

Supervisión de redes con Honeypots e IDSs

(y detección de ellos por un atacante)

Patricia Maldonado Mancilla

Ismael Montesinos Perujo

Jesús M. Pérez Terrón

Introducción	2
Motivación	2
Material usado y metodología	3
Kippo	3
Instalación y configuración de Kippo.	3
Funcionamiento de Kippo.	7
EasyIDS	10
Instalación y configuración	11
Funcionamiento	11
Detección de Honeypots en red	14
Conexión al servidor	14
TOP	14
LOGS	15
Gestión anormal de archivos	15
EXIT	16
Análisis con Wireshark	16
Detección de EasyIDS	17
Conclusiones	18
Referencias	19

Introducción

Vivimos en una época en la que internet está en todos lados y es un avance al que nadie quiere renunciar. De hecho, ocurre justo todo lo contrario: cada vez queremos más dispositivos weareables conectados, coches más conectados que nos informen de diversos eventos por internet, redes sociales que cada vez se integran más en nuestras vidas, etc. Inherente a este avance de Internet está el hecho de que supongamos que es un lugar seguro en todos los sentidos. El hecho es que por mucho que nos esforcemos en hacer Internet y las aplicaciones que de él dependen seguros, siempre va a haber fallas de seguridad. El problema es que muchas veces no sabemos el riesgo real al que nos exponemos y es algo a valorar ya que un sistema nunca se puede afirmar que sea 100% seguro.

Este trabajo versa sobre algunas de las diferentes técnicas que existen para detectar intrusos o, en general, alertas de seguridad en redes conectadas principalmente a internet. Esta detección se hará mediante un IDS (easyIDS) y un honeypot (Kippo, incluido en Honeydrive).

Motivación

Como ya sabemos, la detección temprana de peligros en nuestros equipos y en la red es la clave para reducir al máximo posible un ataque.

Esta detección suele ser pasiva con equipos que monitorizan la red a la escucha de paquetes que le resulten relevantes en algún aspecto.

Uno de los programas más usado para detección de anomalías en una red es SNORT. Pensamos que la configuración y puesta en marcha de un sistema con SNORT podía ser muy interesante por varias razones, como son que esté muy extendido su uso en el ámbito empresarial, que sea muy potente y completo y que esté constantemente actualizado. En nuestro caso, hemos usado una distribución que lleva SNORT instalado además de otras herramientas, llamada EasyIDS y basada en CentOS.

Otro programa que hemos usado es Honeydrive. Honeydrive es un honeypot sencillo y configurable que es muy útil para identificar ataques que se realicen a nuestros sistemas, tanto automatizados por robots como por personas.

Con estos dos programas podemos tener una amplia visión de los ataques hacia y en nuestra red y clasificarlos según su tipo e importancia.

Por último, queremos analizar también si estos sistemas son identificables por nuestros atacantes.

Material usado y metodología

Kippo

Un honeypot [1], es un sistema trampa, dispuesto en una red o sistema informático para ser objetivo de un ataque informático, el cual podemos detectar y obtener información sobre el mismo.

Es una forma de engañar a un atacante haciéndole creer que se encuentra perfilando y analizando un servicio del objetivo, cuando en realidad lo que está haciendo es suministrando información al objetivo sobre las actividades que está realizando.

Existen varios tipos de honeypots, ya sea por su tipo de funcionamiento más básico o más complejo. Podemos encontrarnos honeypots que tratan de emular un servicio en un puerto así como otros que emulan un sistema entero y guardan un log de casi todos los cambios que ocurren.

En nuestro caso hemos probado el funcionamiento de Kippo,[2][3] uno de los más conocidos desarrollado utilizando Python y Twisted (framework de red para programación dirigida por eventos). Este en concreto levanta un servicio SSH en el puerto que se le indique y registra distintos eventos. Es posible utilizarlo para localizar a un atacante e incluso, para que pueda iniciar sesión en el supuesto sistema y permitirle interactuar con un sistema de ficheros ficticio, el cual también es configurable.

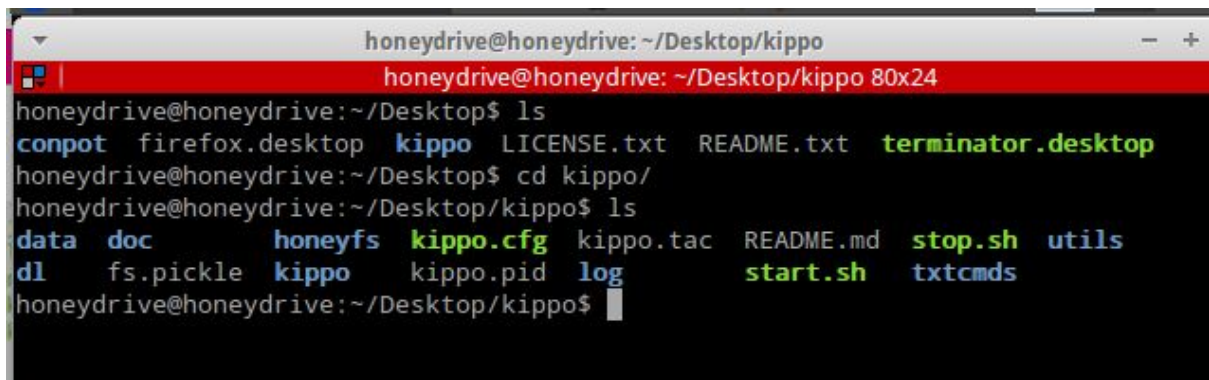
Instalación y configuración de Kippo.

Para la configuración de Kippo lo primero que tenemos que hacer es clonar el repositorio con las siguiente orden en la terminal:

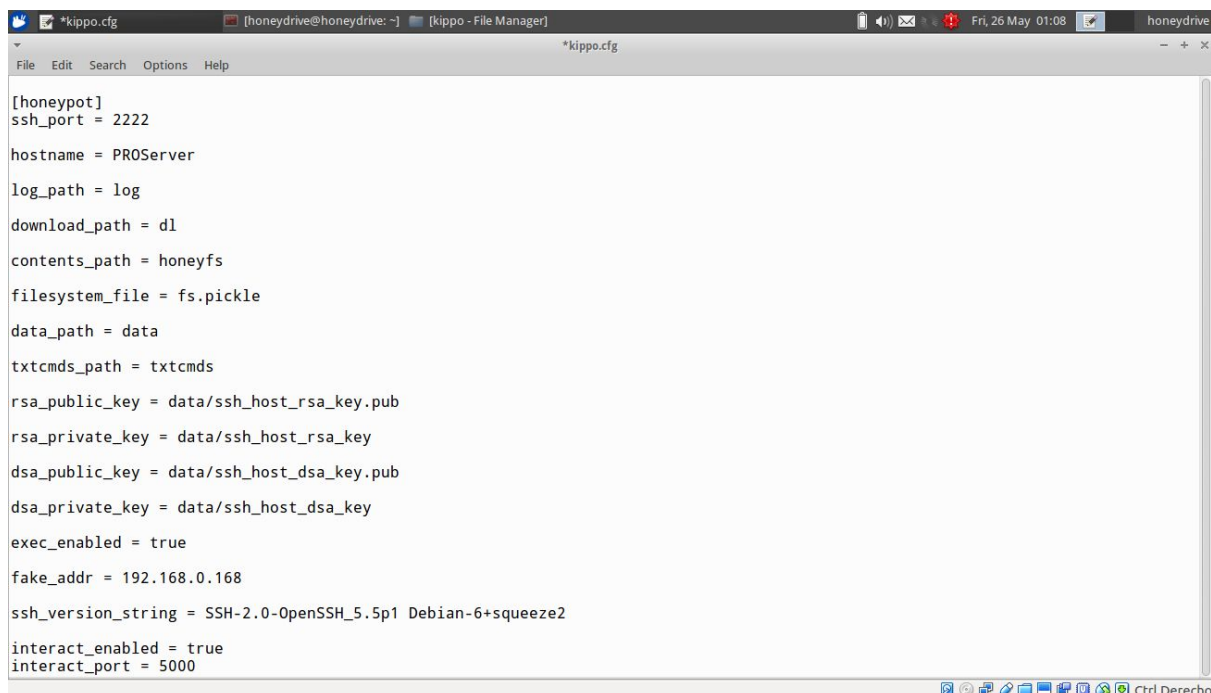
```
>git clone https://github.com/desaster/kippo.git
```

Para el buen funcionamiento de nuestro servicio además tenemos que instalar unas librerías fundamentales como son: Twisted, PyCrypto y service_identity Zope.

Una vez que hemos clonado el repositorio podemos ver en el directorio el archivo de configuración de Kippo, "kippo.cfg.dist" con una configuración por defecto. Pero lo más recomendable es crear un archivo nuevo llamado kippo.cfg en el que estableceremos una configuración básica que hará que sea más difícil que detecten que están atacando a un honeypot.



```
honeydrive@honeydrive: ~/Desktop/kippo
honeydrive@honeydrive: ~/Desktop/kippo 80x24
honeydrive@honeydrive:~/Desktop$ ls
conpot  firefox.desktop  kippo  LICENSE.txt  README.txt  terminator.desktop
honeydrive@honeydrive:~/Desktop$ cd kippo/
honeydrive@honeydrive:~/Desktop/kippo$ ls
data  doc  honeyfs  kippo.cfg  kippo.tac  README.md  stop.sh  utils
dl    fs.pickle  kippo  kippo.pid  log        start.sh  txtcmds
honeydrive@honeydrive:~/Desktop/kippo$
```



```
*kippo.cfg
[honeydrive@honeydrive: ~] [kippo - File Manager]
File Edit Search Options Help
*kippo.cfg
[honeypot]
ssh_port = 2222
hostname = PROServer
log_path = log
download_path = dl
contents_path = honeyfs
filesystem_file = fs.pickle
data_path = data
txtcmds_path = txtcmds
rsa_public_key = data/ssh_host_rsa_key.pub
rsa_private_key = data/ssh_host_rsa_key
dsa_public_key = data/ssh_host_dsa_key.pub
dsa_private_key = data/ssh_host_dsa_key
exec_enabled = true
fake_addr = 192.168.0.168
ssh_version_string = SSH-2.0-OpenSSH_5.5p1 Debian-6+squeeze2
interact_enabled = true
interact_port = 5000
```

A continuación se muestran algunas de las propiedades más importantes:

ssh_addr: Interfaz de red en la que iniciará el honeypot. El valor por defecto es "0.0.0.0"

ssh_port: Puerto en el que iniciará el honeypot. El valor por defecto es "2222"

hostname: Será cadena que verá un atacante cuando consiga una shell en el honeypot.

download_path: Directorio en el que se guardaran todos los ficheros que intente descargar el atacante en el servidor.

filesystem_file: esta propiedad se encarga de definir un sistema de ficheros completo, con sus correspondientes ficheros, permisos, etc. Este fichero se debe encontrar en formato de objetos serializados en Python (Pickle). Se puede utilizar el script "utils/createfd.py" para crear dicho sistema de ficheros partiendo de un sistema elegido por el usuario.

data_path: En el directorio indicado en esta propiedad se debe ubicar el fichero "usersdb.txt" en él se definirá por cada línea, el usuario y sus credenciales de acceso al supuesto honeypot. Es recomendable indicar una contraseña que sea fácilmente predecible para que el atacante pueda acceder.

txtcmds_path: En el directorio indicado en esta propiedad se deben definir los programas y utilidades que podrá ejecutar el atacante una vez se encuentre dentro del honeypot.

ssh_version_string: Será el banner que devolverá el servicio cuando se realicen peticiones sobre el puerto definido en la propiedad "ssh_port"

interact_enabled: Kippo permite controlar las sesiones que el atacante tenga iniciadas contra el honeypot en un servicio independiente de monitoreo. Por defecto dicha propiedad se encuentra desactivada, pero es muy útil para ver en tiempo real lo que está haciendo el atacante en el honeypot.

interact_port: Puerto en el que se iniciará el servicio de monitoreo. Solamente tiene sentido indicar un valor en esta propiedad si "interact_enabled" tiene el valor "true".

Dos archivos importantes de nuestro directorio kippo, son los scripts start.sh y stop.sh para iniciar y para parar el servicio, a los que deberemos de darles permisos.

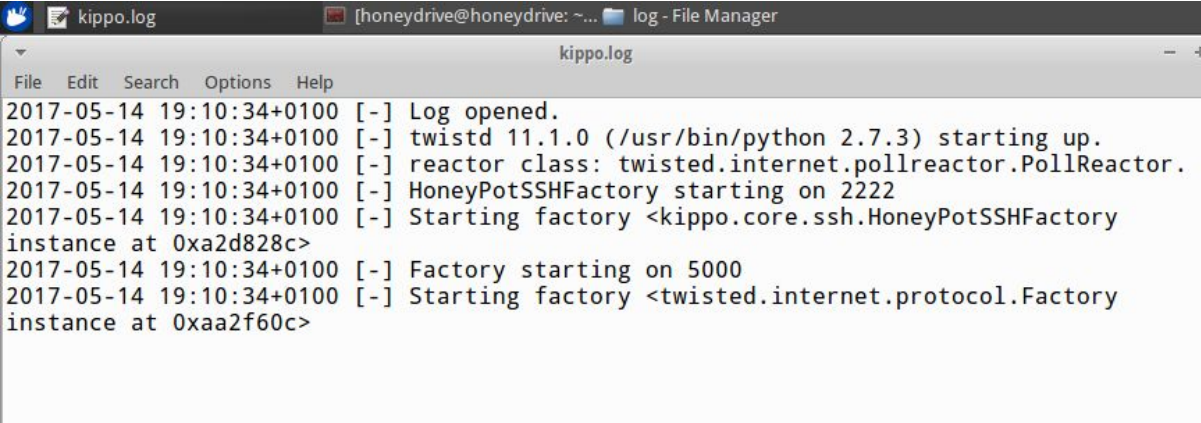
Procedemos a iniciar el script start.sh. Para comprobar que se ha iniciado correctamente ejecutamos la orden: netstat -antp, en la que listamos las conexiones tcp. Este caso visualizamos el puerto 5000 por el que se monitorea al atacante y el puerto 2222 donde haremos la escucha de nuestro honeypot.

```

honeydrive@honeydrive:~/Desktop/kippo$ netstat -antp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
-
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
-
tcp        0      0 0.0.0.0:5000            0.0.0.0:*               LISTEN
2575/python
tcp        0      0 127.0.0.1:27017         0.0.0.0:*               LISTEN
-
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
-
tcp        0      0 0.0.0.0:2222            0.0.0.0:*               LISTEN
2575/python
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
-
tcp        0      0 127.0.0.1:28017         0.0.0.0:*               LISTEN
-
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN
-
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
-
tcp        0      0 192.168.0.175:2222      192.168.0.158:44516     ESTABLISH
ED 2575/python
tcp6       0      0 :::22                   :::*                     LISTEN
-
honeydrive@honeydrive:~/Desktop/kippo$

```

Además en la carpeta log se encuentra el archivo kippo.log en el que podemos ver los logs que ha generado Kippo en el proceso de arranque, podemos observar que el honeypot está a la escucha en el puerto 2222, y el puerto 5000 para administración.



```

kippo.log
[honeydrive@honeydrive: ~... log - File Manager]
kippo.log
File Edit Search Options Help
2017-05-14 19:10:34+0100 [-] Log opened.
2017-05-14 19:10:34+0100 [-] twisted 11.1.0 (/usr/bin/python 2.7.3) starting up.
2017-05-14 19:10:34+0100 [-] reactor class: twisted.internet.pollreactor.PollReactor.
2017-05-14 19:10:34+0100 [-] HoneyPotSSHFactory starting on 2222
2017-05-14 19:10:34+0100 [-] Starting factory <kippo.core.ssh.HoneyPotSSHFactory
instance at 0xa2d828c>
2017-05-14 19:10:34+0100 [-] Factory starting on 5000
2017-05-14 19:10:34+0100 [-] Starting factory <twisted.internet.protocol.Factory
instance at 0xaa2f60c>

```

Para que capte más cantidad de atacantes, definiremos el honeypot con un puerto que no requiera privilegios de root para ser utilizado. Vamos a usar el puerto "2222"

y haremos un port forwarding al puerto 22. Para ello utilizaremos la siguiente regla de iptables:

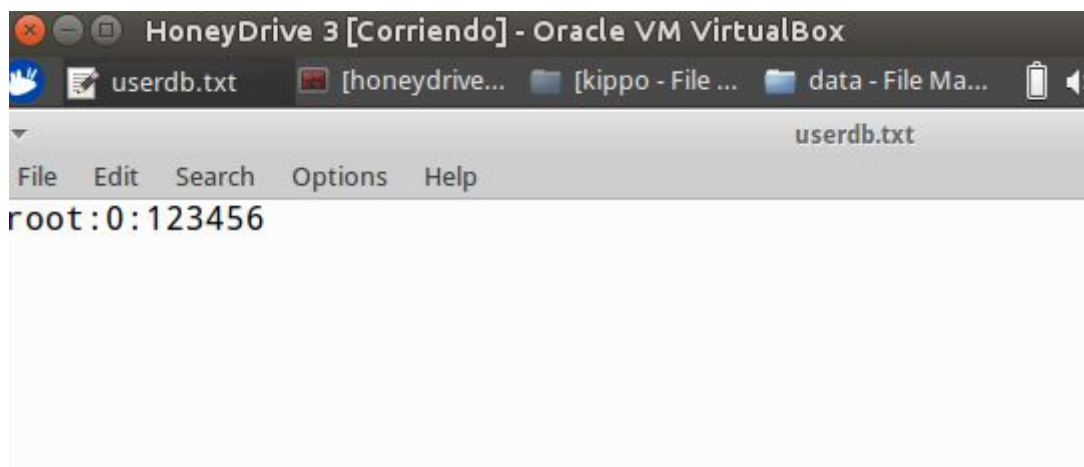
```
sudo iptables -t nat -A PREROUTING -p tcp --dport 22 -j REDIRECT --to-port 2222
```

Llegados hasta aquí ya tendríamos la máquina configurada correctamente para recibir ataques.

Funcionamiento de Kippo.

Una vez que el atacante descubre el equipo obtendremos un log de los intentos de sesión o de las sesiones iniciadas correctamente.

La contraseña y password de nuestro shell están almacenados en data/userdb.txt. En él podemos visualizar que el usuario es root y la contraseña 123456.



Desde la máquina atacante, nos conectamos mediante ssh por el puerto 2222 con la contraseña y password anteriormente mencionados.

```
ssh root@192.168.0.168
```


La máquina atacante solo tendrá disponible una serie de comandos que se definen en nuestro archivo de configuración log en la línea txtcmds_path. Por ejemplo aquí vemos como realiza una serie de comandos.

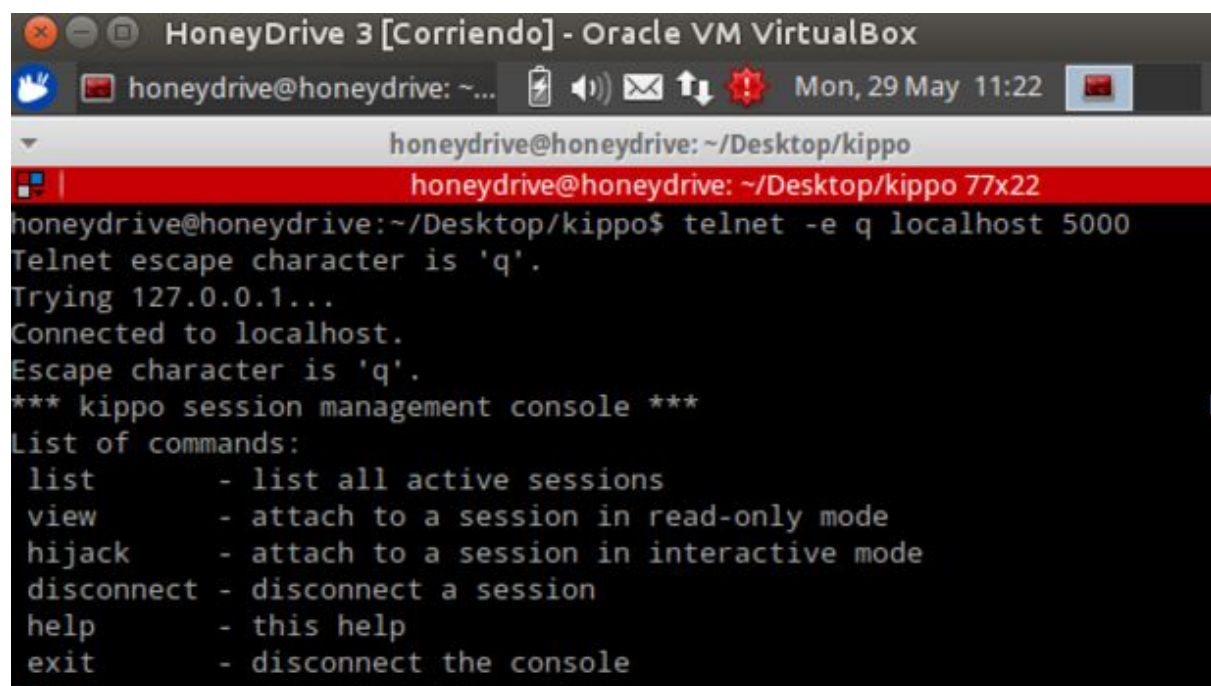
```
root@PROServer:~# whoami
root
root@PROServer:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:4c:a8:ab:32:f4
          inet addr:10.98.55.4  Bcast:10.98.55.255  Mask:255.255.255.0
          inet6 addr: fe80::21f:c6ac:fd44:24d7/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:84045991 errors:0 dropped:0 overruns:0 frame:0
          TX packets:103776307 errors:0 dropped:0 overruns:0 carrier:2
          collisions:0 txqueuelen:1000
          RX bytes:50588302699 (47.1 GiB)  TX bytes:97318807157 (90.6 GiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:308297 errors:0 dropped:0 overruns:0 frame:0
          TX packets:308297 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:355278106 (338.8 MiB)  TX bytes:355278106 (338.8 MiB)

root@PROServer:~#
```

Una vez que el atacante está conectado, desde nuestro honeypot podemos ver en tiempo real lo que está haciendo mediante telnet a través del puerto 5000:

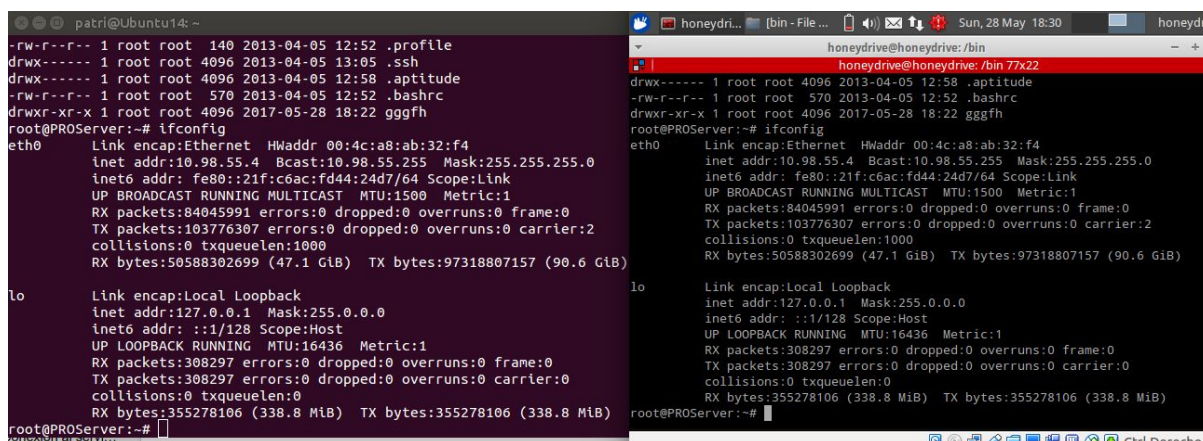
telnet -e q localhost 5000



```
HoneyDrive 3 [Corriendo] - Oracle VM VirtualBox
honeydrive@honeydrive: ~... Mon, 29 May 11:22
honeydrive@honeydrive: ~/Desktop/kippo
honeydrive@honeydrive: ~/Desktop/kippo 77x22
honeydrive@honeydrive:~/Desktop/kippo$ telnet -e q localhost 5000
Telnet escape character is 'q'.
Trying 127.0.0.1...
Connected to localhost.
Escape character is 'q'.
*** kippo session management console ***
List of commands:
list      - list all active sessions
view      - attach to a session in read-only mode
hijack    - attach to a session in interactive mode
disconnect - disconnect a session
help      - this help
exit      - disconnect the console
```

Existen una lista de comandos, por ejemplo con list vemos las sesiones activas asociadas a un id, en este caso vemos que hay una sesión activa con id 3, podremos acceder mediante view 3 para visualizar esa sesión y ver en tiempo real la interacción del atacante.

```
list
ID      clientIP      clientVersion
3       192.168.0.158      SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.8
```

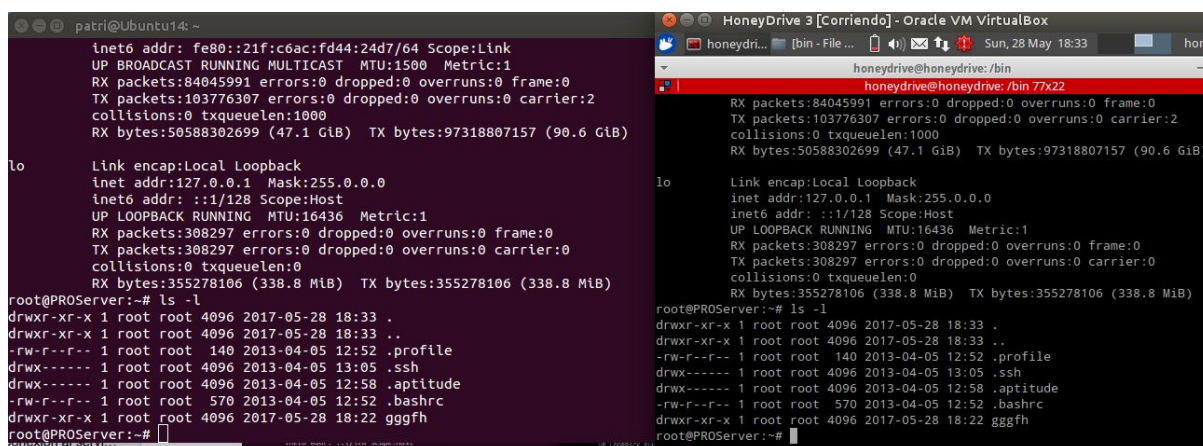


```
patri@Ubuntu14:~
-rw-r--r-- 1 root root 140 2013-04-05 12:52 .profile
drwx----- 1 root root 4096 2013-04-05 13:05 .ssh
drwx----- 1 root root 4096 2013-04-05 12:58 .aptitude
-rw-r--r-- 1 root root 570 2013-04-05 12:52 .bashrc
drwxr-xr-x 1 root root 4096 2017-05-28 18:22 gggfh
root@PROServer:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:4c:a8:ab:32:f4
          inet addr:10.98.55.4  Bcast:10.98.55.255  Mask:255.255.255.0
          inet6 addr: fe80::21f:c6ac:fd44:24d7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:84045991 errors:0 dropped:0 overruns:0 frame:0
          TX packets:103776307 errors:0 dropped:0 overruns:0 carrier:2
          collisions:0 txqueuelen:1000
          RX bytes:50588302699 (47.1 GiB)  TX bytes:97318807157 (90.6 GiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:308297 errors:0 dropped:0 overruns:0 frame:0
          TX packets:308297 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:355278106 (338.8 MiB)  TX bytes:355278106 (338.8 MiB)
root@PROServer:~#
```

```
honeydrive@honeydrive:/bin
honeydrive@honeydrive:/bin 77x22
drwx----- 1 root root 4096 2013-04-05 12:58 .aptitude
-rw-r--r-- 1 root root 570 2013-04-05 12:52 .bashrc
drwxr-xr-x 1 root root 4096 2017-05-28 18:22 gggfh
root@PROServer:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:4c:a8:ab:32:f4
          inet addr:10.98.55.4  Bcast:10.98.55.255  Mask:255.255.255.0
          inet6 addr: fe80::21f:c6ac:fd44:24d7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:84045991 errors:0 dropped:0 overruns:0 frame:0
          TX packets:103776307 errors:0 dropped:0 overruns:0 carrier:2
          collisions:0 txqueuelen:1000
          RX bytes:50588302699 (47.1 GiB)  TX bytes:97318807157 (90.6 GiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:308297 errors:0 dropped:0 overruns:0 frame:0
          TX packets:308297 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:355278106 (338.8 MiB)  TX bytes:355278106 (338.8 MiB)
root@PROServer:~#
```



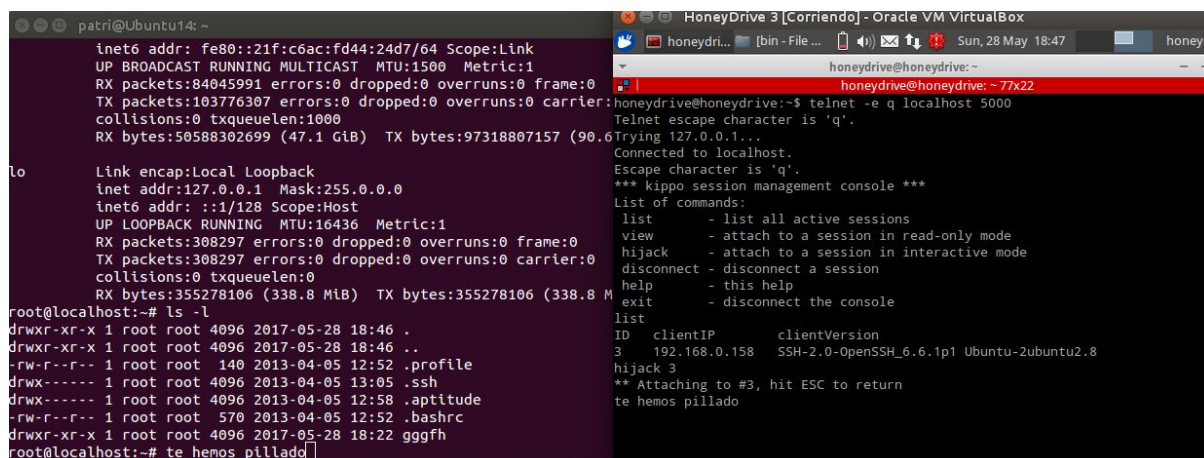
```
patri@Ubuntu14:~
          inet6 addr: fe80::21f:c6ac:fd44:24d7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:84045991 errors:0 dropped:0 overruns:0 frame:0
          TX packets:103776307 errors:0 dropped:0 overruns:0 carrier:2
          collisions:0 txqueuelen:1000
          RX bytes:50588302699 (47.1 GiB)  TX bytes:97318807157 (90.6 GiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:308297 errors:0 dropped:0 overruns:0 frame:0
          TX packets:308297 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:355278106 (338.8 MiB)  TX bytes:355278106 (338.8 MiB)
root@PROServer:~# ls -l
drwxr-xr-x 1 root root 4096 2017-05-28 18:33 .
drwxr-xr-x 1 root root 4096 2017-05-28 18:33 ..
-rw-r--r-- 1 root root 140 2013-04-05 12:52 .profile
drwx----- 1 root root 4096 2013-04-05 13:05 .ssh
drwx----- 1 root root 4096 2013-04-05 12:58 .aptitude
-rw-r--r-- 1 root root 570 2013-04-05 12:52 .bashrc
drwxr-xr-x 1 root root 4096 2017-05-28 18:22 gggfh
root@PROServer:~#
```

```
HoneyDrive 3 [Corriendo] - Oracle VM VirtualBox
honeydrive@honeydrive:/bin
honeydrive@honeydrive:/bin 77x22
          RX packets:84045991 errors:0 dropped:0 overruns:0 frame:0
          TX packets:103776307 errors:0 dropped:0 overruns:0 carrier:2
          collisions:0 txqueuelen:1000
          RX bytes:50588302699 (47.1 GiB)  TX bytes:97318807157 (90.6 GiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:308297 errors:0 dropped:0 overruns:0 frame:0
          TX packets:308297 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:355278106 (338.8 MiB)  TX bytes:355278106 (338.8 MiB)
root@PROServer:~# ls -l
drwxr-xr-x 1 root root 4096 2017-05-28 18:33 .
drwxr-xr-x 1 root root 4096 2017-05-28 18:33 ..
-rw-r--r-- 1 root root 140 2013-04-05 12:52 .profile
drwx----- 1 root root 4096 2013-04-05 13:05 .ssh
drwx----- 1 root root 4096 2013-04-05 12:58 .aptitude
-rw-r--r-- 1 root root 570 2013-04-05 12:52 .bashrc
drwxr-xr-x 1 root root 4096 2017-05-28 18:22 gggfh
root@PROServer:~#
```

y el comando más interesante es hijack para acceder a la sesión e interactuar con el atacante, en el que podríamos enviar un mensaje al atacante desde la consola.



```
patri@Ubuntu14: ~  
inet6 addr: fe80::21f:c6ac:fd44:24d7/64 Scope:Link  
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
RX packets:84045991 errors:0 dropped:0 overruns:0 frame:0  
TX packets:103776307 errors:0 dropped:0 overruns:0 carrier:0  
collisions:0 txqueuelen:1000  
RX bytes:50588302699 (47.1 GiB) TX bytes:97318807157 (90.6 GiB)  
Link encap:Local Loopback  
inet addr:127.0.0.1 Mask:255.0.0.0  
inet6 addr: ::1/128 Scope:Host  
UP LOOPBACK RUNNING MTU:16436 Metric:1  
RX packets:308297 errors:0 dropped:0 overruns:0 frame:0  
TX packets:308297 errors:0 dropped:0 overruns:0 carrier:0  
collisions:0 txqueuelen:0  
RX bytes:355278106 (338.8 MiB) TX bytes:355278106 (338.8 MiB)  
root@localhost:~# ls -l  
drwxr-xr-x 1 root root 4096 2017-05-28 18:46 .  
drwxr-xr-x 1 root root 4096 2017-05-28 18:46 ..  
-rw-r--r-- 1 root root 140 2013-04-05 12:52 .profile  
drwx----- 1 root root 4096 2013-04-05 13:05 .ssh  
drwx----- 1 root root 4096 2013-04-05 12:58 .aptitude  
-rw-r--r-- 1 root root 570 2013-04-05 12:52 .bashrc  
drwxr-xr-x 1 root root 4096 2017-05-28 18:22 gggfh  
root@localhost:~# te hemos pillado
```

```
HoneyDrive 3 [Corriendo] - Oracle VM VirtualBox  
honeydrive@honeydrive:~  
honeydrive@honeydrive:~ 77x22  
honeydrive@honeydrive:~$ telnet -e q localhost 5000  
Telnet escape character is 'q'.  
Trying 127.0.0.1...  
Connected to localhost.  
Escape character is 'q'.  
*** kippo session management console ***  
List of commands:  
list - list all active sessions  
view - attach to a session in read-only mode  
hijack - attach to a session in interactive mode  
disconnect - disconnect a session  
help - this help  
exit - disconnect the console  
list  
ID clientIP clientVersion  
3 192.168.0.158 SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.8  
hijack 3  
** Attaching to #3, hit ESC to return  
te hemos pillado
```

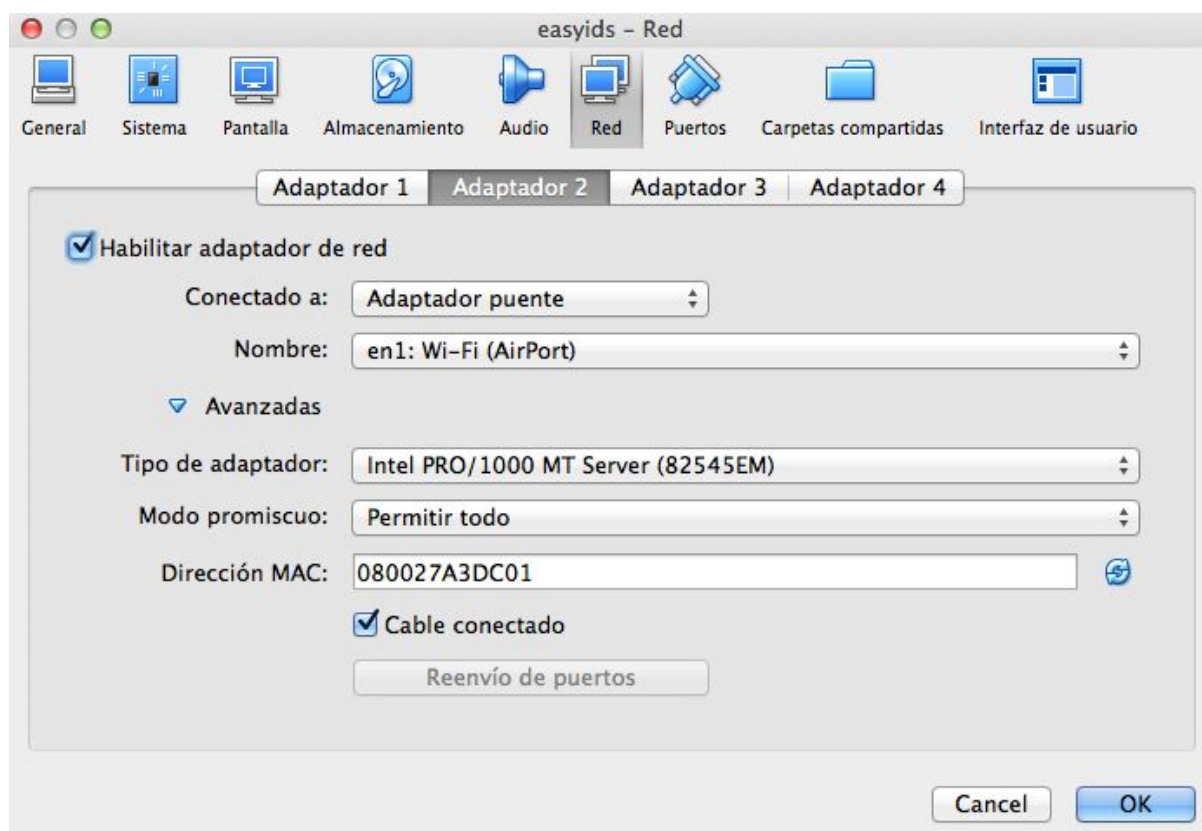
Y finalmente con el comando disconnect interrumpimos la sesión activa entre el atacante y el honeypot.

EasyIDS

EasyIDS es un sistema de detección de intrusos (IDS) equipado de serie con Snort. Según los creadores, la intención era crear una distribución de código abierto con todo lo necesario para montar un IDS rápidamente en cualquier ordenador con arquitectura x86. Hay que destacar que aunque las herramientas en las que se basa están actualizándose constantemente, el sistema en sí no recibe actualizaciones desde 2009.

Aunque tiene una buena documentación tanto en el sitio oficial como en otros foros por la red, la puesta en marcha varía mucho según la red en la que se vaya a incorporar. En nuestro caso hemos hecho una instalación en una máquina virtual que nos ha limitado en varios aspectos.

Para empezar, necesitábamos dos interfaces de red distintas: una para administración y otra para analizar la red en modo promiscuo. Si no se hacía así, Snort no era capaz de arrancar. La única configuración que nos ha funcionado es poniendo las interfaces como “modo-bridge” conectadas a la interfaz Wi-Fi del anfitrión.



Esta configuración hacía que tampoco funcionase la red conectados a eduroam en la facultad porque no nos daba una IP. También nos impedía analizar la red que estuviese fuera de nuestro host porque la tarjeta de red de este no podía entrar en promiscuo.

No obstante, hemos conseguido que funcionase y que detectase algunos eventos. Una vez que sabemos los problemas, si hubiese que instalarlo en un equipo real en producción, ya sabríamos hacerlo.

EasyIDS con la configuración básica de Snort, actúa como un IDS pasivo. Hemos comprobado cómo muestra los resultados de los eventos que genera y cómo se configura.

Instalación y configuración

Para empezar con la instalación nos vamos a la página oficial del proyecto y descargamos la ISO del sistema.

Después creamos una máquina en VirtualBox con los parámetros que he comentado en el campo de "red" (ESTE PASO ES CRUCIAL Y PUEDE DAR LUGAR A OTROS ERRORES).

Iniciamos una instalación típica como si de otra máquina con Linux se tratase.

Una vez iniciado, nos muestra la dirección IP a la que hay que acceder desde el navegador para administración.

Funcionamiento

The screenshot shows the EasyIDS web interface in a browser window. The URL bar shows `https://192.168.1.9`. The interface includes a navigation bar with links: Analysis, Graphs, Settings, Status, Tools, and Thanks. Below this, there's a section for 'Today's alerts' and 'Last 24 Hours alerts' with various filters like 'unique', 'listing', 'Source IP', and 'Destination IP'. A 'Traffic Profile by Protocol' section shows a bar chart with 'Portscan Traffic (94%)' being the most significant. A terminal window on the right shows the command `easyids login: root` and the URL `https://192.168.1.9`.

Si no hemos tenido ningún problema, se nos mostrará algo parecido a lo de la imagen anterior. Estamos dentro de BASE, una herramienta que nos muestra los avisos de Snort de una forma visual.

Aquí podemos seleccionar un aviso para que se nos muestre información relacionada.

The screenshot shows the EasyIDS web interface with a detailed alert view. The 'Queried on' date is 'Sun May 28, 2017 14:55:48'. The 'Summary Statistics' section shows 'Sensors', 'Unique Alerts', 'Unique addresses', 'Unique IP links', 'Source Port', 'Destination Port', and 'Time profile of alerts'. The main table displays alerts with columns: Signature, Classification, Total #, Sensor #, Source Address, Dest. Address, First, and Last. The first alert is '[snort] portscan: TCP PortswEEP' with a classification of 'attempted-recon' and a total of 6 alerts. The second alert is '[snort] portscan: Open Port' with a classification of 'unclassified' and a total of 298 alerts. The third alert is '[local] [snort] ICMP Destination Unreachable Communication with Destination Network is Administratively Prohibited' with a classification of 'misc-activity' and a total of 3 alerts. The fourth alert is '[local] [snort] ICMP Destination Unreachable Communication Administratively Prohibited' with a classification of 'misc-activity' and a total of 18 alerts. The fifth alert is '[local] [snort] ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited' with a classification of 'misc-activity' and a total of 8 alerts. The sixth alert is '[snort] portscan: TCP Portscan' with a classification of 'attempted-recon' and a total of 2 alerts. At the bottom, there's an 'ACTION' section with a dropdown menu and buttons for 'Selected' and 'ALL on Screen'.

Alert #0

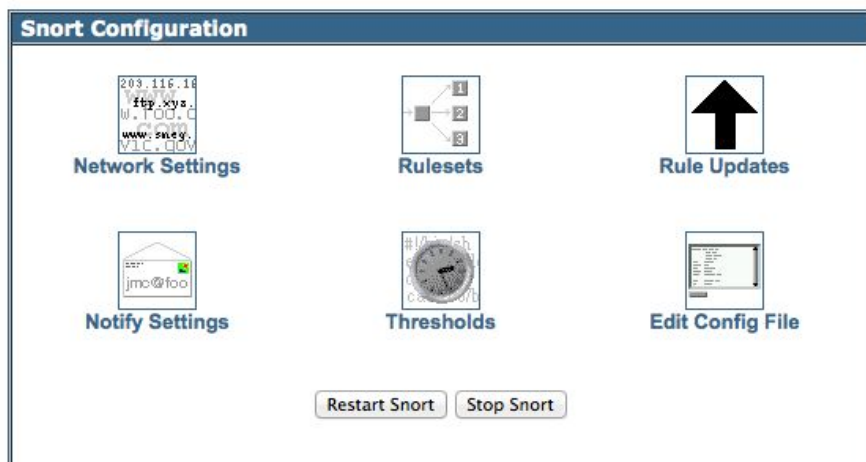
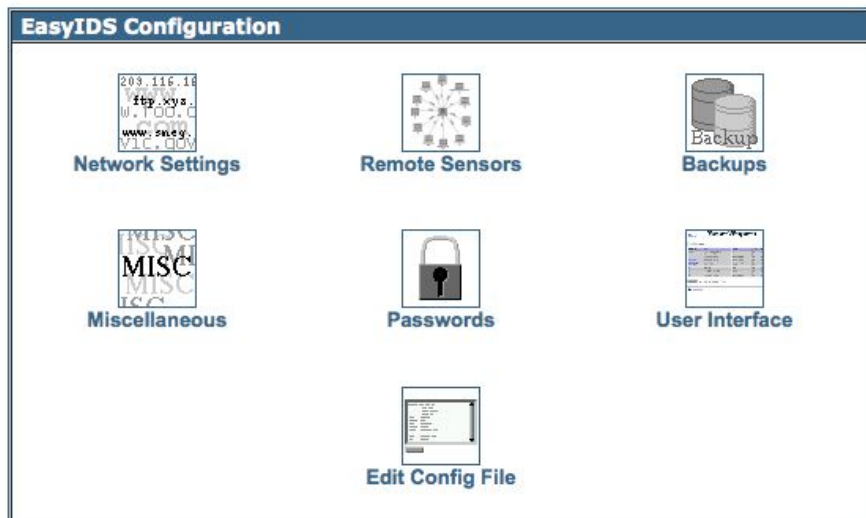
[First]

>> Next #1-(1-2)

Meta	ID #	Time		Triggered Signature								
	1 - 1	2017-05-14 11:59:19		[snort] portscan: TCP Portsweep								
	Sensor	Sensor Address	Interface	Filter								
		easyids	eth1	none								
Alert Group		none										
IP	Source Address	Dest. Address		Ver	Hdr Len	TOS	length	ID	fragment	offset	TTL	chksum
	192.168.1.6	216.58.210.131		4	20	0	162	13375	no	0	0	55473 = 0xd8b1
	Options	none										
Payload	length = 142											
Plain Display	000 : 50 72 69 6F 72 69 74 79 20 43 6F 75 6E 74 3A 20 Priority Count:											
Download of Payload	010 : 37 0A 43 6F 6E 6E 65 63 74 69 6F 6E 20 43 6F 75 7.Connection Cou											
	020 : 6E 74 3A 20 31 0A 49 50 20 43 6F 75 6E 74 3A 20 nt: 1.IP Count:											
	030 : 35 0A 53 63 61 6E 6E 65 64 20 49 50 20 52 61 6E 5.Scanned IP Ran											
	040 : 67 65 3A 20 35 34 2E 32 34 30 2E 31 38 36 2E 31 ge: 54.240.186.1											
Download in pcap format	050 : 33 3A 32 31 36 2E 35 38 2E 32 31 30 2E 31 33 31 3:216.58.210.131											
	060 : 0A 50 6F 72 74 2F 50 72 6F 74 6F 20 43 6F 75 6E .Port/Proto Coun											
	070 : 74 3A 20 32 0A 50 6F 72 74 2F 50 72 6F 74 6F 20 t: 2.Port/Proto											
	080 : 52 61 6E 67 65 3A 20 38 30 3A 34 34 33 0A Range: 80:443.											

[First]

>> Next #1-(1-2)



Aquí vemos información coleccionada de otros programas, en este caso Arpwatch mostrando los host que ha reconocido en la red.

Arpwatch

Known Hosts (27)				
< MAC Address >	< IP Address >	< Last Seen >	Hostname	Action
dc:85:de:c0:a0:69	192.168.0.158	2017-05-14 16:42:09		
0:23:12:d:6e:14	192.168.0.170	2017-05-14 16:45:34		
8:0:27:e5:8d:6a	192.168.0.171	2017-05-14 15:51:10		
8:0:27:a3:dc:1	192.168.0.172	2017-05-14 16:43:56		
8:0:27:e5:8d:6a	192.168.0.173	2017-05-14 16:37:58		
10:68:3f:fd:41:e	192.168.1.3	2017-05-24 22:11:58		
10:68:3f:fd:41:e	192.168.1.4	2017-05-14 13:12:59		
0:23:12:d:6e:14	192.168.1.6	2017-05-14 13:19:21		
8:0:27:e5:8d:6a	192.168.1.7	2017-05-14 13:19:20		
8:0:27:a3:dc:1	192.168.1.8	2017-05-28 14:51:45		
8:0:27:a3:dc:1	192.168.1.9	2017-05-28 14:51:45		
0:23:12:d:6e:14	192.168.43.20	2017-05-14 16:02:36		
a:0:27:0:0:0	192.168.56.1	2017-05-18 11:52:55		
8:0:27:6e:e9:e6	192.168.56.100	2017-05-18 11:59:58		
8:0:27:a3:dc:1	192.168.56.101	2017-05-18 11:52:55		
8:0:27:1e:ec:7e	192.168.56.102	2017-05-18 11:57:31		

Detección de Honeypots en red

Afrontemos ahora la otra perspectiva, estamos intentando atacar un servidor y queremos detectar si estamos ante un servidor real o a un honeypot.

Detectar que estás siendo víctima de un honeypot es algo crucial para cualquier atacante. Esto no solo significa que estás atacando a un servidor falso, también puede llevar connotaciones muy negativas para la seguridad del atacante. La mayoría de honeypots cuentan con un keylogger en la terminal para registrar todo lo que se teclee. También puede ser que tenga un malware que cree un backdoor a tu máquina, esto es una puerta en la que puede entrar a tu ordenador cada vez que se quiera. Esto nos podría delatar y es lo que no queremos.

Antes de seguir con nuestro posible ataque tenemos que detectar el honeypot. La mayoría de honeypots tienen un sistema que evita que detecten que es un honeypot, sin embargo algunas características propias de los honeypots pueden delatarlo, y es en lo que nos centraremos en las próximas páginas.

Conexión al servidor

Lo primero que nos damos cuenta nada más entrar es un tiempo de demora de la conexión excesivamente grande. Cuando en cualquier conexión por ssh no tarda más de 5 segundos, al conectarnos al honeypot nos damos cuenta como la conexión se puede alargar hasta 15 o 20 segundos. Esto puede deberse a que el honeypot tenga que cargar un servidor nuevo. Otro tema que llama la atención al conectarnos, es que cuando nos conectamos al honeypot, el texto de la contraseña es sólo 'Password:', cuando debería tener la estructura 'usuario@ip's password:'

TOP

Lo primero que debería hacer todo atacante, es ocultarse dentro del sistema atacado. Muchos payloads permiten el comando 'migrate', que permite ocultar tu proceso dentro de otro proceso del sistema que estés seguro que no se va a cerrar. Lo primero que debe hacer todo atacante entonces, es ejecutar el comando top para ver el PID del proceso al que quieres migrarte. Como vemos en la siguiente imagen, el comando top no está disponible.

```
root@PROServer:~# top
E82: Cannot allocate any buffer, exiting...
```


LOGS

Complementando un poco el comportamiento raro del punto anterior, ciertas carpetas como por ejemplo /var/log tampoco tienen un comportamiento normal. Si hacemos ls a esa carpeta vemos una lista de archivos de log: syslog, kern.log... En cualquier sistema, al hacer cat a alguno de estos archivos está permitido, sin embargo al hacer cat en, por ejemplo syslog nos aparece el siguiente mensaje de error:

```
root@PROServer:~# cat /var/log/syslog
cat: /var/log/syslog: No such file or directory
root@PROServer:~# ls /var/log/
dmesg.0      installer      mail.log      kern.log      aptitude      news
wtmpt        faillog        auth.log      lpr.log       user.log       fsck
syslog       dmesg          lastlog       mail.warn     mail.info     dpkg.log
daemon.log   alternatives.log mail.err       btmp          debug          apt
messages
root@PROServer:~#
```

Aquí podemos ver como existen distintos archivos, pero al realizar cat sobre uno de ellos, nos dice que no existe el archivo.

Gestión anormal de archivos

Ya visto un poco el comportamiento anormal del servidor, podemos ir pensando que estamos siendo víctimas de un honeypot. Al sospechar algo podemos abrir otro terminal y probar lo siguiente: un honeypot es un sistema aislado, entonces toda modificación sobre una sesión, no modificará el comportamiento en otra sesión que tengamos abierta. Podemos aprovecharnos de esto para borrar un archivo, por ejemplo algún archivo de log. Como podemos observar en la siguiente imagen en la que tenemos dos terminales conectados al mismo servidor, en uno borramos un archivo. En la otra sesión, buscamos ese archivo y efectivamente, no existe el archivo:

```
root@PROServer:~#
dmesg.0      installer      mail.log
kern.log     aptitude      news
wtmpt        faillog        auth.log
lpr.log      user.log      fsck
syslog       dmesg         lastlog
mail.warn    mail.info     dpkg.log
daemon.log   alternatives.log mail.err
btmp         debug         apt
messages
root@PROServer:~# rm /var/log/mail.log
root@PROServer:~# ls /var/log
dmesg.0      installer      kern.log
aptitude     news          wtmp
faillog      auth.log      lpr.log
user.log     fsck          syslog
dmesg        lastlog       mail.warn
mail.info    dpkg.log      daemon.log
alternatives.log mail.err      btmp
debug        apt           messages
root@PROServer:~#
```

```
root@PROServer:~# ls /var/log
dmesg.0      installer      mail.log
kern.log     aptitude      news
wtmpt        faillog        auth.log
lpr.log      user.log      fsck
syslog       dmesg         lastlog
mail.warn    mail.info     dpkg.log
daemon.log   alternatives.log mail.err
btmp         debug         apt
messages
root@PROServer:~#
```

EXIT

Estamos ya casi 100% seguros de que estamos ante un honeypot, y lo primero que queremos es cerrar la conexión lanzando el típico comando 'exit'. Pues bien aquí nos encontramos otro problema. Si bien este comando no solo no cierra la conexión, sino que te hace creer que sí la has cerrado:

```
Connection to server closed.  
root@localhost:~#
```

Una vez nos limpia el terminal, nos dice que la conexión con el servidor se ha cerrado. Como vemos, el usuario sigue siendo root, pero la ip ha cambiado, nos dice que estamos en localhost.

Obviamente es totalmente falso. Seguimos conectados al servidor y mandando con el keylogger propio del honeypot todo lo que escribamos en la terminal.

Análisis con Wireshark

Al ejecutar el comando exit mientras se captura la conexión con wireshark, vemos bastantes diferencias. Esta es una conexión y exit de una conexión normal a un servidor corriente:

39	14.051684686	10.0.2.4	10.0.2.5	SSHv2	102 Client: Encrypted packet (len=102)
40	14.052826291	10.0.2.5	10.0.2.4	SSHv2	102 Server: Encrypted packet (len=102)
41	14.052892182	10.0.2.4	10.0.2.5	TCP	66 32790 → 22 [ACK] Seq=2398 Acl=102
42	14.405752477	10.0.2.4	10.0.2.5	SSHv2	102 Client: Encrypted packet (len=102)
43	14.406826835	10.0.2.5	10.0.2.4	SSHv2	102 Server: Encrypted packet (len=102)
44	14.406860899	10.0.2.4	10.0.2.5	TCP	66 32790 → 22 [ACK] Seq=2434 Acl=102
45	14.572534247	10.0.2.4	10.0.2.5	SSHv2	102 Client: Encrypted packet (len=102)
46	14.573713872	10.0.2.5	10.0.2.4	SSHv2	102 Server: Encrypted packet (len=102)
47	14.573757926	10.0.2.4	10.0.2.5	TCP	66 32790 → 22 [ACK] Seq=2470 Acl=102
48	14.786214713	10.0.2.4	10.0.2.5	SSHv2	102 Client: Encrypted packet (len=102)
49	14.787157420	10.0.2.5	10.0.2.4	SSHv2	102 Server: Encrypted packet (len=102)
50	14.787193527	10.0.2.4	10.0.2.5	TCP	66 32790 → 22 [ACK] Seq=2506 Acl=102
51	15.762766799	10.0.2.4	10.0.2.5	SSHv2	102 Client: Encrypted packet (len=102)
52	15.764900160	10.0.2.5	10.0.2.4	SSHv2	214 Server: Encrypted packet (len=102)
53	15.764950783	10.0.2.4	10.0.2.5	TCP	66 32790 → 22 [ACK] Seq=2542 Acl=102
54	15.764972017	10.0.2.5	10.0.2.4	SSHv2	138 Server: Encrypted packet (len=102)
55	15.764975345	10.0.2.4	10.0.2.5	TCP	66 32790 → 22 [ACK] Seq=2542 Acl=102
56	15.765137513	10.0.2.4	10.0.2.5	SSHv2	102 Client: Encrypted packet (len=102)
57	15.765202859	10.0.2.4	10.0.2.5	SSHv2	126 Client: Encrypted packet (len=102)
58	15.765242995	10.0.2.4	10.0.2.5	TCP	66 32790 → 22 [FIN, ACK] Seq=2639 Acl=102
59	15.765359233	10.0.2.5	10.0.2.4	TCP	66 22 → 32790 [ACK] Seq=3442 Acl=102
60	15.784509230	10.0.2.5	10.0.2.4	TCP	66 22 → 32790 [FIN, ACK] Seq=3442 Acl=102
61	15.784544021	10.0.2.4	10.0.2.5	TCP	66 32790 → 22 [ACK] Seq=2639 Acl=102

Desde el paquete 39 empieza nuestro intento de salida. vemos cómo envía dos paquetes ssh, y recibe un ACK. Cada dos paquetes y ACK, es una letra escrita en la terminal. A partir del paquete 54 empezamos a salir, paquete 58 nuestro sistema con ip 10.0.2.4 pide fin, y en el paquete 60 el servidor cierra la conexión ssh.

Vamos a ver las diferencias ahora con un honeypot:

55	47.777937458	10.0.2.4	10.0.2.15	SSHv2	118 Client: Encrypted packet (lei
56	47.780366930	10.0.2.15	10.0.2.4	SSHv2	118 Server: Encrypted packet (lei
57	47.780408230	10.0.2.4	10.0.2.15	TCP	66 48342 → 22 [ACK] Seq=3774 Acl
58	48.066775213	10.0.2.4	10.0.2.15	SSHv2	118 Client: Encrypted packet (lei
59	48.069213622	10.0.2.15	10.0.2.4	SSHv2	118 Server: Encrypted packet (lei
60	48.069256174	10.0.2.4	10.0.2.15	TCP	66 48342 → 22 [ACK] Seq=3826 Acl
61	48.183809661	10.0.2.4	10.0.2.15	SSHv2	118 Client: Encrypted packet (lei
62	48.186099632	10.0.2.15	10.0.2.4	SSHv2	118 Server: Encrypted packet (lei
63	48.186139130	10.0.2.4	10.0.2.15	TCP	66 48342 → 22 [ACK] Seq=3878 Acl
64	48.380207032	10.0.2.4	10.0.2.15	SSHv2	118 Client: Encrypted packet (lei
65	48.383286490	10.0.2.15	10.0.2.4	SSHv2	118 Server: Encrypted packet (lei
66	48.383329733	10.0.2.4	10.0.2.15	TCP	66 48342 → 22 [ACK] Seq=3930 Acl
67	48.983404597	10.0.2.4	10.0.2.15	SSHv2	118 Client: Encrypted packet (lei
68	48.986218984	10.0.2.15	10.0.2.4	SSHv2	462 Server: Encrypted packet (lei
69	48.986248937	10.0.2.4	10.0.2.15	TCP	66 48342 → 22 [ACK] Seq=3982 Acl
70	51.801318499	10.0.2.4	10.0.2.15	SSHv2	134 Client: Encrypted packet (lei
71	51.801745099	10.0.2.4	10.0.2.15	TCP	66 48342 → 22 [FIN, ACK] Seq=4051 Acl
72	51.802645921	10.0.2.15	10.0.2.4	TCP	66 22 → 48342 [FIN, ACK] Seq=48342 Acl
73	51.802667601	10.0.2.4	10.0.2.15	TCP	66 48342 → 22 [ACK] Seq=4051 Acl

Aparentemente ambos son iguales, empezando nuestro intento de salida en el paquete 59. Al intentar salir vemos la principal diferencia... Desde que nuestro sistema manda fin en un servidor normal, hacen falta 4 paquetes. Un fin de nuestro sistema, un ACK por parte del servidor, el servidor pide cerrar conexión, y nosotros mandamos un ACK. En un honeypot, sólo hacen falta 3 paquetes para "cerrar" conexión. El servidor no confirma que la conexión se va a cerrar, que de hecho no la cierra.

Detección de EasyIDS

Así mismo, también hubiese sido posible detectar a EasyIDS en nuestra red monitorizando paquetes. Si tenemos más de un sensor en la red se generarían paquetes cifrados con Stunnel (tal y como vemos en la siguiente captura) que nos indicarían la muy posible presencia de este IDS. Este es solo un posible ejemplo.

December 7, 2009 - v0.4

- Designed around Centos 5.4 cd1 with updates.
- Upgraded Snort to 2.8.5.1.
- Upgraded Snort rulesets to 2.8.
- Upgraded BASE to customized version 1.4.4.
- Upgraded ntop to 3.3.8.
- Added Arpwatch 2.1a13.
- Upgraded Nmap to 4.11.
- Added stunnel 4.15.
- Added network traffic graphs (Daily, Weekly, etc).
- Added system usage graphs (Daily, Weekly, etc).
- Modified Snort performance graphs (Daily, Weekly, etc).
- Web selectable management/monitoring network NICs.
- Bridging support for inline placement if 3+ NICs.
- Multiple remote sensor support with Stunnel encryption.
- Added auto restart of failed services with notification script.

Conclusiones

Como conclusión principal tras la realización de este trabajo debemos decir que la implantación de un sistema de supervisión de red es costoso.

Costoso en tiempo, pues hemos estado mucho tiempo (más de un mes) intentando que alguno de los sistemas empezase a funcionar.

Costoso también económicamente, ya que han sido tres personas que han dedicado bastante tiempo a empezar a hacer que funcione el sistema. Un sistema con además bastantes limitaciones y problemas detectables por intrusos si no se le dedica más tiempo a su configuración.

Pero quedándonos con lo bueno, hemos aprendido bastante los tres sobre estos sistemas de detección de intrusos ya que nos hemos ayudado los unos a los otros para ver qué tal iba cada parte.

Hemos aprendido que la configuración básica de uno de estos sistemas no puede ser un punto y final a su implantación y que se necesita seguir muy a fondo con su configuración para que sea utilizable en un entorno profesional. De lo contrario, sería rápidamente identificable el sistema y los atacantes tratarían de tomar medidas para esquivarlos que no podríamos detectar.

Por tanto, unos buenos honeypots y un buen IDS bien configurados, pueden ser de gran ayuda para los administradores de sistemas que necesitan estar al tanto de las incidencias que ocurren tanto dentro de su red como de las que vienen del exterior.

Referencias

- [1]Honeypot: 2017. <https://es.wikipedia.org/wiki/Honeypot>. Accessed: 2017- 05- 20.
- [2]HoneyPots Parte 1 – Kippo: 2017.
<https://thehackerway.com/2015/03/24/honeypots-parte-1-kippo/>. Accessed: 2017- 05- 21.
- [3]Probando Kippo, un honeypot de SSH en Ubuntu - Un blog mas: 2017.
<http://pablo.sarubbi.com.ar/instalaciones/probando-kippo-un-honeypot-de-ssh-en-ubuntu>. Accessed: 2017- 05- 21.
- [4]HoneyPots Parte 3 – Detección de un HoneyPot: 2017.
<https://security.stackexchange.com/questions/90642/is-it-possible-to-detect-a-honeypot> . Accessed: 2017- 05- 24.
- [5]HoneyPots Parte 3 – Detección de un HoneyPot: 2017.
<http://ro.ecu.edu.au/cgi/viewcontent.cgi?article=1027&context=adf>. Accessed: 2017- 05- 24
- [6]Skynet Solutions, Inc. | Web Design | E-commerce | Web Programming | Tulsa, OK | Broken Arrow, OK : EasyIDS: 2017.
<https://www.skynet-solutions.net/development/our-software/easyids/>. Accessed: 2017- 05- 27.
- [7]EasyIDS | Enredando con redes ...: 2017. <https://enredandoconredes.com/tag/easyids/>. Accessed: 2017- 05- 14.
- [8]Manual de instalación y configuración de EasyIDS – Snort: 2017.
<https://humanliks.wordpress.com/seguridad-de-la-informacion/manual-de-instalacion-y-configuracion-de-easyids-snort/>. Accessed: 2017- 05- 05.