssh                   off (0.00/0/0)
```

Ahora con un cliente irc que no soporte proxy:

```
$ irssi -c localhost -p 2001
04:17 -!- Irssi: Looking up localhost
04:17 -!- Irssi: Connecting to localhost
[127.0.0.1] port 2001
04:17 -!- Irssi: Connection to localhost established
04:17 !localhost *** Looking up your hostname...
04:17 !localhost *** Checking ident
04:17 !localhost *** Couldn't look up your hostname
04:17 !localhost *** No identd (auth) response
04:17 -!- Welcome to the freenode IRC Network zodman
```

O si no,

```
$ ftp   anonymous@localhost:2001
Buscando localhost
Intentando localhost:2001
Conectado a localhost:2001
220 saens.debian.org FTP server (vsftpd)
USER anonymous
331 Please specify the password.
PASS xxxx
230-
```

```
230-This site is just another one in a worldwide
array of Debian mirrors.
```

Ya sabemos como hacer túneles pero siempre tener abierta una terminal a veces es estorboso por eso es mejor utilizar un manejador de túneles ssh.

http://sourceforge.net/projects/gstm/

No duden en pasar a visitar la web del autor y saludarle. Encontrarán mucha información interesante, hacks, y demas cosas en:

http://www.zod.com.mx

Sandino Networks

TEQUILA, MEZCAL, CHARANDA, VODKA O UN BUEN BOURBON.
Servicios de hosting a la carta.

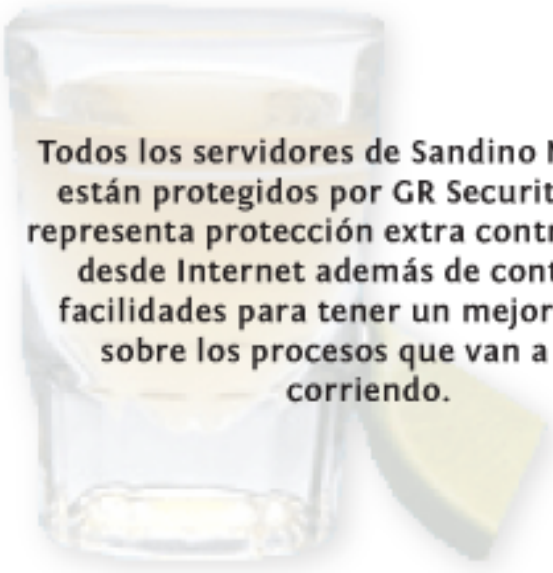
Desde hosting compartido hasta servidores dedicados. En Sandino Networks nos adaptamos a las necesidades de nuestros clientes.

Contamos con un gran soporte técnico y servicios de consultoría especializada que la experiencia del staff nos brinda

Sandino Networks es una compañia formada por personas que viven en Internet y han crecido con él.

Emergency Recovery.
El mejor staff de soporte tecnico a su disposicion

Sistemas inutilizables, perdida de datos, regreso a producción urgente. Nosotros lo ayudaremos a que regrese todo a la normalidad con nuestros servicios de consultoria emergente. Aunque ¿para qué sufrir, si puede prevenir todo ésto?



Todos los servidores de Sandino Networks están protegidos por GR Security lo que representa protección extra contra ataques desde Internet además de contar con facilidades para tener un mejor control sobre los procesos que van a estar corriendo.

La seguridad es nuestra mas grande preocupación.

informes@sandino.net
www.sandino.net.

Problemas detectados en la difusión del Software Libre en las empresas

Santiago Bonet
tic@aimme.es

Según Richard Stallman, en su pasada visita a la Conferencia internacional de software libre 3.0, en Extremadura han llevado a cabo la introducción del software libre en los ámbitos educativos (Linex) y empresariales (LinexPyme), simplemente porque han sido los primeros en pasar de las palabras a la acción. Es decir, todo el mundo habla del software libre pero pocos ámbitos los están incorporando en realidad en la vida cotidiana. En el caso de la Comunidad Valenciana, también se empezó a extender hace unos años en el ámbito educativo (Lliurex) y en el empresarial se está llevando a cabo desde finales de 2005 a través del Proyecto SourcePyme.

Durante los primeros meses de la puesta en marcha del proyecto SourcePyme, han ido aflorando los diversos problemas que suponen un freno a la innovación (hacer viable la tecnología comercialmente, es decir que exista oferta y demanda) y a la difusión (hacer viable la tecnología económicamente, es decir que se gane dinero con ello):

DEMANDA

Los problemas identificados en el caso de la demanda (empresas usuarias de software libre) son relativos a:

1) Piratería: Según la consultora IDC, en España el índice de piratería se sitúa en el 46%, lo que representa para el sector TIC unas pérdidas de 600 millones de Euros. La industria del software en España está formada por cerca de 12.000 empresas que dan trabajo a aproximadamente 80.000 personas, además de propiciar otros 300.000 empleos indirectos.

RECOMENDACIÓN: Eliminar por completo el software pirata, y utilizar el 60% de software libre y el 40% de software propietario con licencia (sólo para el caso de no existir una solución de software libre con unos niveles de calidad similar). De esta forma se pueden beneficiar de los mejor de ambas tecnologías y a la vez ante una posible caída de los sistemas debido a la propagación de un nuevo virus, tendríamos la mitad de los sistemas sin verse afectados.

2) Miedo al cambio: Por lo general, entre las empresas hay bastante confusión en relación al software libre (open source). Se confunde con el software gratuito (freeware) o directamente se asocia a Linux, siendo ésta una de las posibles plataformas sobre las que ejecutar software libre, junto con Windows.

RECOMENDACIÓN: Migrar a software libre decenas de aplicaciones ejecutándolas sobre Windows para el caso de tener miedo a cambiar a linux. De esta forma el impacto del cambio es menor, tanto en formación del personal como soporte técnico posterior.

3) Desconfianza: Algunas empresas descartan la implantación de software libre en su organización, dado que o no hay software libre con similares niveles de calidad en algunas áreas o hay pocas implantaciones (software CAD/CAM, gestión ERP, etc.), o bien los ahorros en licencias del resto de software se destinarían a formación y soporte, o bien desconfían de que haya empresas detrás que garanticen el mantenimiento y soporte de dicho software, o que el formato de los ficheros sea compatible con los generados con software propietario.

RECOMENDACIÓN: Empezar por los sistemas servidores, luego migrar las aplicaciones de ofimática y por último los programas de producción o ERPs. La ventaja de empezar ya, es que conforme pase el tiempo, cada vez habrá mayor número de aplicaciones, serán de mayor calidad, y a la vez existirán mayor número de empresas dispuestas a garantizar el mantenimiento y soporte. De esta forma, se tendrá camino recorrido cuando se migren las aplicaciones de mayor coste. En cuanto al formato, se recomienda usar formatos basados en estándares abiertos.

OFERTA

Los problemas identificados en el caso de la oferta (empresas

proveedoras de aplicaciones y servicios de software libre) son relativos a:

1) Miedo al cambio: Por lo general, las empresas de informática están acostumbradas a desarrollar software a medida, sin suministrar los fuentes a sus clientes, debido al incremento de coste que supondría el no poder rentabilizar sus desarrollos con otros clientes. Con la aparición del software libre, algunas empresas creen que la única solución es liberar parte del código que hasta la fecha ha sido cerrado.

RECOMENDACIÓN: Dedicar recursos humanos a estudiar y analizar la extensa oferta de aplicaciones de software libre (www.sourceforge.net), de forma que en unos meses se pueda ofertar las dos soluciones a sus clientes, la propietaria (con coste de licencia) y la libre (sin coste de licencia), cada una con sus peculiaridades. De esta forma será el mercado el que decidirá cual adopta según precio y prestaciones. En una segunda fase cabe también plantearse el aportar horas de desarrollo a la comunidad con la que se está colaborando, así como a proponer nuevos proyectos si se cuenta con amplio respaldo técnico (colaborando con una red de empresas similares) y económico (generalmente con una aportación de capital riesgo).

2) Falta de espíritu colaborativo: Por lo general las empresas de informática están acostumbradas a desarrollar software de forma individual.

RECOMENDACIÓN: Apostar por el modelo colaborativo para desarrollo de software, con el fin de reaprovechar el trabajo de otros y a su vez colaborar con ellos, y trasladar al cliente final los ahorros en licencias.

En resumen, nos encontramos ante la revolución del conocimiento abierto (Open Knowledge) que está cambiando los paradigmas de crecimiento de la mayoría de sectores empresariales, entre ellos el sector informático. Las empresas que sepan adaptarse y vean en ello una oportunidad, mediante la cooperación competitiva (coopetition) pueden llegar a alcanzar metas más lejanas que de forma individual. Tal y como decía John Perry Barlow, en la pasada edición del PowerfulIdeasSummit en la Universidad Politécnica de Valencia, “en una jerarquía el poder lo tiene quien guarda secretos; en una red el poder lo obtiene quien disemina información”.

Para más información sobre el artículo anterior puede ver o descargar el vídeo "Más allá de la noticia - UPTV: Software libre en la empresa. Proyecto SOURCEPYME" (21-Junio-2007):

http://www.aimme.es/formacion/teleformacionVideo/ficha.asp?id=56

Sourcepyme es un proyecto coordinado por AIMME en cooperación con AIMPLAS, ITI y UPV, promovido por el IMPIVA, que tiene como finalidad fomentar y facilitar el uso de software libre en las pymes con el objetivo de ganar capacidad tecnológica y reducir costes. Este proyecto coordina, a nivel de la Comunidad Valenciana el desarrollo y la adaptación de aplicaciones basadas en software libre para el uso en empresas.

Bondades de una auditoría de seguridad informática

Enrique Alfonso Sánchez Motellano
enrique.sanchez@revista-sl.org

Una Auditoría Informática es una parte central e importante de la seguridad, ya sea una aplicación comprada o una aplicación hecha en casa; la información contenida es el verdadero valor de la aplicación.

Una aplicación contiene errores debido a que el programador no fue enseñado con seguridad en mente, en realidad, no habla nada de su experiencia sino que es un campo nuevo que se lleva rápidamente en la expansión de técnicas y formas de explotación.

Libros y clases están llenas a veces de errores. El reuso de código así como toma de ejemplos y políticas de programación puede afectar hasta el punto de crear una aplicación vulnerable, estas vulnerabilidades pueden ir desde suplantación de identidades, robo de contraseñas, toma de control de la aplicación, toma de control del servidor, etc.

Estas aplicaciones se ejecutan en servidores que contienen información como correos, planes de ventas, mercadotecnia, asuntos personales de altos ejecutivos, etc. Los cuales conllevan un riesgo y un valor añadido a la red.

Un Penetration Testing, también llamado hackeo ético, es un ataque controlado por consultores que conocen técnicas hackers o hackers de sombrero blanco (Whitehats) que descubren y explotan fallos de seguridad dentro de la red y los reportan, reduciendo el riesgo de la red así como de los aplicativos e información dentro de la misma.

¿Inversión o gasto?

En mi experiencia he tenido que contestar la siguiente pregunta mas de una vez:

• ¿Por qué si mi aplicación costó X el costo de auditarla es Y? (Donde X < Y)

Creo que la respuesta más obvia es:

• El valor real de la aplicación puede ser definida por la siguiente ecuación de riesgo:

COSTO APLICACIÓN

+ COSTO INFORMACIÓN
+ COSTO DOWNTIME
– AMORTIZACIÓN COSTO APLICACIÓN
+ MANTENIMIENTO

_____
= COSTO TOTAL DE APLICACIÓN

Como se puede ver en la ecuación anterior el costo de la aplicación es solo la inversión inicial, intangibles como COSTO INFORMACIÓN y COSTO DOWNTIME deben ser generados en un análisis de riesgo informático ya que son los costos generalmente más grandes debido a que el COSTO DOWNTIME es:

COSTO OPERACIÓN

+ PÉRDIDAS MERCADOTECNIA (MARCA, ETC)
+ VENTAS PÉRDIDAS (En caso de ventas por Internet o sistemas críticos abajo)

_____
= COSTO DOWNTIME

La importancia de una auditoría de código radica en la reducción de los factores de riesgo haciendo que la probabilidad de ocurrencia baje a lo mínimo. Esto nos da un ejemplo de riesgo:

Una aplicación de $50,000 MXN maneja $3,000,000 MXN el costo de downtime es de $50,000 por hora y el mantenimiento es de $5,500 MXN al mes.

Si la aplicación tiene un riesgo del 75% de ser atacada y perder los datos, ser modificados o comprometer el servidor, el riesgo real que

debe de reportarse es de:

($50,000 + $3,000,000 + ($50,000 * 3) + $5,500) * 75% = $2,404,125 MXN

Este riesgo tiene una probabilidad de generarse en un 100% en un tiempo cercano (debido a la interacción de la variable del 75% está ya prorrateado al tiempo de 1 año).

Este tipo de riesgo es simplemente inaceptable en la mayoría de las compañías, al auditar la aplicación se encuentran los fallos, se arreglan y por lo tanto la probabilidad de ser atacada y perder datos, modificarlos o comprometer el servidor baja, por ejemplo un 25% del ejemplo anterior seria:

($50,000 + $3,000,000 + ($50,000 * 3) + $5,500) * 75% = $801,375 MXN

Lo cual es una reducción de riesgo considerable, tomando en cuenta que la auditoría costará $1,000,000 MXN aun asi habría un ahorro de aproximadamente $600,000 MXN neto.

Esto es importante debido a que la mayoría de las compañías están tomando la opción del “open source” como una opción barata, pero descuidan que no es gratis su mantenimiento y que el riesgo sigue estando vigente y que debe mantenerse de manera administrativamente responsable para la empresa.

Invasión de los clientes ligeros en las empresas

Jesus Antonio Álvarez Cedillo
jesus.alvarez@revista-sl.org
Elizabeth Acosta Gonzaga
agonzaga@ipn.mx

Las organizaciones, empresas y escuelas, estan siguiendo una tendencia iniciada por las agencias Federales de Los Estados Unidos de America y es el eliminar las PC de los escritorios y las están reemplazando por sistemas de "terminales ligeras". Se da a cada empleado una pantalla de computadora, un teclado y un ratón, pero es la computadora central la que almacena todos los datos y efectúa la mayor parte de los procesos, algo que reduce los costos de mantenimiento y vuelve mucho más fácil hacer un seguimiento y restringir cómo los trabajadores utilizan sus máquinas.

Introducción.

A lo largo de los años las terminales ligeras han aparecido y desaparecido de las oficinas. Ahora vuelven debido a los crecientes costos de mantenimiento de las redes y las exigencias a las empresas de tener más seguridad y mantener mejores registros. La firma de investigación de mercado IDC pronostica que para 2008 las terminales ligeras representrarán casi el 10% del mercado de computadoras de escritorio en empresas grandes y medianas, cifra que contrasta con el 5,4% de este año.

IDC prevé un crecimiento aún más rápido para otros sistemas llamados "PC ultrafinas" (Blade PC). Al contrario de las terminales, que dependen de grandes servidores centrales, los sistemas de PC ultrafinas dan a cada empleado una versión reducida de una PC ordinaria, pero las máquinas se almacenan en una sala con un servidor central para facilitar su mantenimiento. IDC predice que las ventas de blades crecerán de 350.000 este año a 6,5 millones en 2008.

El número y variedad de las terminales ligeras ha estado creciendo. En abril Neoware Systems Inc. presentó una que cuesta US$199 por terminal, una ganga frente a los US$800 que suele costar una PC. La fabricante californiana de chips PMC-Sierra Inc. anunció que estaba organizando un grupo de empresas de microprocesadores para que trabajen con fabricantes chinas para crear una computadora de red de US$150 que contenga software de fuente abierta.

Pero, en general, el atractivo de las terminales ligeras no es su bajo precio. Algunas cuestan tanto como las PC de escritorio, dependiendo, por ejemplo, de si utilizan el sistema operativo Windows o uno menos caro. Las PC ultrafinas suelen costar más que las PC tradicionales debido al software y hardware que las conecta a la red central. Los ahorros se producen en la gestión de las computadoras, ya que los costos de mantenimiento bajan radicalmente. [KOKI2006]

Las terminales ligeras también están llegando a los hogares. Hace poco Sun Microsystems Inc. anunció que sus empleados que trabajan desde casa pueden utilizar sus terminales ligeras Sun Ray para acceder a los servidores centrales.

Definición y conceptos básicos

Un Cliente Ligero (Thin client) es una computadora (cliente) en una arquitectura de red cliente-servidor que tiene muy poca o ninguna lógica del programa, por lo tanto depende principalmente del servidor central para las tareas de procesamiento. La palabra ligero se refiere a lo pequeña que es la imagen de arranque, quizá no más grande que la requerida para conectar a la red y arrancar un navegador web. Ver figura 1.

En el diseño de una aplicación cliente-servidor, hay una decisión que hay que tomar: qué parte de la aplicación debe ser hecha por el cliente y cuál por el servidor. Esta decisión puede afectar crucialmente el costo del servidor y el cliente, la robustez, la seguridad de toda la aplicación.

Las ventajas de los lientes Ligeros son las siguientes:

Informacion Centralizada: Como la informacion se encuentra en un solo lugar facilita la realizacion de backups y evita que se guarden

Figura 1: Un sistema comercial básico de cliente ligero.

archivos que no sean del negocio.

Menor costo de hardware: El hardware de los Clientes Ligeros es generalmente más barato ya que estos no cuentan con disco duro, memoria para las aplicaciones, o un procesador poderoso.

También tienen un periodo de funcionamiento más grande antes de necesitar actualizarse o quedar obsoletos.

Menor IT costo de administración: Estos Clientes Ligeros son manejados enteramente desde el servidor, el hardware tiene menos lugares donde puede fallar, y el entorno local es altamente restringido, por lo tanto provee protección contra el cargado y la ejecución de malware.

Más barato y seguro: Los Clientes Ligeros pueden ser diseñados para que ninguna información de las aplicaciones resida en los clientes, entonces la protección contra el malware es centralizada

Sin valor para los ladrones: El hardware de los Clientes Ligeros es poco útil fuera de un entorno cliente-servidor. Ladrones interesados en equipamiento de computadoras tardan mucho más tiempo en revender el hardware de los Clientes Ligeros y este es mucho menos valioso.

En la práctica, parece que hay poco donde elegir para decantarse entre una y otra arquitectura para la mayoría de las aplicaciones. Pocas situaciones se decantan claramente hacía una u otra. Los proyectos de computación distribuida como SETI@home (que utilizan una gran cantidad de ordenadores remotos para realizar un análisis computacional intensivo) son aplicaciones que requieren clientes pesados. Por otro lado los sistemas de difusión de entretenimiento multimedia o la difusión de material educativo a muchos clientes puede ser realizada mejor con clientes ligeros, ya que se difunde el mismo material a todos los clientes. Los protocolos más comunes son los siguientes:

• XML sobre HTTP usado por XHTML y BXML de Backbase para definir aplicaciones ricas de Internet.

• X11 usado esencialmente por variantes de Unix tecnología.

• Computación en Red Virtual o VNC

• Citrix ICA con MetaFrame

• RDP el mecanismo por defecto de acceso remoto al escritorio para Windows

• HTML sobre HTTP usado por un gran grupo de aplicaciones web.

• Tarantella

Empresas en clientes ligeros:

Hoy en día, para una empresa invertir en tecnología, tanto en hardware como en servicios, es necesario que estos productos tengan dos cualidades: Tiene que crear nuevos flujos de ganancias y básicamente, ahorrarle dinero a la compañía. La computación basada en servidores pertenece a la exclusiva categoría de tecnología con propuestas de valor claros, tanto en términos de ahorro de costos como también aumento de la productividad del usuario final. El concepto detrás de Thin Client es simple: Porque poner equipos PC caros y con alto mantenimiento en cada escritorio cuando ellos pueden tener la misma funcionalidad por medio de un “thin client” sin partes móviles? Los poderosos servidores de hoy permiten correr aplicaciones populares como Microsoft Word, Power Point, Excel, Outlook, Internet Explorer, Adobe Acrobat (con excepción de aplicaciones de video y fotografía). Todos estos procesos se corren en un servidor central, solo actualizaciones de pantallas, input del keyboard y del mouse son transferidos entre el servidor y la terminal que puede estar localizada en la oficina, casa, o remoto.

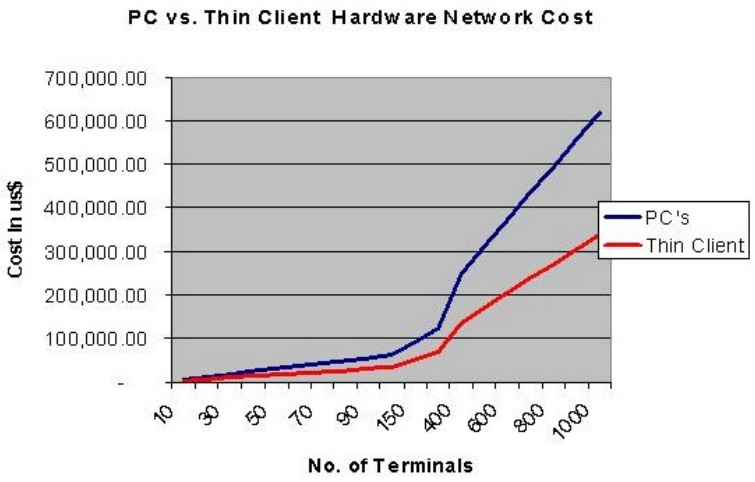


Figura 2: Comparativo en costo de una Terminal ligera contra el hardware de red Tradicional (Fuente IDC).

Cuando un usuario comienza una cliente Ligera o “Thin Client”, el concepto de trabajo basado en servidor rápidamente se vuelve familiar y conveniente ya que no importa donde el usuario este, en la casa, en la oficina o en alguna oficina remota e inclusive en un aeropuerto con software de emulación thin client corriendo en una notebook. El ambiente familiar de LAN de la oficina es replicado y hay rápido acceso a almacenamiento, aplicaciones, email y todo lo que resida en el servidor.

Con Computadoras convencionales y con Computadoras portatiles, el intentar conectarse a un servidor desde un sitio remoto puede ser frustrante y requerir de tiempo. Documentos, presentaciones, emails tienen que ser bajados del servidor antes de comenzar a trabajar con ellos.

El mayor problema de trabajar en forma remota se puede demostrar con simples matemáticas, una típica oficina tiene una LAN con ancho de banda entre 128k a 512kbits por segundo, mientras que una conexión dial-up esta restringida a solo 56kbits por segundo. Con email, PowerPoint, hojas de Excel y documentos de Word ocupando varios megabytes y múltiples usuarios compartiendo ese ancho de banda se forma rápidamente un cuello de botella generando frustración y desesperación por parte del usuario.

Cada documento, hoja de calculo, pagina web, presentación e emails tiene que ser “bajado” del servidor antes de ser abierto y/o modificado. Una vez que los documentos son modificados tiene que ser “subidos” nuevamente al servidor, lo anterior se muestra en figura 3.

Figura 3: Arquitectura de funcionamiento de una Terminal ligera.

La única solución del problema anterior, se resuelve instalando servidores locales en cada oficina remota y replicar la información en forma diaria o restringir sesiones remotas a tareas que utilizan un menor ancho de banda. [ORFA2006]

Este concepto de “bajar documentos antes de trabajar” es un mal necesario en un ambiente tradicional que rápidamente lleva a otro problema: Versiones múltiples de documentos que se comienzan a multiplicar ya que múltiples usuarios pueden editar localmente el mismo archivo y no los vuelven a “subir” al servidor para ahorrar tiempo ya que lo hacen mas tarde. Este concepto también genera otro problema, Computadoras robadas con información confidencial o critica. [MICR2006]

Aplicaciones como Outlook o Power Point son abiertos y corridos en el servidor, solo el display y el input del keyboard/Mouse es comunicado al terminal Thin Client. El resultado es que usuarios remotos pueden conectarse al servidor corporativo y tener sus aplicaciones visualmente replicadas en sus pantallas de Thin Client o en la pantalla del notebook con acceso total a aplicaciones, discos duros como si estuvieran sentados en su propia oficina. [TODD2005]

Esto no quiere decir que todos los usuarios en una corporación tienen que utilizar tecnología Thin Client. Hay ciertos perfiles de usuario que pueden requerir procesamiento o almacenamiento local.

Protocolos avanzados Thin Client como el Citrix ICA que soporta la transferencia de display y keyboard/Mouse típicamente requieren un mínimo de 20Kbits por segundo.

Como resultante los usuarios pueden experimentar un rendimiento similar a un LAN sobre líneas regulares de teléfono.

La arquitectura general de estos sistemas es mostrada en la figura 4.

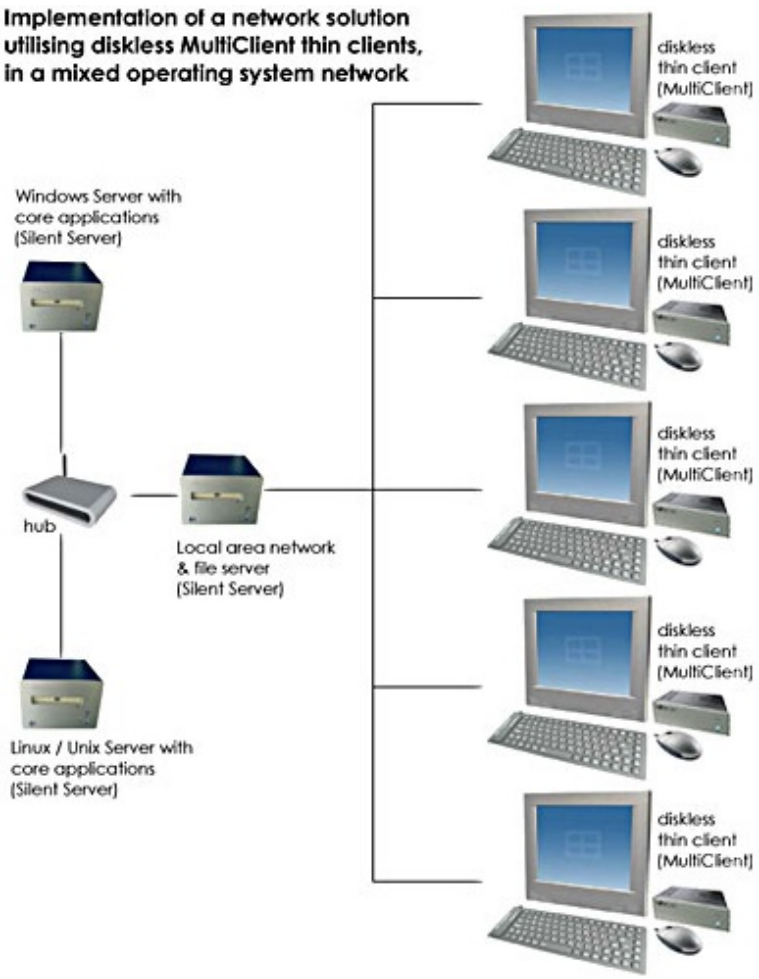


Figura 4. Arquitectura Básica.
Fuente: MULTICLIENT

Conclusiones

La aparición del ordenador personal, o PC, revoluciona el mundo de la tecnología permitiendo, en un entorno como el empresarial, el incremento de la productividad por parte de los empleados. Ahora bien, dicho incremento se asocia inmediatamente a un costo de mantenimiento importante.

El cliente ligero es tan sólo un dispositivo de escritorio, que permite a cada usuario disponer de un entorno informático en su puesto de trabajo. Este cliente ligero depende obligatoriamente de un equipo servidor, o estación de trabajo principal, que es la encargada de distribuir escritorios entre todos los clientes conectados. Basa su eficacia en la utilización de los recursos mínimos para su funcionamiento. Un cliente ligero no procesa ningún tipo de datos, por

lo que no se requiere una máquina potente, dejando ese trabajo al equipo servidor.

Puesto que un cliente ligero no procesa datos, y por lo tanto no requiere de un equipo complejo, su precio es económico.

El carácter interactivo, simultáneo y aleatorio de los clientes permite el aprovechamiento al máximo de los recursos del equipo servidor.

La actualización de todos los clientes (puestos de trabajo) se realiza desde el servidor. Y el mantenimiento es el de un único equipo, el servidor.

BIBLIOGRAFIA.

[ORFA2006] Orfali, CLIENTE/SERVIDOR GUÍA DE SUPERVIVENCIA. 2.ª ED.Editorial McGrawHill,2006

[MICR2006] Microsoft Corporation,SMART CLIENT ARCHITECTURE AND DESING GUIDE
Microsoft , Editorial McGraw-Hill

[TODD2005] Todd W. Mathers, WINDOWS SERVER 2003/2000 THIN CLIENT SOLUTIONS, Editorial SAFARI.

[KOKI2006] Kokichi Matsumoto, THIN CLIENTS TRANSFORM CORPORATE IT INFRASTRUCTURE, Editorial HP JAPAN.

Automatización en web con WWW::Mechanize

David Moreno Garza
damog@espiral.org.mx

WWW::Mechanize es un módulo muy interesante. Básicamente nos permite automatizar o mecanizar una conversación a través de una página web. Es decir, poder automatizar el procesamiento de enlaces, imágenes, formas, etc. En realidad WWW::Mechanize actúa como una subclase de LWP::UserAgent, que es el mítico LWP en Perl.

Hay que entender que todo lo que hacemos en un navegador web, o casi todo, es posible automatizarlo por medio de scripts, debido a que lo que en realidad sucede es que trabajamos con cabeceras y cuerpo HTTP, tanto en petición como en respuesta. Todo es a través del protocolo HTTP.

Mi trabajo actual requiere mucha interacción automatizada con páginas web y WWW::Mechanize me ha permitido hacer desarrollos muchos más rápidos, mucho mejor encapsulados y con un nivel de abstracción menor.

En esta ocasión, quiero hacer algo para mostrar de qué es capaz WWW::Mechanize y que ustedes, amables lectores, vean su utilidad real. Lo que haré será un proceso de autenticación en el wiki del proyecto Debian y la obtención de una de sus páginas. Básicamente es loguearme en el wiki y obtener el valor de uno de los campos de una forma. Lo interesante en este asunto viene al automatizar el login en un sistema basado en web, escribir en una forma y enviar forma, obtener campos, etc: WWW::Mechanize lo hace muy simple.

Manos a la obra.

Script

Vamos a empezar por invocar a nuestro intérprete de Perl activando warnings, así como el pragma strict.

```
#!/usr/bin/perl -w
use strict;
```

Como necesitaremos un nombre de usuario y contraseña para acceder el wiki en Debian (es necesario para editar páginas), vamos a tomarlos como argumentos:

```
my $username = $ARGV[0] || die "ERR: Es necesario
especificar un nombre de usuario", "n";
my $password = $ARGV[1] || die "ERR: Es necesario
especificiar una contraseña", "n";
```

Los argumentos vienen en el arreglo @ARGV: El primer argumento de nuestro programa será el nombre de usuario y el segundo será la contraseña. Si no existen, entonces nuestro programa morirá.

A continuación llamamos al módulo WWW::Mechanize y creamos un objeto de ese tipo.

```
use WWW::Mechanize;
my $mech = WWW::Mechanize->new;
```

Bastante simple.

Ahora tenemos que ver qué es lo que haríamos desde un navegador convencional para hacer lo que queremos. Lo más lógico es que entráramos a la página del wiki de Debian, cuya URL es http://wiki.debian.org/. Vamos a hacerlo de la misma manera en el script.

```
$mech->get('http://wiki.debian.org/‘);
unless($mech->success) {
 die “Tuvimos problemas al acceder la página de
Debian”, “n”;
}
```

Básicamente estamos utilizando dos métodos disponibles en Mechanize: get() y success(). El primero hará que nuestro objeto viaje hasta la página que le indiquemos. En cualquier caso, el método success() nos indica por medio de un valor booleano si la última operación que se realizó en Mechanize, fue exitosa o algo falló, lo cual

nos sirve para tener más control del flujo de nuestra aplicación. Si no tenemos problemas con get(), entonces el programa no morirá con el mensaje especificado; sin embargo si sí hubo problema, es decir, success() regresa falso, entonces entrará en acción el die(). Continuemos.

Una vez que estamos en Firefox (u Opera, o Internet Explorer, no sé) en la página http://wiki.debian.org/, ¿qué es lo siguiente? Bueno, lo normal, que es al no estar logueados, nos aparezca un link que dice "Login". En un proceso común tendríamos que dar clic en él para continuar con el proceso de autenticación. Eso mismo haremos:

```
$mech->follow_link(text => 'Login') or die "ERR:
No pude dar clic en login", "n";
```

Así de simple. Con el método follow_link() y el parámetro 'text', podemos indicarle a Mechanize que donde estamos actualmente, que sería la página frontal del wiki, siga el enlace cuyo texto es 'Login'. Mechanize nos permite incluso acomodar regex en estos campos, lo cual nos facilitaría aún más al lidiar con contenido dinámico o situaciones más complejas. Como follow_link() regresa realmente un objeto HTTP::Response en caso de éxito, también provee un retorno en caso de falla, que es undef. Así que si hubo una falla al intentar seguir el enlace, moriremos con el mensaje de error especificado.

Una vez que dimos clic en 'Login', podemos ver en nuestro navegador convencional que la página donde nos encontramos es aquella con la URL 'http://wiki.debian.org/UserPreferences'. En realidad pudimos habernos evitado algo de código al empezar nuestro primer get() con esta URL pero intentaba mostrar un poco de las bondades de Mechanize. ¿Qué tal si en tu script no sabes en qué URL se encuentra tu navegador de Mechanize? Puedes usar el método uri() e incluso el método response(). Eso te ayudará a analizar bien el flujo de tu programa.

En fin, en esa página de UserPreferences podemos ver que tenemos una forma para introducir "Name", "Password", etc. En realidad esos dos campos son los que realmente nos interesan. Para ésto, nos puede servir mucho la extensión WebDeveloper de Firefox, pues con ella podemos ver la información detallada de las formas en las páginas. Lo que queremos saber es cuál es nombre de la forma para ingresar el nombre de usuario y la contraseña, y además, el nombre de estos dos campos (hay que recordar que comúnmente las formas en HTML llevan un parámetro "name"). O incluso puedes intentar buscar estos parámetros viendo el código HTML directo desde la página.

Luego de ver el HTML (o de usar la extensión de Firefox) nos damos cuenta que en la página de autenticación de Debian hay tres formas, sin embargo, la forma donde se introducen el nombre de usuario y la contraseña, ¡no tiene nombre! Mechanize sabe que algunos webmasters no colocan esta información, así que provee otra forma en que puedes usar esas formas: Por número. Puedes especificar si trabajarás con la primera, segunda, tercera o enésima forma. En nuestro ejemplo, la forma de autenticación es la tercera:

```
eval {
 $mech->form(3);
 $mech->submit_form('username' => $username,
'password' => $password);
};
die "ERR: Problemas al enviar la forma: $@", "n"
if $@;
```

Un bonito pedazo de código aquí. Lo primero que notamos es que usamos un bloque eval{} para encapsular dos métodos, form() y submit_form(). form() selecciona la forma con la que trabajaremos, en este caso la tercera y submit_form() envía esa forma con los campos indicados. Es bastante intuitivo, ¿no crees? Y ya, básicamente eso es lo único que necesitamos para autenticarnos. En caso de falla con submit_form(), la aplicación muere, sin embargo, eso lo intentamos evitar usando eval{}. Si algo dentro del bloque de eval fallara y muriera, eval{} llena una variable especial, $@ con el mensaje de error y ya después nosotros podemos morir o hacer lo que queramos. Bonito, ¿no es así?

Una vez que mandamos la forma, tenemos que asegurarnos que no nos haya mandado mensaje de password erróneo o algo así. Una cosa es que el envío de la forma con sus parámetros sea exitoso o no, y otra diferente que la información de usuario y contraseña sean incorrectos. ¿Cómo saber lo que necesitamos para identificar ésto? Podemos intentar en nuestro Firefox poniendo información errónea en estos campos e intentar loguearse. Al hacer pruebas de este tipo veo que es diferente si el usuario no existe o si la contraseña es errónea.

Si el usuario no existe, la página nos manda un error que dice: "Unknown user name: ''sdfsdfsdf". Please enter user name and password.". Si la contraseña está mal (o sea, el usuario existe, pero la contraseña es errónea), la página despliega: "Sorry, wrong password.". Vamos a ver cómo controlaríamos dichos mensajes:

```
die "Unknown user name!", "n" if $mech->content =~
/Unknown user name/;
die "Wrong password!", "n" if $mech->content =~
/Sorry, wrong password/;
```

¡Qué sencillo! Morimos en caso de que el método content(), que nos regresa la cadena con todo el HTML de la página actual, contiene alguna de ambas cadenas. Código simple para regexs simples. Si no tenemos ninguno de ambos, asumimos que estaremos ya bien logueados.

Ahora, ¿qué es lo que queremos hacer? Digamos que queremos obtener lo que nuestra propia página tiene escrito. Las páginas de los usuarios son simples páginas del wiki con nuestro nombre de usuario, en mi caso, "http://wiki.debian.org/DavidMorenoGarza", o lo que es lo mismo "http://wiki.debian.org/$username":

```
$mech->get("http://wiki.debian.org/$username");
```

Estando en dicha página hay en enlace que dice "Edit", ese es en el que tendríamos que dar clic para entrar a la forma directa de nuestra página:

```
$mech->follow_link(text => 'Edit');
```

Y una vez en la página de edición, nos encontramos con una forma nuevamente, que es básicamente la caja de edición de la página. Al leer el HTML o usando WebDeveloper, nos encontramos con que la forma sí tiene nombre y la caja con el texto de la página se encuentra en un campo llamado "editor-textarea". Pues vamos a obtener ese texto:

```
$mech->form_name('editor');
my $pagetext = $mech->value("editor-textarea");
```

Con eso seleccionamos la forma cuyo nombre es "editor" y con value(), seleccionamos el valor de "editor-textarea" y lo guardamos en la variable $pagetext.

¿Qué más podemos hacer? Lo que queramos, quizás queremos usar ese valor y escribirlo en un archivo, modificarlo y volverlo a guardar, obtener el preview luego de modificarlo, etc, etc. En realidad desde

quí la imaginación es el límite.

Fin

Una regla de oro que he aprendido luego de usar mucho WWW::Mechanize es que casi todo lo que puedas hacer en un navegador convencional, lo puedes hacer también con este módulo. No, no puedes interpretar JavaScript, pero muchas veces no lo necesitas si saber leer bien el HTML de una página y entiendes lo que estás realmente haciendo. A final de cuentas si entiendes perfectamente lo que pasa en una conversación por HTTP quizás ni siquiera necesites Mechanize, pues como un amigo me comentaba, Mechanize es simplemente una rompecabezas armado de muchos módulos alrededor de LWP.

Hazme llegar tus preguntas o comentarios a mi correo, damog@espiral.org.mx. Estaré encantado de saber qué usos le das a Mechanize.

Referencias Útiles

http://search.cpan.org/~petdance/WWW-Mechanize-1.30/lib/WWW/Mechanize.pm

http://search.cpan.org/~gaas/libwww-perl-5.808/lib/LWP/UserAgent.pm

http://wiki.debian.org/

http://wiki.debian.org/UserPreferences

http://search.cpan.org/~petdance/WWW-Mechanize-1.30/lib/WWW/Mechanize/Examples.pod

INSTITUTO POLITÉCNICO NACIONAL

Centro de Innovación y Desarrollo Tecnológico en Cómputo

10º Aniversario

3er Congreso Internacional: Tendencias Tecnológicas en Computación 2007

del 12 al 16 de Noviembre

Conferencias y Talleres

"Unidad Profesional Adolfo López Mateos", Av. Juan de Dios Bátiz s/n, casi esq. Miguel Othón de Mendizábal, Edificio CIDETEC. Colonia Nueva Industrial Vallejo, Delegación Gustavo A. Madero C.P 07700, México D.F

www.ipn.mx www.sep.gob.mx

SECRETARÍA DE EDUCACIÓN PÚBLICA

SEP

Django el framework para perfeccionistas con tiempos límite

Julio Mauricio Acuña Carrillo
julio.acuna@revista-sl.org

Django Reinhardt es considerado uno de los mejores guitarristas de todos los tiempos, tuvo un accidente que lo dejó con dos dedos paralizados, sin embargo con sólo dos dedos podía tocar mejor que muchos y hacer música simplemente maravillosa. Django también es el nombre de un framework para desarrollo web escrito en Python, con el que podemos tener sitios web funcionando en muy poco tiempo; el código simple y elegante de las aplicaciones web nos hace recordar la música de Reinhardt.

Originalmente Django fue creado para funcionar en un ambiente periodístico donde los tiempos límite son cruciales para una publicación periódica, así que los desarrolladores tuvieron que adaptarse al ritmo de los reporteros y editores para poder sacar el contenido a tiempo en el sitio web del periódico.[1]

Adrian Holovaty, Simon Willison y Jacob Kaplan-Moss son el equipo de desarrolladores detrás de Django. En el 2005 después de asistir al PyCon, decidieron liberar el código de Django bajo licencia BSD y ahora muchos voluntarios ayudan en su desarrollo.

Django sigue el modelo MVC (Model View Controller - Modelo Vista Controlador), bueno realmente es MTV (Model Template View - Modelo Plantilla Vista), separando la programación del diseño con lo que podemos tener trabajando tanto a los desarrolladores como a los diseñadores casi simultaneamente ahorrando mucho tiempo, o bien se puede cambiar en cualquier momento la imagen del sitio sin preocuparse porque la aplicación pueda resultar dañada. También se apega al principio DRY (Don't Repeat Yourself - No te repitas a ti mismo) por lo que si se tiene que realizar un cambio en la aplicación, no se tiene que hacer dicho cambio muchas veces.

Para empezar a desarrollar nuestra aplicación web necesitamos Python 2.3 en adelante[2], algún manejador de base de datos (sqlite3, PostgreSQL, MySQL, Oracle), el adaptador para el manejador de base de datos correspondiente (pysqlite, psycopg, MySQLdb, cx_Oracle), un servidor web (Apache, cherokee, lighttp) y finalmente Django, ya sea su versión en desarrollo o la versión oficial.[3] Los detalles de la instalación puede consultarse en el sitio oficial del proyecto.

Creando nuestro proyecto

Django viene con una herramienta que facilita la creación de proyectos y aplicaciones, django-admin.py, veremos como funciona. En el proceso de instalación django-admin.py debió ser copiado (como root) en algún lugar donde pueda ser llamado como por ejemplo en /usr/local/bin/ así que desde la línea de comandos tecleamos:

```
$django-admin.py startproject prueba
```

Esto nos creará un directorio que contiene varios archivos que nos ayudarán en la creación de nuestras aplicaciones y debe lucir algo como esto:

```
__init__.py    manage.py    settings.py    urls.py
```

En este punto ya podemos ver nuestro primer sitio, por supuesto que no tenemos nada todavía mas que una bonita página en colores pastel. Corremos el servidor de desarrollo de Django con el comando

```
$python manage.py runserver
```

Y abrimos nuestro navegador con nuestra dirección local en el puerto 8000 http://127.0.0.1:8000

Creación de nuestra base de datos

Antes de empezar a modificar los archivos hay que poner la base de datos sobre la cual funcionarán nuestras aplicaciones, en lo personal prefiero PostgreSQL y es el gestor de bases de datos que utilizaré

para ejemplificar el uso de Django creando un simple sistema de blogs. Nuestro usuario debe tener permiso de crear bases de datos en nuestro sistema, si no es así nos logeamos como postgres y teclemos en la línea de comandos:

```
$createuser nombre_de_usuario
```

Y le damos permiso a nuestro usuario de crear bases de datos. Una vez con los permisos necesarios creamos nuestra base de datos pruebadb:

```
$createdb pruebadb
```

Ejecutamos la shell de PostgreSQL

```
$psql pruebadb
```

Dentro de la shell tecleamos

```
pruebadb=# ALTER USER usuario WITH password 'contraseñasecreta';
pruebadb=#^D
```

Eso es todo lo que tenemos que hacer con respecto a la base de datos, Django se encargará de escribir las tablas y nosotros no nos tendremos que meter con sentencias SQL, a menos que lo necesitemos, gracias a su ORM(Object-Relational Mapper).

Configuración

Con nuestro editor favorito modificamos settings.py donde encontramos casi al principio del archivo las opciones de nuestra base de datos

```
DATABASE_ENGINE = 'postgresql_psycopg2'
DATABASE_USER = 'usuario'
DATABASE_PASSWORD = 'contraseñasecreta'
```

nuestra zona horaria

```
TIME_ZONE = 'America/Mexico_City'
```

el idioma, Django tiene soporte multilenguaje

```
LANGUAGE_CODE = 'es'
```

y eso es todo por ahora, regresaremos mas tarde a este archivo cuando tengamos nuestra aplicación hecha.

Creando la aplicación

Dentro del directorio prueba tecleamos

```
$django-admin.py startapp blog
```

esto nos creará un directorio como el siguiente

```
__init__.py   models.py   views.py
```

con esto empieza lo divertido, abrimos models.py con nuestro editor y creamos nuestros modelos.


```
class Post(models.Model):
        title = models.CharField(maxlength=50)
        slug = models.SlugField(unique=True,
                                prepopulate_from=('title', ))
        date = models.DateTimeField('published on')
        body = models.TextField('Body', blank=True)

def __str__(self):
        return self.title
class Admin:
        pass
```

este es un modelo de lo más sencillo que nos servirá para darnos cuenta de cómo trabaja Django. La función __str__ y la clase Admin nos servirán mas adelante para manipular nuestras entradas desde la interfaz de administración de Django.

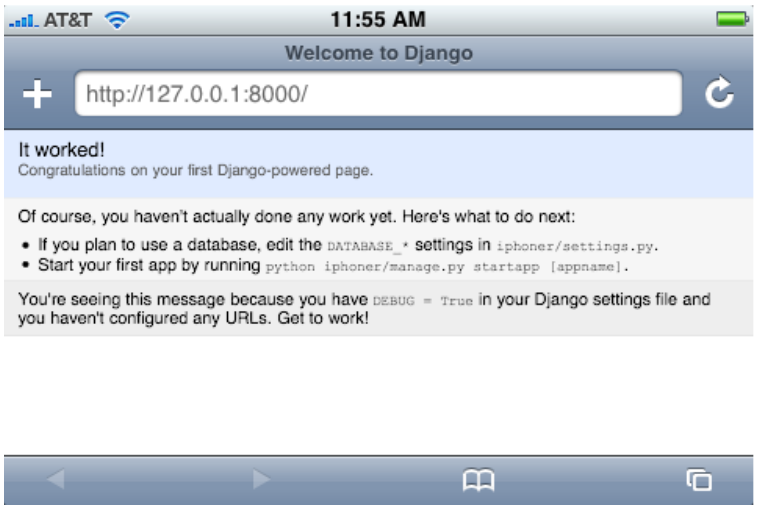
Modificamos una vez mas nuestro archivo settings.py para agregar la aplicación que hemos creado y habilitar la aplicación de administración.

```
INSTALLED_APPS = (
        'django.contrib.auth',
        'django.contrib.contenttypes',
        'django.contrib.sessions',
        'django.contrib.sites',
        'prueba.blog',
        'django.contrib.admin',
)
```

Creamos las tablas en la base de datos al ejecutar

```
$python manage.py syncdb
```

automágicamente ya tenemos nuestro esquema en la base de datos y nos pregunta si queremos definir un superusuario, le decimos que si, mas adelante veremos lo fácil que es manejar nuestros sitios desde la



Servidor de desarrollo de Django corriendo sobre el iPhone de Apple

Diseñando urls

Diseñar urls tal vez no sea lo mas divertido del mundo, pero las ventajas de tener urls bonitas permite a los usuarios encontrar las páginas más facilmente y a los buscadores la indexación.

El diseño de urls se hace a través de expresiones regulares, al editar el archivo urls.py se tendría algo como esto:

```
from django.conf.urls.defaults import *

urlpatterns = patterns('',
        # Example:
        # (r'^myproject/', include('myproject.foo.urls')),

        # Uncomment this for admin:
        (r'^admin/', include('django.contrib.admin.urls')),
        (r'^blog/', include('myproject.blog.urls')),
)
```

Convenientemente hemos incluido otro archivo urls.py dentro de nuestra aplicación blog, esto es para facilitarnos tanto el mantenimiento como el intercambio de aplicaciones entre diferentes proyectos.

```
from django.conf.urls.defaults import *

urlpatterns = patterns('',
        (r'^$', 'blog.views.front'),
        #(r'^(?P<post_id>\d+)/$', ''),
)
```

Views y Templates

Las vistas (views) son funciones que reciben peticiones y devuelven respuestas de cualquier tipo, como puede ser HTML, XML, imágenes, etc. En nuestras urls habíamos llamado a una vista front y lo que queremos que nos muestre es las últimas cinco entradas de nuestro blog, así es como se vería:

```
import django.shortcuts as shortcuts
from models import Post

def front(request):
        entries = Post.objects.all().order_by('-date')[:5]
        return shortcuts.render_to_response("blog.html",
                                dict(entries=entries))
```

Django cuenta con vistas genéricas (generic views) que pueden ser usadas en determinados casos por lo que nos podríamos ahorrar este paso con sólo agregar un diccionario y los patrones correspondientes a urls.py .

A continuación tenemos que crear nuestros templates, uno para el sitio en general y otro para nuestro blog; aunque en esta ocasión sólo tendremos una aplicación, si queremos agregar aplicaciones, el template base.html se puede usar para las demás y debe lucir algo como esto:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">

<head>
        <title>Mi Sitio</title>
        <meta http-equiv="content-type"
         content="text/html; charset=utf-8" />
</head>
<body>
<div id="header">
        <h2>Titulo del Sitio</h2>

</div>
<div id="content">
        {% block content %}
        {% endblock %}
</div>
<div id="footer">
        Powered by Django
</div>

</body>
</html>
```

En este template podremos añadir mas adelante hojas de estilo o JavaScript para que luzca mejor. Incluso podemos crear una aplicación AJAX ligando la aplicación con la vista y utilizando una de las tantas bibliotecas JavaScript que existen como Mochikit, Dojo o Prototype, por mencionar algunas.

Django tiene su propio lenguaje de template, fijemonos en los bloques {% %} dentro de los cuales podemos manipular el comportamiento de nuestra vista con palabras como for o if .

El siguiente template es el de nuestro blog y como podemos ver extiende base.html por lo que ya no tenemos que volver a escribir html.

```
{% extends "base.html" %}
{% block title %}Mi Blog{% endblock %}
{% block content %}
<h2>Entradas Recientes</h2>
{% for post in posts %}
<h3>{{post.title}}</h3>
<h4>{{post.date|date:"j/n/Y H:i"}}</h4>
{{post.body}}
{% endfor %}
{% endblock %}
```

Por supuesto que para que Django sepa dónde encontrar nuestros templates tenemos que indicarle la ruta, una vez mas en settings.py

```
TEMPLATE_DIRS = (
        # Put strings here, like "/home/html/django_templates" or
            #"C:/www/django/templates".
        # Always use forward slashes, even on Windows.
        # Don't forget to use absolute paths, not relative paths.
        "/home/usuario/django_projects/prueba/templates"
)
```

De igual forma tenemos que indicarle dónde encontrar nuestras hojas de estilo, imágenes, JavaScript, etc.

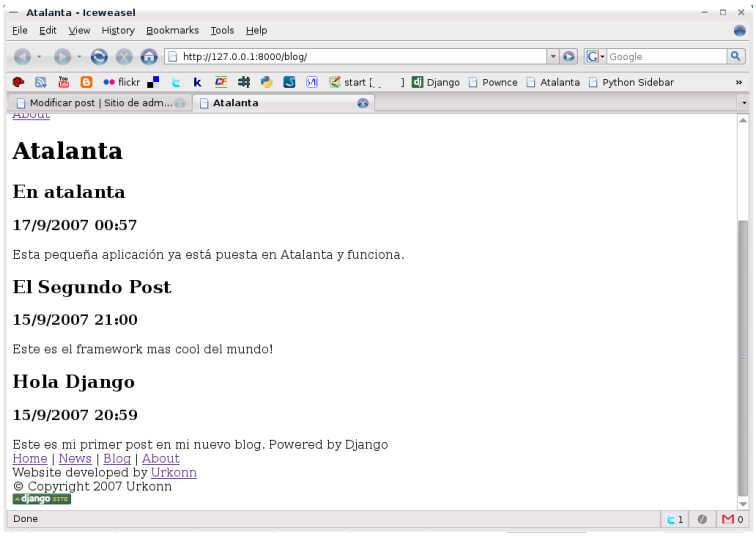
```
MEDIA_ROOT = '/home/usuario/django_projects/prueba/media/'
```

Probando nuestro blog

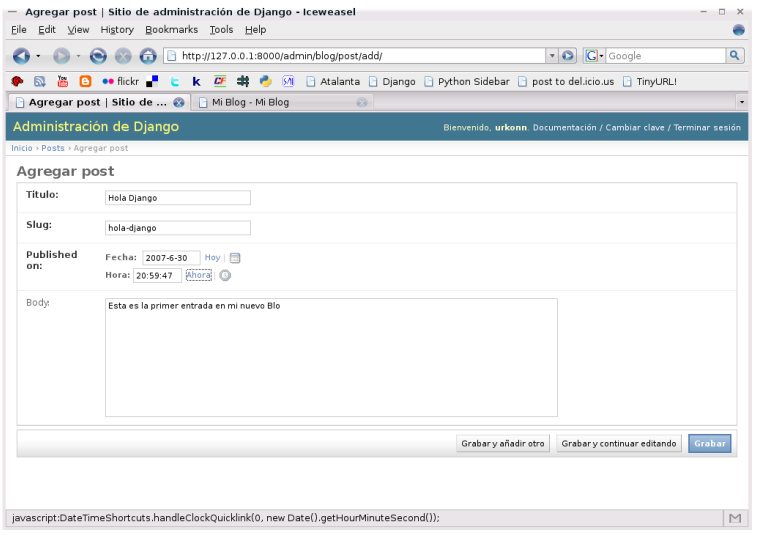
Para comprobar que todo marche bien corremos el servidor de desarrollo de Django.

```
$python manage.py runserver
```

al ir a http://127.0.0.1:8000/admin ya podemos ver la aplicación de administración de Django a la que podemos acceder dándole el nombre de usuario y contraseña que asignamos al crear el



Probando el blog desarrollado en Django



Interfaz de administración de Django

administrador, en http://127.0.0.1:8000/blog vemos nuestro blog como habíamos indicado en urls.py

Despliegue

Para poner nuestra aplicación en producción tenemos la opción de servir nuestro contenido con FastCGI, SCGI o de una forma mas sencilla con mod_python en Apache.

El archivo de configuración httpd.conf de Apache tendría un aspecto como este:

```
<Location "/">
SetHandler python-program
PythonHandler django.core.handlers.modpython
SetEnv DJANGO_SETTINGS_MODULE prueba.settings
PythonPath "['/var/www/django/'] + sys.path"
PythonDebug On
</Location>

<Location "/media">
SetHandler default
</Location>

Alias /media "/var/www/django/media"
```

donde podemos ver que lo que está en el directorio media (imágenes por ejemplo) está servido directamente por Apache sin tener que pasar por Django, esto es recomendable para no gastar recursos del servidor de manera innecesaria.

Para finalizar.

La creación de aplicaciones web con Django es muy rápida y sencilla, claro que este ejemplo es muy sencillo y muchas de las características de este framework no se pudieron ver aquí como lo son las formas y su validación o las pruebas unitarias que son de utilidad para los programadores en la puesta en producción de una aplicación, sin embargo espero haber dado un buen vistazo a este gran framework que ha ganado merecidamente su reputación dentro de la comunidad pythonista y fuera de ella por su rapidez y eficiencia frente a otros frameworks de desarrollo web.

[1] Lawrence Journal-World

[2] En su rama estable, Python 3.0 o Python 3000 es una reimplementación del lenguaje y rompe con la compatibilidad hacia atrás por lo que no se puede usar con Django en este momento.

[3] Hasta hace poco la versión de desarrollo de Django se podía considerar como estable para ambientes de producción pero como forma de alcanzar la versión 1.0 mas rápidamente se ha cambiado esta política. La última versión estable es la 0.96

Referencias:

http://www.djangoproject.com Sitio oficial del proyecto
http://www.djangobook.com Libro sobre Django con licencia libre
http://django.es Django en español

CodeIgniter

Marco Alfonso Ocampo
me@marcoalfonso.net

¡Hola!, así que, ¿está buscando un buen Framework con el cuál desarrollar sus proyectos?, se ha dado cuenta de los grandes beneficios de utilizar uno, pero, ¿no sabe cual elegir?, bien, aquí le presentamos uno de los mejores frameworks para PHP: CodeIgniter.

¿Y porqué CodeIgniter es mejor que otros frameworks?, en realidad esta es una mala pregunta, como la gran mayoría de aplicaciones en el software libre, no es cuál debería elegir como la mejor, sino, cual es la que mejor se adapta a sus necesidades. Una vez establecido esto, CodeIgniter (CI) es para usted si:

• requiere un framework ligero

• requiere que sea rápido

• necesita excelente documentación

• desea una activa comunidad que lo soporte

• necesita que corra con los estándares de hosting disponibles, ningún requerimiento especial

• no quiere tener que aprender reglas de codificación, diseño, flujos de trabajo, etc., para poder usar el framework

• no desea tener que aprender un lenguaje de template nuevo para sus diseños, o peor aún, enseñárselo a sus diseñadores

• usted no requiere tener que crear forzosamente la tripleta MVC, tal vez, sólo vaya a tener VC, o MV. (MVC, Model View Controller)

• sobre todo, no le gusta que haga "magia", que le genere código que no entenderá o no está seguro si lo necesita del todo, le gusta programar sus propias herramientas, tener completo control, y saber que pasa exactamente en cada función, plugin, biblioteca, etc.

Entonces, ¿CI es para usted?... ¿sí?, ¡bien! empecemos a ver como funciona, y para ello desarrollaremos una pequeña aplicación, no, no es otro "blog en 20 minutos", pero si haremos algo igual de sencillo un pequeño ToDo engine ;-).

Empiece bajándose la última versión de CI en http://codeigniter.com, y descomprímalo en el DocumentRoot de su apache (/var/www por ejemplo) puede crear un vhost local para trabajar más cómodamente (puede ver instrucciones aquí: http://tinyurl.com/33ywzg), en nuestro caso, hemos creado el vhost: http://ci.lo, una vez instalado, vamos al browser, abrimos http://ci.lo y veremos una página de bienvenida con alguna información extra, y un link a la documentación oficial que ya viene junto a CI (¿ya había mencionado la excelente documentación de CI?), vamos a hacer algunas configuraciones:

• Edite PATH_A_CI/system/application/config/config.php y ponga la base_url que corresponda a su instalación.

• Edite PATH_A_CI/system/application/config/routes.php y vamos a cambiar el default_controller de 'welcome' a 'todo', y el valor de scaffolding_trigger será 'sca'.

• ¡Es todo!

Necesitamos crear una base de datos, puede crearla con el nombre que prefiera, y dentro de ésta una tabla, como los programadores somos muy ingeniosos se llamara 'todo', con la siguiente estructura:

```
CREATE TABLE todo (
  id serial primary key,
  title text
);
```

Ahora editemos el archivo PATH_AL_CI/system/application/config/database.php y si, como lo imagina, aquí ponemos nuestros datos para que CI se conecte a nuestra BD.

Si en este momento abrimos nuestro browser en http://ci.lo veremos un error 404, ya que aún no hemos creado nuestro controller 'Todo', así que proseguimos con eso, creamos un nuevo archivo en system/application/controllers/todo.php y vamos a meter el siguiente código:

```
<?php
    class Todo extends Controller {

        function Todo()
        {
            parent::Controller();
            $this->load->scaffolding('todo');
            $this->load->database();
            $this->load->helper('url');
        }

        function index()
        {
            $this->db->orderby('id');
            $query = $this->db->get('todo');
            $data['todos'] = $query->result();
            $this->load->view('front', $data);
        }
?>
```

Aquí, declaramos la clase Todo, y en su constructor (function Todo) cargamos todos los plugins, helpers, bibliotecas, etc, que ocupe nuestra aplicación, en nuestro caso, activamos el scaffolding para este controller y para la tabla 'todo', cargamos la biblioteca para el manejo de la BD, y por último el helper 'url' que nos ayudará a hacer algunas redirecciones. La función index, es el método default de la clase, CI maneja las URL's de la siguiente manera: www.tusitio.com/index.php/clase/funcion/argumento1/argumento2/.../ argumentoN, por lo que podremos acceder a nuestra clase con http://ci.lo/index.php/todo o http://ci.lo/index.php/todo/index, sí quisieramos pasarle argumentos a index, sería http://ci.lo/index.php/todo/index/pag/13 y en nuestra función: function index($pag,$id){...}. Ahora veamos el significado de cada línea en la función index:

```
$this->db->orderby('id');
```

Establecemos que en la siguiente query se ordene por el campo id

```
$query = $this->db->get('todo');
```

Hacemos una query a la BD, get('tabla') equivale a un select * from tabla;

```
$data['todos'] = $query->result();
```

Preparamos el array asociativo con los datos que le pasaremos a la vista

```
$this->load->view('front', $data);
```

Invocamos a la vista, y le pasamos $data

Y como se observa, necesitamos tener una vista, llamada 'front' así que creamos el archivo PATH_AL_CI/system/application/views/front.php y dentro ponemos:

```
<html>
<head>
<title>ToDo X</title>
</head>
<body>

<h1>ToDo List:</h1>

<table border=1 cellpadding=10 cellspacing=0>
<tr>
    <th>Hecho</th>
    <th>Descripcion</th>
</tr>

<?php foreach($todos as $item): ?>
<tr>

    <td>
    <a href="/index.php/todo/done/<?=$item-
>id;?>">Hacer!</a>
    </td>

    <td>
<?=$item->title?>
    </td>

</tr>
<?php endforeach;?>

</table>

<p><br />Page rendered in {elapsed_time} seconds</p>

</body>
</html>
```

Y ahora estamos listos para ir a http://ci.lo o http://ci.lo/index.php ó http://ci.lo/index.php/todo o http://ci.lo/index.php/todo/index ;-), ¿porqué tantas url's? bien tiene su explicación, http://ci.lo y http://ci.lo/index.php es cuestión del propio servidor web (apache en nuestro caso), recuerda que editamos routes.php para poner el default_controller? bueno, esto es lo que nos permite acceder a index.php/todo desde index.php y como la función 'index' es el default de todo controller, podemos acceder a index.php/todo/index con sólo index.php/todo, y claro está usted puede controlar todas las rutas, elaborar las suyas propias con regexp, o con alias, dentro del routes.php de CI, refiérase al user_guide para más información sobre ello.

Si ya está viendo nuestra pequeña aplicación en el browser, seguramente observará 0 ToDo items, bien vamos a crear algunos para nuestros fines, ¿recuerda que editamos routes.php para establecer la palabra clave que activa el scaffolding? y ¿recuerda que en nuestro controller "Todo" cargabamos la biblioteca de scaffolding? bien vaya a http://ci.lo/index.php/todo/sca y juegue un poco con la tabla, agregue/edite/borre elementos, cada vez regrese a http://ci.lo/index.php y observe como nuestra aplicación ya está obteniendo datos de la BD y reflejándolos a través de la vista.

Si observa en la vista, tenemos la siguiente línea: <a href="/index.php/todo/done/<?=$item->id;?>">Hacer!, este enlace nos servirá para borrar los items ya realizados, así que como verá estamos apuntando a la función done del controller todo y pasándole el parámetro $item-id, es decir: quedará de la forma: todo/done/1, todo/done/2, etc... bueno, vamos a crear esa función, en nuestro controller todo.php agregamos:

```
function done($id)
{
$this->db->where('id', $id);
```

Establecemos que la siguiente query debe cumplir esta condición where

```
$query = $this->db->delete('todo');
```

Ejecutamos el query de delete sobre la tabla "todo"

```
redirect('');
```

Redireccionamos al usuario al controller default

```
}
```

Así de sencillo!, vamos de nuevo a nuestro browser demos click en "Hacer!" y veremos como nos refrescará la página ya sin elemento borrado. Bueno, ya hemos visto como seleccionar elementos ($this->db->get()) y como borrar ($this->db->delete()), ahora vamos a agregar elementos a la BD, para ello, insertemos un pequeño form a nuestra vista justo debajo del tag </table>:

```
<form action="/index.php/todo/add" method='POST'>
<input type='text' name='item' />
<input type='submit' value='Agregar'>
</form>
```

Como vemos en el action, necesitamos un nuevo método en nuestro controller: /todo/add. Nuevamente agregamos a nuestro controller todo.php lo siguiente:

```
function add()
{
    $item = $this->input->post('item');
    $this->db-
>insert('todo',array('title'=>$item));
    redirect('');
}
```

Y aquí aprendemos un par de cosas nuevas, primero, el $this->db->insert('tabla', array('campo1'=>'valor', 'campo2'=>'valor', ...) donde campo1, campo2, etc, son los nombres de los campos de su BD. Y vemos la función: $this->input->post('item'), donde 'item' es el nombre de mi <input> en la vista, como vemos es extremadamente sencillo acceder a los valores quer recibimos por POST, también existen: $this->input->cookie y $this->input->server(). La ventaja de usar la función $this->input->post() es que está ya trae validación integrada, es decir, equivaldría a tener:

```
if ( ! isset($_POST['something']))
{
$something = FALSE;
}
else
{
$something = $_POST['something'];
}
```

CI ya viene con una serie de filtros contra XSS, puede usar:

```
$data = $this->input->xss_clean($data);
```

Para aplicar los filtros para todas sus entradas de datos. Bueno, ya tenemos un form nuevo en la vista para mandar datos al servidor, ya tenemos el método para insertar esos datos a la base de datos, sólo falta probarlo :-). Con esto terminamos nuestra pequeña aplicación de demostración. Puede encontrar el código usado aquí en http://maop.tk/files.

Así que, como ve, no corre nada en línea de comandos, no necesita tener que declarar modelos a diestra y siniestra, aún y cuando sólo hará una simple aplicación, no necesita aprender un nuevo lenguaje de template, puede usar simple php (aunque en este ejemplo utilizamos la sintáxis alternativa de php), no necesita aprender a manejar un ORM (Object Relational Manager) para manipular su BD, CI le da la flexibilidad de añadir sus propias bibliotecas, si ya las tiene, o trabajar con las que CI le provee, así mismo añadir los que la comunidad genera (http://codeigniter.com/wiki), cualquier función mostrada aquí tiene su respectiva documentación (http://codeigniter.com/user_guide) usted sólo imaginelo, siéntese y escríbalo ;-).

NOTA: Para efectos de demostración, esta aplicación se escribió de manera sencilla pero también insegura (como borrar directamente en el método done), hay algunos aspectos a revisar, como por ejemplo la explotación de una vulnerabilidad como esta, que sería fácilmente "parchada" insertando una validación al método done.

GIMP Galaxy

Fabio Sasso
fabio.sasso@gmail.com

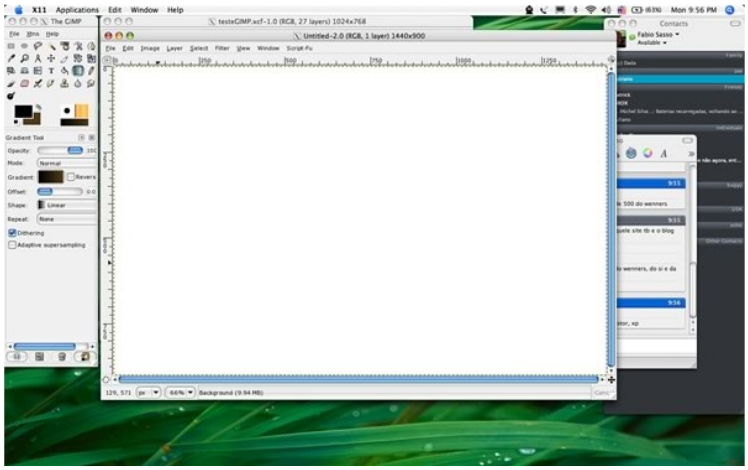
Me encanta el producto estrella de Adobe. Pero eh visto muchos artículos sobre GIMP, El manipulador de imagenes GNU, con el cual la gente crea efectos padrísimos usando esta herramienta. Por esto decidí probarla.

Tengo que expresar que es posible obtener buenos resultados usando GIMP, tiene gran variedad de filtros, algunos ya los quisiera Photoshop. Aunque la interfaz no es muy amigable, es muy parecido a la interfaz del Mac OS X . No quiero revisar , ni comparar software en este momento, yo solo quiero mostrarles mi primera creación usando GIMP, La idea de la imagen fue inspirada en el juego de Mario Galaxy.

1. Abrimos un nuevo documento:

En Archivo > Nuevo

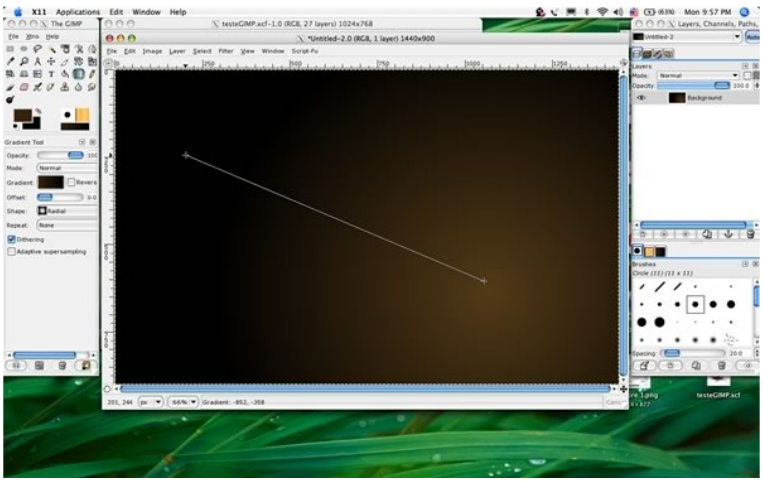
Seleccionamos un tamaño en pixeles 1440x900.



2. Creamos el degradado de Fondo

Seleccionamos la herramienta para mezclar, con las opciones "Forma: Radial" . Y escogemos los colores de fondo negro y otro color para el fondo (esto se seleciona en colores de frente y fondo)

Aplicamos la mezcla en el documento como se muestra en la siguiente imagen.

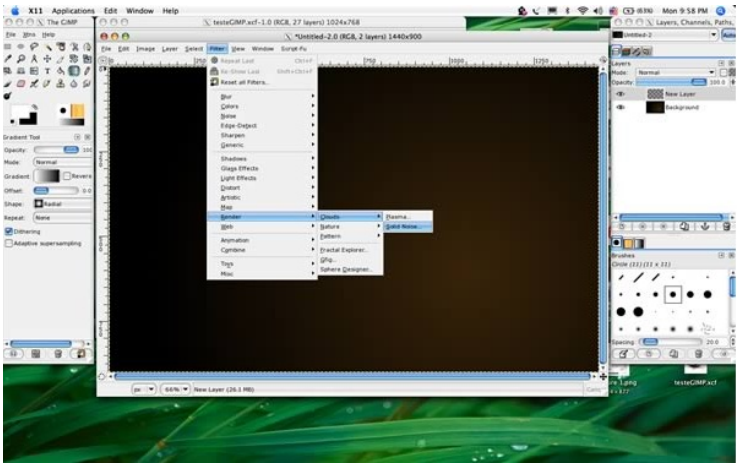


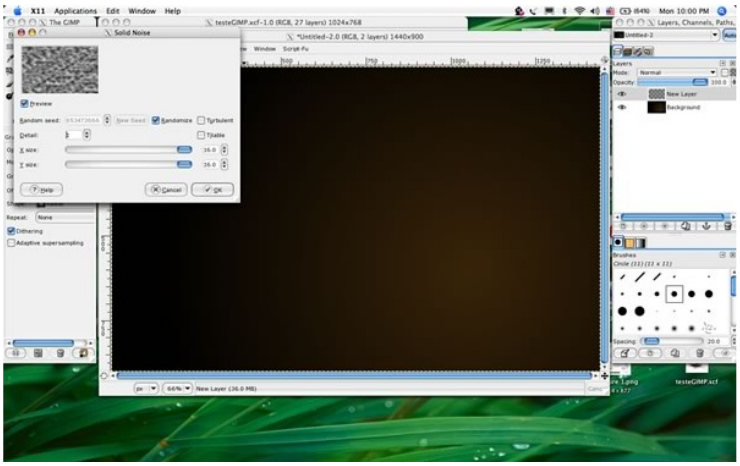
3. Añadimos un poco de humo:

Agregamos una nueva capa, establecemos el primer plano en blanco y nos dirigimos a Filtros > Render > Nubes > Ruido Sólido. Colocamos la capa en Overlay al 44%

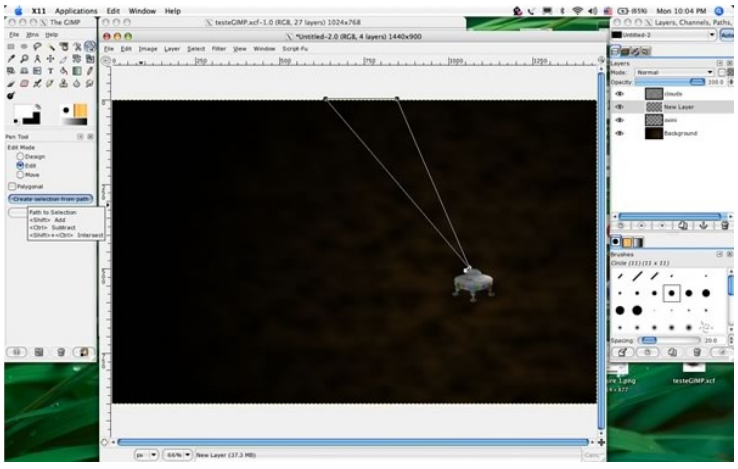
4. Creando los efectos de la luz y los detalles

Para hacer esto, es necesario crear una nueva capa, escoger la herramienta Crear y editar lineas (B) y dibujar un triángulo que será el haz de luz. Después de crear la seleción será necesario presionar click derecho sobre el botón CREAR SELECCIÓN de líneas. Llena la selección usando la herramienta de Relleno (shift+B), usamos





Sistema de renderización

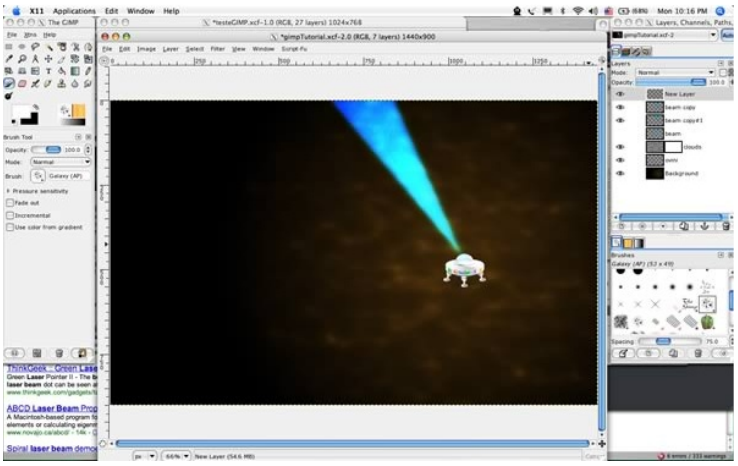


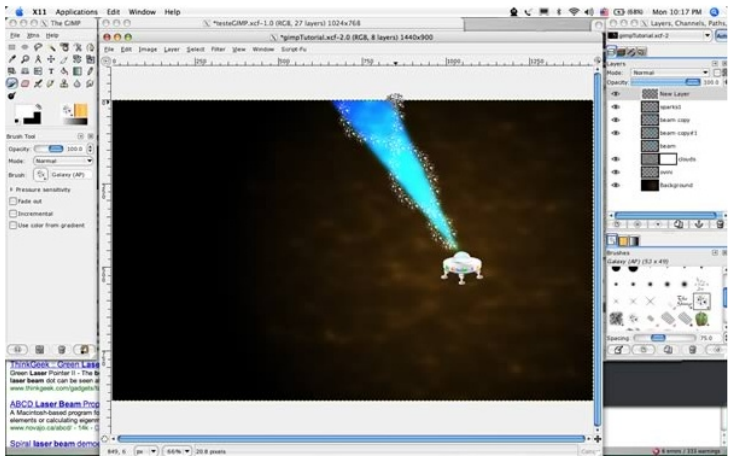
5. Agregamos efecto Gausiano.

Creamos dos copias del haz de luz, y le aplicamos el efecto Gausiano. Coloca la capa en el fondo, y dezplaza las anteriores. Esto debrá de producir el efecto de la LUZ.

6. Creado el efecto de luminosidad.

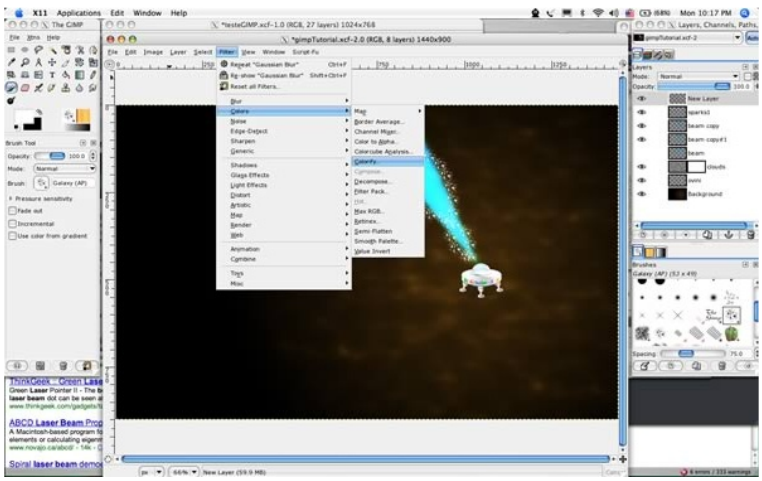
Ahora agregaremos una nueva capa y usando la Herramienta de Pintura Difusa, crearemos los efectos de la luz, bajo el haz de Luz del Objeto.





7. -Agregar colores al efecto de luz.

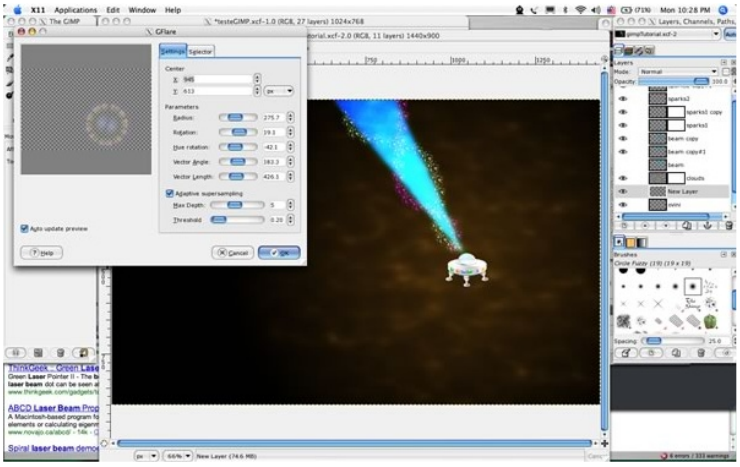
Usando la herramienta de Filtros > Colores > Colorizar cambiaremos los efectos de color de los haces de luz.



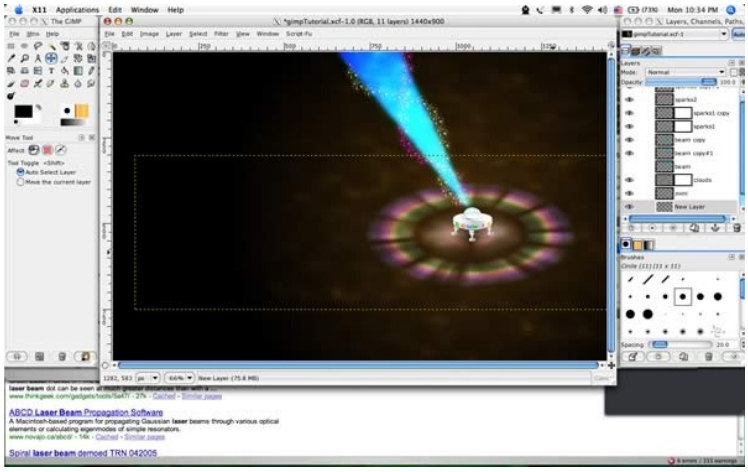
Tip: Duplica las capas y agrega un pequeño haz luminoso para crear un efecto de movimiento.

8. Creando el circulo de luz:

Crea una nueva capa, usa la herramienta Filtros > Efectos de Luz > GFlare, y escoge Planeta Oculto. Intenta algunos valores diferentes hasta obtener el más adecuado al objeto o a lo que queramos expresar.



9. Redimensiona la capa y solo agrega la perspectiva

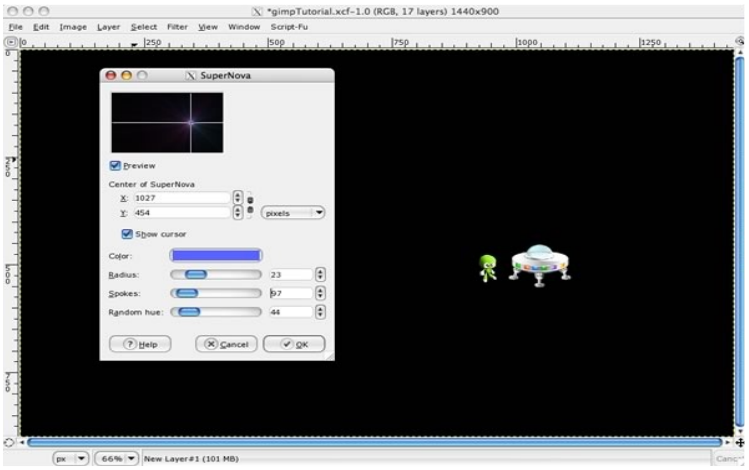


10 - Últimos Efectos.

Duplica el círculo de luz y agrega efectos usando la herramienta Filtros > Blur > Animación. Escribe lineal, y el ángulo a 90 grados.

11. La Supernova.

Agrega una nueva capa, rellenala con negro, aplicale el efecto de Supernova en Filtros > Efectos de luz > Supernova. Otra vez intenta con diferentes valores.

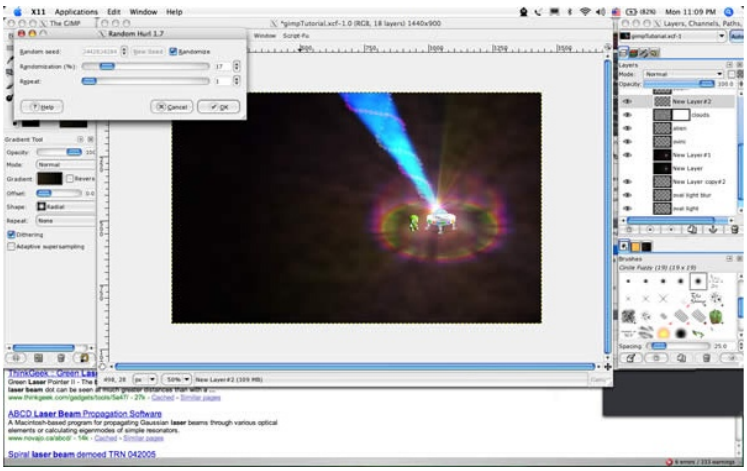


12. Más detalles.

En una nueva capa, utiliza el gradiente en azul, esta vez para utilizarlo en el fondo con tonalidades hacia el negro, crea un gradiente radial de la esquina superior izquierda a la inferior derecha. Establece la capa al modo pantalla con el 65%.

13. Algo de ruido.

Agrega ruido usando Filtros > Ruido > Hurl en la primera capa donde se encuentra el fondo con gradiente de la imagen.



El resultado Final

Si deseas encontrar mas información sobre diseña gráfico usando herramientas libres, visita:

http://abduzeedo.com

Encuentro Nacional Linux 2007

Sonia Sánchez Díaz
sonia.sanchez@revista-sl.org

Los pasados 18,19 y 20 de octubre se realizó el Encuentro Nacional de Software Libre, ENLI 2007, en la ciudad de Puebla, Puebla.

La Benemérita Universidad Autónoma de Puebla (BUAP) fue la Universidad sede del evento, en su tercera emisión este evento demuestra el crecimiento e importancia que está teniendo dentro de la comunidad del Software Libre en nuestro país. Dicho crecimiento se notó en el número de asistentes al evento, quienes desde temprana hora llenaron en su totalidad el espacio físico asignado al área de registro.

Con ponentes internacionales de la talla de Álvaro López y Marcela Tiznado, y nacionales como Gunnar Wolf y Sandino Araico; el congreso tuvo un programa muy diverso para el deleite de todos los gustos. No solamente se trataron temas sobre lenguajes de programación, entre los que figuraron PHP y Python, sino además se escucharon charlas sobre frameworks como Django y RoR.

Área de registro del ENLi

Representantes de distintos grupos de usuarios se dieron cita en el evento, compartiendo un poco de sus conocimientos con todos los asistentes, tal es el caso de Ivan de Jesús Pompa de grupo LIDSOL, el grupo de ICEnetx Team; por parte de Revista-SL: Andrés Vargas, Héctor Leal, Julio Mauricio Acuña demostrando activismo por parte de sus miembros.

Ponentes y organizadores del ENLi

Entre camaradería, como ya es costumbre en este tipo de eventos, la tercera emisión de ENLI fue un éxito, demostrando la madurez que está adquiriendo conforme el paso de experiencias y porque no, del tiempo.

CONSOL

CONGRESO NACIONAL DE SOFTWARE LIBRE

Una Nueva Visión Sobre las Tecnologías

2008

DEL 19 AL 22 DE FEBRERO DEL 2008

TEMAS

- COMUNIDAD, FILOSOFÍA Y NEGOCIOS
- DESARROLLO DE SOFTWARE
- APLICACIONES
- ADMINISTRACIÓN, SEGURIDAD Y REDES
- EDUCACIÓN Y ACADEMIA

www.consol.org.mx

UACM Universidad Autónoma de la Ciudad de México

CFE Comisión Federal de Electricidad

Política digital Innovación Gubernamental

Ingeniero en informática, especializado en inteligencia artificial y doctorado en telecomunicaciones en la Universidad Politécnica de Madrid. Desarrollador del proyecto GNU, líder del proyecto Cherokee y de GNU/MacChanger; además colaborador de proyectos como GNOME. Actualmente ingeniero de Sun Microsystems en donde trabaja en el desarrollo de OpenSolaris y JDS.

RSL. ¿Cómo te llamas?

Alo. Álvaro López Ortega, aunque hay mucha gente que me llama "Alo". Al principio era un nick, pero últimamente me he dado cuenta de que es mucho mas fácil de pronunciar para los angloparlantes.

RSL.- Describe tu vida en el mundo del Software Libre

Alo. Soy un desarrollador que ha tenido la grandísima suerte de hacer de ello su forma de vida.

Trabajo para Sun Microsystems como "Technical Lead", así que mi día a día esta dividido entre la comunidad de OpenSolaris y el resto de comunidades con las que he estado trabajando por los últimos años como GNOME o Cherokee.

Aparte intento aceptar las invitaciones para congresos que recibo, aunque muy a mi pesar no siempre tengo el tiempo suficiente para asistir a todos ellos. Siempre es muy interesante estar en esa clase de eventos para tener interacción real con la comunidad.

RSL. ¿Dónde trabajas y cuál es tu función?

Alo. Trabajo en Sun Microsystems, como Technical Lead dentro de Open Platform Group.

RSL. ¿Es económicamente rentable el Software Libre para SUN?

Alo. Por supuesto. :-)

Sun es una compañía, y como tal su objetivo es ganar dinero (un fin completamente ético). Una de las mejores características de la compañía es que una parte muy importante de su trabajo se basa en el desarrollo de Software Libre.

Los clientes y usuarios valoran muchísimo que la tecnología en la que basan sus redes sean abiertas y libres. Es por ellos que OpenOffice, Java, Solaris, NetBeans, Sun Cluster, etc. es ahora Software Libre.

RSL. ¿Qué es Cherokee?

Alo. Cherokee es un servidor web en el que he estado trabajando los últimos años. Básicamente es un servidor web de altas prestaciones que implementa todas las características que la gran mayoría de gente utiliza hoy en día.

RSL. ¿Por qué hacer Cherokee?

Alo. ¿Por que no? :-) Si no hubiera trabajado en Cherokee ahora la gente no tendría un servidor web muchísimo mas rápido que los que estaban usando hasta entonces.

Cuando comencé a trabajar en Cherokee creía que los servidores web que existían en ese momento era técnicamente mejorables, y estaba seguro de que con algo de trabajo podíamos escribir un nuevo servidor HTTP rápido, estable y con las características necesarias para poderlo utilizar en la mayoría de entornos.

RSL. ¿Dónde nace Cherokee?

Alo. En un viejo 486 que era demasiado lento para servir mi sitio web escrito en PHP a una velocidad decente.

Cherokee es un proyecto que comencé cuando aún estaba estudiando en la universidad, lo cual fue una gran suerte. Como cualquiera que haya salido de la universidad sabe, durante ese periodo tienes muchísimo mas tiempo libre para trabajar en esa clase de proyectos.

RSL. ¿Quiénes hacen Cherokee?

Alo. Un grupo fantástico de hackers. :-) Hay gente que ha contribuido por años, y otra que manda algún parche, mejora o plug-in eventualmente; básicamente lo mismo que en el resto de proyectos libres.

RSL. ¿Cuáles son las espectativas de Cherokee para un futuro?

Alo.- Consolidarse como uno de los principales servidores web libres. Cherokee no pretende ocupar el espacio que hoy tiene Apache. Cherokee intenta satisfacer al 90% de los usuarios, pero somos muy conscientes de que si queremos seguir teniendo un servidor rápido no podemos hacerlo crecer hasta el punto en el que soporte todas las características que soporta Apache.

La idea es: "Si tienes un sitio web 'normal' y quieres alto rendimiento, entonces utiliza Cherokee. Si tu sitio web utiliza tecnología poco habitual es mejor que sigas con tu viejo servidor web". De esta forma el 90% de los usuarios disfrutaran de un servidor mucho mas rápido.

RSL. ¿Cuál es mejor Cherokee o Apache? :-)

Alo. Ninguno es mejor. Cherokee es mas moderno y rápido, mientras que para Apache tienes toda clase de módulos disponibles. Cherokee no esta sujeto a los intereses de ninguna empresa, mientras que en Apache hay muchísimos (y por lo tanto mas dinero).

Son dos proyectos diferentes, con diferentes objetivos, forma de trabajar y en muchas ocasiones incluso audiencia. Cada usuario es quien tiene que decidir que servidor se ajusta mas a sus necesidades. Personalmente - y es por eso por lo que trabajamos en el servidor - creo que Cherokee es mejor opción para la gran mayoría de la gente.

RSL. ¿Qué fue primero el huevo o la gallina? :-)

Alo. El huevo. Esta científicamente demostrado, no es broma :-)

RSL. Muchas gracias por contestar nuestras preguntas, sabemos que tienes
una reunión que asistir, ¿algo que desees agregar?

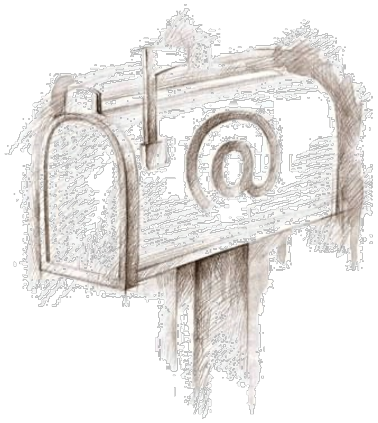
Alo. Gracias a todos los que estáis leyendo esto, porque eso

significa que estais de una forma u otra interesados en el mundo de Software Libre y por lo tanto contribuyendo a su éxito!! :-)

RSL. Bueno eso esto esto estodo amigos, nos leemos la próxima edición.

Si deseas conocer el proyecto Cherokee, encontraras mucha información en el sitio web:

http://www.cherokee-project.com



BuzónSL

Tips, tricks... y mucho mas.

Linux se bloqueo, tecla 'SysRq' o 'Pet Sis'

En ciertas ocasiones Linux se bloquea al punto de congelar la pantalla, el teclado y el mouse, así que no hay forma de cambiar a consola con Ctrl+Alt+Fx o matar las "X" con Ctrl+Alt+BkSp (excepto que puedas entrar mediante ssh).

Antes de darle duro al botón del reset, lee lo que sigue:

Muchos no saben que el kernel de Linux siempre responde a la combinación de "magic keys", a menos que se haya colgado por un grave problema de hardware.

El 'SysRq' (System Request, Petición al Sistema) está incorporado en el kernel Linux desde su versión 2.1.

Pero, ¿dónde está la tecla 'SysRq'?

Es la bien ponderada "Imprimir Pantalla", situada comunmente a un lado de la tecla F12. También conocida como "PrintScreen", "Impr Pant", "PrntScr", "SysRq", etc...

Cuándo se nos cuelgue pulsaremos la siguiente combinación:

Alt + SysRq + R: El kernel responde a esta combinación de teclas descargando todo controlador que tenga tomado el teclado, por ejemplo muy útil cuando se cuelgan las "X" o algún programa en alguna terminal, con esto podremos usar la combinación Control+Alt+Fx para irnos a otra terminal y matar el proceso.

Alt + SysRq + S: Con esta orden el kernel Intenta sincronizar nuestras particiones montadas, para que en caso de un reinicio, se pierda la menor cantidad de datos posibles.

Alt + SysRq + E: El kernel manda la señal de terminar todos los procesos, menos el init, o sea envía un SIGTERM general al sistema que cierre todos los procesos.

Alt + SysRq + I: Se usa en caso que el anterior no funciona, ya que si por alguna razón el sistema no pudo terminar los procesos y no nos lleva a la consola del init, este manda la señal de matar todos los procesos, algo así como un "kill -9" para todo lo que este corriendo (menos el init) SIGKILL.

Alt + SysRq + U: Intentará desmontar todas las particiones montadas para luego montarlas nuevamente en modo de sólo lectura, para seguir previniendo cualquier daño a nuestros ficheros a la hora de reiniciar.

Alt + SysRq + B: reinicio del equipo.

Alt + SysRq + O: apaga el equipo.

Si pusiéramos: Alt+SysRq+RSEIUB se producirían los pasos descritos y se reiniciaría el equipo. Más información en http://en.wikipedia.org/wiki/Magic_SysRq_key

Gracias al usuario "Dragonauta" de http://preguntaslinux.org por proporcionarnos este trick.

Proyecto Wikiaula

Hoy hace 24 años Richard M. Stallman iniciaba un ciclo y nosotros conocíamos el Software Libre.

Hoy el Proyecto wiki-aula quiere continuar este proyecto e iniciar el propio después de un poco mas de un mes de planeación.
y mas de un año de discusiones sobre el tema.

Con esta iniciativa intentamos informar a más personas acerca de lo que es el Software Libre, esto desde una perspectiva distinta a lo que otros han intentado.
Estamos apoyándonos en distintos medios para realizar las conferencias, buscamos incluir la experiencias y las ideas de todos los que gusten participar, es claro que queremos dejar de lado el enfoque tradicional donde la conferencias se realizan en el mismo lugar, queremos experimentar con una forma distinta.

Deseamos llevar las conferencias a distintas sedes cada semana, en distintos estados, inclusive en distintos países.

Por lo anterior se les invita a participar, ya sea como ponentes, o como asistentes, recuerda que es un nuevo estilo de vivir el Software Libre.

Faltan algunas fechas por llenar, algunos eventos no están completos y hacemos lo posible por llenarlos, mas información en http://wiki.wikiaula.org/site/Ciclo_de_conferencias

P.D. Si quieres participar como SEDE solo es necesario que nos asegures un lugar adecuado para el evento, es preferible
que exista la manera de transmitir video en tiempo real, o la grabación en algún formato para su publicación posterior.

Es necesario material, para su entrega en las conferencias, como estampas y discos compactos.

Ya hay gente en distintos estados en México y 2 personas en Venezuela y Chile como parte del proyecto. que están organizando algunas fechas, todavía por confirmar.

Se invita a cualquiera a ayudar. La infraestructura de wiki-aula esta en completa disposición para esto.

****
"Libres en la vida, libres en la mente, libres en el software".
Información de contacto
Jesús Christian Cruz Acono
http://wikiaula.org
jccruza@wikiaula.org
Registred Linux user:403636
Miembro del grupo de usuarios Linux Tlaxcala
Administrador del proyecto Wikiaula

RevistaSL

El software libre hecho revista

Colaboradores SL:

Miguel J. Hernández y López
Carlos Francisco Lermna Résendez
Elizabeth Acosta Gonzaga
Santiago Bonet
David Moreno Garza
Marco Alfonso Ocampo
Fabio Sasso
Alvaro López
"Dragonauta"
Jesús C. Cruz Acono

Proyectos:

HoneyNet México
Cherokee
Columna PL

Instituciones:

CIDETEC - IPN

Eventos:

Congreso CIDETEC
Encuentro Nacional Linux
Congreso Nacional de Software Libre
Latinoware
Simposium de la privatización del conocimiento

Usuarios de Flickr:

Tambako the jaguar cc by-nc
ubyside cc by-nc
stick_xchange_woman cc by-nc
Erik_Silva cc by-nd
comfed1975 cc by-nc
leonardohss cc by-nc
bhikku cc by-nd
JeffMaurone cc by-nd
peoasap cc-by-nc
jacobian cc by-nc

Contacto:
buzon@revista-sl.org

Sitio web:
www.revista-sl.org

~# Guide Lines SL

RevistaSL es una revista creada por la comunidad de Software Libre para la comunidad de Software Libre, por lo que la participación de la comunidad es fundamental.

Es por ello que RevistaSL invita a la comunidad de Software Libre a participar; puedes hacerlo de diferentes formas:

1. Escribe un artículo

La mayor parte del contenido de RevistaSL es aportado por colaboradores que desean compartir sus conocimientos, experiencias, opiniones, etc. Te recomendamos sigas las recomendaciones para que tu artículo quede perfecto y sea publicado de inmediato.

a) Debes liberar tu artículo bajo alguna licencia libre, en caso de no hacerlo adoptará la licencia usada por RevistaSL (Creative Commons 2.5 MX).

b) Existen diferentes tipos de artículos:

*Artículo: Un artículo abarca un tema, software o tecnología en específico. Consta de una breve introducción, desarollo del tema, recomendaciones, etc. y generalmente viene acompañado de capturas de pantalla, gráficos, imágenes, etc. Consta de una longitud entre 450 y 600 palabras
*Tutorial. Los tutoriales son textos explicativos, acompañados de capturas de pantalla, código e instrucciones detalladas sobre cierto tema. No consta de un límite de palabras debido al detalle que se llega a tener en este tipo de artículos.
*Reseñas: Las reseñas refieren a eventos ocurridos como congresos, festivales de instalación, talleres, ciclos de conferencias, etc. Constan de una longitud de entre 150 a 200 palabras.

c) El artículo debe ser enviado en formato Open Document para su mejor manejo.

d) En caso de requerirse el equipo editorial realizará modificaciones mínimas al artículo, si el equipo editorial considera que se requieren modificaciones mayores se pondrá en contacto con el autor para pedirle su opinión.

2. Patrocina

Si ofreces algún tipo de producto o servicio, RevistaSL es un excelenete medio para publicitarlo. Ponte en contacto con nosotros para mas información.

3. Divulga

Cuéntale a todo mundo del proyecto RevistaSL, anímalos a escribir y leernos.