



ESCUELA
COLOMBIANA
DE INGENIERÍA
JULIO GARAVITO

INSTALACIÓN DE KISMET

INSTALACIÓN DE KISMET

Para la instalar kismet en Linux solo es escribir los siguientes comandos en la terminal:

- **Sudo apt-get update**
- **Sudo apt-get Install kismet**

Como el IDS Kismet fue instalado en una Raspberry Pi 3, es importante saber que la tarjeta de red de este dispositivo no se deja cambiar a modo monitor, por lo tanto fue necesario implementarle la siguiente antena USB:



Configuración de la antena: Para cambiar la tarjeta a modo monitor se escriben los siguientes comandos en la consola:

- **sudo ifconfig wlan0 down**
- **sudo iwconfig wlan0 mode monitor**
- **sudo ifconfig wlan0 up**

*La interface es a la que está conectada la antena, para mirar escriba el comando: ifconfig

Configuración de kismet:

Vamos al archivo de configuración de kismet `kismet.conf` que generalmente se encuentra en la ruta `/etc/kismet/kismet.conf` y la abrimos con el siguiente comando:

- **nano kismet.conf**

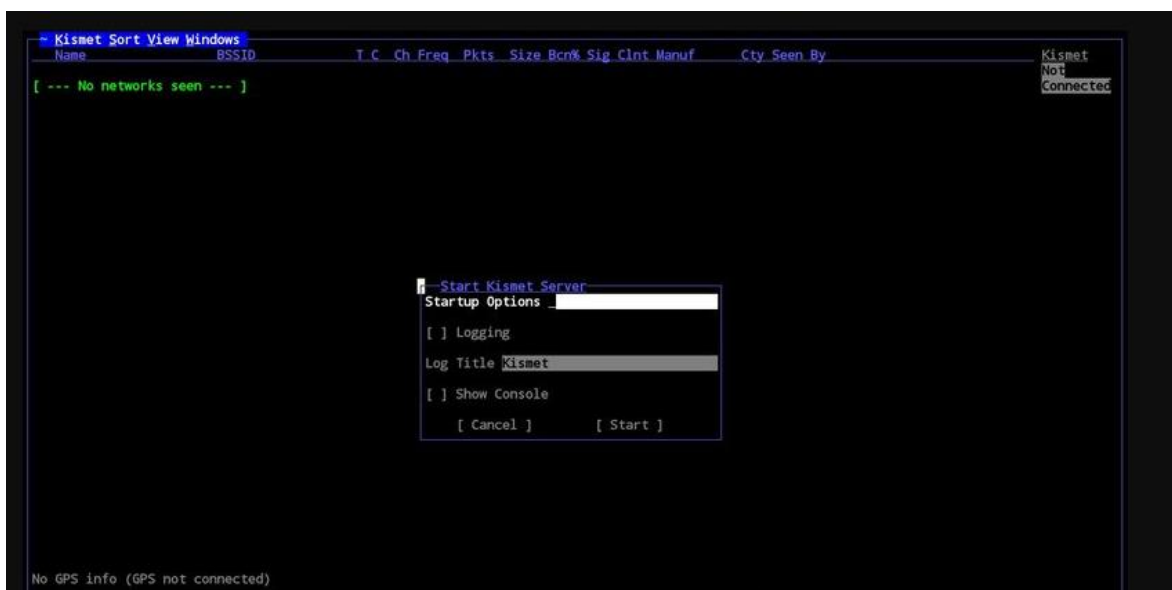
Añadimos al final del archivo `"ncsource = wlan0"` sin las `"`, para que kismet reconozca por de interface se va hacer el escaneo.

Corriendo Kismet:

Para correr kismet ese escribe el comando:

- **kismet**

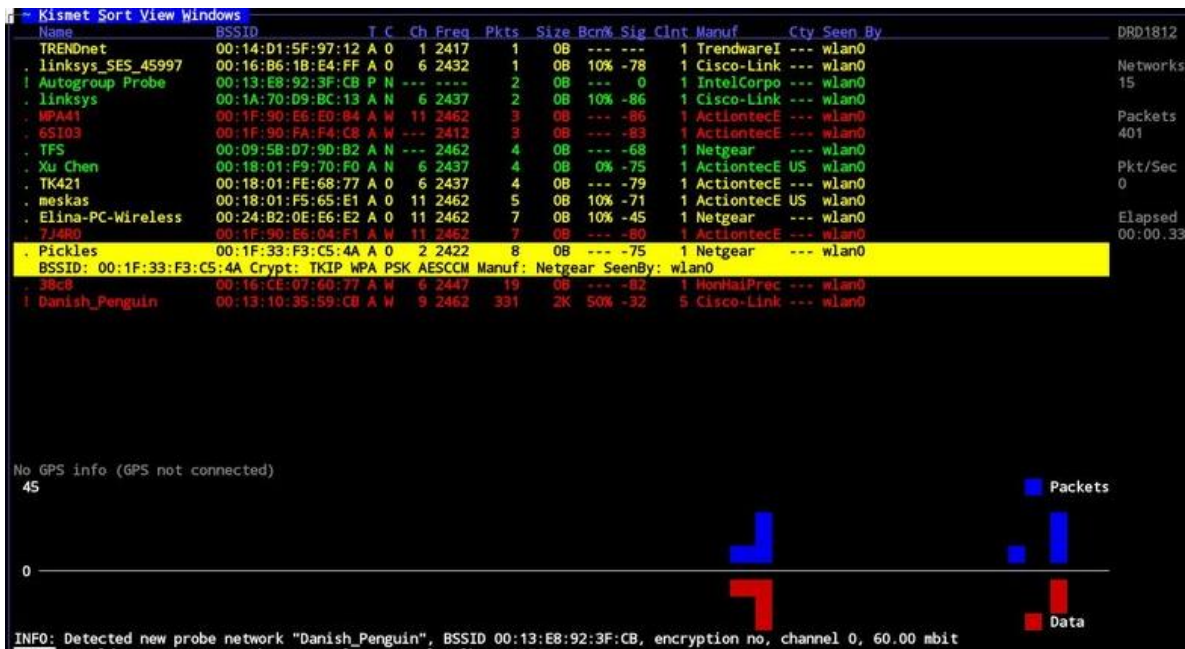
Se abrirá una ventana de kismet:



Si queremos ver la información por la consola de kismet le damos click en **[]Show Console**

Le damos click en **Start**.

Y kismet empieza a monitorer las redes WIFI.



CONFIGURACIÓN KISMET EN OSSIM

Para cuando Kismet genere la alerta y la pueda enviar a OSSIM se debe tener corriendo el programa que está en el siguiente Github

<https://github.com/andresvega82/SIEM-IoT/tree/master/Software/Kismet>

en el archivo [Syslogkismet.zip](#), el cual envía el syslog al OSSIM para que este pueda recibir el evento.

En la siguiente imagen podemos ver en color amarillo el código donde se envía el syslog:

```
// Initialise sender
UdpSyslogMessageSender messageSender = new UdpSyslogMessageSender();
messageSender.setDefaultMessageHostname(""); // some syslog cloud services may use this field to t
messageSender.setDefaultAppName("kismet");
messageSender.setDefaultFacility(Facility.USER);
System.out.println("lo que tiene USER"+Facility.USER.name());
messageSender.setDefaultSeverity(Severity.ALERT);
//messageSender.setSyslogServerHostname("10.2.78.8");
messageSender.setSyslogServerHostname("192.168.0.5");
messageSender.setSyslogServerPort(514);
messageSender.setMessageFormat(MessageFormat.RFC_3164);

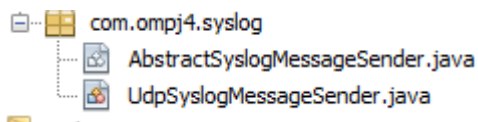
try {
    FileReader fr = new FileReader(fichero);
    BufferedReader br = new BufferedReader(fr);

    String linea;
    while((linea = br.readLine()) != null){
        lineas.add(linea);
        //System.out.println(linea);
    }

    for(int i= 0; i<lineas.size(); i++){
        System.out.println(lineas.get(i));
        // send a Syslog message

        try{
            messageSender.sendMessage(lineas.get(i));
            System.out.println("el mensaje enviado fue:"+ lineas.get(i));}catch (IOException ex) {
        Logger.getLogger(ex.getMessage());
    }
}
```

****Es necesario incluir las clases:**



****Para correr el programa se genera un .jar**

**** y se ejecuta con el comando `java -jar <el nombre del archivo >.Jar`.**

Ya con lo anterior se envía el syslog al OSSIM , pero para indicarles que campos del syslog queremos que se vea en el evento, es necesario agregar en la plataforma de OSSIM el plugin [kismetloTPlugin.cfg](#), el cual se encuentra en el Github mencionado anteriormente, el cual contiene la expresión regular:

```
[syslog - KismetSyslog]
event_type=event
regex="([\\s\\S]+) (?<src_ip>\\d{1,3}\\d{1,3}\\d{1,3}\\d{1,3}) (kismet: )([\\s\\S]+)(?<date>\\w+ \\d\\d \\d\\d:\\d\\d:\\d\\d)([\\s\\S]+)(BCASTDISCON
plugin_sid=1
src_ip={resolve($src_ip)}
username={$date}
userdata2={$channel}
userdata3={$src_mac}
userdata4={$dest_mac}
userdata5={$chan}
```

Continuación de la expresión regular

```
(?<channel>\\d+) (?<src_mac>\\w\\w(:)\\w\\w(:)\\w\\w(:)\\w\\w(:)\\w\\w(:)\\w\\w) (?<dest_mac>\\w\\w(:)\\w\\w(:)\\w\\w(:)\\w\\w(:)\\w\\w) (?<chan>[\\s\\S]+)"
```

La cual hará que el OSSIM reciba y muestre el evento de la siguiente forma:

```
Nov 25 11:45:57 10.2.67.240 -i Nov 25 16:10:53 10.2.67.240 kismet: Sat Nov 25 16:10:53 2017 BCASTDISCON 0 90:F6:52:89:7C:B2 90:F6:52:89:7C:B2
FF:FF:FF:FF:FF:FF 00:00:00:00:00:00 Network BSSID 90:F6:52:89:7C:B2 broadcast deauthenticate/disassociation of all clients, possible DoS
```

PRUBAR KISMET

Generar una alerta en Kismet:

En este escenario vamos a generar la alerta **BCASTDISCON** la cual se dispara cuando detecta que hay un ataque de desasociación de la red de un cliente o de varios, causando una posible denegación de servicio.

Debemos tener el Kismet corriendo:

Comando : **Kismet**

Para generar el ataque vamos a utilizar la herramienta **aireplay-ng**, en nuestro caso vamos a desconectar a todos los clientes conectados a la red:

Escribimos el siguiente comando:

- **aireplay-ng -deauth 0 -a < BSSID> wlan1**

- * el 0(cero) indica que va a realizar el ataque indefinidamente, si queremos que solo se realice 1 vez, cambiamos el 0 por 1.
- * wlan1 es la interface de red.
- * El BSSID es el punto de acceso por el cual se va a realizar el ataque.

BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER
BC:CA:B5:D2:A5:B0	-63	590	63	0	7	54e	WPA2 COMP
98:6B:3D:1D:94:F0	-67	583	583	0	8	54e	WPA2 COMP
A4:7E:39:AA:00:AB	-77	514	26	0	6	54e	WPA COMP
58:23:BC:83:CE:34	-81	513	5211	0	6	54e	WPA2 COMP
5C:B9:01:6E:13:4E	-73	465	0	0	6	54e	WPA2 COMP
CC:96:A0:02:B5:4B	0	450	3	0	11	54	WPA TKIP

Cualquiera que se encuentre en la red para visualizar mejor estas BSSID podemos utilizar el comando: **airodump-ng wlan1**, utilizando la interface de red que tengamos en el momento.

Luego le damos Enter y se empieza el ataque, se demora un poco ya que tiene que coincidir el canal(CH) del BSSID con el de la interface de red(wlan1).

```

05:27:16 wlan1 is on channel 2, but the AP uses channel 11
root@kali:~# airoplay-ng --deauth 0 -a CC:96:A0:02:B5:4B wlan1
05:27:17 Waiting for beacon frame (BSSID: CC:96:A0:02:B5:4B) on channel 11
NB: this attack is more effective when targeting
a connected wireless client (-c <client's mac>).
05:27:18 Sending DeAuth to broadcast -- BSSID: [CC:96:A0:02:B5:4B]
^[[05:27:18 Sending DeAuth to broadcast -- BSSID: [CC:96:A0:02:B5:4B]
05:27:19 Sending DeAuth to broadcast -- BSSID: [CC:96:A0:02:B5:4B]
05:27:19 Sending DeAuth to broadcast -- BSSID: [CC:96:A0:02:B5:4B]
05:27:20 Sending DeAuth to broadcast -- BSSID: [CC:96:A0:02:B5:4B]
05:27:20 Sending DeAuth to broadcast -- BSSID: [CC:96:A0:02:B5:4B]
05:27:21 Sending DeAuth to broadcast -- BSSID: [CC:96:A0:02:B5:4B]
05:27:21 Sending DeAuth to broadcast -- BSSID: [CC:96:A0:02:B5:4B]
05:27:22 Sending DeAuth to broadcast -- BSSID: [CC:96:A0:02:B5:4B]
05:27:22 Sending DeAuth to broadcast -- BSSID: [CC:96:A0:02:B5:4B]
05:27:23 Sending DeAuth to broadcast -- BSSID: [CC:96:A0:02:B5:4B]
05:27:23 Sending DeAuth to broadcast -- BSSID: [CC:96:A0:02:B5:4B]
05:27:24 Sending DeAuth to broadcast -- BSSID: [CC:96:A0:02:B5:4B]
05:27:24 Sending DeAuth to broadcast -- BSSID: [CC:96:A0:02:B5:4B]

```

En la imagen anterior ponemos ver que empieza a enviar paquetes de desasociación por el punto que acceso que definimos y de este modo generar una denegación de servicio.

Kismet detecta el ataque:

```
INFO: Detected new probe network "HOLLYWOOD", BSSID 7C:8B:C6:FA:9F:FA,  
encryption no, channel 6, 72.28 mbit  
ALERT: BCASTDISCON Network BSSID CC:96:A0:02:B5:4B broadcast deauthenticate  
/disassociation of all clients, possible DoS  
ALERT: BCASTDISCON Network BSSID CC:96:A0:02:B5:4B broadcast deauthenticate  
/disassociation of all clients, possible DoS  
ALERT: BCASTDISCON Network BSSID CC:96:A0:02:B5:4B broadcast deauthenticate  
/disassociation of all clients, possible DoS  
ALERT: BCASTDISCON Network BSSID CC:96:A0:02:B5:4B broadcast deauthenticate  
/disassociation of all clients, possible DoS  
ALERT: BCASTDISCON Network BSSID CC:96:A0:02:B5:4B broadcast deauthenticate  
/disassociation of all clients, possible DoS
```

Informado que tipo de ataque es, por donde se está generando el ataque y cual la consecuencia de este.

Cuando kismet recibe la alerta esta es reportada al OSSIM y el OSSIM por medio del siguiente script, realiza una acción:

```
import paramiko
import sys
def sshCommand(hostname,port,username,password,command):
    sshClient = paramiko.SSHClient()

    sshClient.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    sshClient.load_system_host_keys()
    sshClient.connect(hostname,port,username,password)
    stdin, stdout, stderr = sshClient.exec_command(command)
    print(stdout.read())

if __name__=='__main__':
    sshCommand(sys.argv[0],22,'root','toor','reboot')
```

En nuestro caso lo que hace el reiniciar el equipo cerrando el punto de acceso del atacante. En el OSSIM se programa de la siguiente manera:

You can use the following keywords within any field which will be substituted by its matching value upon action execution:

- DATE
- PLUGIN_ID
- PLUGIN_SID
- RISK
- PRIORITY
- RELIABILITY
- SRC_IP_HOSTNAME
- DST_IP_HOSTNAME
- SRC_IP
- DST_IP
- SRC_PORT
- DST_PORT
- PROTOCOL
- SENSOR
- BACKLOG_ID
- EVENT_ID
- PLUGIN_NAME
- SID_NAME
- USERNAME
- PASSWORD
- FILENAME
- USERDATA1
- USERDATA2
- USERDATA3
- USERDATA4
- USERDATA5
- USERDATA6
- USERDATA7
- USERDATA8
- USERDATA9

NAME *	CentinelaRegla1
DESCRIPTION *	Se envía la ejecución de un comando SSH correspondiente
TYPE *	Execute an external program
CONDITION	<input checked="" type="radio"/> Any <input type="radio"/> Only if it is an alarm <input type="radio"/> Define logical condition
COMMAND: *	sh /opt/scripts/prueba3.sh USERDATA2 SRC_IP
TO: *	email;email;email

SAVE

El COMMAND:* se indica el scripts que se va a utilizar.

En TO* se puede poner un correo para informar al administrador de que ocurrió el evento.