



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Hacker Tools, Techniques, and Incident Handling (Security 504)"
at <http://www.giac.org/registration/gcih>

Remote Access Point/IDS

GIAC (GCIH) Gold Certification

Author: Jared Kee, Jkee@alertlogic.com

Advisor: Tim Proffitt

ACCEPTED: AUGUST 29TH 2011

Abstract

Access points are a modern day commodity that are widely accepted and used. Many businesses use Access points to allow their customers to connect wirelessly to the internet while other businesses may use them to allow employees access to resources that may be inaccessible for a variety of reasons. While Access points have come a long way and many people are eager to deploy them, not a lot has been done to upgrade the security of these devices which can leave large gaping holes in an otherwise secure environment. This paper will show readers how they can use open source technologies to monitor and secure their wireless networks without spending thousands of dollars or significantly altering their network.

1. Introduction

You may be wondering what a Remote Access Point is since access points are already wireless. A Remote Access Point is simply an access point or AP with no cables attached to it (aside from the power cable.) This sort of setup will allow someone to simply plug it into a power outlet and it will work as long as an access point is in range. Of course you will have to at least setup the wireless portion of the AP before you can go plugging into any outlet but with the right configuration and a little bit of technical prowess, you will be monitoring WLAN traffic in no time.

If you plan on using the Wired Equivalent Privacy or WEP encryption for whatever reason, you will also have the ability to monitor layer 3 traffic as the server has the ability to decrypt WEP traffic if the WEP key is entered into the configuration file on the Kismet server. This can be very useful if you have a guest or public AP that anyone can join so you have the ability to see exactly what is going on inside your network. In any other circumstance, however, you will probably want to use WPA2 encryption since it is more secure than WEP.

The wireless lab used in this setup is equipped with a RT73 USB card, an Atheros AR7161 processor chip, and 2 Atheros AR922X wireless chipsets found in the D-link DIR-825 which allows the access point to operate in one frequency while the wireless management interface can operate in the 5 GHz range. This allows both wireless interfaces to operate independently while avoiding channel interference from each other. The USB card in this lab is used to capture traffic from the 2.4 GHz AP, but the two 2.4 cards can switch roles fairly easily if desired. The upside to having multiple frequencies in this configuration is that if someone performs a DOS attack against the AP, the monitoring interface will still function normally and users will be notified of the DOS attack instead of wondering what happened to the AP until the attack has subsided. A sample network configuration setup can be seen in the figure below.

Wifi







Networks									
Link	ESSID	BSSID	Channel	Protocol	Mode	Encr.	Power	Scan	
-	Wireless WAN connection	-	36		sta	psk2	-		
-	Monitor	-	11		monitor		-		
-	Access Point	-	11		ap	wep	15 dBm		

Figure 1

2. Hardware Requirements

- A dual band AP capable of running OpenWrt with an available USB port. A list of compatible APs can be found here <http://wiki.openwrt.org/toh/start>. D-link DIR-825 Rev. B and Netgear WNDR3700 versions 1 and 2 are popular and economical choices. (\$50-\$150)
- Compatible wireless chipsets can be found in section 12 of Kismet's Documentation located at <http://www.kismetwireless.net/documentation.shtml#old> (Newer versions of Kismet auto detect wireless cards, so refer to the old readme for compatibility) (\$20-50 depending on the features you want in your wireless card.)
- Server capable of running Snort or Wireshark and Kismet server software
- Optional hardware includes an active USB hub, a USB Bluetooth interface, a 3G dongle card for backup WAN connectivity, and a GPS or Global Positioning System chip for tracking and location info

2.1 Software Requirements

- A current version of OpenWrt found at <https://openwrt.org>. The latest version is Backfire 10.03.1. Don't worry if you can't get the latest version to work with your device as you may have to go back a revision or two to find one that works with your hardware
- Kismet Drone which can be found in OpenWrt's software repository for easy installation if desired. If you decide to compile your own image to run the latest version of Kismet Drone, the library requirements for the drone are: libpcap, libncurses, uclibcxx, libpcrc, &

libnl (Kershaw, 2011)

- Kismet server, which processes the raw data sent by the Kismet drone and logs it.

Optionally, a virtual interface can be created for the Drone so Snort or Wireshark can see traffic from this interface.

- Snort <http://www.snort.org/> or Wireshark <http://www.wireshark.org/>, which is the IDS/IPS software used that can utilize both the pcap files and virtual interface(s) created by Kismet.
- Linux/GNU build environment based on <http://wiki.openwrt.org/doc/howto/buildroot.exigence> if you wish to compile your own image or have an up to date version of kismet-drone without using OpenWrt's Software Development Kit.

3. OpenWrt

OpenWrt is described as a Linux distribution for embedded devices and provides a fully writable file system with package management (OpenWrt, 2012). This allows for a modular setup where only chosen packages are installed. It is updated often and is compatible with many network devices. Installation varies from device to device but the compatibility list has the current status, limitations, and installation options for each device.

Once you've found your device listed, the next step will be determining if you want to use a precompiled image or compile your own. Each option has its own advantages and disadvantages but by compiling your own image you will have the most up-to-date packages as well as having a smaller/leaner install image which may be necessary if your device has very limited disk space.

3.1 Compiling your own image

Using a precompiled image is recommended for most users but if you are unable to install the precompiled image onto your device due to space limitations or if the precompiled image is using outdated or buggy drivers and newer drivers exist, you may want to compile your own image. This may seem intimidating at first but it is actually a pretty simple and straightforward process. Before you begin you may want to take a look at your devices wiki

page located at <http://wiki.openwrt.org/toh/start>, which should list any additional packages your device may require or caveats that you may need to be aware of before compiling your image.

There are a couple of different options available for compiling your image. The recommended method is to create your own build environment on which to compile all the packages you want installed into your image but if you are unable to meet the prerequisites for having your own build environment you may wish to use the image builder instead found at <http://wiki.openwrt.org/doc/howto/obtain.firmware.generate>. The image builder is like the precompiled image from the last section in that the image builder already has the majority of the tools you need to compile your image but the image builder itself does not utilize the latest code and is not as customizable as having your own build environment so using your own build environment will be covered.

Once you have downloaded your source and met all the prerequisites of the build environment listed here <http://wiki.openwrt.org/doc/howto/buildroot.exigence>, you will want to update your source and packages by running the ‘svn update’ command in the build root directory followed by ‘./scripts/feeds update -a’ and ‘./scripts/feeds install -a’. Now that all of your files are up to date, it is time to start the build process by running the ‘make menuconfig’ command from the build root directory which will take you to the build menu. The most important thing you need to set here is the ‘Target Profile’ which is the chipset/device you will be installing this image on which also may select some packages necessary to ensure that this image runs smoothly on the selected device. Once the profile has been selected, proceed to go through the menus and select the packages you want to install. Spacebar will cycle through the install options with blank meaning the package will not be installed, M meaning it will make the selected package only (for future installation), and * means that the selected package will be compiled into the final image. If you wish to have an operational web graphical user interface or GUI after installing this image, ensure that these packages are installed as well: liblua, libuci, libuci-lua, lua, luci-app-diag-core, luci-app-firewall, luci-app-qos, luci-i18n-english, luci-lib-core, luci-lib-fastindex, luci-lib-ipkg, luci-lib-lmo, luci-lib-nixio, luci-lib-sys, luci-lib-web, luci-mod-admin-core, luci-mod-admin-full, luci-sgi-cgi, luci-sgi-uhttpd, luci-theme-base, luci-theme-openwrt, uci, uhttpd, uhttpd-mod-lua. For USB support on your AP, install the package kmod-usb2 for USB 2 support and kmod-usb-core with kmod-usb-ohci or kmod-usb-uhci for USB 1.1 support as well as drivers for the device(s) you plan to plug into the USB port(s).

Jared Kee, jkee@alertlogic.net

This configuration should allow for basic connectivity into the AP once installed. If you are installing this image on a system with limited space, like a system with only 4mb of flash, verify that you have all of your packages you need installed because you may not have enough room to install any other packages using the OpenWrt package manager later due to the space needed to compile those missing packages on the fly. Please note if you are installing this image on a device with limited space, you will need to install the Kismet drone and all supporting packages (libpcap, libncurses, uclibcxx, libpcrc, & libnl) into the image. If you wish to install the latest version of Kismet into the image, you will have to edit the Makefile located at `/packages/net/kismet` in your build directory to read `PKG_VERSION:=$LATEST_VERSION`. The latest version is currently '2011-03-R2' and can be found at <http://kismetwireless.net/download.shtml> under the 'Kismet Release' section. If you are installing this image on a system that has 8 MB or greater of flash space, you will be able to install the latest version of Kismet drone easily using OpenWrt's package manager at a later time, so compiling kismet-drone into the image is not required. Now that you have your Target profile set and your packages selected to install, go back to the main menu and select exit at the bottom and save your configuration when it asks. Now that the configuration is saved, you can proceed to the next step of compiling your image by typing 'make' into the command line interface or CLI. Note that this process may take quite a while to complete and requires an active internet connection to download the files necessary for each package to compile. Once the images have been generated, you can find them in the 'bin' folder of your buildroot directory.

3.2 Using a precompiled image

If you are looking to get your device up and running in the shortest amount of time possible, using a precompiled image is the way to go. All precompiled images can be found at <http://downloads.openwrt.org>. Once there, select the latest version of OpenWrt, which is currently Backfire, and navigate to the chipset folder that your device utilizes. Each folder will have several images of other devices that use the same chipset so you will have to locate your model AND HARDWARE VERSION (if applicable). Different versions of the same model can use different hardware so you may be looking at the right device but the wrong hardware revision, which may have disastrous results if used.

3.3 Choosing your image

Once you have navigated to the correct folder you will notice that there are several different files available. The file system that your device will use depends on the file that you choose. A file with 'Jffs2' in the filename will use the Jffs2 filesystem while a file with 'squashfs' in the filename will use the Squash file system. Differences between the two file system types can be found here <http://downloads.openwrt.org/whiterussian/rc3/00-README>, but unless you have a specific requirement to use the JFFS2 file system, it is recommended that you use the Squash version instead since it requires less space and can often boot itself where a JFFS version would fail to boot.

Now that you have decided on the file system you wish to utilize, your available image choices should be reduced by half. Every make and model listed should have a minimum of 2 images, which will have either 'factory' or 'sysupgrade' in the file name. If you are running your manufacturers stock firmware, you will want to select the factory image but if you are already running OpenWrt and just need to upgrade, you will want to select the image with 'sysupgrade' in the name. Other image choices may be available to you depending on your chipset so you will have to refer to your devices wiki page for descriptions and installation instructions.

3.4 Finalizing the installation

Now that you have successfully upgraded your firmware, it is time to finalize the install so you can start utilizing the full feature set of OpenWrt. Once the firmware has been successfully installed, the device will have a default IP address of 192.168.1.1, which you need to telnet to. After telnetting into the device, set a password by using the 'passwd' command which also enables the web interface (if you installed the appropriate packages mentioned above) and SSH service. To ensure that the web GUI is fully operational, run the following commands: `/etc/init.d/luci_fixtime enable`, `/etc/init.d/luci_dhcp_migrate enable`, `/etc/init.d/luci_bwc enable`, `/etc/init.d/uhttpd enable`. Now reboot the device with the 'reboot' command and you should be able to login using SSH or <http://192.168.1.1> with the password you provided.

3.5 Setting up your network

The next step in getting your device working is to setup the wireless network. This can be done from the web GUI by going to the 'network' tab and selecting the wireless interface you

wish to modify from the 'wifi' subtab. You will be able to setup your AP here as well as your monitoring interface. If you are going to use WPA2-PSK encryption, make certain that the package 'wpa-mini' is installed before you choose this option or your device may freeze up. In order for this setup to be completely wireless, you are going to have to use the 5 GHz interface as a client so it can connect to another access point on your network and ultimately connect with the Kismet server. To do this, you will need to set the interfaces 'Mode' to client, specify the 'ESSID' of the AP on your network that you wish to connect to, and specify the encryption used along with the wireless key. Finally, to put an interface into monitor mode so that kismet drone will be able to utilize it, select 'Monitor' from the 'Mode' dropdown box in the monitoring interfaces configuration page.

If you are using the CLI to setup your network, you can edit the wireless configuration file by entering the command 'vi /etc/config/wireless'. More information about the different wireless configuration options in the wireless file can be found at <http://wiki.openwrt.org/doc/uci/wireless>. It is also recommended to put this interface in its own zone, which will help prevent anyone from intercepting traffic locally, but this will be covered in greater detail in section 8.

4. Drone

The drone is a lightweight piece of software whose sole purpose is to forward traffic it captures from a wireless interface to a server. The server can then store this traffic or create a virtual interface that can be monitored in real-time by Snort or Wireshark for malicious activity. Installation is a breeze in most cases and if you included it in your image from the previous section, installation is already complete. If you did not include Kismet-drone in your image and are ok with using the version of Kismet-drone included in your release of OpenWrt, all you need to do is navigate to 'System -> Software', click 'update package lists', and select kismet-drone to install it. Alternatively, you can go to the CLI and type 'opkg update' and 'opkg install kismet-drone'. If you want to use the latest version of kismet-drone and did not include it in your compiled image or you used a precompiled image, another option is still available.

You will need to create a build environment and follow the instructions in section 3.2 up to creating your image and unselect everything while marking kismet-drone with a 'M' so that

only the Kismet-drone package will be created. This includes editing the Makefile after you have updated your packages located at `/packages/net/kismet` in your build directory to read `PKG_VERSION:=$LATEST_VERSION`. Once you have saved your configuration and exited the build menu, run the `'make'` command which will start the process of creating your kismet-drone package. After this completes, you will find your newly created kismet-drone package in the `'bin'` folder of your build directory labeled as `'kismet-drone_($version).ipk'`. You can now SCP this file to the `/tmp/` folder of your device using the command `'scp kismet-drone_($version).ipk root@device:/tmp/kismet-drone_($version).ipk'` and install it using the CLI with the command `'opkg install /tmp/ kismet-drone_($version).ipk'`.

4.1 Drone configuration

Now that the drone is installed, it is time to configure the drone so it knows which interface to monitor and who can connect to it. First, you will need to connect to your device via SSH and run the command `'iwconfig'` so you can verify which interface your monitoring wireless card is assigned to. With that information, you can proceed to edit the kismet configuration file located at `/etc/kismet/kismet_drone.conf`. Proceed to the line that starts with `'dronelisten=tcp:/'` which configures the device to listen for a Kismet server on the IP/interface and port number you specify. The next line you want to configure is the `'droneallowedhosts='` line which specifies the host or network segment allowed to connect to the Kismet drone. You may use netmasks for network segments and can also specify multiple networks as long as a comma separates them.

The last line that needs to be edited is the most important line, which tells the kismet drone which interface the monitoring wireless card is attached to. Scroll down to the line that begins with `'ncsource='` and change it to `'ncsource=$WIRELESS_INTERFACE'`. Once again a comma will separate any additional options on this line. The next option we need to append to this line is `'forcevap=false'` which will prevent the device from creating its own virtual interface and communicate directly with the monitoring interface. This option is not needed since we manually put this interface into monitor mode in section 3.6; nor do all devices support it. The last option we need to configure on this line is the channel that the drone will listen on. To set this option, append `'channel=X'` to the interface configuration line, and `'hop=false'`, if your AP does not use channel hopping. If you are wanting to capture layer 3 traffic from the drone as

well, add the line ‘hidedata=false’, which will allow the drone to start forwarding IP header and packet data to the Kismet server. This is disabled by default because it would allow anyone to sniff layer 3 traffic from the drone straight out of the box. This is all that is needed for Kismet drone to successfully start, so you should have a line that looks similar to ‘ncsource=wlan1:type=rt73usb,forcevap=false,hop=false,channel=6’. Note that the ‘type=’ is optional in most cases since the drone auto detects the chipset, but auto detect does not always use the best chipset available so you may want to manually enter it in. For a full list of available options and features, refer to section 5 of the current readme file <http://www.kismetwireless.net/documentation.shtml#readme>. After you have saved your configuration, start the drone by running the command ‘kismet_drone’.

5. Kismet server installation

Now that the drone is setup, it’s time to configure the kismet server so it can start communicating with the drone. Kismet server is able to run on both Mac OS and Linux and has the same prerequisites as the drone. The first thing you will need is to obtain the latest Kismet source to compile onto your server which can be obtained here <http://www.kismetwireless.net/download.shtml> or here <http://www.kismetwireless.net/code>. It is recommended that you compile the same version of Kismet that is running on your device to reduce the chance of having incompatible versions, which shouldn’t happen unless you are running a version that’s older than 2009. Once you have downloaded the .tar file with the Kismet source in it, extract it, navigate to the created folder and run ‘./configure’. This command will check for compatibility and verify that all required libraries and headers needed to compile Kismet are available and will notify you if they aren’t. Upon successful completion, you need to run the commands ‘make dep’ and ‘make’ followed by ‘make install’ or ‘make suidinstall’. Running ‘make suidinstall’ allows Kismet to be run by local users with only the required privileges, which is safer than having the program run as root. If you use ‘make suidinstall’, make sure that the user you will use to run the Kismet server is added to the Kismet group so that Kismet will have the permissions needed in order to run properly.

There are many different plugins available for Kismet. If you want to use any plugins with the Kismet server, ensure that all plugins you wish to install are located in the Kismet source directory that was recently created. Now run the command “make plugins”, which will

compile all plugins found in this folder. You can install these plugins by running `make plugins-install` to install them at the root level or `make plugins-userinstall` to install them at the user level. These commands will compile all native and third party plugins you downloaded using the version of Kismet source you are compiling. One such plugin that you can use as a frontend replacement for Kismet is called qKismet and after it is installed you will be able to run it alongside the Kismet server instead of using the default client as discussed in section 6.

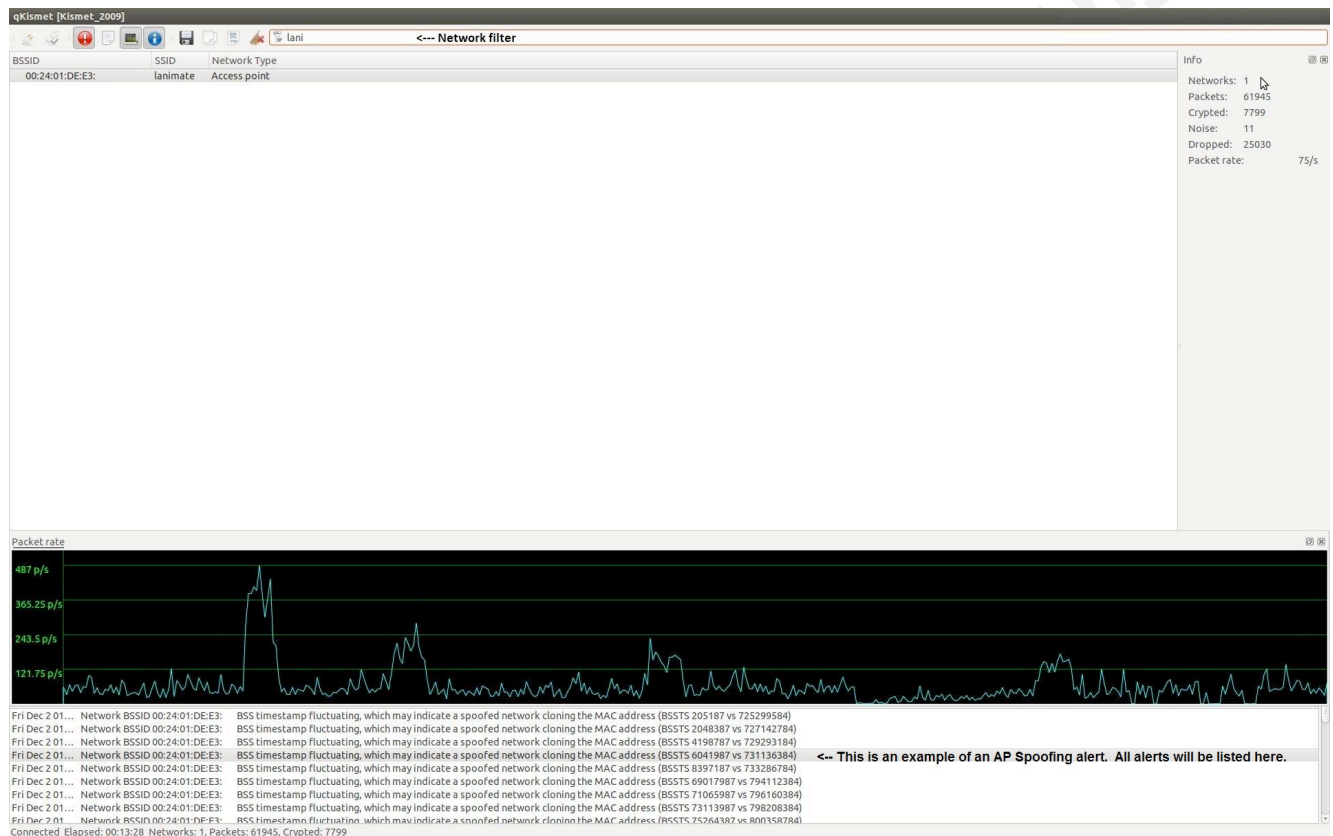


Figure 2: qKismet client

5.1 Kismet server configuration

All configuration options for the Kismet server are located in `/usr/local/etc/kismet.conf`. There are considerably more options available here than in the drone's configuration file because the purpose of the server is to capture, log, and decode data it receives from the drone. First, open this file in your favorite editor and note the version at the top. At the time of this writing, the most current version is "2009-newcore". The server hasn't changed much since that release

because the server is fairly stable and does not need to be built for many platforms, unlike the client or drone.

You will notice that in the configuration file, each option will have configuration notes and a little description about that option. While only one of these options is truly necessary for getting the server up and running, there are other options that may be required depending on your setup like ‘WEKey’ if you wish to view wireless layer 3 traffic. You may recognize some of the other options as well since they are the same options used in the drone’s configuration. The first option we will cover will be the “ncsource” option since it is vital for server and drone communication. This option allows you to specify where the server will collect its data from. If you are trying to gather data from a wireless card on the same system you are running your server on, all you may need to do is specify the wireless card as the source (default is `ncsource=wlan0`) and it will treat your wireless card like a drone. This can be especially useful if you want an all in one mobile wireless detection setup, but for our purposes we only need to specify the drone as the source with the following command.

`ncsource=drone:host=[IP_ADDRESS],port=PORT_NUMBER`. Note that the default port number is 2502, but you can change the port number the drone is listening on by modifying the “`dronelisten=`” for additional security. Furthermore, if you are wanting to fine tune the traffic you are seeing down to one (or multiple) BSSIDs, enable the line `filter_tracker=BSSID(AA:BB:CC:DD:EE:FF,!11:22:33:44:55:66)`, which logs and process any traffic seen from the first SSID and ignores any traffic seen from the second BSSID since “!” was used in front of that SSID.

The first set of optional parameters we will discuss are the server/client and server/drone options. These options allow you to specify if the server will act as a drone, in case you need to export the captured data to another server, and who can connect to the server via a Kismet client. To enable these features, proceed to the “client/server listen config” or “drone + server config” and fill in the appropriate IPs, ranges, and ports to allow connectivity with the server. Other options in these sections are max clients, which limits the amount of clients allowed to connect to the server, and max backlog, which limits the amount of alerts created per kismet client. These options may need to be fine tuned on servers with limited resources. The drone + server options should not need to be configured if you are already using a drone for capturing traffic, and the client/server options will not need to be configured unless you want to connect to the

server using a remote Kismet client. Note that if you will be logged into the server remotely, running “kismet” will launch the server as well as a local client. This will allow you to utilize the features of the client without having to open a port on the server.

The next section of optional parameters will cover WEP configuration since Kismet has the capability to decrypt WEP traffic if you want to see all layer 3 traffic from that wireless network. WEP is the only encryption standard that Kismet has the ability to decrypt which means that the most processing will be done if you utilize this option since another layer of traffic will be analyzed. Due to WEP not being inherently secure, it is not recommended that you use this standard but it is better than having no encryption at all. The first option you will need to configure is the `wepkey` option, which will provide the key to Kismet so that it will be able to decrypt this type of traffic. The format is `wepkey=BSSID,$HEXADECIMAL_WEP_KEY`. Once configured, you should be able to detect all wireless network traffic on the specified WEP enabled access point. The next couple of options deal with security of the data and key transmission. If clients are connecting to your server, you can either choose to have the server transmit the WEP key to these clients or you can disable this option and program the key into each client. If you choose to program each client with the WEP key, you will need to set the “`allowkeytransmit=`” option from true to false to prevent the key from being sent to each client. If you wish to enable the WEP key to be transmitted, you may want to consider having a secure connection to the server, like a VPN tunnel, so that unauthorized individuals will not be able to detect the WEP keys or data packets being sent by the server. Note that if you have SSH access to the server, you can simply login to the server and securely obtain the WEP key from the servers configuration file if you need it. The last option here is the `hidedata` option. This option truncates all of the data frames into the header, which is more secure to transmit, but it will disable IP detection. To enable this option, simply uncomment the “`hidedata=true`” field.

Alerts have already been configured to work with Kismet because they are a root function of Kismet, but since they are the generic default values, you may want to look over them in case you need to do some fine tuning in the future. The alerts and throttling rates can be found in the alerts section in the servers configuration file. Each alert will follow the format “`alert=name,throttle/unit,burst`”. The throttle option controls how many alerts are allowed per time unit (Valid time units are second, minute, hour, and day), while the burst option controls

how many alerts are allowed in a row (Kershaw, 2011). There are currently 22 alerts that can be configured by Kismet, and a description of each of these alerts can be found online at www.kismetwireless.net/documentation.shtml under section 12. One final alert worth noting is the APSPOOF alert, which detects if there is another AP spoofing the MAC address of your AP using timestamps to look for discrepancies. This alert has its own option [apspoof=] so that you can specify the MAC address of your AP and if Kismet detects a beacon or probe response from another MAC address, an alert will be created. This can be configured by using the following format: `apspoof=Name:ssid="SSID",validmacs="aa:bb:cc:dd:ee:ff`. Rogue AP detection will be covered in greater detail in Section 9

One of the newest features of Kismet is its exporting of data packets from a drone into a virtual interface on the local server. This allows programs like Wireshark or Snort the ability to monitor the layer 3 traffic being seen in real time without having to analyze pcap files later. To enable this feature, set the line `tuntap_export=true` and a virtual interface will be created the next time you start the server.

Logging is the last main function of the server we will discuss. When considering pcap files, there are two formats that Kismet can follow. The standard 802.11 header, and the more recent PPI header. The Per-Packet Information (PPI) Header is a general and extensible meta-information header format originally developed to provide 802.11n radio information but can handle other information as well, (CACE Technologies, 2007) such as GPS and spectrum data. Kismet will default to this header since it is more up-to-date, has better documentation than the 802.11 header, and contains more information. To change the default pcap header from PPI to the 802.11 standard header, change the line `'pcapdumpformat=ppi'` to `'pcapdumpformat=80211'`. After you have decided on a log template and format, there are a few options available to you when you are deciding what you want to log. These options allow you to save networks, GPS info, strings of clear text detected, Kismet generated alerts, entire PCAPs, as well as logs that may be generated by your plugins like the Bluetooth scanner plugin. You can configure what you want logged by finding the line `'logtypes='` and adding the log types listed there you want to be saved. The different log types are alert (text file of alerts), gpsexml (GPS data in XML format), nettxt (networks seen in text), netxml, pcapdump, and string (All strings detected). You can specify how and where you want your logs stored by using the `"logtemplate="` line which will allow you to change the location of where logs are stored (via the

“logprefix=” at the very top of the configuration file) as well as change the name and titles of the logs that are being stored. All usable options can be found in the log template section in the server’s configuration file. These options can be especially useful if you have a program that can read these log files but only if they are in a specific location or using a specific naming convention.

At this point now, you should have everything setup properly and can start the server by running /usr/local/bin/kismet_server (or /usr/local/bin/kismet, which also launches the local client) and should see output similar to the following figure.

```

lemondrop@lemondrop:~$ kismet_server <- Kismet Server Started
INFO: Not running as root - will try to launch root control binary (/usr/local/bin/kismet_capture) to control cards.
INFO: Started kismet_capture control binary successfully, pid 2276
INFO: Reading from config file /usr/local/etc/kismet.conf
debug - 2275 - child creating ipc fd
INFO: No 'dronelisten' config line and no command line drone-listen argument given, Kismet drone server will not be enabled.
INFO: Created alert tracker...
INFO: Creating device tracker...
INFO: Registered 80211 PHY as id 0
ERROR: Failed to open primary plugin directory (/usr/local/lib/kismet/): No such file or directory
INFO: Loaded info for plugin 'spectool_net.so': Plugin name: 'SPECTOOL' Plugin version: '2011-03-R2' Plugin description: 'Spectool-Net'
INFO: Loaded info for plugin 'dot15dot4.so': Plugin name: 'DOT1SD4' Plugin version: '2011-03-R2' Plugin description: '802.15.4 protocol plugin'
INFO: Pcap logging for type pcap15d4
INFO: Activated plugin '/home/lemondrop/.kismet//plugins/dot15dot4.so': 'DOT1SD4' '2011-03-R2'
INFO: Loaded info for plugin 'autowep-kismet.so': Plugin name: 'AUTOWEP' Plugin version: '2011-03-R2' Plugin description: 'AutoWEP Plugin'
INFO: Activated plugin '/home/lemondrop/.kismet//plugins/autowep-kismet.so': 'AUTOWEP' '2011-03-R2'
INFO: Loaded info for plugin 'aircrack-kismet.so': Plugin name: 'AIRCRAK-PTW' Plugin version: '2011-03-R2' Plugin description: 'Aircrack PTW Plugin'
INFO: Activated plugin '/home/lemondrop/.kismet//plugins/aircrack-kismet.so': 'AIRCRAK-PTW' '2011-03-R2'
INFO: Loaded info for plugin 'btscan.so': Plugin name: 'BTSCAN' Plugin version: '2011-03-R2' Plugin description: 'Active Bluetooth scanning plugin'
INFO: Activated plugin '/home/lemondrop/.kismet//plugins/btscan.so': 'BTSCAN' '2011-03-R2'
INFO: Kismet will spend extra time on channels 1,6,11
INFO: Kismet will attempt to hop channels at 3 channels per second unless overridden by source-specific options
INFO: No specific sources named on the command line, sources will be read from kismet.conf
INFO: Matched source type 'drone' for auto-type source 'drone'
INFO: Using default channel list 'n/a' on source 'drone'
INFO: Created source drone with UUID 00bf4aae-1cc0-11e1-8957-30040c171802
INFO: Disabling channel hopping on source 'drone' because it is not capable of setting the channel.
INFO: Will attempt to reopen on source 'drone' if there are errors

INFO: Created TCP listener on port 2501
INFO: Kismet drone framework disabled, drone will not be activated.
INFO: Inserting basic packet dissectors...
INFO: Using key 7137137137 length 5 for BSSID 84:C9:B2:7A:3A:
INFO: Allowing Kismet frontends to view WEP keys <- WEP key loaded
INFO: Starting GPS components...
INFO: GPS support disabled in kismet.conf
ERROR: Could not open OUI file '/etc/manuf': No such file or directory
ERROR: Could not open OUI file '/usr/share/wireshark/wireshark/manuf': No such file or directory
ERROR: Could not open OUI file '/usr/share/wireshark/manuf': No such file or directory
ERROR: No OUI files were available, will not resolve manufacturer names for MAC addresses
INFO: Creating network tracker...
INFO: Creating channel tracker...
INFO: Registering dumpfiles...
INFO: Pcap logging for type pcapdump
INFO: Opened pcapdump log file 'Kismet-20111202-02-31-17-1.pcapdump'
INFO: Opened nettxt log file 'Kismet-20111202-02-31-17-1.nettxt'
INFO: Opened string log file 'Kismet-20111202-02-31-17-1.string'
INFO: Dumpfile_String - forced string extraction from packets at all times
INFO: Opened alert log file 'Kismet-20111202-02-31-17-1.alert'
INFO: No spectools= line in config file, will not try to use spectools for spectrum data
INFO: Activated plugin '/home/lemondrop/.kismet//plugins/spectool_net.so': 'SPECTOOL' '2011-03-R2'
INFO: Kismet starting to gather packets
INFO: Started source 'drone'
INFO: kismet_capture pid 2276 synced with Kismet server, starting service loop
INFO: Opened tun/tap replicator 'kistap0' <- Virtual interface loaded
INFO: Kismet drone client connected to remote server "Kismet-Drone" using protocol version 1
INFO: Detected new managed network "lanimate", BSSID 00:24:01:DE:E3: encryption yes, channel 11, 54.00 mbit
AP Detected & Alerting
ALERT: BSSTIMESTAMP Network BSSID 00:24:01:DE:E3: BSS timestamp fluctuating, which may indicate a spoofed network cloning the MAC address (BSSTS 3414323587 vs 4139417984)
ALERT: BSSTIMESTAMP Network BSSID 00:24:01:DE:E3: BSS timestamp fluctuating, which may indicate a spoofed network cloning the MAC address (BSSTS 3416371587 vs 4141465984)
ALERT: BSSTIMESTAMP Network BSSID 00:24:01:DE:E3: BSS timestamp fluctuating, which may indicate a spoofed network cloning the MAC address (BSSTS 3417907587 vs 4143001984)
  
```

Figure 3: Kismet server startup

There are many options available to you once you have a drone collecting traffic and your server logging and processing data, so we will move on to the next section detailing how to use and configure the Kismet client.

6. Client

Kismet client is the frontend of Kismet and allows you to view the traffic and statistics from the server in real time. The installation procedure is the same as the Kismet server since Kismet server, client, and drone are all installed at the same time. Once installed, executing “kismet_client” will launch the client, while executing “kismet” will launch the server and client together. If the server and drone are already configured properly, you will be dumped at the main client screen and should see a monitoring terminal similar to the following figure:



Figure 4: Kismet client

If you are running the client on a machine outside of the server, you will first have to connect to the server by either clicking on the Kismet menu and selecting ‘connect’ or using ‘’, tab, and the arrow keys to navigate through the menu. Once you enter in the server’s IP and port number to connect to, you should start seeing data that the server has received if the drone is (or has been) connected to the server.

Now that the client is running and you are connected to the server, you may wish to configure some parameters to customize your view. The first thing you will want to do if you

installed plugins is setup which plugins you want to load (either on startup or one time only). This is accomplished by clicking on Kismet, plugins, and select plugins. Now you can use the tab button and arrow keys to select the plugin(s) you wish to be loaded. Pressing enter will enable or disable a plugin (Note that plugins can NOT be unloaded without restarting the server). The other menu option you will want to take a look at is the preferences option located under the Kismet menu. These options will allow you to customize the startup options, default Kismet server, as well as the look and feel of the client including colors, sounds, and network views.

Since the client is now fully configured, we can start taking a look at the wireless stats and graphs inside Kismet. To view more details about a specific network, go to the 'Sort' menu and set it to anything other than 'Auto-fit' since this will allow networks to be selectable. Now you can click on the network or use the tab, arrow keys, and enter to view additional details about the selected network. The details you will be able to find on this screen are a breakdown of wireless packets seen, the time the network was first seen, signal strength, type of wireless network, and an individual graph of the current network traffic which can be useful in identify trends in traffic over time. The last thing that you will want to be aware of is the alerts screen, which can be found in the main menu under 'Windows' and 'Alerts'. Each alert will have a summary that you will be able to select which displays more information concerning that alert.

7. Alerting and Detection

One of the leading causes of a missed attack is that a monitoring device was misconfigured or not configured at all which is why this section is paramount to the success of this setup. This section assumes you already have a working Kismet server and client connection established so we will start with layer 2 detection since if you aren't seeing layer 2 traffic, you aren't going to be seeing layer 3 traffic.

7.1 Layer 2 alerting and detection

By default, Kismet reports any traffic it can detect. In order to properly detect your wireless network, you must first set your monitoring channel(s), which can be done in the drone's configuration file, and set your access point's bssid and MAC address, which can be set in the server's configuration file using the "filter_tracker=BSSID" option. Once set, you should start to logging traffic from the specified devices, which can be found by using the client or

analyzing the pcap file that will be created by default from the Kismet server. If you set the “logprefix” to a local folder, all of your log files created by Kismet will be located there, otherwise they will be found in the location where you are running Kismet. If you have set the server to automatically start on boot without setting the logprefix variable, the logs will be located in the home directory of the user that’s set to run Kismet. If any alerts were generated, they will be found in the file with the extension ‘.alert’ and will have output similar to the alerts generated below.

```
Sun Dec 18 20:37:04 2011 BCASTDISCON 0 00:24:01:DE:E3:57 00:24:01:DE:E3:57
FF:FF:FF:FF:FF:FF 00:00:00:00:00:00 Network BSSID 00:24:01:DE:E3:57 broadcast
deauthenticate/disassociation of all clients, possible DoS
```

```
Sun Dec 18 20:37:44 2011 BSSTIMESTAMP 11 00:24:01:DE:E3:57 00:24:01:DE:E3:57
FF:FF:FF:FF:FF:FF 00:00:00:00:00:00 Network BSSID 00:24:01:DE:E3:57 BSS timestamp
fluctuating, which may indicate a spoofed network cloning the MAC address (BSSTS
5273600384 vs 5642240675)
```

7.2 Layer 3 alerting and detection

If you are using WEP encryption (or none), you have the ability to view all wireless layer 3 traffic as well as the layer 2 traffic. If you use any other encryption, you will only be able to view layer 2 802.11 traffic. However, if you are only interested in seeing layer 2 traffic, that means that you will have less overhead, more space available, and faster throughput of your captured traffic. There are 2 different ways you can capture layer 3 traffic depending on what your needs are. If you need to view real-time traffic, you will want to utilize the virtual interface created by the Kismet server, but if you do not need to see real time traffic you can just configure Kismet to log pcap files. However, there needs to be enough space on the server for those pcap files since they can become quite large in a small amount of time.

If you decide to use pcap files to save your wireless traffic for later analysis, you can view the pcap file in snort by typing “snort -r <pcap file>” or selecting “File -> Open” if you are using Wireshark. Below is a screenshot of a pcap of a wireless network being analyzed in Wireshark.

ath9k pc and phone [Wireshark 1.7.1-SVN-40247 (SVN Rev Unknown from unknown)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: wlan.bssid == 00:1e:58:2e:24: <- BSSID filter

No.	Time	Source	Destination	Protocol	Length	Info
29294	126.144735010	D-Link 2e:24:	00:11:11:11:11:11	802.11	344	PROBE Response, SN=4030, FN=0, Flags=....., BI=100, SSID=AbrahamLinksys
29295	126.157962010	192.168.0.198	192.168.0.197	UDP	74	Source port: wizard Destination port: openvpn
29296	126.163028010	D-Link 2e:24:	Broadcast	802.11	224	Beacon frame, SN=4029, FN=0, Flags=....., BI=100, SSID=AbrahamLinksys
29297	126.166983010	192.168.0.197	192.168.0.1	DNS	106	Standard query 0x253f A north-america.pool.ntp.org
29298	126.182368010	192.168.0.198	192.168.0.197	UDP	62	Source port: emce Destination port: pcanywherestat
29309	126.253345010	192.168.0.198	192.168.0.197	TFTP	84	Read Request, File: nessus125705465, Transfer type: octet <- Selected packet
29300	126.265356010	D-Link 2e:24:	Broadcast	802.11	224	Beacon frame, SN=4031, FN=0, Flags=....., BI=100, SSID=AbrahamLinksys
29301	126.367849010	D-Link 2e:24:	Broadcast	802.11	224	Beacon frame, SN=4032, FN=0, Flags=....., BI=100, SSID=AbrahamLinksys
29302	126.386626010	192.168.0.197	239.255.255.250	SSDP	282	NOTIFY * HTTP/1.1
29303	126.387154010	192.168.0.197	239.255.255.250	SSDP	280	NOTIFY * HTTP/1.1
29304	126.445125010	192.168.0.198	192.168.0.197	TCP	80	2007 > 49777 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK PERM=1
29305	126.445902010	192.168.0.197	192.168.0.197	TCP	80	2007 > 49777 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK PERM=1
29306	126.446580010	192.168.0.198	192.168.0.197	TCP	80	2007 > 49777 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK PERM=1
29307	126.446678010	192.168.0.198	192.168.0.197	TCP	80	2007 > 49777 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK PERM=1
29308	126.470292010	D-Link 2e:24:	Broadcast	802.11	224	Beacon frame, SN=4033, FN=0, Flags=....., BI=100, SSID=AbrahamLinksys
29309	126.572679010	D-Link 2e:24:	Broadcast	802.11	224	Beacon frame, SN=4035, FN=0, Flags=....., BI=100, SSID=AbrahamLinksys
29310	126.675121010	D-Link 2e:24:	Broadcast	802.11	224	Beacon frame, SN=4036, FN=0, Flags=....., BI=100, SSID=AbrahamLinksys
29311	126.675147010	Motorola 8a:ce:	D-Link 2e:24:	802.11	16	Power-Save poll, Flags=...P....
29312	126.676272010	192.168.0.198	192.168.0.197	UDP	76	Source port: wizard Destination port: openvpn
29313	126.676913010	Motorola 8a:ce:	D-Link 2e:24:	802.11	16	Power-Save poll, Flags=...P....
29314	126.676238010	192.168.0.198	192.168.0.197	UDP	64	Source port: emce Destination port: pcanywherestat
29315	126.677949010	Motorola 8a:ce:	D-Link 2e:24:	802.11	16	Power-Save poll, Flags=...P....
29316	126.678993010	192.168.0.197	192.168.0.198	ICMP	104	Destination unreachable (Port unreachable)
29317	126.679350010	192.168.0.197	192.168.0.197	TFTP	86	Read Request, File: nessus125705465, Transfer type: octet
29318	126.679680010	192.168.0.197	192.168.0.198	ICMP	102	Destination unreachable (Port unreachable)

[Protocols in frame: wlan:llc:ip:udp:tftp] <- Note the listed protocols

[Coloring Rule Name: UDP]

[Coloring Rule String: udp]

IEEE 802.11 Data, Flags:T

Type/Subtype: Data (0x20)

Frame Control: 0x0108 (Normal)

Duration: 44

BSS Id: D-Link 2e:24: (00:1e:58:2e:24:) <- Access point mac address

Source address: Alfa la:bf: (00:c0:ca:la:bf:) <- Source mac address

Destination address: Motorola 8a:ce: (98:4b:4a:8a:ce:) <- Destination mac address

Fragment number: 0

Sequence number: 3351

Logical-Link Control

Internet Protocol Version 4, Src: 192.168.0.198 (192.168.0.198), Dst: 192.168.0.197 (192.168.0.197) <- Source and destination IP addresses

Version: 4

Header length: 20 bytes

Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))

Total Length: 52

Identification: 0xda41 (55873)

Flags: 0x00

0000 08 01 2c 00 00 1e 58 2e 24 27 00 c0 ca 1a bf 8cX.\$'..... <- Payload

0010 98 4b 4a 8a ce 36 70 d1 aa aa 03 00 00 00 08 00 .KJ..6p.

0020 45 00 00 34 da 41 00 00 40 11 1d 9c c0 a8 00 c6 E..4.A..@.....

0030 c0 a8 00 c5 2a 01 00 45 00 20 a3 a0 00 01 6e 65*.E.....ne

0040 73 73 75 73 31 32 35 37 30 35 34 36 35 00 6f 63 ssus1257 05465.oc

0050 74 65 74 00 tet.

Protocols carried by this frame (fr... Packets: 31632 Displayed: 31632 Marked: 0 Load time: 0:01.235

Figure 5: Events being analyzed using Wireshark

If you decide to use the virtual interface that Kismet creates, we first need to verify a few settings to ensure that the virtual interface will be successfully created upon launch. Before viewing or troubleshooting anything related to the virtual interface, you may want to use a local Kismet client (simply run “kismet”) to ensure that you are seeing all the traffic you are expecting from your access point before it is logged or exported out of Kismet. Once you have verified those settings, ensure that the 2 lines “tuntap_export=true” and “tuntap_device=\$name” are in the Kismet server's configuration file. In the figure below, we are able to see the virtual interface that Kismet successfully created.

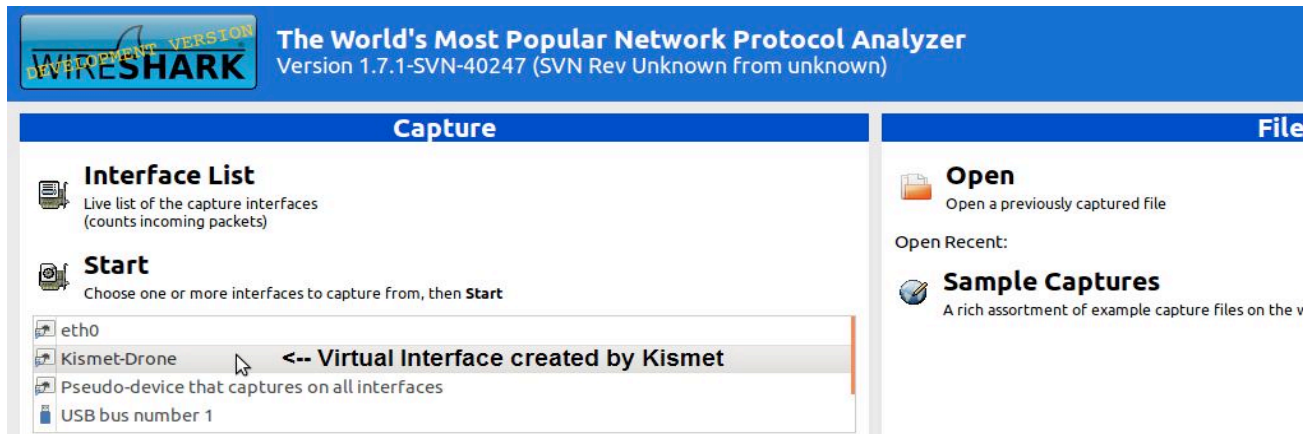


Figure 6: Virtual interface detected in Wireshark

You should now be able to click on the Kismet interface and start, which takes you to a live view of captured wireless traffic. From here you can specify a filter like the one seen in the screenshot earlier in this section, since Wireshark has the ability to view all layer 2 traffic which does not get filtered out by Kismet. Snort, however, does not analyze traffic at layer 2 (unless you use the arpspoof preprocessor), so no additional filters are needed. In fact, if you have the Kismet server already running, all you need to do is append “-i <\$Kismet_virtual_interface>” command to your Snort command, and you should start see output similar to the figure below.

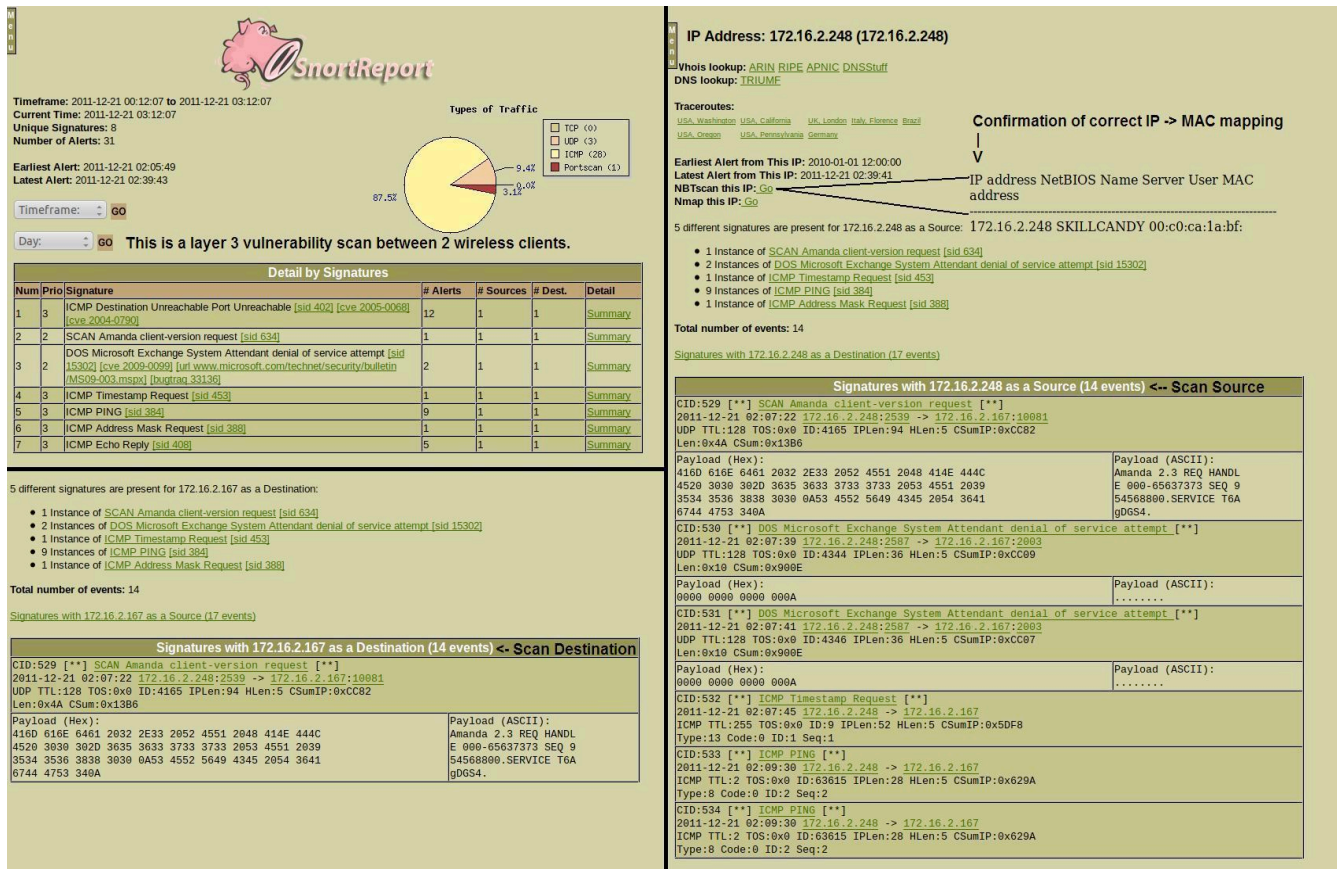


Figure 7: Events being analyzed using SnortReport

This screenshot above is using a front-end for Snort called SnortReport (Symmetrix Technologies, 2012). If you wish to use SnortReport in your configuration, you may want to take a look at the Snort installation guide by David Gullet listed below in the references section.

8. Implementation and Security

Everything up to this point has had focus on setup and operation but now that your device is operational, we need to ensure that it's secure. There are four basic configurations you can easily implement and secure depending on your hardware and objective. The first one is a Remote access point/IDS or RAPIDS configuration, which only requires a power cable to operate when setup. If you followed all of the preceding sections correctly, you currently have this configuration. The next two options are based on the original configuration, except that the monitored access point is independent of your drone device so it functions more like a traditional wireless IDS. The last setup option is also based on the original configuration except it uses a wired network connection. This gives you the option to disable the additional interface or use it

as a 5Ghz monitoring interface so you have the ability to completely monitor a dual band access point. Using one of the wired WAN configurations will be inherently more secure since your WAN connection will not be susceptible to the same type of attacks as your monitored access point, but it's not quite as portable so you will have to decide which is more important to you.

Before we start implementing a configuration, we need to change the ports of hosted services on your device and create a key so that you are able to securely connect to your device. You can change your SSH port in the GUI if you go to Services and Dropbear SSHd or modify the file `/etc/config/dropbear` from the CLI. If you only want SSH to be available on a specific interface, you can change "0.0.0.0" to the IP address of that interface. Alternatively, you can add a firewall statement in `/etc/config/firewall` similar to the one listed below which will allow only specified interfaces or zones access to your SSH port.

```
config 'rule'
    option 'src' 'wan'
    option 'proto' 'tcp'
    option 'dest_port' '22'
    option 'target' 'ACCEPT'
```

SSH keys can be added by going to System and SSH-Keys which are stored in the `/etc/dropbear` folder. The settings for the http server are located at `/etc/config/uhttpd` and allow you to change the port for http or https as well as add a SSL certificate. Alternatively, you can disable this service if you are comfortable using the CLI by typing 'opkg remove uhttpd'. If you have an OpenVPN server you wish to use to connect to your server, you can add this package by typing `opkg update` and `opkg install luci-app-openvpn`. This will add a menu under Services ->

OpenVPN where you can add your OpenVPN certificate and key. Additional OpenVPN options can be found at `/etc/config/openvpn`. Disabling ping responses on the WAN interface is also recommended so that this device will be harder to locate. However, if you use any network tools or monitoring software that may use the ping response as a metric, you may wish to keep this option enabled. Disabling the ping responses can be accomplished by editing the file `/etc/config/firewall` to reject ping responses on the WAN interface.

```
config 'rule'
  option 'src' 'wan'
  option 'proto' 'icmp'
  option 'icmp_type' 'echo-request'
  option 'target' 'ACCEPT'
```

Change option 'target' 'ACCEPT' to 'REJECT'

```
config 'rule'
  option 'src' 'wan'
  option 'proto' 'icmp'
  option 'icmp_type' 'echo-request'
  option 'target' 'REJECT'
```

Figure 8: Disabling ping responses in the firewall

Next we are going to enable layer 2 security by using MAC address filtering so only specified MAC addresses will be able to connect to the device on the LAN ports. Before we enable layer 2 security, we need to add exceptions for your MAC address by adding the following statement to the /etc/config/firewall:

```
config rule
  option 'src' 'lan'
  option 'src_mac' '00:11:22:33:44:55'    (← your MAC address)
  option 'target' 'ACCEPT'
```

If you do not have any MAC address exceptions you wish to add, you can add the following statement which will reject all connections on the LAN ports.

```
config rule
  option 'src' 'lan'
  option 'src_mac' '00:00:00:00:00:00'
  option 'target' 'REJECT'
```

Now that your device is secured, we can move on to choosing a configuration. Because we are going to be disabling services that will not be used as well as configuring existing services, we will start with the RAPIDS configuration since it utilizes the most services. Seeing that this configuration uses its second wireless interface as a client for connectivity, we can disable the WAN port in the firewall. Modify the interface line seen in the previous paragraph from “option ‘src’ ‘lan’” to “option ‘src’ ‘wan’” which will reject any incoming connections on that port. If you wish to use the GUI to disable the physical WAN port, you can accomplish this by going to firewall, zones, WAN, and simply changing the WAN zone to your new WAN interface. You can also specify any additional firewall rules here you wish to be enforced on

your WAN port.

In two of these configurations, the 5 GHz interface is used for network connectivity. Since we are not concerned with monitoring traffic on this interface, you will want to make sure that you use WPA2 encryption. WPA2 personal with a good password will be good enough for most people, but it is recommended that you use WPA2 enterprise if you are able to meet the hardware requirements since it uses additional authentication methods. You can do this by going to your interface configuration and making sure WPA2 is selected under encryption. Note that using WPA2 on your AP interface is recommended, but you will not be able to detect layer 3 traffic.

As long as you are hosting an access point on your device, you may wish to have the wireless users in their own zone so that wired and wireless users are secluded from each other. This can be achieved by going to the interface configuration in the GUI and selecting your zone under the network tab. Alternatively you can go to firewall, zones, and create a new zone with that wireless interface specified. Once you have created a zone for your access point, you can add a rule to allow WAN access under firewall and forwarding rules by setting the access point zone as the source and your WAN interface as the destination. On the CLI, this can be accomplished by modifying the `/etc/config/firewall` file. Additional information and a full list of available firewall commands can be found at <http://wiki.openwrt.org/doc/uci/firewall>

8.1 Backup connection

In case of a primary WAN link failure, you can still maintain a connection to your server by utilizing a 3G USB connection with an encrypted network connection. This connection can also be used as a primary connection, but it is not inherently secure as a wired connection and also subject to additional charges if you go over your allowed data usage. If you have more than one USB port available and want to use more than one 3g card for redundancy or load balancing, you can install multiwan. It is complete with load balancing, failover, and an easy to manage traffic ruleset (OpenWrt, 2012). A list of compatible cards and options utilizing one or multiple USB 3G cards can be found at <http://wiki.openwrt.org/doc/recipes/3gdongle>.

9. Locating Rogue APs

A Rogue Access Point is a wireless access point installed on a wired enterprise network without authorization from the network administrator (AirTight Networks, 2009). Recently, the

topic of Rogue AP's has drawn a lot of attention, as many manufacturers like Cisco (Cisco Systems, 2012) are starting to introduce new detection techniques and protocols that will successfully detect these devices. Once detected, tracking them down can sometimes be a bit of a task since detection software may have trouble distinguishing the real AP from the spoofed AP. You can try and make use of the signal levels since they can be useful sometimes, but they may not always be a good indicator in every instance for many reasons. There are, however, a number of tools you can use to assist you in locating these devices. If your environment is a relatively quiet one, you may be able to utilize tools like Wififofum to simply walk around and identify rogue APs based on signal level alone as demonstrated in the figure below.



Rogue AP detection using WiFiFoFum

Since Kismet uses timestamps to detect rogue APs, you may want to consider having your APs synchronize with your own NTP server if you have multiple wireless devices. Synchronizing with a NTP server can also be used to detect rogue APs during a vulnerability test by offsetting the time slightly so that any APs not updated to this abnormal time will immediately be detected (of course there will be an initial alert for your network when you change the time). However, this is not recommended for day-to-day use as all clients connected to the AP may start attempting to authenticate with the rogue AP. Another method you can use if you have detected a rogue AP on your network is to simply turn off all wireless networks so that only the rogue AP can be detected, but once again this is not recommended under normal

circumstances as all wireless clients will start trying to authenticate with the rogue AP.

Currently, the best way to locate rogue APs without altering settings on your wireless network require multiple APs for triangulation or expensive hardware, but opensource tools are available and constantly being improved to help better assist in locating rogue APs. Airodump-ng (Aircrack-ng, 2012) is considered one of the best programs to use to detect rogue APs since it can be setup quickly and is compatible with many mobile platforms that support linux. It comes bundled inside the Aircrack-ng suite and can filter out networks based on MAC address, channel, encryption, and ESSID just like Kismet. It also has a very large support forum that can assist you if you run into any problems or have questions. If you have a compatible wireless card, all you need to do to start basic monitoring is run the command 'airodump-ng \$WIRELESS_INTERFACE' as root. You can append filters to that line as needed like --encryption wep or --channel X depending on your needs. It is also very useful in detecting unauthorized clients that may be on your network. This program is constantly being worked on and additional for support for locating rogue Aps is in development, but I don't expect to see any updates on it until the later part of this year. Additionally, if you want a more visual frontend for Airodump-ng you may want to take a look at the fern-wifi-cracker located at <http://code.google.com/p/fern-wifi-cracker/> which is a security auditing application written in python that supports features like GPS mapping to make future wireless network assessments easier (Ekiko, 2011).

Some other noteworthy AP detection applications are Netstumbler for Windows and Ekahau for Android which utilizes a HeatMapper that can be very effective in locating wireless devices.

10. Conclusion

In conclusion, as popularity grows and advances in wireless technologies are made so does the risk of a wireless based attack on your network. I believe that by properly implementing one of the configurations mentioned in this paper, detection of malicious wireless attacks can be safely and easily mitigated. New attacks and vulnerabilities are constantly being detected and updated to the Kismet database, and within a couple of years, I expect to see IDS devices that detect both wired and wireless attacks that may occur on your network straight out of the box.

Jared Kee, Jkee@alertlogic.net

11. References

Hallinan, C. (2011). *Embedded Linux Primer*. Boston, Massachusetts: Pearson.

Cache, J., Wright, J., & Liu, V. (2010). *Hacking Exposed Wireless: Wireless Security Secrets & Solutions, Second Edition*. New York, New York: McGraw-Hill.

Haines, B., Schearer, M., & Thornton, F. (2008). *Kismet Hacking*. Burlington, Massachusetts: Syngress.

Gullet, D. (2011, November 5). *Snort 2.9.1.2 on Ubuntu 10.04 LTS*. Retrieved from <http://www.snort.org/assets/158/013-snortinstallguide2912.pdf>

Kershaw, M., (2011, April 4). Kismet. Retrieved from <http://www.kismetwireless.net/documentation.shtml>

Renderman (2010, November 28). The Renderlab: Kismet Newcore Drone Build Hacking. Retrieved from <http://www.renderlab.net/projects/newcore/newcore-drone/drone.html>

D-Link DIR-825. (n.d.). Retrieved September 7, 2011 from the OpenWrt Wiki: <http://wiki.openwrt.org/toh/d-link/dir-825>

Rasmus (2004, March 5). Kismet on the Linksys WRT54G. Retrieved from <http://toys.lerdorf.com/archives/20-Kismet-on-the-Linksys-WRT54G.html>

Kismet Server/Drone. (n.d.). Retrieved September 3, 2011 from the DD-WRT Wiki: http://www.dd-wrt.com/wiki/index.php/Kismet_Server/Drone

OpenWrt Buildroot - Usage. (n.d.). Retrieved October 1, 2011 from the OpenWrt Wiki: <http://wiki.openwrt.org/doc/howto/build>

Jared Kee, jkee@alertlogic.net

Firewall Configuration. (n.d.). Retrieved January 4th, 2012 from the OpenWrt Wiki:

<http://wiki.openwrt.org/doc/uci/firewall>

Use 3g/UMTS USB Dongle for WAN Connection. (2011, November 12). Retrieved January 4th, 2012 from the OpenWrt Wiki: <http://wiki.openwrt.org/doc/recipes/3gdongle>