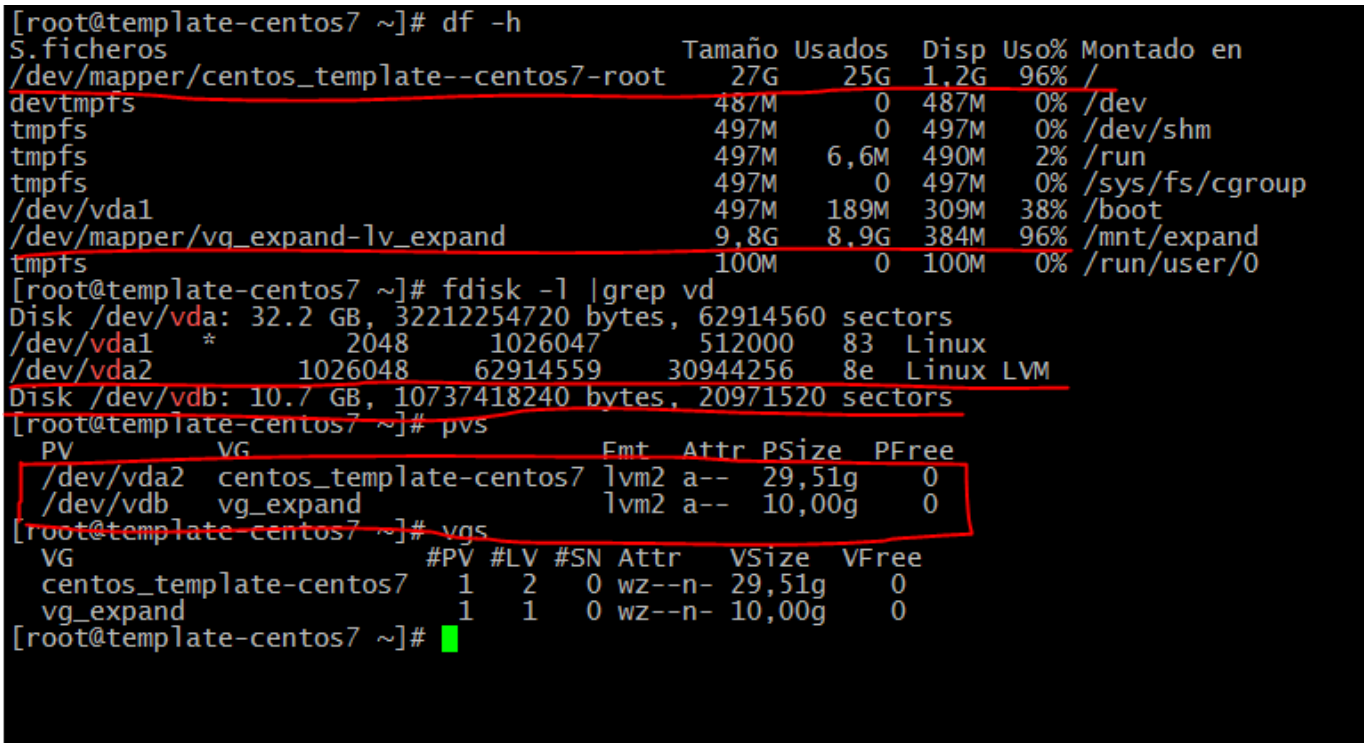
# Gestión de particiones en discos y LVM

Siguiendo con la aclamada serie “vfmBOFH haciendo malabares con motosierras”, vamos a ver cómo saltarnos un par de pasos en la gestión de las ampliaciones de particiones / discos con capa LVM por encima.

Vamos a ver el siguiente supuesto:




Vemos que tanto la partición raíz como el punto de montaje /mnt/expand están próximos al 100% de ocupación. En el caso de la partición raíz, el PV que aloja al VG/LV de sistema está en la partición física /dev/vda2 y el PV del LV/VG expand ocupa el dispositivo físico /dev/vdb sin usar particionado. Un ejemplo de cada, qué apropiado 

Vamos a aumentar el tamaño de los discos virtuales en 10GB cada uno y luego vamos a redimensionar “al vuelo” la capa LVM sin modificar el esquema de particionado. Empezemos por el dispositivo /dev/vdb que como hemos visto, es un PV directo sin una tabla de particiones por debajo:

```
[root@template-centos7 ~]# fdisk -l /dev/vdb

Disk /dev/vdb: 21.5 GB, 21474836480 bytes, 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

```
[root@template-centos7 ~]# pvresize /dev/vdb
Physical volume "/dev/vdb" changed
1 physical volume(s) resized / 0 physical volume(s) not resized
[root@template-centos7 ~]# vgs
VG #PV #LV #SN Attr VSize VFree
centos_template-centos7 1 2 0 wz--n- 29,51g 0
vg_expand 1 1 0 wz--n- 20,00g 10,00g
[root@template-centos7 ~]# lvextend -l +100%FREE /dev/mapper/vg_expand-
lv_expand
Size of logical volume vg_expand/lv_expand changed from 10,00 GiB (2559
extents) to 20,00 GiB (5119 extents).
Logical volume lv_expand successfully resized.
[root@template-centos7 ~]# resize2fs /dev/mapper/vg_expand-lv_expand
resize2fs 1.42.9 (28-Dec-2013)
Filesystem at /dev/mapper/vg_expand-lv_expand is mounted on /mnt/expand; on-
line resizing required
old_desc_blocks = 2, new_desc_blocks = 3
The filesystem on /dev/mapper/vg_expand-lv_expand is now 5241856 blocks long.
[root@template-centos7 ~]# df -h
S.ficheros Tamaño Usados Disp Uso% Montado en
/dev/mapper/centos_template--centos7-root 27G 25G 1,2G 96% /
devtmpfs 487M 0 487M 0% /dev
tmpfs 497M 0 497M 0% /dev/shm
tmpfs 497M 6,6M 490M 2% /run
tmpfs 497M 0 497M 0% /sys/fs/cgroup
/dev/vda1 497M 189M 309M 38% /boot
tmpfs 100M 0 100M 0% /run/user/0
/dev/mapper/vg_expand-lv_expand 20G 8,9G 9,9G 48% /mnt/expand
[root@template-centos7 ~]#
```

Ahora la chicha. Toca redimensionar una partición. Cuidadín y atentos:

```
[root@template-centos7 ~]# fdisk -l /dev/vda

Disk /dev/vda: 42.9 GB, 42949672960 bytes, 83886080 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Identificador del disco: 0x000af804

Disposit. Inicio    Comienzo      Fin      Bloques  Id  Sistema
/dev/vda1   *        2048     1026047      512000   83  Linux
/dev/vda2         1026048    62914559    30944256   8e  Linux LVM
[root@template-centos7 ~]# fdisk /dev/vda
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.
```

```
Orden (m para obtener ayuda): p

Disk /dev/vda: 42.9 GB, 42949672960 bytes, 83886080 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Identificador del disco: 0x000af804

Disposit. Inicio    Comienzo      Fin      Bloques  Id  Sistema
/dev/vda1   *        2048     1026047      512000   83  Linux
/dev/vda2         1026048    62914559    30944256   8e  Linux LVM

Orden (m para obtener ayuda): d
Número de partición (1,2, default 2): 2 <--Ojito con la partición que borramos
Partition 2 is deleted
Orden (m para obtener ayuda): n
Partition type:
   p   primary (1 primary, 0 extended, 3 free)
   e   extended
Select (default p): p
Número de partición (2-4, default 2):
Primer sector (1026048-83886079, valor predeterminado 1026048):
Se está utilizando el valor predeterminado 1026048
Last sector, +sectors or +size{K,M,G} (1026048-83886079, valor predeterminado
83886079):
Se está utilizando el valor predeterminado 83886079
Partition 2 of type Linux and of size 39,5 GiB is set
##########Recreamos la partición como estaba, pero asignándole el espacio extra
##########
##########En algunos casos, fdisk se lía con los sectores de inicio y final de
###########
##########las particiones, será cuestión de ponerle como inicio el sector que
############
##########nos aparece en la línea 30, campo "Fin"
(1026047)+1############################
Orden (m para obtener ayuda): w
¡Se ha modificado la tabla de particiones!

Llamando a ioctl() para volver a leer la tabla de particiones.

WARNING: Re-reading the partition table failed with error 16: Dispositivo o
recurso ocupado.
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)
```

```
Se están sincronizando los discos.
[root@template-centos7 ~]# partx -a /dev/vda
partx: /dev/vda: error adding partitions 1-2
[root@template-centos7 ~]# fdisk /dev/vda
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.



Orden (m para obtener ayuda): p

Disk /dev/vda: 42.9 GB, 42949672960 bytes, 83886080 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Identificador del disco: 0x000af804

Disposit. Inicio    Comienzo      Fin      Bloques  Id  Sistema
/dev/vda1   *        2048     1026047      512000   83  Linux
/dev/vda2         1026048    83886079    41430016   83  Linux <--Ojocuidao, que
tiene que ser tipo LVM

Orden (m para obtener ayuda): t
Número de partición (1,2, default 2): 2
Hex code (type L to list all codes): 8e
Changed type of partition 'Linux' to 'Linux LVM'

Orden (m para obtener ayuda): w
¡Se ha modificado la tabla de particiones!

Llamando a ioctl() para volver a leer la tabla de particiones.

WARNING: Re-reading the partition table failed with error 16: Dispositivo o
recurso ocupado.
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)
Se están sincronizando los discos.
[root@template-centos7 ~]# partx -a /dev/vda
partx: /dev/vda: error adding partitions 1-2
```

Vemos que he entrado en fdisk dos veces: una para re-crear la partición y otra para cambiarle el tipo. Puede hacerse en un único paso, pero yo prefiero escribir las tablas de particiones a cada cambio. Cada uno es cadascual.

```
[root@template-centos7 ~]# pvs
  PV         VG                     Fmt  Attr PSize  PFree
  /dev/vda2  centos_template-centos7 lvm2 a--  29,51g    0
  /dev/vdb   vg_expand               lvm2 a--  20,00g    0
[root@template-centos7 ~]# fdisk -l /dev/vda

Disk /dev/vda: 42.9 GB, 42949672960 bytes, 83886080 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Identificador del disco: 0x000af804

Disposit. Inicio    Comienzo      Fin      Bloques  Id  Sistema
/dev/vda1   *        2048     1026047      512000   83  Linux
/dev/vda2         1026048    83886079    41430016   8e  Linux LVM
[root@template-centos7 ~]# pvresize /dev/vda2  <-- Una vez re-creada la
partición, procedemos igual
  Physical volume "/dev/vda2" changed
  1 physical volume(s) resized / 0 physical volume(s) not resized
[root@template-centos7 ~]# pvs
  PV         VG                     Fmt  Attr PSize  PFree
  /dev/vda2  centos_template-centos7 lvm2 a--  39,51g 10,00g
  /dev/vdb   vg_expand               lvm2 a--  20,00g     0
[root@template-centos7 ~]# vgs
  VG                     #PV #LV #SN Attr   VSize  VFree
  centos_template-centos7   1   2   0 wz--n- 39,51g 10,00g
  vg_expand                 1   1   0 wz--n- 20,00g     0
[root@template-centos7 ~]# lvextend -l +100%FREE /dev/mapper/centos_template--
centos7-root
  Size of logical volume centos_template-centos7/root changed from 27,51 GiB
(7042 extents) to 37,51 GiB (9602 extents).
  Logical volume root successfully resized.
[root@template-centos7 ~]# resize2fs /dev/mapper/centos_template--centos7-root
resize2fs 1.42.9 (28-Dec-2013)
Filesystem at /dev/mapper/centos_template--centos7-root is mounted on /; on-line
resizing required
old_desc_blocks = 4, new_desc_blocks = 5
The filesystem on /dev/mapper/centos_template--centos7-root is now 9832448
blocks long.

[root@template-centos7 ~]# df -h
S.ficheros                                Tamaño Usados  Disp Uso% Montado en
/dev/mapper/centos_template--centos7-root    37G    25G   11G  70% /
devtmpfs                                    487M      0  487M   0% /dev
```

```
tmpfs                                    497M      0  497M   0% /dev/shm
tmpfs                                    497M   6,6M  490M   2% /run
tmpfs                                    497M      0  497M   0%
/sys/fs/cgroup
/dev/vda1                                497M   189M  309M  38% /boot
tmpfs                                    100M      0  100M   0% /run/user/0
```

Tacháaaan!

---

# Bloqueando intentos de cifrado en shares Samba

Holaquetal.

Andamos algo revolucionados en el trabajo, así que no puedo estar mucho ante las teclas.

Una de las revoluciones con la que nos hemos encontrado es la proliferación de ransomwares, que en el curro llamamos *Cristolockers*, juego de palabras a cuenta de los cipostios que lían y que ahora no viene al caso.

En fin, por Twitter me llegó hoy un enlace con un método para evitar en la medida de lo posible el cifrado de archivos en servidores de ficheros Windows. Como complemento, aquí dejo una posible manera de lograr lo mismo en servidores linux con el rol de servidor de ficheros vía Samba.

El tinglado consiste en unas directivas de auditoría Samba para los shares

que el servidor va a presentar, y un filtro a medida para fail2ban que examine el log del sistema y banee al listoferias que haya abierto un mail de correos sobre un paquete certificado que no le han podido entregar.

Al lío.

No es muy complicado, empezaremos añadiendo las directivas de auditoría en el smb.conf:

```
# Recurso de acceso R/W  publico
[publico]
        comment = Directorio compartido
        path = /ruta/al/share/publico
        vfs objects = full_audit
        full_audit: failure = none
        full_audit: success = pwrite write rename
        full_audit: prefix = IP = %I | USER = %u  | MACHINE = %m | VOLUME = %S
        full_audit: facility = local7
        full_audit: priority = NOTICE
        valid users = @winusers
        read only = No
        create mask = 0664
        directory mask = 0775
        force group = winusers
        force directory mode = 0775
        wide links = Yes
```

En negrita, las distintas directivas de auditoría. En concreto y por orden, qué operaciones se van a registrar en caso de que fallen (failure), las que se registrarán en caso de realizarse correctamente (success, pwrite -subida-, write -escritura-, rename -renombrado-), La cadena que se escribirá en el log (prefix: detallamos la ip, el usuario, la máquina y el volumen), el "facility" (local7, /var/log/syslog) y su prioridad. El resto del snip, vemos que son configuraciones bastante normales de un share samba.

Bien. Con estas directivas de auditoría, y una vez reiniciado samba, cada vez que un usuario que tenga mapeada la unidad de red \\servidor\publico escriba, suba ficheros o renombre los mismos, dejará una traza en el log similar a ésta:

```
Apr 21 21:02:50 srv1 smbd[31353]: IP = 192.168.1.35 | USER = fulano | MACHINE =
pote | VOLUME = publico|pwrite|ok|ruta/al/archivo/archivo.extension
```

Lo cual nos da un par de patterns chulos para revisar el log con fail2ban.

Para ello, una vez instalado creamos un fichero de filtro en

/etc/fail2ban/filter.d/. Llamémosle samba.conf:

```
[Definition]
failregex = smbd.*\:\ IP\ =\ \ \|.*\.0x0$
            smbd.*\:\ IP\ =\ \ \|.*\.1999$
            smbd.*\:\ IP\ =\ \ \|.*\.*obleep$
            smbd.*\:\ IP\ =\ \ \|.*\.LOL!$
            smbd.*\:\ IP\ =\ \ \|.*\.aaa$
            smbd.*\:\ IP\ =\ \ \|.*\.abc$
            smbd.*\:\ IP\ =\ \ \|.*\.bleep$
            smbd.*\:\ IP\ =\ \ \|.*\.ccc$
[...]
ignoreregex =
```

Esto es un fragmento recortado, pero se deja entender bastante. Básicamente es una compilación de extensiones conocidas de archivos cifrados por los ransom más *populares*. Yo me la he pillado de aquí y adaptado al resto de línea de log para que cuadre con la regex. Ahora mismo, me rondan las 50 extensiones. También te lo puedes currar y añadir los nombres de archivo conocidos con las instrucciones para pagar el rescate, pero ya empezaría a ser un poco demasiado para andar comprobando.

Bueno, hora de añadir nuestro filter al final de /etc/fail2ban/jail.conf:

```
[samba]
filter = samba
enabled = true
action = iptables-multiport[name=samba, port="135,139,445,137,138",
protocol=tcp]
         mail[name=samba, dest=vfmbofh@mi.correo.no.te.dire]
logpath = /var/log/syslog
maxretry = 1
findtime = 600
bantime = 86400
```

Baneamos 24 horas al primer intento, y enviamos un correo avisando del tema. El findtime, lo dejamos cuadrado con el que haya por default en la instalación.

Iniciamos /etc/init.d/fail2ban start y listos:

```
root@srv1:~# iptables -nL -v
Chain INPUT (policy ACCEPT 3697 packets, 303K bytes)
 pkts bytes target     prot opt in     out     source               destination
```

```
    0     0 fail2ban-samba  tcp  --  *      *       0.0.0.0/0
0.0.0.0/0           multiport dports 135,139,445,137,138
    0     0 fail2ban-ssh  tcp  --  *      *       0.0.0.0/0           0.0.0.0/0
multiport dports 22

Chain FORWARD (policy ACCEPT 1047 packets, 157K bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 3175 packets, 995K bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain fail2ban-samba (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain fail2ban-ssh (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

Y con ésto y un bizcocho…

---

# LVM: Mover volúmenes lógicos entre discos físicos.

Asumámoslo, shit happens.

A veces pasa que tu magnífico disco de 2TB lleno de ~~porno~~ interesantísimos proyectos, máquinas virtuales y archivos sensibles (porno) empieza a fallar. Si tienes backup de todo ésto, pues tampoco es ningún drama, excepto la parte de configurar de cero un nuevo sistema en otro disco y restaurar los datos. Cuando tienes el sistema más tuneado que los gorgoritos de Justin Bieber, pues es un peñazo insufrible.

Así que si eres mínimamente listo (o te ha pasado con anterioridad la mierda ésta), te lo montas para que el sistema vaya avisando (con smartctl, por ejemplo) conforme al disco duro de tus entretelas se le van acabando las horas de vuelo. Si además configuras todo el sistema con LVM (Fedora, Debian y otras ya te lo hacen por defecto al instalar), lo tienes hecho.

Asumimos un sistema con todos sus archivos y particiones dentro de un LVM basado en /dev/sdb1(excepto /boot, que sigue dando problemas). Asumiremos también que ya hemos pinchado el disco de recambio y que es correctamente reconocido por el sistema.

Creamos un nuevo volumen físico en /dev/sdc1 y echamos un vistazo a cómo está el patio a nivel de volúmenes físicos, grupos de volúmenes y volúmenes lógicos:

```
root@bestiaparda:/mnt# pvcreate /dev/sdc1
  Physical volume "/dev/sdc1" successfully created
root@bestiaparda:/mnt# pvscan
  PV /dev/sdb1   VG vg_deb   lvm2 [1,82 TiB / 1,55 TiB free]
  PV /dev/sdc1               lvm2 [1,82 TiB]
  Total: 2 [3,64 TiB] / in use: 1 [1,82 TiB] / in no VG: 1 [1,82 TiB]
root@bestiaparda:~# pvs
  PV         VG     Fmt  Attr PSize PFree
  /dev/sdb1  vg_deb lvm2 a--  1,82t 1,55t
  /dev/sdc1         lvm2 a--  1,82t 1,82t
root@bestiaparda:~# vgs
  VG     #PV #LV #SN Attr   VSize VFree
  vg_deb   2   5   0 wz--n- 3,64t 3,10t
root@bestiaparda:~# lvs
  LV          VG     Attr       LSize   Pool Origin Data%  Meta%  Move Log [...]
  lv_ERP      vg_deb -wI-a-----  20,00g
  lv_home     vg_deb -wI-ao---- 200,00g
  lv_root     vg_deb -wI-ao----  46,56g
  lv_swap     vg_deb -wI-ao----   1,86g
  vfmbofh.net vg_deb -wI-a-----   8,00g
root@bestiaparda:~#
```

Cuidadín con el pvcreate, que como nos equivoquemos de partición, la liaremos parda. Por otra parte, si el disco es grande (en este caso de 2TB, conviene crear una partición y luego crear el pv en ella. Cosas del GPT.

Bueno, vemos que nuestro LVM reside en /dev/sdb1, que contiene un grupo de volúmenes (VG) llamado vg_deb y éste diversos volúmenes lógicos (LV), llamados lv_*WHATEVER*. También vemos que el sistema reconoce correctamente el nuevo volumen físico (PV) en /dev/sdc1.

Toca extender el VG con nuestro flamante PV nuevo:

```
root@bestiaparda:/mnt# vgextend vg_deb /deb/sdc1
  Volume group "vg_deb" successfully extended
root@bestiaparda:~# pvs
  PV         VG     Fmt  Attr PSize PFree
```

```
  /dev/sdb1  vg_deb lvm2 a--  1,82t 1,55t
  /dev/sdc1  vg_deb lvm2 a--  1,82t 1,82t
root@bestiaparda:~#
```

Vemos que ya tenemos el nuevo PV asignado al VG, así que podemos proceder a mover todos los LV de un disco a otro del tirón. Sin desmontar ni nada. Fácil para todos.

```
root@bestiaparda:/mnt# pvmove /dev/sdb1 /dev/sdc1
/dev/sdb1: Moved: 0,0%
[...]
/dev/sdb1: Moved: 99,9%
```

Fácil. No rápido. Si tenéis tabaco, es un buen momento para un piti. O diez. Es un movimiento bastante seguro. Internamente, LVM no elimina nada que no se haya copiado correctamente antes. Así que la cosa es *a prueba de fallos*. MUY a prueba de fallos. He visto sistemas recuperarse de un apagado a las bravas con un proceso como éste a medio hacer y no sólo arrancar tan ricamente, sino continuar el proceso allí donde se quedó.
Por cierto, no avisa con ningún mensaje en caso de éxito. Simplemente llega al 100% y para.

Veamos qué tenemos al acabar:

```
root@bestiaparda:/mnt# pvs
  PV         VG     Fmt  Attr PSize PFree
  /dev/sdb1  vg_deb lvm2 a--  1,82t 1,82t
  /dev/sdc1  vg_deb lvm2 a--  1,82t 1,55t
```

Como podemos ver, los datos se han movido sin más. Aunque la partición raíz del sistema que estemos moviendo esté en uso. Ole y ole.

Ya podemos eliminar el disco moribundo del LVM activo y respirar tranquilos:

```
root@bestiaparda:/mnt# vgreduce vg_deb /dev/sdb1
  Removed "/dev/sdb1" from volume group "vg_deb"
root@bestiaparda:/mnt# pvremove /dev/sdb1
  Labels on physical volume "/dev/sdb1" successfully wiped
root@bestiaparda:/mnt#
```

Ahora, en la próxima parada del sistema, podremos eliminar físicamente el disco sin más problemas y darle un final PePero, a martillazos.

:wq

# Utilizar un host como proxy para apuntar un dominio hacia una ip dinámica

Buenas.

Empezamos con una historia para afinar nuestras habilidades como técnicos de Poor’s man IT(r)

Asumimos que, por diferentes motivos, tenemos control (y permiso para usar) un vps con ip pública fija.

Asumimos también que queremos ofrecer servicios (web, correo) desde un host con ip pública dinámica. Hoy en día, para un webserver normalito sin demasiado stress, un buen enganche de fibra da el pegote a la hora de servir contenidos, por ejemplo.

También asumimos que tenemos un dominio registrado en algún ISP que no permite la actualización en tiempo real de la ip donde apuntan los registros DNS de los servicios que queramos publicar a través de un host con ip pública. A veces pasa. La mayoría de veces, en realidad, a no ser que estés con DonDominio, que ofrece un estupendo script para ir actualizando la ip de los registros DNS que queramos de los dominios que tengamos contratados con ellos.

Bueno, pues lo que vamos a montar aquí tiene esta estructura:

ISP (zona DNS) –> host proxy (ip fija) –> host destino (ip dinámica)

A través de la herramienta que nos proporcione el ISP, fijaremos las entradas DNS que nos interese dirigir (para este ejemplo el registro A de dominio.com y un CNAME para el www.dominio.com. Para el correo, la filigrana es un poco más elaborada. La dejamos para otro día.

En el momento en que configuremos ésto, nos podemos olvidar de mantener los registros DNS al día (a no ser que cambiemos de *host proxy*). Sólo tendremos que asegurarnos de que el host proxy transfiera correctamente las peticiones al host destino, y que siempre sepa dónde enviarlas, en el caso de que la ip pública de destino cambie.

Para llevar a cabo ésta tarea, nos valdremos de tres scripts. Uno de ellos correrá en el host destino a intervalos regulares (pongamos cada 5 minutos) y en función de determinadas condiciones (que haya cambiado la ip pública que se le asigna) llamará a otros dos scripts que correrán en el host proxy. Para la ejecución de éstos scripts, configuraremos un acceso root por ssh vía clave pública (dejar el ssh de un host expuesto a logins como root a toda

internet es suicida, niños), y ya que estamos, nos montaremos un sistema de alertas por correo.

Lo primero va delante: creamos el par de claves pública/privada y configuramos el acceso al host proxy desde el destino (en este momento aún tendremos el aceso ssh para root en host proxy habilitado)

Host destino:

```
vfmbofh@destino:~/.ssh$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/vfmbofh/.ssh/id_rsa): hostproxy_rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in hostproxy_rsa.
Your public key has been saved in hostproxy_rsa.pub.
The key fingerprint is:
e6:01:be:87:41:df:ea:5b:0e:9d:08:16:e0:1f:10:e4 vfmbofh@destino.dominio.com
The key's randomart image is:
+--[ RSA 2048]----+
|   .=.           |
|   o o           |
|    E =          |
|     + = .       |
|      * S .      |
|     . B = .     |
|      o * +      |
|       o +       |
|        o..      |
+-----------------+
vfmbofh@destino:~/.ssh$
```

Procedemos a copiarlo con ssh-copy-id en el host proxy:

```
vfmbofh@destino:~$ ssh-copy-id -i /home/vfmbofh/.ssh/hostproxy_rsa.pub
root@dominio.com
root@dominio.com's password:
Now try logging into the machine, with "ssh 'root@dominio.com'", and check in:

  ~/.ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.
##### Comprobamos que podemos hacer login #####
vfmbofh@destino:~$ ssh -i /home/vfmbofh/.ssh/hostproxy_rsa root@dominio.com
Linux hostproxy 3.2.0-4-amd64 #1 SMP Debian 3.2.68-1+deb7u5 x86_64
```

```
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
No mail.
root@hostproxy:~# logout
Connection to dominio.com closed.
vfmbofh@destino:~$
```

Ea, cuando os recupereis de la lisergia de los colorines (estoy buscando otro plugin para el resaltado de código) ya podemos cerrar el acceso por password a root por ssh en el host proxy y continuar.

Vamos ahora con los scripts que correrán en el host proxy. Uno de ellos es bastante simple: sólo dos lineas. Informará al host destino sobre qué ip tiene almacenada para pasar peticiones que lleguen a dominio.com. Si las dirección almacenada difiere de la actual del host destino, llamaremos al segundo script para realizar los cambios pertinentes.

En host proxy:

```
#!/bin/bash
ALMACENADA=`grep dominio.com /etc/hosts | awk '{print $1}'`
echo $ALMACENADA
```

Buscamos en el /etc/hosts la entrada que contiene la ip de nuestro dominio y la escupimos con un echo que recogerá el script de host destino para evaluarla. No era tan difícil.

El segundo script, que cambia cuando se lo ordenan la dirección ip de host destino:

```
#!/bin/bash
REAL=$1
LINEAHOSTS="$REAL dominio.com subdominio.dominio.com whatever.dominio.com"
sed -i '$ d' /etc/hosts
echo $LINEAHOSTS >> /etc/hosts
/etc/init.d/nginx stop >/dev/null 2>&1
/etc/init.d/nginx start >/dev/null 2>&1
```

Este tampoco es un prodigio de dificultad: Recogemos un parámetro (que nos entrega el script de host destino) y construímos con él la línea apropiada para incluir en /etc/hosts. Luego con sed eliminamos la última línea de

hosts, y realizamos la inclusión de la línea nueva. Luego reiniciamos nginx para que refresque la ip internamente. A mí me gusta pararlo y levantarlo, aunque con un restart también vale.

Vamos ahora con el script en host destino. Un poco más grande, ya que es el que hace todo el trabajo:

```
#!/bin/bash
function correo () {
        correos=('vfmbofh@xxxx.com')
        for i in "${correos[@]}"
        do
                echo "$1" | mailx -s "Información IP host" $i
        done
}
REAL=""
ALMACENADA=`ssh -i /home/vfmbofh/.ssh/hostproxy_rsa.pub root@dominio.com
'/ruta/script/checker'`
RESUL=$?
if [ $RESUL -ne 0 ]
then
        msg="Error al tratar de contactar con el servidor proxy"
        correo "$msg"
        exit 0
fi
#Usamos el servicio de icanhazip.com para que nos devuelva la ip real
REAL=`curl  --connect-timeout 60 icanhazip.com`
RESUL=$?
if [ $RESUL -ne 0 ]
then
        msg="Error al tratar de obtener la IP real"
        correo "$msg"
        exit 0
fi
if [ "$ALMACENADA" == "$REAL" ]
then
        logger "IP actualizada, no es necesario actuar."
        exit 0
else
        #Enviamos la ip que nos ha devuelto icanhazip.com
        #para actualizar el /etc/hosts
        ssh -i /home/vfmbofh/.ssh/hostproxy_rsa.pub root@dominio.com
"/ruta/script/changer $REAL"
        RESUL=$?
        if [ $RESUL -ne 0 ]
```

```
        then
                msg="Error al tratar de fijar la nueva IP revisar
funcionamiento. La ip actualmente asignada al host es $REAL"
                correo "$msg"
                logger "$msg"
                exit 0
        else
                msg="La IP del router ha cambiado de $ALMACENADA a $REAL."
                correo "$msg"
                logger "$msg"
                exit 0
        fi
fi
```

Por partes. Primero definimos una función para enviar correos mediante mailx. Aquí sólo hay un destinatario definido, pero podemos poner tantos como queramos, entre comillas simples y separados por espacio. Recogemos la ip que host proxy tiene registrada con el script checker que hemos visto antes, y la que tenemos realmente asignada a través de una llamada con curl a icanhazip.com. Almacenamos en $ALMACENADA y $REAL respectivamente. Si hay errores, notificamos y paramos.

Seguidamente evaluamos la dirección real. Si coincide con la almacenada, mensaje al log y paramos.

En el caso de que difieran, llamamos al script changer en host proxy para que actualice la ip en su /etc/hosts con la que hemos obtenido y almacenado en $REAL. Si todo va bien, notificamos el cambio, si no, notificamos el error, añadiendo la ip por si podemos conectarnos y tratar de ver qué pasa.

Si este script lo metemos en el cron de root para que se ejecute cada 5 minutos, tendremos una comunicación más o menos estable entre proxy y destino. Bueno, todo lo estable que sea la conexión de host destino.

Arrementonses, asumiendo que en host destino tengamos una web sirviéndose por el puerto 80/443, en host proxy tendremos que montar una cochinadita en nginx tal que ésta:

En host proxy, /etc/nginx/sites-enabled/dominio.com:

```
server {
        listen 80;
        server_name dominio.com www.dominio.com;
        access_log /var/log/nginx/dominio_access.log;
        error_log /var/log/nginx/dominio_error.log;
        client_max_body_size 100M; #Por decir algo
        location / {
```

```
                #nginx resuelve como un navegador de la vida
                #de ahí el cambio en /etc/hosts
                proxy_pass http://dominio.com;
                proxy_next_upstream error timeout invalid_header http_500
http_502 http_503 http_504;
                proxy_redirect off;
                proxy_buffering off;
                proxy_set_header Host dominio.com;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                add_header X-Frame-Options SAMEORIGIN;
        }
}
server {
        listen 443;
        server_name dominio.com www.dominio.com;
        access_log /var/log/nginx/dominio_ssl_access.log;
        error_log /var/log/nginx/dominio_ssl_error.log;
        client_max_body_size 100M;
        ssl on;
        #No he llegado a determinar qué webserver acaba sirviendo
        #el certificado, sólo con servirlo desde el proxy debería bastar,
        #pero recomiendo que esté en los dos
        ssl_certificate /etc/nginx/ssl/dominio.com.crt;
        ssl_certificate_key /etc/nginx/ssl/dominio.com.key;
        location / {
                proxy_pass https://dominio.com;
                proxy_next_upstream error timeout invalid_header http_500
http_502 http_503 http_504;
                proxy_redirect off;
                proxy_buffering off;
                proxy_set_header Host dominio.com;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                add_header X-Frame-Options SAMEORIGIN;
        }
}
```

Yastá. No era tan complicado. O sí.

La chulada de ésta config de nginx es que con los set_header que hay definidos, en el access_log de host destino, también se verán las IP reales de los visitantes. Útil si no siempre tienes acceso al proxy.