How to use Monitor Mode to Sniff Wi-Fi Packets

June 1, 2017

1 Setting Up Monitor Mode

There are two methods for setting up monitor mode. Either using Linux's built in `iwconfig` command, or via the Aircrack-ng suite's `airmon-ng` command. It is also important to note that not every WiFi interface is capable of entering monitor mode. However, using either of these commands is enough to check whether or not your WiFi device is capable of monitor mode.

1.1 Using `iwconfig`

- Open the Linux terminal
- Turn off the interface using `sudo ifconfig` <interface> `down`
- Set the device to monitor mode using `sudo iwconfig` <interface> `mode monitor`
- Turn on the interface using `sudo ifconfig` <interface> `up`
- Confirm the device is in monitor mode using `sudo iwconfig` <interface>

1.2 Using `aircrack-ng`

- Download the Aircrack-ng suite using your Linux distro's package manager, or download and compile the Aircrack-ng source code. This can typically be done with `sudo apt-get install aircrack-ng`
- Set the device to monitor mode using `sudo airmon-ng start` <interface>. Airmon is a part of the Aircrack package.
- Confirm a new monitor mode interface has been created using `iwconfig`

2 Scanning the network

There are many different ways to handle network packet analysis, and these would vary between languages, with some having libraries and APIs which help make this task easier. As these many different methods would be too numerous, we'll only be tackling the method used in ROCMAN, which is to use `Python 2.7` with a library called `scapy` (this library is not available on `Python 3.x` and above at least at the time of making this tutorial). This was chosen due to it's capabilities in network sniffing and packet analysis.

- First of all, you'll want to install `scapy` (obviously). The easiest way to install `scapy` is using the command `sudo pip install scapy`. You could, for whatever reason, download the `scapy.py` file online instead, but this offers no clear benefits.
- Afterwards, you'll want to start using it. Although `scapy` has a variety of different uses, we'll only be discussing how `scapy` can be used to sniff packets on the network. For this purpose, `scapy` uses a function `sniff()` to capture a stream of incoming packets on some interface. The function takes a number of arguments, some of which are:
  - `iface`, a string corresponding to the interface you wish to scan on. (e.g. mon0, wlan1).
  - `prn`, the function which is used to handle the incoming packets. Essentially, the function is called every time a new packet is received, which allows you to work on the incoming data on a packet-by-packet basis. Any value you return in this function will be displayed as well. *Defaults to None.*
  - `lfilter`, a function for deciding whether or not you should operate on a packet. For example, you could set `lfilter=lambda p: p.haslayer(Padding)`. *Defaults to None.*
  - `count`, the number of packets you want to capture before stopping. *Defaults to 0*, which corresponds to infinity in this case.
  - `store`, a boolean which decides whether or not you should store the packets. If you set this to 1, the sniff function will return a list of those packets. Otherwise it won't. It is very much recommended to set this to 0 if you scan infinitely. *Defaults to 1.*
  - `timeout`, which corresponds to how long you want the script to run. *Defaults to None*, which lets you scan infinitely (or until you stop due to some other parameter)
  - `stop_filter`, a function that stops the capture when it returns True. *Defaults to None.*

- `offline`, a pcap file you can provide that it will read packets from instead. For the purposes of scanning a network this is fairly useless, but might be useful if you want to rerun the scan again. *Defaults to None*.

If there is something you're unclear of regarding scapy or simply isn't tackled in this tutorial, you can always check the documentation at `scapy.readthedocs.io/en/latest/index.html` for further clarification.