H-Zine

Revista H-Zine Volumen #1 Septiembre 2008 - Hackerss.com

Redes y Comunicaciones

IPv6. *(P23-P26)*
Aspectos de configuración para firewall central *(P21-P22)*

GNU / Linux

IpTables *(P2-P10)*
Clustering en Linux *(P10-P20)*

#1

Miselánea

Concursos
Editorial
Agradecimientos
(P38)

Seguridad Informática

Troyanos *(P27-P28)*

Progamación

Clases en C++ *(P29-P34)*
Evitar SQL Injection. *(P35-P37)*

hackerss.com PORTAL INFORMÁTICO

Iptables *por SoulLost*

¿Qué es Iptables?

Sirve para filtrar paquetes tcp/ip mediante reglas estrictamente establecidas (es pocas palabras es un cortafuegos o firewall).

¿Para qué me demonios me sirve un firewall? Para controlar el tráfico de información o paquetes que pasan por nuestro ordenador e Internet (incluyendo redes locales).

Bien, ¿cómo obtengo un firewall en Linux? Fácil, solo debes cargar los "módulos" necesarios que te permitirán realizar cierta actividad con los paquetes que especifiques (Para información sobre los módulos consultar google.com o la ayuda del kernel), en conjunto con el kernel y el programa "Iptables" (http://www.iptables.org/ http://www.linuximq.net/ http://l7-filter.sf.net/).
Para descargar iptables se puede usar nuestro gestor de paquetes, algunos de los que conozco:

Debian: apt-get install iptables
Gentoo: emerge iptables

Claro también puede optar por compilar el programa a mano o usar paquetes precompilados de ciertas distros.

¡Demonios!, pero ¿cómo levanto esos mentados módulos? Básicamente estos módulos están ya disponibles en la mayoría de las distribuciones GNU/Linux que existen en la actualidad..
Para darlos de alta utilizamos la herramienta modprobe, primero antes que nada hay que mirar que módulos tenemos disponibles con modprobe -l | grep netfilter :

```
UnderHouse linux # modprobe -l | grep netfilter
/lib/modules/2.6.14-gentoo-r5/kernel/net/ipv4/netfilter/ip_conntrack.ko
/lib/modules/2.6.14-gentoo-r5/kernel/net/ipv4/netfilter/iptable_nat.ko
/lib/modules/2.6.14-gentoo-r5/kernel/net/ipv4/netfilter/iptable_mangle.ko
/lib/modules/2.6.14-gentoo-r5/kernel/net/ipv4/netfilter/iptable_filter.ko
/lib/modules/2.6.14-gentoo-r5/kernel/net/ipv4/netfilter/ipt_state.ko
/lib/modules/2.6.14-gentoo-r5/kernel/net/ipv4/netfilter/ipt_limit.ko
/lib/modules/2.6.14-gentoo-r5/kernel/net/ipv4/netfilter/ipt_REDIRECT.ko
/lib/modules/2.6.14-gentoo-r5/kernel/net/ipv4/netfilter/ipt_MASQUERADE.ko
/lib/modules/2.6.14-gentoo-r5/kernel/net/ipv4/netfilter/ipt_LOG.ko
/lib/modules/2.6.14-gentoo-r5/kernel/net/ipv4/netfilter/ip_tables.ko
/lib/modules/2.6.14-gentoo-r5/kernel/net/ipv4/netfilter/ip_nat.ko
```

Se preguntaran qué módulos necesito. Para esto les recomiendo levantar los módulos principales para el funcionamiento:

```
modprobe ip_tables iptable_filter iptable_nat
```

Conforme estemos asignando nuestras reglas de iptables puede que pida algún modulo y mande algún error, solo quedaría montar dicho módulo que nos pida.

¿Y si no tengo los módulos? Buena pregunta, pues habrá que modificar el kernel (tema que no nos corresponde en este momento). Solo como nota hay que entrar a la carpeta del kernel, modificar el kernel y hacer los cambios correspondientes en nuestro gestor de arranque si es necesario, sería algo como esto:

```
cd /usr/src/linux
make menuconfig

##Dejar la siguiente configuración (es la que recomiendo):
```

```
Networking --->
Networking options --->

[*] TCP/IP networking
[*] IP: multicasting
[*] IP: advanced router
[*] Network packet filtering (replaces ipchains) --->
IP: Netfilter Configuration --->
<M> Connection tracking (required for masq/NAT)
<M> IP tables support (required for filtering/masq/NAT)
<M> limit match support
<M> Connection state match support
<M> Packet filtering
<M> LOG target support
<M> Full NAT
<M> MASQUERADE target support
<M> REDIRECT target support
<M> Packet mangling
--------------------------------------------------------------------------------
##############################################

make && make modules_install && make install
```

Conceptos básicos

En primera necesitamos saber las políticas que iptables usa por defecto:

```
UnderHouse soullost # iptables -L
Chain INPUT (policy ACCEPT)
target prot opt source destination

Chain FORWARD (policy ACCEPT)
target prot opt source destination

Chain OUTPUT (policy ACCEPT)
target prot opt source destination
```

Por así definirlas, tenemos tres secciones: INPUT (entrada), OUTPUT (salida) y FORWARD (Redireccionamiento). Estas por defecto están puestas para aceptar todo el trafico (ACCEPT), ¡bien!, pues aquí necesitamos saber bien que es lo que queremos:

- Denegar todo y aceptar solo lo que necesitamos
- Aceptar todo y denegar solo lo que se necesita

En este punto ustedes tendrán que idear una estrategia de filtrado, en mi caso les recomiendo denegar la entrada y el redireccionamiento (INPUT Y FORWARD) y aceptar todo lo que yo mande al exterior (OUTPUT). Para esto, hacemos lo siguiente:

```
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
```

Quedando:

```
UnderHouse soullost # iptables -L
Chain INPUT (policy DROP)
target prot opt source destination

Chain FORWARD (policy DROP)
target prot opt source destination

Chain OUTPUT (policy ACCEPT)
target prot opt source destination
```

Nomenclatura de Iptables

```
iptables -A Agrega una regla
iptables -D Elimina una regla
iptables -P Politica por defecto
iptables -R Reemplazar una ragla
iptables -L Lista las reglas actuales
iptables -F Elimina todas las reglas
```

En las reglas podemos hacer uso de ciertos parámetros que son los siguientes:

- -p Potocolo (tcp,icmp,udp,etc)
- -s Ip fuente
- -d Ip destino
- -o Interfaz de salida (ethx, lo, etc)
- -i Interfaz de entrada (ethx, lo, etc)
- --sport Puerto o rango de puertos fuente
- --dport Puerto o rango de puertos destino
- -j Acción a tomar (ACCEPT o DROP, Aceptar o Denegar)

Denegar todo y aceptar solo lo que necesitamos

Quiero creer que esta es la mejor manera de entender las reglas de iptables y por obvió la opción mas segura para nuestro equipo de computo.

Les muestro mi ip de la tarjeta Ethernet:

```
soullost@UnderHouse ~ $ su -c "ifconfig eth0"
Password:
eth0 Link encap:Ethernet HWaddr 00:0B:6A:8E:94:D8
inet addr:192.168.1.2 Bcast:192.168.1.255 Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:137 errors:0 dropped:0 overruns:0 frame:0
TX packets:137 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:92963 (90.7 Kb) TX bytes:15155 (14.7 Kb)
Interrupt:11 Base address:0x6f00
```

Asignamos políticas por defecto

```
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
```

¿Haciendo ping?

(vale, es como si la tarjeta de red estuviera desconectada).

```
UnderHouse soullost # ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted

--- 192.168.1.1 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2012ms
```

Aceptando ping al exterior

```
UnderHouse soullost # iptables -A OUTPUT -s 192.168.1.2 -d 192.168.1.1 -p icmp -j ACCEPT
```

¿Qué demonios dice lo de arriba? Aceptamos paquetes de salida de la ip 192.168.1.2 (nuestra ip) hacia la ip 192.168.1.1 del protocolo icmp..

¡¡Excelente!! ¿Hacemos el ping?

```
UnderHouse soullost # ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.

--- 192.168.1.1 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1000ms
```

¿Pero qué sucede? Nada, el problema esta en que aceptamos paquetes para enviar pero recuerden que tenemos cualquier paquete de entrada denegado. En este caso el ping usa el protocolo icmp el cual envía un paquete para saber si el host esta conectado a la red (echo-request), el host destino una ves recibido el paquete de pregunta, reenvía otro paquete con el cual contestara a nuestra solicitud (conocido como echo-reply)…

Agregamos la regla faltante

```
UnderHouse soullost # iptables -A INPUT -s 192.168.1.1 -d 192.168.1.2 -p icmp -j ACCEPT
```

Hacemos ping

```
UnderHouse soullost # ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=30 time=0.286 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=30 time=0.328 ms

--- 192.168.1.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1006ms
rtt min/avg/max/mdev = 0.286/0.307/0.328/0.021 ms
```

Miramos nuestras primeras dos reglas para que se familiaricen con estos reportes

```
UnderHouse soullost # iptables -L
Chain INPUT (policy DROP)
target prot opt source destination
ACCEPT icmp -- 192.168.1.1 192.168.1.2

Chain FORWARD (policy DROP)
target prot opt source destination

Chain OUTPUT (policy DROP)
target prot opt source destination
ACCEPT icmp -- 192.168.1.2 192.168.1.1
```

Permitir ping desde el exterior

¿Haciendo telnet?

```
UnderHouse soullost # telnet 192.168.1.1
Trying 192.168.1.1...
```

Malo malo, manos a la obra:

```
UnderHouse soullost # iptables -A OUTPUT -s 192.168.1.2 -d 192.168.1.1 -p tcp --sport 1024:65535 --dport 23 -j ACCEPT
```

Aceptamos paquetes de salida de cualquier puerto privilegiado (1024:65535 son puertos que por default están asignados a ciertos programas o servicios, paquetes enviados fuera de este rango puede ser inseguros) de mi ip hacia el puerto telnet (23) de la ip 192.168.1.1.

```
UnderHouse soullost # iptables -A INPUT -s 192.168.1.1 -d 192.168.1.2 -p tcp --sport 23 --dport 1024:65535 -j ACCEPT
```

Aceptamos paquetes de entrada del puerto 23 de la ip 192.168.1.1 hacia nuestra ip en cualquiera de los puertos privilegiados.

```
UnderHouse soullost # telnet 192.168.1.1 Trying 192.168.1.1...
Connected to 192.168.1.1.
Escape character is '^]'.
SpeedStream Telnet Server


login: as
password:
password: Connection closed by foreign host.
```

Ejercicio: Permitir que entre desde telnet del exterior

¿Obteniendo Internet?

Vamos estoy atrás de un router el cual me comparte Internet a la red local, por ende necesito tener comunicación con él...

```
UnderHouse soullost # iptables -L -n
Chain INPUT (policy DROP)
target prot opt source destination

Chain FORWARD (policy DROP)
target prot opt source destination

Chain OUTPUT (policy ACCEPT)
target prot opt source destination
```

```
####Primero que nada quiero conexión entre ellos asi que habilitamos los paquetes icmp
UnderHouse soullost # iptables -A INPUT -s 192.168.1.1 -i eth0 -p icmp -j ACCEPT
```

Aceptamos paquetes de entrada icmp de la ip 192.168.1.1 (router) hacia la interfaz eth0

```
##Resolviendo mis DNS
UnderHouse soullost # iptables -A INPUT -i eth0 -p udp --sport 53 -j ACCEPT
```

Aceptamos paquetes de entrada del puerto 53 de nuestros servidores de nombres hacia mi interfaz eth0

```
####Accediendo a paginas WEB
UnderHouse soullost # iptables -A INPUT -i eth0 -p tcp --sport 80 -j ACCEPT
```

Aceptamos paquetes de entrada del puerto 80 de cualquier ip hacia mi interfaz eth0

```
###Aceptar cualquier otra conexión del exterior ya establecida o relacionada anteriormente con
nosotros:
iptables -A INPUT -i eth0 -p tcp -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Esto en gran medida puede reemplazar a las otras dos reglas anteriores.
Aquí se presentan parámetros nuevos:
-m --state se utiliza para comparar el tipo de conexión existente, puede obtener los siguientes estados:

- NEW Conexión nueva
- ESTABLISHED Conexión ya establecida anteriormente
- RELATED Conexión relacionada
- INVALID Conexión invalida

Les dejo las reglas realizadas:

```
UnderHouse soullost # iptables -L --line-numbers
Chain INPUT (policy DROP)
num target prot opt source destination
1 ACCEPT icmp -- 192.168.1.1 anywhere
2 ACCEPT tcp -- anywhere anywhere tcp spt:http
3 ACCEPT udp -- anywhere anywhere udp spt:domain
4 ACCEPT tcp -- anywhere anywhere state RELATED,ESTABLISHED

Chain FORWARD (policy DROP)
num target prot opt source destination

Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
```

Si apreciaron con el parámetro --line-numbers enumeramos las reglas, esto es útil al momento de reemplazar una o al querer eliminarlas.

Para eliminar la regla "ACCEPT icmp -- 192.168.1.1 anywhere" hacemos:

```
UnderHouse soullost # iptables -D INPUT 1
```

A estas altura ya sabrán como abrir y cerrar puertos, ¿no?

Funcionar como gateway con iptables

¿Qué es un gateway? También es conocido como encaminador, esto permite reenviar paquetes que pasa por una tarjeta de red hacia una dirección específica.

¿Qué necesitamos? Básicamente disponer de dos tarjetas de red, digo dos por que una será utilizada para obtener Internet y otra para hacer funcionar la red local (LAN)..

Para activar nuestro sistema como gateway hay que permitir el forward entre interfaces de red. Hay dos formas de hacerlo, una temporal y otra permanente (la temporal puede servir para hacer pruebas nada más):

Temporal (programación bash):

echo "1" > /proc/sys/net/ipv4/ip_forward

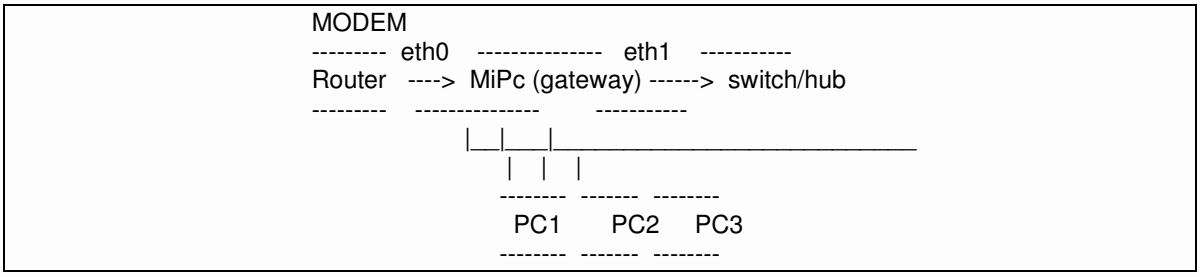
Permanente

Editar /etc/sysctl.conf (es necesario reiniciar):

```
net.ipv4.ip_forward = 1
```

Hay otras maneras claro, pero son las más comunes, las cuales depende en gran medida de la distribución. Igual puede agregar el primer comando al momento de levantar su configuración de su tarjeta al iniciar el sistema.

Aquí hay algo importante de mencionar, para poder compartir Internet deberán tener su MODEM en modo bridge o monopuesto conectado a la pc que funcionará como gateway (supongamos a la interfaz eth0) y la otra tarjeta de red (eth1) conectada a un switch o hub (o a otra pc si solo cuentan con dos). El esquema es algo así:



Un gateway funciona mas o menos como un router..

¿Cómo le digo a mis otros host que mi pc es el gateway al que deben de consultar? Regresamos al mismo tema, depende de la distribución:

En debian y derivados: Editar /etc/sysconfig/network y agregar GATEWAY=ip

En gentoo: Editar /etc/conf.d/net y agregar routes_eth0=("default gw ip")

Temporal: Teclear el siguiente comando (funciona en todas las distros):

```
UnderHouse soullost # route add default gw ip
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
192.168.1.0 * 255.255.255.0 U 0 0 0 eth0
Loopback localhost 255.0.0.0 UG 0 0 0 lo
Default 192.168.1.1 0.0.0.0 UG 0 0 0 eth0
```

Empezando a filtrar paquetes con la tabla NAT (Ejemplos)

Básicamente esta tabla se encarga de dos cosas:

1. Modificar la dirección ip fuente de los paquetes
2. Modificar la dirección ip destino de los paquetes

Por este motivo se divide en dos secciones:

1. SNAT (Source NAT): Cuando se altera la ip fuente del paquete (paquetes de salida)
2. DNAT (Destination NAT): Cuando se altera la ip destino del paquete (paquetes de entrada)

¿Cómo maneja iptables estas tablas?
En primera hay que usar la opción -t NAT.
Seguido de una de estas dos opciones:

1. POSTROUTING: Se usa para SNAT
2. PREROUTING: Se usa para DNAT

Obteniendo Internet en la red local (SNAT)

Supongamos que tenemos salida hacia Internet con una ip fija.

¿Qué necesitamos? Editar el paquete de salida a nombre de la ip que nos ofrece Internet, el ¿por qué? es fácil, cuando un host de la red local haga ping a google.com estará tratando de enviar paquetes con una ip privada, lo cual ni de risa nos permitirá hacer conexión con tal host (google.com), tengan en cuenta que siempre tenemos que tener una ip publica con acceso a Internet.

```
UnderHouse soullost # iptables -t NAT -A POSTROUTING -o eth0 -j SNAT --to 200.33.146.217
```

Que bien, ¿pero qué coño dice?

-t NAT Especificamos que usamos la tabla NAT para hacer forward (que por defecto esta activado)

-A POSTROUTING Indicamos que vamos a modificar la ip fuente del paquete

-o eth0 Es la interfaz de la cual vamos a enviar el paquete (deberá ser la interfaz con conexión a Internet)

-j SNAT --to 200.33.146.217 Señalamos que todos los paquetes que salgan de la interfaz eth0 tendrán la ip 200.33.146.217, la cual tiene acceso a Internet

Supongamos que tenemos salida hacia Internet con una ip dinámica.

La mayoría de los ISP dan a sus cliente acceso a Internet mediante un rango de ip's (ip's dinámicas), por lo tanto es difícil decidir cual será la ip fuente con acceso a la red mundial. Para esto hacemos uso del Enmascaramiento (similar a SNAT) en donde no se especifica la ip fuente, la cual tomará la ip del host emisor el cuál pertenece a otra subred (en este caso la ip con salida a Internet):

```
UnderHouse soullost # iptables -t NAT -A POSTROUTING -o eth0 -j MASQUERADE
```

Servidor Web (DNAT)

Supongamos tenemos la siguiente situación: Internet (200.33.146.217) y un host con servidor WEB dentro de la red local (192.33.20.3)

Con esto deseamos que puedan acceder a nuestro servidor WEB desde cualquier parte del mundo, así que agregamos la siguiente regla al gateway.

```
UnderHouse soullost # iptables -t NAT -A PREROUTING -p tcp --dport 80 -i eth1 -j DNAT --to 192.33.20.3:80
```

- -t NAT Especificamos que usamos la tabla NAT para hacer forward (que por defecto esta activado)
- -A PREROUTING Indicamos que vamos a modificar la ip destino del paquete
- -p tcp --dport 80 Los paquetes con el protocolo tcp y puerto destino 80
- -i eth1 Es la interfaz de la cual vamos recibir los paquetes mencionados arriba (ip publica a Internet)
- -j DNAT --to 192.33.20.3:80 Redireccionamiento al host 192.168.20.3 en el puerto 80

Se lee: Todos aquellos paquetes tcp que vengan de Internet con destino al puerto 80 serán reenviados a nuestro servidor Web que se encuentra en el host y puerto 192.33.20.3:80

Redireccionamiento de puertos localmente (DNAT)

```
UnderHouse soullost # iptables -t NAT -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 113
```

Se lee: Todos los paquetes que se reciban con destino al puerto local 80 del gateway serán redireccionados al puerto 113.

Articulo realizado por: SoulLost
Email: soullost [at] gmail [ot] com

Clustering en Linux *por Dinosaurio*

Bueno aquí les dejo lo de Clustering en Linux.

Las estrategias empresariales nos han llevado a tomar la tecnología como herramienta estratégica y diferenciadora en el mundo de los negocios. Vivimos la era de la Informática y de las Comunicaciones, o lo que Jhon Nasbit en su libro de Mega tendencias 2000 denomino la cuarta Ola. Gracias al rápido y avanzado desarrollo de la Tecnología, la Informática y las comunicaciones, nos encontramos con Plataformas y estructuras adecuadas para soportar de manera adecuada las estrategias y alianzas en el mundo empresarial y de negocios. De eso se trata el Clustering, diría ¡YO!. En tener al alcance herramientas que nos permitan mantener y gestionar información al instante, veraz, oportuna, intuitiva y de alto nivel de valor, que nos permita a los Administradores tomar decisiones estratégicas y oportunas para llevar a cabo la misión de nuestro negocio y por ende alcanzar la visión del mismo. Encontrar ese norte u horizonte que Anhelamos.

Eso es clustering son servicios de alta disponibilidad o de alto rendimiento, con sistemas redundantes e inmediatos, que nos permitan productividad 7x24x30. Administración y monitoreo constante, que un sistema sea respaldado por otro, o sencillamente que afinen su perfomance a un 99.999%, compartir almacenamiento de Datos y Aplicativos, auto soporte, y transparencia total para el usuario, ya que este nunca sabrá si uno de los sistemas entro en modo de respaldo, y levanto servicios e hizo el intercambio nuevamente cuando se aplique la solución del principal, devolviéndole el control. ¡Ni debe saber que paso!. Esto por que no es su objetivo dentro del negocio.

¿Se imaginan que seria de los sistemas de la Bolsa, los Grandes Almacenes Detallistas, Sistemas del Gobierno, Sistemas Financieros, Sistemas Educativos, Sistemas de Transporte, B2B, E-commerce, Web Services, etc. Si no contaran con dicha tecnología?.

Los sistemas de Clustering son un conjunto de Servidores que se unen con un medio de transmisión de alta velocidad, permitiendo compartir recursos, almacenamiento, monitoreo, detección y corrección de fallas de Hardware y Software, deshabilitar, iniciar y reiniciar servicios del NOS, integridad de Datos, balanceo de cargas, tolerancia a fallos, escalabilidad, optimización y el mantenimiento en si del sistema.

Se pueden lograr Cluster para:

- Alta disponibilidad

- Alto rendimiento

- Escalabilidad

- Equilibrio de carga

Qué necesitamos para tener un sistema de Clustering
1. Servidores (como mínimo dos).
2. NOS (Sea software privativo o Free).
3. Que exista una Conexionde Red.
4. Middleware (Sistema que nos permitirá levantar los servicios de Clustering).
5. Protocolos y servicios.
6. Aplicaciones que funcionen en modo Paralelo (no necesariamente).

Bueno, pero creo que no mas preludio. Entremos al tema de Clustering sobre Linux.

En la actualidad existen diversos proyectos y Distros que aplican para la Gestión y Administración de Clustering, vamos a citar el proyecto MOSIX, OPENMOSIX, OPENSSI, Beouwulf, en Red Hat Enterprise, SUSE Enterprise, Debian, etc.

En la siguiente sección utilizaremos como referencia la tecnología Kimberlite de la Compañia Mission Critical Linux, la cual esta bajo licencia GPL.

Descripción del Hardware.

Todos los equipos que componen un cluster de alta disponibilidad (dos equipos en caso de utilizar Kimberlite) han de estar preparados para ejecutar cualquiera los servicios ofertados por el cluster, y además, utilizando información totalmente actualizada. Una manera de conseguir este requerimiento es mediante el uso compartido de los mismos discos de datos por parte de ambos equipos. Cada servicio ofertado por el cluster dispone de su propia partición dentro de estos discos compartidos, y el equipo que tenga arrancado el servicio monta la partición correspondiente. Para evitar la posible corrupción de los datos, por ejemplo por que ambos equipos intenten modificar simultáneamente los datos, el software de alta disponibilidad nos garantiza que sólo uno de los equipos monta a la vez cada partición.

Los discos compartidos también son utilizados por el software de alta disponibilidad para monitorizar el estado de los equipos del cluster. Para ello se utiliza una partición especial denominada partición de quorum. Cada miembro del cluster escribe periódicamente su estado (si está activo, que servicios está ofertando, etc.) en la partición de quorum, y lee la información escrita por el otro miembro del cluster. Si un equipo falla durante cierto tiempo al escribir esta información, posiblemente se trate de un problema con el hardware o software de ese equipo. En este caso, se utiliza un segundo mecanismo (llamado canal de latido o heartbeat) para comprobar si realmente el equipo está fallando. Este canal puede ser un enlace Ethernet, o simplemente un cable serie entre ambos equipos.

Podemos optar por varias configuraciones hardware a la hora de instalar Kimberlite. Estas configuraciones van desde una instalación mínima, con aquellos elementos hardware mínimo que requiere un cluster, hasta una configuración de máxima seguridad, que proporcione redundancia de todos los componentes hardware. El administrador del sistema tiene libertad, según las necesidades y el presupuesto disponible, de seleccionar la configuración más adecuada. En cualquier caso, se recomienda utilizar hardware de calidad, ya que los fallos de hardware suelen ser, en la mayoría de las ocasiones, la causa de la parada de un sistema

Instalación Mínima
La configuración mínima de hardware requerida para instalar un cluster bajo Kimberlite se compone de los siguientes elementos.

Dos equipos, encargados de ejecutar las aplicaciones ofertadas por el cluster,
uno o más discos SCSI, compartidos por ambos equipos del cluster, y que almacenan los datos de las aplicaciones, además de contener las particiones de quorum, y
un canal de comunicación, por ejemplo Ethernet, que se utiliza para comunicar a los equipos con el exterior, y para que se puedan monitorizar entre sí (canal de latido).

Para los discos SCSI compartidos se suele utilizar un sistema RAID (Redundant Array of Independent Disks - array redundante de discos independientes) que garantice la disponibilidad de los datos en caso de fallo de alguno de los discos. Sin embargo, y para ahorrar costes, en lugar de un RAID se puede utilizar simplemente una pila de discos sin ningún tipo de redundancia. Nótese que las capacidades RAID que implementa el núcleo de Linux no se pueden utilizar junto con Kimberlite.

Otra manera de ahorrar algunos costes consiste en utilizar un conmutador (switch) KVM que con un solo monitor, teclado y ratón nos permita gestionar ambos equipos.

Instalación Avanzada.
La configuración hardware descrita en el apartado anterior presenta numerosos puntos débiles que pueden hacer que nuestro sistema falle, incluso utilizando un software de alta disponibilidad. Por ejemplo, si alguno de los discos SCSI compartidos falla, los servicios alojados en dicho disco no se pueden seguir ofertando.

Una configuración hardware más avanzada, tolerante a fallos, incluiría los siguientes elementos:
Discos SCSI compartidos con soporte RAID por hardware, que proporcione un sistema de redundancia de datos, dobles controladoras SCSI, que permitan al bus seguir funcionando incluso en el caso de que una de las controladoras falle, single-initiator SCSI, que nos aísle ambos segmentos del bus, evitando posibles interferencias, una tarjeta de red Ethernet adicional, usada como canal de latido, un cable serie conectado a ambos equipos, usado como canal serie de latido, sistemas de alimentación ininterrumpida UPS, para prevenir posibles cortes en el suministro eléctrico, dos power switch, que permitan a cada equipo del cluster reiniciar al otro equipo mediante hardware, de esta manera se garantiza la integridad de los datos, ya que tenemos la seguridad de que sólo uno de los equipos monta a la vez cada partición del disco compartido.

Consideraciones sobre el bus SCSI
Como hemos visto, los discos compartidos SCSI son el elemento central de toda instalación de alta disponibilidad bajo Kimberlite. Por ello, debemos poner especial énfasis en cómo configurar correctamente estos elementos. Aspectos que hay que tener en cuenta a la hora de configurar el bus SCSI son:

1. los discos compartidos han de estar conectados al segmento LVD de las controladoras
SCSI
2. Ambos extremos del bus han de estar terminados físicamente, es decir, tenemos

que terminar el bus por hardware usando terminadores, no podemos utilizar la opción de terminar automáticamente el bus que viene con el software de configuración de la tarjeta
3. Los equipos al arrancar no deben reiniciar el bus, por lo que las tarjetas deben tener esta opción desactivada.
4. todos los dispositivos del bus han de tener un identificador único, tanto los discos como las propias controladoras.
5. no es conveniente que los discos del sistema se encuentren en el mismo bus SCSI que los discos de datos compartidos.
6. y en cualquier caso, los discos compartidos han de tener el mismo nombre hardware
(¿¿/dev/sd??) en ambos equipos.

Este último punto es importante. Lo que nos está diciendo es que hay que tener cuidado cuando se utilizan dos equipos con distinto hardware en un mismo cluster. Si por ejemplo uno de los equipos utiliza una controladora SCSI para el sistema operativo, y el otro una controladora IDE, es posible que en primer equipo los discos de datos figuren bajo el nombre /dev/sdb, mientras que en el segundo figuren como /dev/sda. En este caso, el software Kimberlite no funcionaría correctamente. Lo que se recomienda es usar discos IDE para el sistema operativo y el software de aplicación, y dejar los discos SCSI para los datos compartidos.

Instalación y Configuración del Software

Una vez nos hemos decidido por una configuración hardware, comprado los equipos, e instalado físicamente los mismos, podemos continuar con la instalación del software en el cluster. Primero tenemos que instalar y configurar una distribución Linux en ambos equipos del cluster, a continuación hay que instalar y configurar el software Kimberlite, y finalmente, hay que definir y configurar los distintos servicios ofertados por el cluster. En las siguientes secciones se describen en detalle estos pasos.

Configuración del Sistema Operativo

El primer paso para instalar y configurar el software de alta disponibilidad en nuestro cluster es instalar una distribución estándar Linux. Para este artículo, hemos probado a instalar Kimberlite 2.0.0 bajo la distribución RedHat Linux 7.3, no habiendo encontrado ningún tipo de problemas de incompatibilidades software. Si se quiere utilizar otra versión de la distribución RedHat Linux, o incluso otra distribución de Linux distinta, se recomienda antes consultar la página web de Mission Critical Linux para evitar potenciales problemas de incompatibilidad.

Finalizada la instalación de la distribución Linux, resulta conveniente revisar que se han reconocido y configurado correctamente todos los dispositivos hardware de nuestros equipos. Por ejemplo, podemos revisar la lista de mensajes producidos durante el arranque del sistema (con el comando dmesg), o ejecutando cat sobre fichero /proc/devices y comprobando que existen entradas para los dispositivos SCSI (sd), y puertos serie (ttyS). También es importante que exista una entrada para los dispositivos de tipo raw (raw).

Nótese que para que nuestro cluster funcione de manera adecuada, resulta fundamental tener correctamente configuradas las tarjetas de red en ambos equipos. Antes de continuar, se recomienda revisar que ambos equipos tienen una dirección IP

correcta, una máscara de subred correcta (esto es importante), y que disponemos de un servidor DNS (o de algún otro mecanismo) que nos asigne correctamente los nombres de máquinas (hostnames) a las correspondientes direcciones IP. En caso de utilizar una segunda tarjeta de red como canal de latido, se recomienda configurar esta tarjeta utilizando las direcciones IP de alguna de las subredes privadas existentes (por ejemplo 192.168.0.0), y utilizar el fichero /etc/hosts para asignar los nombres de máquina a éstas direcciones privadas. Para no complicar la configuración de los servicios de red, se recomienda que la tarjeta de red externa está asociada al dispositivo /dev/eth0 y la interna a /dev/eth1.

Kimberlite basa su funcionamiento en la tecnología IP-Aliasing (para más información sobre IP-Aliasing y cómo funcionan las direcciones IP flotantes consultar la ``IP-Aliasing Mini-HOWTO'' por Harish Pillay, disponible en la página web de The Linux Documentation Project [4]). En la serie 2.4 del núcleo de Linux se puede activar el soporte de direcciones IP flotantes en tiempo de compilación, de hecho, el núcleo incluido con la distribución RedHat 7.3 Linux trae este soporte ya activado. Aun así, si queremos estar completamente seguros de que nuestro núcleo soporta direcciones IP flotantes, podemos realizar un sencillo experimento, podemos configurar nuestra tarjeta de red eth0 para que responda a una dirección IP flotante. Para ello probar a ejecutar:

```
ifconfig eth0:0 192.168.0.1 \
netmask 255.255.255.
0
```

Cuando tengamos instalado y configurado el sistema operativo, tenemos que crear las particiones en los discos SCSI compartidos. Hay que crear una partición por cada uno de los servicios ofertados, y dos particiones adicionales de unos 10Mb cada una para los servicios de quórum. A continuación hay que crear los correspondientes sistemas de ficheros en las nuevas particiones, excepto en las particiones de quórum, que son tratadas como dispositivos de tipo raw. Para que las particiones de qurum sean vistas como dispositivos raw, tenemos que utilizar el comando raw, por ejemplo:

```
raw /dev/raw/raw1 /dev/sda1
raw /dev/raw/raw2 /dev/sda2
```

Podemos comprobar que los dispositivos raw han sido asignados correctamente ejecutando: raw -qa.

Finalmente tenemos que modificar el fichero /etc/sysconfig/rawdevices para que cuando arranquemos nuestro equipo, se configuren correctamente los dispositivos raw recién creados. Añadir al fichero las siguientes líneas:

```
/dev/raw/raw1 /dev/sda1
/dev/raw/raw2 /dev/sda2
```

Opcionalmente también podemos modificar la configuración del sistema de arranque

(grub o lilo) para reducir el tiempo que tardan las máquinas en arrancar (opción boot time). De esta manera conseguimos que las máquinas recuperen los servicios perdidos más rápidamente.

Instalación de Kimberlite

Para instalar el cluster de alta disponibilidad descrito en este artículo hemos utilizado la versión 2.0.0 de Kimberlite. La mejor manera de instalar el software Kimberlite es compilándolo desde el código fuente original. Para ello nos descargamos el fichero kimberlite-2.0.0.tar.gz de la página web de Mission Critical Linux, y lo compilamos e instalamos (en ambas máquinas del cluster) utilizando el conocido procedimiento:

```
./configure
make
make install
```

Este procedimiento nos instala el software bajo el directorio /opt/cluster, y los ficheros de configuración bajo /etc/opt/cluster. Como es habitual, si queremos cambiar el directorio de destino del software, en lugar de la orden configure, deberemos utilizar:

./configure	*prefix=/otro/directorio*

Una vez compilado e instalado el software, tenemos que crear un nuevo grupo de usuarios llamado cluster, y hacer que todos los ficheros y subdirectorios que cuelgan de /opt/cluster pertenezcan a este nuevo grupo. Por ejemplo ejecutar:

chgrp	*R cluster /opt/cluster*

A continuación tenemos que configurar correctamente en ambas máquinas el software recién instalado. La configuración de Kimberlite reside en el fichero /etc/opt/cluster/cluster.conf, sin embargo, se desaconseja editar a mano este fichero. El problema reside en que ambos equipos han de estar configurados exactamente de la misma manera, y que una copia de dicha configuración ha de ser almacenada en las particiones de quórum.

Para simplificar el proceso de configuración, Kimberlite nos proporciona la utilidad member_config, residente en el directorio /opt/cluster/bin. Esta utilidad nos hará algunas preguntas sobre cómo queremos configurar nuestro cluster, y realizará la configuración por nosotros. Conviene tener preparada la información que se nos solicita antes de ejecutar la utilidad. La información solicitada es: hostname y dirección IP de ambos equipos del cluster, dispositivos raw asociados a las particiones de quórum, número de canales de latido utilizados y tipo (Ethernet o serie), y puerto serie asociado al power switch (en caso de existir).

Cuando hayamos terminado de configurar el software, sólo nos queda inicializar las particiones de quórum. Para ello utilizamos la orden diskutil -I. Y una vez inicializadas las particiones, podemos arrancar el software de cluster.

El procedimiento de configuración descrito en los párrafos anteriores puede resultar un poco confuso, sobre todo la primera vez que se realiza. Por ello, para simplificar la tarea del administrador de sistemas, se propone seguir el siguiente procedimiento paso a paso:

En el servidor 1:

ejecutar la utilidad member_config y responder a las preguntas que nos haga, inicializar la partición de quórum,

diskutil	*I*

comprobar que la partición ha sido inicializada correctamente,

diskutil	*t*
diskutil	*p*

arrancar los servicios de cluster,

service cluster start

En el servidor 2:

ejecutar el la utilidad member_config indicándole que lea la configuración ya existente en la partición de quórum,

member_config	init=/dev/raw/raw1

arrancar los servicios de cluster,

service cluster start

Para comprobar que los servicios de cluster están funcionando correctamente, podemos ejecutar la orden clustat -i 2, que nos mostraró en pantalla, cada 2 segundos, el estado de nuestro cluster.

Configuración de los Servicios

El objetivo final de instalar un cluster bajo Kimberlite es poder configurar un conjunto de servicios que ofrezcan datos y aplicaciones en alta disponibilidad. Para crear estos servicios tenemos que indicar los recursos que utilizan las aplicaciones y las propiedades de las mismas. Durante el proceso de configuración, la información que se nos solicitará es el nombre del servicio, el script de arranque y parada, la partición o particiones (en los discos compartidos) donde residen los datos, los correspondientes

puntos de montaje, y la máquina en la que se va a ejecutar por defecto la aplicación. A un servicio también se le puede asignar una dirección IP (flotante) que proporcione a los clientes un acceso transparente.

Kimberlite nos permite configurar nuestro cluster de dos maneras distintas, utilizando una configuración de tipo activo-activo, o una configuración de tipo activo-espera. En la configuración de tipo activo-activo ambos miembros del cluster ofertan servicios y ejecutan aplicaciones, mientras que en la configuración de tipo activo-espera sólo uno de los miembros del cluster ejecuta las aplicaciones, mientras que el segundo equipo queda en espera, preparado para arrancar el software en caso de que el otro equipo falle.

Son muchas las aplicaciones que se pueden utilizar bajo Kimberlite, por ejemplo un servidor de páginas web, una base de datos, un servidor de colas de impresión, etc. En esta sección veremos cómo instalar y configurar una de estas aplicaciones; vamos a ver cómo instalar un servidor web de alta disponibilidad.

Instalación de un servidor web Apache

Como ya hemos indicado, antes de definir el servicio tenemos que identificar algunos elementos de configuración. Para este caso concreto, estos elementos son:

nombre del servicio: httpd
script de arranque y parada: /etc/init.d/httpd
partición de datos en el disco compartido: /dev/sda1
punto de montaje: /var/www
máquina en la que va a ejecutar el servicio por defecto: servidor1
dirección IP flotante: 192.168.1.101

Una vez tenemos identificada toda la información necesaria para definir el servicio podemos empezar con la instalación y configuración del mismo. El primer paso consiste en instalar y configurar el software Apache en ambos equipos del cluster. Podemos comprobar que el servidor web está funcionando correctamente accediendo a la página de prueba que se instala por defecto. Cuando está instalado el servidor web en ambos equipos, paramos el servicio (por ejemplo con service httpd stop), y realizamos los siguientes pasos:

En el servidor 1:

montar la partición correspondiente a Apache en el disco compartido en un punto de montaje temporal,

```
mount /dev/sda1 /mnt/tmp
```

copiar los datos de nuestra página web al disco compartido,

cp	*r /var/www/* /mnt/tmp*

borrar los datos originales

rm	*rf /var/www*

desmontar la partición y volverla a montar en su destino final,

```
umount /mnt/tmp
mount /dev/sda1 /var/www
```

comprobar que el servidor web sigue funcionando, y

```
service httpd start
```

parar el servicio y desmontar la partición.

En el servidor 2:

borrar los datos originales,
montar la partición en el directorio destino, y comprobar que el servicio funciona correctamente, y
parar el servicio y desmontar la partición.

En el servidor 1:

arrancar la herramienta de configuración de Kimberlite,

/opt/cluster/bin/cluadmin
añadir un nuevo servicio, y

cluadmin> service add
responder a las preguntas que se nos hacen, con la información sobre la configuración que habíamos preparado.
Cuando definimos un nuevo servicio con la herramienta cluadmin, Kimberlite añade la información de configuración del servicio a una base de datos de configuración del cluster, residente en el disco compartido y accesible para ambos equipos. Y a partir de ese momento podemos empezar a trabajar con el nuevo servicio.

Incompatibilidad con las scripts de arranque y parada de RedHat 7.3.
Existe un problema de incompatibilidad entre Kimberlite y las scripts de arranque y parada de servicios instalados por RedHat 7.3. El problema reside en que estas scripts consideran como un error intentar parar un servicio que ya está parado. Por otro lado, cuando Kimberlite arranca un servicio, primero se asegura de que está correctamente parado invocando explícitamente una orden de parada. Al recibir un código de error por parte del script, Kimberlite asume que no se pudo realizar correctamente la parada, y por tanto, que el servicio no puede ser arrancado.

Para solucionar este problema, se puede crear un script puente que nos filtre esta condición de error. Por ejemplo, el script puente en caso del servicio httpd podría ser:

```
#!/bin/sh
/etc/init.d/httpd $1
exit 0
```

Mantenimiento
Para las tareas de mantenimiento del cluster, Kimberlite proporciona la herramienta en modo texto cluadmin. Con esta herramienta podemos administrar de una manera fácil y cómoda tanto el software de alta disponibilidad, como los servicios que hayan sido definidos.

La herramienta cluadmin nos permite arrancar y parar el software Kimberlite, crear una copia de seguridad de la configuración actual del cluster, y recuperar esta copia en caso de que algo haya salido mal. En cuanto a los servicios, cluadmin nos permite parar o arrancar cualquier servicio, modificar su configuración, o incluso eliminarlo. Además de lo anterior, cluadmin también nos permite monitorizar el estado del cluster.

Junto a la herramienta cluadmin, Kimberlite también nos proporciona una herramienta gráfica, basada en páginas web, para monitorizar de manera dinámica el estado de nuestro cluster. Sin embargo, en este artículo no vamos a entrar en los detalles de cómo configurar y usar esta herramienta gráfica (para más información, consultar la documentación que acompaña a Kimberlite).

También es interesante destacar la posibilidad que ofrece Kimberlite de migrar los servicios desde un equipo del cluster hacia el otro, de una manera totalmente transparente para los usuarios. Esto nos permitiría, por ejemplo, actualizar el sistema operativo, o incluso el hardware, de los equipos del cluster sin necesidad de parar los servicios, y sin que exista el peligro de que se corrompan los datos.

Conclusión
Kimberlite es uno de los mejores paquetes software de alta disponibilidad, distribuidos bajo licencia GPL, que existen actualmente para Linux. Es cierto que presenta algunas limitaciones importantes, como por ejemplo que no se garantice la alta disponibilidad de servicios individuales (sólo se garantiza la disponibilidad para el caso de fallos de las máquinas). Aun así, Kimberlite resulta adecuado para muchas aplicaciones, tales como servidores de base de datos, o servidores de World Wide Web con contenidos dinámicos.

Por último, es interesante mencionar que Kimberlite puede ser utilizado junto con otros paquetes software de alta disponibilidad para Linux, como el proyecto Linux Virtual Server [5], para proporcionar servicios de red de alta disponibilidad que sean escalables.

Articulo realizado por: Dinosaurio
Email: Dino[at]hackerss[dot]com

Aspectos de configuración para instalar un firewall central, Reglas Básicas, ***Por Alluz***

La implementación de un firewall central dentro de una corporación debe ir de la mano con las políticas de seguridad existentes, el objetivo del negocio y el modelo de seguridad informática. Sin embargo, existen algunos lineamientos básicos que pueden ser considerados o no por el administrador de seguridad al momento de implementar un firewall.

De antemano, se debe tener en cuenta que un firewall solo es un control para mitigar riesgos de seguridad informática; la efectividad de estos controles depende de las reglas que se tengan en el mismo. Algunos aspectos y reglas básicas recomendadas son las siguientes:

Documentar y definir clara y precisamente de zonas militarizadas, desmilitarizadas, Vlans y clientes VIP.
Cuando se encuentran bien definidas las zonas en una red de datos, ajustar las reglas de firewall a las necesidades del negocio es una tarea más sencilla. La documentación debe incluir el nombre de cada segmento y sus reglas básicas. Ejemplo:

La vlan1 es de usuarios corporativos; estos usuarios deben tener acceso a Internet y a los servidores internos. No deben tener acceso a los ambientes de pruebas (vlan2)

Enumerar, sustentar y documentar los servicios requeridos entre zonas, vlans e Internet (mapa de servicios requeridos)
Un firewall central no puede cerrar un puerto en un host, pero puede controlar el acceso desde otras redes a determinado número de puertos. Algunos servicios deben ser visibles entre vlans (por ejemplo ftp, ssh, smtp, http, smtp etc), pero los patrones de servicios comunes no son suficientes para constituir reglas de firewall exitosas, por lo que se hace necesario establecer cuales son los servicios requeridos de acuerdo con las necesidades de la corporación y entre cuales redes. Un mapa de servicios puede ayudar a generar reglas más granulares.

Denegar el acceso a todos los puertos excepto los documentados. Generar un log cuando ocurra un evento de acceso denegado
Al existir un mapa de servicios se pueden evitar reglas de firewall que dificulte el normal desempeño de la organización y al mismo tiempo se puede llegar a implementar controles en niveles de detalle. Por ejemplo, si entre la vlan1 y la vlan2 los equipos Windows deben ?verse?, se debe permitir el tráfico en los puertos 135, 139 y 445; una regla granular puede ser ?Permitir el acceso ssh a los hosts xyz desde el host zyx?

Deshabilitar la traducción de direcciones (NAT) hacia Internet
El uso de NAT hacia Internet puede permitir riesgos como las shells inversas y uso de proxys no autorizados; Al deshabilitar NAT, se obliga a los clientes a pasar por el Proxy corporativo.

Permitir paquetes ICMP del tipo 3, 8, 11 (Destino inaccesible, eco, tiempo excedido) entre Vlans

El uso de esta regla permite el diagnostico de estado de hosts dentro de la red interna. Bajo ciertas circunstancias, controles más efectivos deben considerarse. Alternativamente, algunos firewalls permiten controlar la cantidad de tráfico ICMP que puede permitirse.

Articulo realizado por: Marcos Julian Gutierrez A
Email: marcosjgutierrez[at]gmail[dot]com

IPv6, *por Spawn*

Descripción

IPv6, también llamado IPng (next generation internet protocol) es la nueva versión del conocido protocolo IP, el cual viene a reemplazar la versión anterior (IPv4) de forma GRADUAL. El principal motivo de la creación de esta versión es ampliar el número de direcciones IP, que las que se tenían
pensadas en la versión 4. IPv4, con la que trabajamos actualmente es una dirección de 32 bits formada por 4 grupos de 8 bits cada uno, con esta versión de ip se tenían como máximo 2^32 direcciones IP (4,294,967,296) y los creadores de ésta pues creían que con esto era suficiente para siempre :-D, pero actualmente se están saturando el número de direcciones y pues en poco tiempo ya no quedarán direcciones para más equipos montados a la red.
En cambio, el formato de dirección de IPv6 es de 128 bits, la cual está
formada por 8 grupos de 16 bits cada uno (cada grupo de 16 bits en valor hexadecimal), con esto tenemos que el total de direcciones ip es 2^128 (3.402823669 e38, o sea sobre 1,000 sixtillones), ahora si podremos estar seguro que las direcciones IP nos durarán un ¡BUEN TIEMPO! ¿HE? No solo para ordenadores, ya que también podremos conectar otros dispositivos a la red tales como PDA's, teléfonos, lavadoras, neveras... todo en Internet.

He aquí una comparación de IPv4 con IPv6

IPv4

8 8 8 8
xxx.xxx.xxx.xxx
---
4 grupos de 8 bits
total de direcciones= 4,294,967,296
dirección de 32 bits

IPv6

16 16 16 16 16 16 16 16
xxxx.xxxx.xxxx.xxxx.xxxx.xxxx.xxxx.xxxx
---
8 grupos de 16 bits (en valor hexadecimal)
total de direcciones= 3.402823669 e38
dirección de 128 bits

Características Principales

Como el tema lo dice: unas de las principales características de este protocolo, aparte de la ya mencionada antes (que es la principal) sobre la expansión en el número de direcciones IP son las siguientes:

SEGURIDAD: Este aspecto es muy importante, y pues en esta versión de ip se introdujo IPSec el cual permite autenticación y encriptación del protocolo base, y pues con esto todas las aplicaciónes de red funcionando con este protocolo se beneficiarán de esta nueva característica. OJO hee, ya que IPSec ES UN REQUERIMIENTO DE IPv6.

SIMPLIFICACION DE ENCABEZADO (HEADER): Pues esta sección también se ve afectada con el cambio de versiones, ya que en los encabezados de la versión anterior habían muchos campos muy deficientes o inservibles, y pues en la versión 6 simplemente se QUITARON esos campos o se hacen opcionales.

AUTOCONFIGURACION: IPv6 cuenta con un sistema, mmmm llamémosle PLUG & PLAY, ya que al iniciar un equipo en red, pues este detecta automáticamente una configuración, como si tuviera un sistema DHCP. Pues aqui lo que hace es que INTENTA AUTOCONFIGURARSE y encontrar la salida o el gateway a internet, a esta nueva funcionalidad se le es llamada ROUTING DISCOVERY (Juar ! por fin encuentra el camino a internet por si solo XD).

Nomenclatura

Recordemos pues que las direcciones ip versión 4 eran de este y único tipo, fáciles de recordar, con única forma de escribir y en valor decimal:

xxx.xxx.xxx.xxx ejemplo: 200.65.30.139

Ahora, esto se hace un poco más complejo heee :(, ya que hay varios, digámosle tipos de simplificaciones de la dirección. El tipo de una ip versión 6 es la siguiente:

xxxx.xxxx.xxxx.xxxx.xxxx.xxxx.xxxx.xxxx ejemplo: FEDC:BA98:7654:3210:FEDC:BA98:7654:3210

Como mencioné antes, existen algunas formas de simplificar direcciones de la nueva generación, por ejemplo, si en una dirección hay 0000 (ceros) se puede simplificar con un simple 0 ya que no es necesario escribir todos los ceros a la izquierda. Cuando 2 o más campos consecutivos están llenos de 0's
(ceros) se puede simplificar de la siguiente manera:

...:0000:0000:0000:... ---> ...:0:0:0:... ---> ....::.... (todos los campos con 0 = ::)

esta regla se usa siempre y cuando los campos con valor = 0 sean consecutivos y SOLO UNA VEZ SE PUEDE USAR :: EN UNA DIRECCION IP. Por ejemplo:

2002:0450:0009:0010:0000:0000:0000:0071 ---> 2002:450:9:10::71

Y cuando nos encontremos con direcciones en donde hayan campos consecutivos con 0's en diferentes partes, SOLAMENTE SE PODRá? REPRESENTAR UN CONJUNTO DE 0's CON :: UNA SOLA VEZ, Y LOS DEMÁS CAMPOS CON 0's SE TIENEN QUE REPRESENTAR Así :0:0:0... bla bla bla. Por ejemplo, la dirección:

FFFF:0:0:0:FFFF:0:0:0 solamente se podrá comprimir en FFFF::FFFF:0:0:0 ó en FFFF:0:0:0:FFFF:: pero NUNCA EN FFFF::FFFF:: ya que como dije antes, solamente

un conjunto de 0's puede ser compreso en una dirección y los demás serán representados. (Repásenle varias veces ;-))

Como última aclaración sobre simplificación de ceros, les digo que cuando vean :: en una dirección, para conocer la dirección COMPLETA simplemente se llenan los campos faltantes con 0's hasta completar la dirección de 8 campos.

Ejemplos:

```
FFFF::12 ---> FFFF:0000:0000:0000:0000:0000:0000:0012
::5 ---> 0000:0000:0000:0000:0000:0000:0000:0005
1080::8:800:200C:417A ---> 1080:0000:0000:0000:0008:0800:200C:417A
1::1 ---> 0001:0000:0000:0000:0000:0000:0000:0001
```

Otros Aspectos.

Bueno, ya hablé sobre algo fundamental de la versión que nos espera en el futuro (muy cercano) y pues cabe aclarar, que esta nueva implantación de ips nos brindará muchas ventajas.. Prestaciones y un sin fin de cosas buenas más :), pero no significa que sea 100 % excelente. También, los desarrolladores de Internet Protocol han hecho algo para mejorar el trabajo de los administradores de redes, ya que el routeo de paquetes es mucho mejor, los routers tendrán menos sobrecarga, en fin, una red corriendo
bajo IPv6 será más fácil de administrar pero no significa que el administrador estará en el PARAÍSO :P, ya que el trabajo del administrador es administrar y por ahí leí que todo lo que se debe administrar, a veces tiende a ser DESADMINISTRADO, entonces, no por implantar IPv6 en una red signifique que estamos 100% estables y seguros hee.

Un ejemplo en la actualidad de una falla de seguridad (no un bug) con IPv6 es en un firewall de WinXP (era de suponerse XD), ya que los chicos Microsoft® (el tio Billy y sus discípulos) admiten que "...las funciones Internet Connection Firewall y Basic Firewall de Windows XP y Windows Server 2003 sólo están en condiciones de bloquear paquetes de datos IPv4, por lo que el tráfico IPv6 pueden entrar SIN RESTRICCIONES AL SISTEMA.
El software cortafuegos de Microsoft contiene, en principio, una cantidad relativamente limitada de funciones de bloqueo. Por ello, no sustituye cabalmente una solución completa de cortafuegos y está dirigida básicamente a los usuarios de Windows, que de otra forma no instalarían una aplicación
de ese tipo.
Considerando que Internet Connection Firewall y Basic Firewall no protegen contra el nuevo protocolo de Internet, IPv6, recomendamos a los usuarios de Windows XP y Windows Server 2003 instalar una aplicación cortafuegos específica, en caso de necesitarla...", así que OJO.

Finalmente aclaro que técnicamente ya se puede hacer uso de esta nueva versión, pero lo que hace falta es $$$, y pues las principales organizaciones o empresas que pueden hacer este cambio de versiones son por ejemplo las Universidades con $$$ o las que en verdad estén interesadas en implantar esta nueva tecnología, otras, principalmente las ISP's (T3LM3X en México), grandes empresas ($$$), y bla bla bla.

En la actualidad mucho países han anunciado su cambio de IP, están listos para migrar a IPv6, aunque como se dijo desde un princio el cambio de versiones es de forma GRADUAL (asi lentamente, no de un día para otro ;-))
y hoy en día, el primer país del mundo en cambiarse de IPv4 a IPv6 fue Taiwán, Uorale !.

Links

http://www.ipv6.org/ - Web Oficial de IPv6
http://www.ipv6.unam.mx/ - Página del proyecto IPv6 de la UNAM
http://www.rau.edu.uy/ipv6/queesipv6.htm - Descripciones fundamentales
http://www.lugro.org.ar/biblioteca/articulos/ipv6.html - Instalar IPv6 en Red Hat 7.3
http://www.redes-linux.com/manuales.php?catId=IPv6 - Mucha info sobre IPv6 para Linux
http://imasd.elmundo.es/imasd/ipv6/cfg/router-freebsd.html - Config. de router FreeBSD en IPv6
http://www.rfc-editor.org/cgi-bin/rfcsearc...all&filefmt=txt - Todos los RFC's sobre IPv6

Bueno saludos espero que les sirva de algo este texto.

Articulo realizado por: Spawn
Email: spawn[at]hackersfire[dot]com

Troyanos, *por GerCat.*

¿Que son?

Un troyano es reconocido como malware *(MALicious softWARE)*, es decir, un programa para fines maléficos, pero estos a diferencia de los virus (la mayoría) no tiene porque ser destructivos (esto depende de quien lo utilice y para que fin).
Un troyano es entonces, un programa que se conecta a ordenadores (de igual manera que un navegador se conecta por el puerto 80, el troyano también lo hace, aunque sea por otros puertos). Suelen estar formados por un cliente y un servidor.
Es cierto también que existen muchos troyanos, incluso pueden llegar a ser indetectables, así es, pueden llegar a no ser reconocidos como maliciosos por los antivirus.

¿Para que sirven?

Tienen más de una función, puede ser utilizado para el espionaje, para hacer daño, o simplemente para conseguir el control remoto de un PC no alcanzable físicamente.
Actualmente, hay muchos troyanos, entre ellos, muchos solo sirven para hacer bromas (la típica de que se habrá la bandeja del cd-rom, o también la de mover el ratón). Y otros utilizados en el espionaje, que pueden incluir keyloggers, capturador o vista de la pantalla en tiempo real, etc.. Y por ultimo funciones como eliminar archivos...

¿Cómo funcionan?

Los troyanos, son simples programa que se conectan a otros ordenadores a través de un puerto determinados*.
Como hemos dicho, está formado por dos partes, el cliente y el servidor. El servidor se instala en el ordenador que queremos tomar el control, y desde el cliente controlamos las diferentes opciones que pueda incorporar el programa (algunas ya han sido dichas en el apartado

¿Para qué sirven?

La funcion de password, sirve para si as infectado a tu victima, otro que tenga el mismo clinte no se pueda aprovechar de eso. Me explicare, digamos que yo infecto a mi victima, entonces qualquier persona puede escanear a esta victima i ver los puertos que parezca sospechosos. De esta manera poniendo password, solo puede utilizarse los que sepan la password.
Hay muchas funciones, suelen ser parecidas en la mayoria de troyanos.

Métodos de instalación.

Existen varios métodos, pero el objetivo es lograr tener instalado el servidor en el ordenador víctima, para ello se puede hacer de varias formas.
Engañando a la víctima para que lo ejecute, o también teniendo acceso físico al ordenador, ejecutándolo tu mismo.
También se puede enviar a través de un e-mail, o con alguna vulnerabilidad poder llegar a instalar el servidor en el PC victima.
Es cuestión de aprovechar también la ignorancia de tu victima (¿Como hago eso? Ingeniería Social).

¿Vale la pena? ¿Es ético? ¿Es útil?

Atención: En esta parte quiero dejar claro que es totalmente subjetiva, pueden no estar de acuerdo.

Vale la pena? Hacer que, ¿violar la intimidad de alguien, escuchar/leer/ver las cosas de los demás? ¿Es correcto espiar a alguien?, ustedes juzguen. Usar troyanos es una cosa relativamente fácil, cuanto menos conocimientos tenga la víctima más fácil será (esto está relacionado con la ingeniería social).

Es ético? Como ya se ha dicho, el programa en si no tiene porque dañar a nadie, pero ya estas violando la intimidad teniendo acceso remoto.

Es útil? Puede ser muy útil, en esto no estoy en desacuerdo, puede servirte para mucho, o para bien poco. Tú decides para que lo usas.

Links:

Puertos que suelen utilizar los Troyanos

¿Donde encuentro Troyanos?

Articulo realizado por: GerCat

Email: cat_aok[at]gmail[dot]com

Clases en C++, *Por Marioly.*

Antes que nada especifico que es una ligera introducción y ya iré escribiendo mas sobre el tema ya que es un poco extenso o mas que para un solo post, escribo primero la base para gente que no tiene experiencia en C++ pero ya iré escribiendo cosas mas "técnicas" (amenos que tenga algo mas que hacer o se me olvide).

Clases en C++

C++ es un lenguaje orientado a objetos, su gran virtud es usar este prototipo para hacer software más depurado y reutilizable. La base de la programación orientada a objetos es el uso de clases, una clase es un tipo de datos abstracto con la cual podemos definir objetos, un objeto es un tipo especial de variable con atributos propios y funciones miembro (métodos).

Los que tengan experiencia con estructuras no les será muy ajeno el concepto, una clase es muy parecida a las estructuras solo con funciones propias y un sistema de protección.

Escribiendo la primera clase.

```
class Ejemplo
{
        public:
                Void diHola();
};
```

Siempre los ejemplos visuales son los mejores, una clase se define con la palabra reservada class, en el ejemplo definimos una clase con el nombre de **Ejemplo** con una función miembro publica (que explicaremos en breve) llama **diHola**, el cuerpo de la clase se introduce entre llaves y termina con punto y coma (;)

Metodos Públicos y Privados.

C++ cuenta con las palabras reservadas Public, private y protected que forman el sistema de protección, al establecer un método como publico puede ser accesado desde cualquier lugar del programa, este es el nivel de seguridad mas bajo.
Private determina que solo puede ser usado dentro de la misma clase (este es el nivel mas restrictivo)
Proteced: indica que solo puede ser usado por la clase y clases padre

Esto a grandes rasgos ya que lo e explicado en otros posts, si no quedo claro pueden leer mas acá:
http://www.conclase.net/c/curso/index.php?cap=032

Para establecer métodos como públicos solo basta con indicar la palabra public seguida de dos puntos y añadir los métodos y atributos bajo de eso, todo lo que se encuentre bajo de eso será establecido como publico hasta encontrar otro nivel de protección por ejemplo:

```
class Ejemplo
{
    public:
        void viHola();
        void otroMetodoPublic();
    private:
        void primerMetodoPrivate()
};
```

Constructores para inicialización.

Un constructor es un método especial que es llamado apenas se crea una nueva instancia de la clase, los constructores deben llevar el mismo nombre de la clase y no debe devolver ningún valor ni debe tener un tipo, ahora añadiremos un constructor a la clase:

```
class Ejemplo
{
    public:
        Ejemplo();
        void diHola();
};
```

Ahora el constructor es llamado cada que inicializamos un objeto Ejemplo. Si no es establecido un constructor el compilador creara uno vació por defecto, aunque es buena practica de programación añadir un constructor a nuestras clases.

Una clase como dijimos es un prototipo de datos , y puede funcionar tan bien como cualquier definido por el estándar como double, las clases nos pueden servir para crear nuestras librerías de tipos y exportarlas a cualquier programa que los pudiese utilizar.

A continuación pondremos un cuerpo a nuestro método diHola ya que lo escrito es solo el prototipo del método, los prototipos son declaraciones que indican el tipo y parámetros del método (o función)

```
void Ejemplo::diHola()
{
  cout << "Hola!";
}
```

Al nombre del método se antepone el nombre de la clase a la que pertenece y su tipo, aquí agregamos un método llamado “diHola” del tipo void, un metodo void no retorna ningún valor, hace su proceso y muere.

Ahora con esto añadiremos un cuerpo al constructor:

```
class Ejemplo
{
	public:
		Ejemplo();
		void diHola();
};
```

Ejemplo:

```
Ejemplo()
{
	Cout << "Inizializando objeto Ejemplo" << endl;
}

Ejemplo::diHola()
{
	cout << “Hola!☐?;
}
```

Inicializando un objeto

Ahora declararemos un objeto de Ejemplo, esto se logra anteponiendo el tipo:

```
Ejemplo objetoEjemplo;
```

En este ejemplo se llama al constructor que especificamos y al compilar deberíamos poder ver el mensaje del constructor, ahora el objeto pude acceder al método di Hola, así que seria correcto lo siguiente:

```
Ejemplo objetoEjemplo;
ObjetoEjemplo.diHola();
```

Para acceder a los métodos podemos usar el operador de acceso**.** (punto), con los métodos y atributos públicos, recordemos que los métodos privados son solo accesibles desde la misma clase.

Error común de inicialización

Para crear un objeto podemos usar distintas formas, como:

```
Ejemplo ejemplo;
Ejemplo ejemplo = Ejemplo(); //Llama implícita al constructor

Pero lo siguiente:
Ejemplo ejemplo();
```

Es incorrecto, de esta forma se esta creando un prototipo de función que retorna un objeto Ejemplo, y la sintaxis es totalmente correcta pero claro no hará lo que deseamos.

Un ejemplo practico

Escribí un pequeño programita para uso didáctico, algo simple y rápido que abajo explico:

```
#include <iostream>

using namespace std;

class Prestamo
{
      public:
            double prestamo, pagos_mes, intereses, taza_interes;
            int meses;
            Prestamo( );
            void fijar_pagos_mes(double prestamo, int meses);
            void actualizar_cuenta();
            void mostrar_informacion(ostream& flujo, int num_mes);
      private:
                  double obtener_interes();
};

int main (int argc, char *argv[])
{
    Prestamo pr;

    cout << "Cantidad del prestamo:" << endl;
    cin >> pr.prestamo;
    cout << "a meses:" << endl;
    cin >> pr.meses;

    pr.fijar_pagos_mes(pr.prestamo, pr.meses);

    cout << "Mes\tPago\tIntereses\tRestante" << endl;

    for( int i = 0; i < pr.meses; i++ )
    {
            pr.actualizar_cuenta();
            pr.mostrar_informacion(cout, (i+1));
    }

    return 0;
}

Prestamo::Prestamo()
{
     taza_interes = 0.10;
}

void Prestamo::fijar_pagos_mes(double prestamo, int meses)
{
     pagos_mes = (prestamo/meses);
```

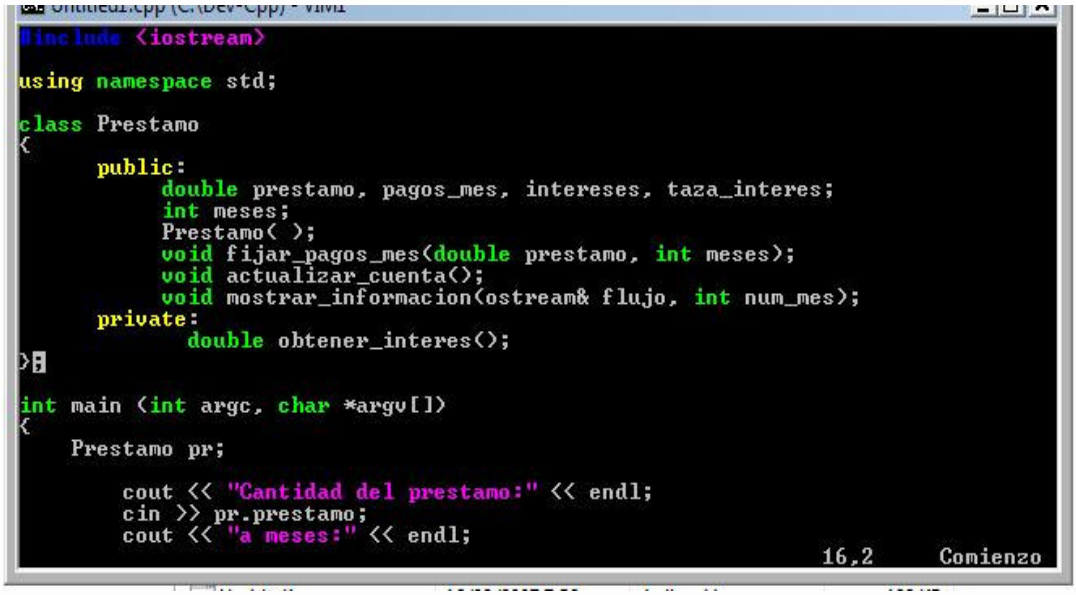
```
}

void Prestamo::actualizar_cuenta()
{
     intereses = obtener_interes();
     prestamo = (prestamo-pagos_mes)+intereses;
}

double Prestamo::obtener_interes()
{
       return (prestamo*taza_interes)/12;
}

void Prestamo::mostrar_informacion(ostream& flujo, int num_mes)
{
     flujo.setf(ios::fixed);
     flujo.setf(ios::showpoint);
     flujo.precision(3);
     flujo << num_mes << "\t" << (double)pagos_mes << "\t" << intereses <<
"\t" << prestamo << endl;
}
```

Algo simple que es para estudio, es un programita para calcular los pagos de un préstamo ficticio (muy ficticio), en el ejemplo, el usuario puede indicar el dinero a recibir y los meses a pagar, con una taza de interés anual del 10% sobre el pago insoluto de la deuda, es decir, los intereses serian: (insoluto*taza_interes)/12 (taza anual), la salida con un préstamo de 20.000 a 20 meses seria algo como:

```
Cantidad:
20.000
a meses:
20
Mes                    Pago          Intereses          Restante
 1                     1.000         0.167              19.167
 2                     1.000         0.160              18.320
.......
```

Algo trivial y se pudo hacer en un bucle pero es para que se entienda el concepto ^_^ , (se que faltan cosas en el calculo de la cuenta pero esto no es foro de contabilidad u.u)

Me decidí hoy por escribir sobre C++ por que se me hace un buen lenguaje para empezar a familiarizarse con la programación orientada a objetos, es bastante limpio y claro y sin tantas complicaciones como empezar con Java o C# , además que existen diversas herramientas para desarrollo (sin mencionar que linux tiene su compilador por defecto en muchas distros), en mi caso para edición de archivos utilizo Vim (http://www.vim.org/download.php en windows y Linux



```
#include <iostream>

using namespace std;

class Prestamo
{
      public:
           double prestamo, pagos_mes, intereses, taza_interes;
           int meses;
           Prestamo( );
           void fijar_pagos_mes(double prestamo, int meses);
           void actualizar_cuenta();
           void mostrar_informacion(ostream& flujo, int num_mes);
      private:
             double obtener_interes();
};

int main (int argc, char *argv[])
{
    Prestamo pr;

        cout << "Cantidad del prestamo:" << endl;
        cin >> pr.prestamo;
        cout << "a meses:" << endl;
                                                    16,2      Comienzo
```

(y si, lo que sea en consola de ms-dos sux, pero vim se ve lindo :P), y para compilar gcc y en windows el port de gcc pero si les parece mejor el desarrollo con IDE pueden utilizar Dev C++ http://www.bloodshed.net/devcpp.html]http://www.bloodshed.net/devcpp.html en windows ó Anjuta en lin (http://anjuta.sourceforge.net/)

Saludos.

Articulo realizado por: Marioly Garza Lozano
Email: marioly[at]Hackerss[dot]com

Evitar SQL Injection, Un poco de seguridad. *Por Ironic.*

Buenas... bueno, este artículo básicamente era para comentar un poco algunas técnicas de programación, referentes a la seguridad en nuestras aplicaciones (tanto web como de windows).

¿Qué es SQL Injection?:

SQL Injection es una técnica que se utiliza para introducir o modificar las llamadas a una base de datos explotando la falta de validacion de las variables.

¿Qué me refiero con esto?

Cuando uno hace una llamada a una base de datos, para verificar un login, para obtener ciertos productos o información por lo general le envía condiciones que se tienen que cumplir. Al Utilizar SQL Injection lo que uno hace es forzar esa llamada para modificar esas condiciones o para ejecutar otra llamada a la base de datos.

Por Ejemplo, el Clásico:

```
SELECT
        *
FROM
        Usuarios /* Acá, sin el where nos trae todos los vendedores...*/
WHERE
        /*Nos traera todos los vendedores con el nombre Jose  y la contraseña
proveida*/
        Usuario='Jose' AND Password='josecapo'
```

El string para ejecutar la llamada seria el siguiente

```
"SELECT * FROM Usuarios WHERE Usuario='" + variableUsuario + "' AND
Password='" + variablepassword +"'"
```

el tema es que si en la variable variableUsuario ponemos lo siguiente, (sin las comillas dobles) "daniel' or

Usuario='manuel" la sentencia quedaria:

```
"SELECT * FROM Usuarios WHERE Usuario='daniel' or Usuario='manuel' AND
Password='" + variablepassword +"'
```

"

por lo que nos traerá todo de los usuarios Manuel y Daniel con la contraseña proveida.

Digamos por ejemplo que en variablepassword le pasamos lo siguiente, (sin comillas dobles) "a' or 'a'='a".
la sentencia entonces quedaria...

```
"SELECT * FROM Usuarios WHERE Usuario='daniel' or Usuario='manuel' AND Password='a' or 'a'='a'"
```

Lo que nos traería al usuarios Daniel o Manuel, sin importar la contraseña ya que se cumple la condición de 'a'='a'.

Ahora escribiríamos lo siguiente, para acceder con cualquier usuario:

```
“SELECT * FROM Usuarios WHERE Usuario='a' or 'a'='a' AND Password='a' or 'a'='a'"
```

Creo que mas o menos se entendió el punto...
Básicamente es lo mismo para el resto de las operaciones, solo hay que saber SQL y utilizar el ingenio.

Ahora lo que nos importa:

¿Cómo evitar que se utilice en nuestras aplicaciones?

Es simple, una buena técnica para implementar seguridad en nuestras aplicaciones, consiste en diferentes técnicas:

1. Toda variable ingresada es dañina, así es como debemos pensar sobre todas las variables que nos pasan por medio de post, get, link y etc. Incluso de BD diría yo.
2. No dar por sentado que el usuario hará lo que nosotros queramos, el usuario hará lo que el quiere, nuestro código debe ser así de flexible o impedírselo. Es decir, permitirle accesos y darle permisos para entrar o ejecutar ciertas acciones. Es bueno utilizar perfiles y permisos junto a los usuarios.
3. Llevar un registro de las acciones de los usuarios, guardar un registro es muy útil al momento de identificar errores y problemas.

Referente al punto **1**:
La mejor forma para evitar esto, es utilizar parámetros para realizar las consultas, al menos esto en SQL Server es

aplicable... sinceramente no se como es con el resto de las bases de datos... pero tambien hay una solucion.

por ejemplo:

```
"SELECT * FROM Usuarios WHERE Usuario=@Usuario AND
Password=@Password"
```

y se le pasa por parámetro los valores ingresados. Eso evita la mayoría de las inyecciones.

Otra manera de hacerlo, es reemplazando los valores por ejemplo reemplazando las comillas simples por dobles comillas simples

o por "\'". Dependiendo del carácter que queramos evitar y de la sintaxis de nuestra base de datos.

Referente al punto **2**:
El concepto es simple, es un poco complicada el diseño de la bd para esto, pero bueno vale la pena.
Esto se puede realizar de diferentes maneras, no lo voy a explicar aquí porque es muy extenso.
Básicamente consiste en permitir acceso a ciertas páginas dependiendo de cada perfil.
Cada usuario pertenece a un perfil determinado.
Esto lo que nos brinda es un control general de quienes pueden acceder a que paginas y que pueden hacer en ellas.

Referente al punto **3**:
Cada vez que un usuario se loguea o intenta loguearse, se guarda esa información. Ya sea en un txt, xml, base de datos o notificando por mail.
Y no solo cuando se loguea, sino cuando realiza una consulta, un insert, un update o entra a una pagina o modifica algo... siempre es útil para detectar intrusiones, problemas o cualquier tipo de diagnostico. también podríamos realizar estadísticas del sitio para saber cual es la pagina mas visitada o cual es la herramienta mas utilizada de nuestra aplicación.
En caso de las aplicaciones windows, siempre es conveniente que exista alguna opción para poder enviar el txt, xml o el output del error a nuestro mail o a nuestro equipo de desarrollo para poder solucionarlo.

Concluyendo
Si, lo se no es un "tutorial" largo..., tampoco lo iba a ser... pero bueno.
Basicamente queria exponer esto para que si alguien no tenia nocion de esto, que ahora al menos tenga un minimo de idea.
De todas maneras abajo dejo un par de links para que la gente tenga mas informacion confiable...

Siempre es bueno validar todo el input de datos que hacemos, sobretodo si es web que nuestros datos estan mas expuestos que en una aplicacion windows.

_____
[+] Mas Info: http://es.wikipedia.org/wiki/SQL_injection
[+] Mas Info: http://www.nextgenss.com/papers/advanced_sql_injection.pdf
[+] Mas Info: Google: sql injection
[+] Mas Info: Google: sql inyeccion

_____

Articulo realizado por: Ironic
Email: ironic[at]ironicnet[dot]com

Concursos & Torneos.

Ganadores del Concurso de Wallpapers de Agosto:
1.- Bucio & RioT (empate)

Ganadores del Concurso de tutoriales:
1.- **Pescaodeth:** Por Su artículo de manejo de sockets y DDOS. Premiado con un Webhosting + Dominio
2.- **Goldenozaro:** Por su traducción del artículo "Desbloqueando el iphone" y constante colaboración en artículos. Premiado con un Rango en los foros por un mes.
3.- **Crispunk:** Por su artículo de 'reconocimiento de voz en linux' y constante colaboración en artículos de ese subforo.

Los artículos de los cinco ganadores aparecerán en la siguiente E-Zine y serán colocados como Post-It en sus subforos.

Editorial, Autores y Agradecimientos. *Por Farid Murzone.*

Finalmente fue lanzada la primera H-Zine. La primera E-Zine de Hackerss.com, es realmente un orgullo finalmente lanzarla ya que varias veces se había planeado o empezado a desarrollar pero nunca se había llegado a éste momento. Esperemos que pueda ser con entregas continuas, al menos mensuales y no tengamos que esperar otros 7 años para lanzar la próxima (Es chiste, quédense tranquilos).

No me queda mucho más que decir, solo expresar el orgullo de ser quien la edite y anuncie, aunque los que realmente trabajaron son otros, muchísimas gracias a todos ellos:

- GerCat: Editor y Productor de la H-Zine y Autor del Artículo "Troyanos".
- RPM: Diseñador gráfico (Portada y plantillas).
- Marioly: Autora del artículo "Clases en c++".
- Ironic: Autor del artículo "Evitar SQL Inyection".
- SoulLost: Autor del artículo "iptables".
- Dinosaurio: Autor del artículo "clustering en Linux".
- Alluz: Autor del artículo "Aspectos de configuración que deben ser evaluados al momento de implementar un firewall central, Reglas Básicas" Cuyo título fue acortado para la E-Zine.
- Spawn: Autor del artículo "IPv6"
- Tec-Diego, Bucio, Klan, Pro-Cool, franz, Benek: Por apoyarnos en el desarrollo de la E-Zine… o Spamear, por estar ahí.

Atte.: El Staff de **Hackerss.com**