Desde hace tiempo soy un feliz usuario de DenyHosts . Este programa Python, lee un archivo de logs del sistema como "messages" y detecta los intentos de acceso ilegales al demonio SSH , ya sea mediante ataques de diccionario, o mediante fuerza bruta, tomando las acciones necesarias para impedirlos.

La configuración es muy sencilla e implica modificaciones en los archivos de configuración de DenyHost, mediante la edición del archivo /usr/share/denyhosts/denyhosts.py, y del demonio SSH, sobre todo, para especificar un nivel adecuado de logs para SSH y para comprobar que nuestro servidor SSH está compilado con soporte para tcp_wrappers. Para no tener problemas durante la configuración del programa, también es conveniente empezar con unos archivos /etc/hosts.deny y /etc/hosts.allow vacíos. Yo tenía algunas configuraciones por omisión en los mismos, que eran incompatibles con DenyHosts, por lo que me costó un poco su configuración...

El funcionamiento del programa es sencillo. Cada cierto tiempo, programable por el usuario y que es conveniente que sea de unos 30 segundos o inferior, DenyHosts comprueba el contenido del archivo messages (o de cualquier otro archivo de logs del sistema que se le indique en su configuración) y si los intentos de acceso al servidor SSHD sobrepasan unos umbrales predefinidos por el usuario, DenyHosts introduce la IP atacante en el archivo /etc/hosts.deny del servidor. Con ello, ya no se podrá volver a intentar la conexión desde esa IP. El programa permite filtrar solamente los siguientes accesos al demonio SSHD, o a todos los servicios del servidor con soporte de tcp_wrappers mediante el uso del mandato "**BLOCK_SERVICE = ALL** " en la configuración de DenyHosts.

Si se configura el demonio SSHD, mediante el archivo /etc/ssh/sshd_config, para que solamente sea accesible mediante claves RSA, es factible configurar DenyHost para que reaccione al más mínimo intento de acceso ilegal al sistema. Por ejemplo podemos usar los siguientes mandatos:

```
 RhostsRSAAuthentication yes
RSAAuthentication yes
PubkeyAuthentication yes
PermitEmptyPasswords no
PasswordAuthentication no
```

El proceso de configuración de las claves RSA en el servidor es muy sencillo. Basta utilizar los siguientes mandatos en nuestro ordenador cliente, usando para ello el usuario con el que accederemos posteriormente al servidor. Como es lógico, para que esto funcione también es necesario tener acceso remoto al servidor, ya que el último mandato de la secuencia envía nuestra clave pública al servidor:

```
 $ ssh-keygen -t rsa
(intro a todo)
$ ssh-copy-id -i ~/.ssh/id_rsa.pub  usuario@miservidor.com
```

DenyHosts permite definir distintos umbrales para los intentos de acceso realizados con usuarios válidos, inválidos, restringidos o root. Como es lógico, es conveniente configurar SSH para que no permita el acceso desde el usuario root usando el mandato "**PermitRootLogin no** " en /etc/ssh/sshd_config.

Si accedemos desde una IP fija, también es conveniente, para evitar problemas, introducir una línea en el archivo hosts.allow del servidor con el contenido "ALL : xxx.xxx.xxx.xxx : ALLOW", en las que xxx.xxx.xxx.xxx deberá ser sustituido por la IP del ordenador cliente. De esta forma nuestro sistema no será incluido nunca en la lista negra. Si tenemos IP variable, también cabe la posibilidad, aunque lógicamente es menos seguro, de indicar un rango de IP's válidas, que no podrán ser prohibidas por DenyHosts.

Además, DenyHost permite limpiar las IP's contenidas en el archivo /etc/hosts.deny si se cumplen determinadas condiciones programables por el usuario, por ejemplo si no se realizan más intentos de acceso desde esa IP en un determinado tiempo. Los tiempos para proceder a la limpieza se pueden configurar en base a si los intentos se hicieron con usuarios válidos, inválidos, restringidos o como root. Por ejemplo, podemos decir que las IP's que han sido introducidas en /etc/hosts.deny por usar un determinado número de veces un usuario restringido se mantengan 30 días, y 15 días solamente si se intentó el acceso con un usuario válido. También puede ser configurado para que una IP no pueda ser borrada de nuevo de /etc/hosts.deny si después de haber sido borrada se produce un nuevo intento de acceso ilegítimo desde la misma IP. Asimismo, el programa permite poner a cero el contador cuando se realiza un acceso con éxito desde una determinada IP.

DenyHosts tiene otras muchas cualidades interesantes, como la posibilidad de recibir información mediante correo electrónico de las IP's introducidas en /etc/hosts.deny, o la de poder enviar y recibir de un servidor las IP's que hemos capturado, o que han sido capturadas por otros usuarios. En la actualidad, el número de usuarios suscritos a este servicio supera los 27.000, lo que nos puede permitir adelantarnos a algunos atacantes demasiado "conocidos".

Una de las capacidades más interesantes de DenyHosts es la posibilidad de definir cadenas para identificar los ataques, lo que nos permite ampliar el uso de este programa a otros servicios, siempre que la información de estos servicios aparezca en el archivo de logs que

hemos seleccionado (por ello es interesante usar messages) y seamos capaces de definir las cadenas que ha de identificar DenyHosts como ataques. Para configurarlo, se usan los mandatos "**USERDEF_FAILED_ENTRY_REGEXn=**" en el archivo /usr/share/denyhosts/denyhosts.conf. Por ejemplo, podemos usar estos mandatos para que DenyHosts identifique otras situaciones de ataque no contempladas inicialmente por el programa:

```
  USERDEF_FAILED_ENTRY_REGEX1=Did not receive identification string from (?P.*)
USERDEF_FAILED_ENTRY_REGEX2=User (?P.*) from (?P.*) not allowed because
not listed in AllowUsers
USERDEF_FAILED_ENTRY_REGEX3=Address (?P.*) maps to (?P.*), but this
does not map back to the address - POSSIBLE BREAK-IN ATTEMPT!
```

Es importante ver que las partes variables de las cadenas van entre los caracteres (?P.*), pudiendo definir las variables "<user>", "<host>" y "<message>"

UN ERROR DEL PROGRAMA Y SU SOLUCIÓN

A pesar de que el programa funcionaba bastante bien y añadía la mayor parte de los ataques detectados al archivo /etc/hosts.deny, en algunas ocasiones no lo hacía y entre los logs del sistema encontré este mensaje de error repetido un montón de veces:

```
  Sep 14 01:02:14 - denyhosts : ERROR regex pattern ( User (?P.*)
not allowed because not listed in AllowUsers ) is missing 'host' group
```

Como es lógico, para que el programa funcione adecuadamente, lo más importante es que pueda identificar la IP atacante, es decir, que la cadena ha de tener siempre un grupo del tipo (?P<host>.*), algo que falta en esta cadena ya que los mensajes son del tipo:

```
  Sep 14 01:02:41 casa sshd[2771]: User  from
not allowed because not listed in AllowUsers
```

Para eliminar este error y lograr que el programa funcione adecuadamente, tenemos que modificar el archivo regex.py, que en mi sistema está en el directorio /usr/lib/python2.5/site-packages/DenyHosts. En otros sistemas, con versiones anteriores de Python, lo tendrán en /usr/lib/python2.4/site-package /DenyHosts.

Hay que cambiar la cadena:

```
  FAILED_ENTRY_REGEX7 = re.compile(r"""User (?P<user>.*)
not allowed because not listed in AllowUsers""")
```

Por esta otra, con cuidado de escribirla exactamente tal como se ve aquí (pero en una sola línea):

```
  FAILED_ENTRY_REGEX7 = re.compile(r"""User (?P<user>.*) .*from (?P<host>.*)
not allowed because not listed in AllowUsers""")
```

Como veremos, hay disponibles otras cadenas para versiones futuras, que podemos añadir o cambiar si queremos; por ejemplo:

```
  FAILED_ENTRY_REGEX8 = re.compile(r"""Address (?P<host>.*) maps to (?P<message>.*),
but this does not map back to the address - POSSIBLE BREAK-IN ATTEMPT!""")
```

Después de modificar este programa, bastará con que vayamos a /etc/init.d y reinicialicemos DenyHosts mediante el mandato "**sudo denyhosts restart**" y si es necesario, el demonio sshd, mediante el mandato " **sudo sshd restart** ".

Como se puede ver, este magnífico programa es muy potente y flexible y si lo tenemos configurado adecuadamente podremos ver cosas como ésta en nuestro log, que sinceramente me encantan:

```
  Sep 14 01:02:41 casa sshd[2771]: User root from xxx.xxx.xxx.xxx
not allowed because not listed in AllowUsers
Sep 14 01:02:42 casa sshd[2771]: User root from xxx.xxx.xxx.xxx
not allowed because not listed in AllowUsers
Sep 14 01:02:42 - denyhosts : INFO new denied hosts: ['xxx.xxx.xxx.xxx']
Sep 14 01:02:44 casa sshd[2771]: refused connect from
xxxx.com (::ffff:xxx.xxx.xxx.xxx)
Sep 14 02:00:01 - sync : INFO sent 1 new host
Sep 14 02:00:01 - sync : INFO received 50 new hosts
```

Evidentemente, el host con IP xxx.xxx.xxx.xxx no podrá volver a intentarlo en nuestro sistema mientras esté su IP en /etc/hosts.deny. También hay que decir que desde que este sistema está funcionando en mi servidor han descendido los ataques por fuerza bruta y diccionario de forma considerable.

Un resumen normal de conexiones denegadas para un periodo de 24 horas es el siguiente, y teniendo en cuenta que el sistema introduce la IP en la lista negra con solamente un único intento fallido, la cosa no está nada mal:

::ffff:xxx.xxx.xxx.xxx (::ffff:xxx.xxx.xxx.xxx): 2 Time(s)
::ffff:xxx.xxx.xxx.xxx (::ffff:xxx.xxx.xxx.xxx): 11 Time(s)
::ffff:xxx.xxx.xxx.xxx (::ffff:xxx.xxx.xxx.xxx): 1 Time(s)
::ffff:xxx.xxx.xxx.xxx (::ffff:xxx.xxx.xxx.xxx): 2 Time(s)
::ffff:xxx.xxx.xxx.xxx (::ffff:xxx.xxx.xxx.xxx): 2 Time(s)
::ffff:xxx.xxx.xxx.xxx (::ffff:xxx.xxx.xxx.xxx): 2 Time(s)
::ffff:xxx.xxx.xxx.xxx (::ffff:xxx.xxx.xxx.xxx): 2 Time(s)
xxx.com (::ffff:xxx.xxx.xxx.xxx): 1 Time(s)

Como se puede ver, el programa -cuando se ha configurado para que lo haga así- también resuelve el dominio del atacante cuando existe resolución inversa del mismo, lo que también es muy útil.

Alguno se sorprendería de ver de qué sitio vienen algunos intentos ;-) .