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.