**Analyzing Deauthentication Packets with Wireshark**

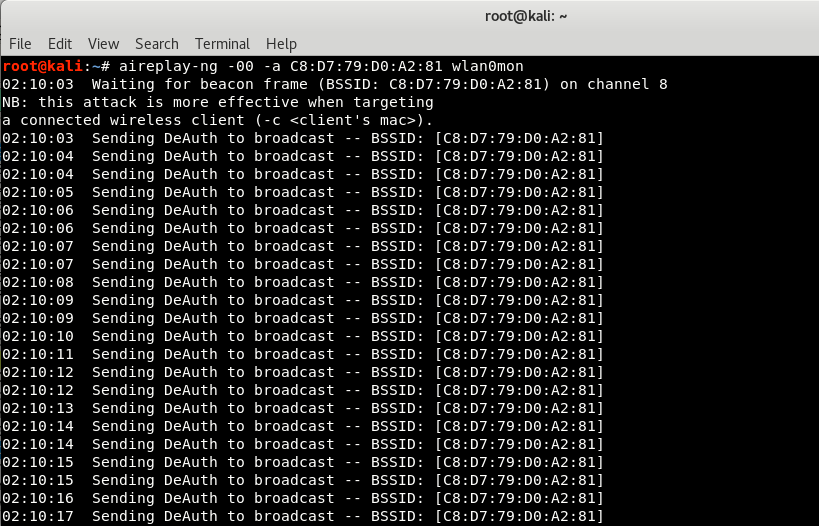
Regardless of whether you are reading a packet capture from a stored file or from a live interface on a Windows or Linux host, Wireshark’s analysis features are nearly identical. Wireshark offers many useful features for analyzing wireless traffic, including detailed protocol dissectors, powerful display filters, customizable display properties, and the ability to decrypt wireless traffic.

Let’s start with analyzing the [*Deauthentication Packets/Frames*](https://www.yeahhub.com/wi-fi-deauthentication-attack-against-802-11-protocol/) with Wireshark.

Deauthentication request can be send either with **aireplay-ng** or with [**mdk3**](https://www.yeahhub.com/ddos-wifi-network-mdk3-tool-kali-linux/) tool. But make sure that your card must listen on same channel as the AP is operating on.

**For Deauthentication with Aireplay-ng, the command is:**

**Command:** aireplay-ng -00 -a <BSSID> wlan0mon



**Where,**

* -0 specifies the number of times the attack has to replay and -00 means no limit which will flood the AP and station with deauth frames.
* -a is the target’s BSSID
* wlan0mon is your monitor interface

*Sending a few deauth frames are enough to successfully disconnect the stations in case of performing tests and capturing handshakes. A long attack will keep the device from connecting and the end would be the device user will have to connect manually from Wi-Fi Manager.*

### ****About Wireshark –****

Wireshark is the best open-source network analyzer available. It is packed with features comparable to commercial network analyzers, and with a large, diverse collection of authors, new enhancements are continually developed.

Wireshark is a stable and useful component for all network toolkits, and new features and bug fixes are always being developed. A lot of progress has been made since the early days of Wireshark (when it was still called Ethereal); the application now performs comparably (and in some regards) better than commercial sniffing software.

One of the most powerful and useful features in Wireshark is the ability to apply **inclusive** or **exclusive** display filters to a packet capture, in order to narrow down the number of packets to those containing useful data. When capturing traffic on a wireless network, it is easy to become overwhelmed by the sheer quantity of data that is captured.

### ****Starting Wireshark –****

Start Wireshark by running the **wireshark** executable with no command line arguments as the root user, and initiate a new packet capture by pressing **Capture | Options**. This opens the “**Wireshark Capture**” options dialog box. Choose the wireless interface which is wlan0mon (in our case) that has been placed in monitor mode by selecting the drop-down box labelled “**Interface:**” and then specify the desired capture options.

In order to begin sniffing wireless traffic with Wireshark, your wireless card must be in monitor mode. Wireshark does not do this automatically; you have to manually configure your wireless card before starting your packet capture.

Next, click **Start** to initiate the packet capture. At this point, you’ve configured your system to capture wireless traffic in monitor mode.The next step is to utilize the information contained in the packets you are capturing. Fortunately,Wireshark has sophisticated analysis mechanisms that can be used for wireless traffic analysis.

Using display filters, you can exclude uninteresting traffic to reveal useful information, or search through a large packet capture for a specific set of information.

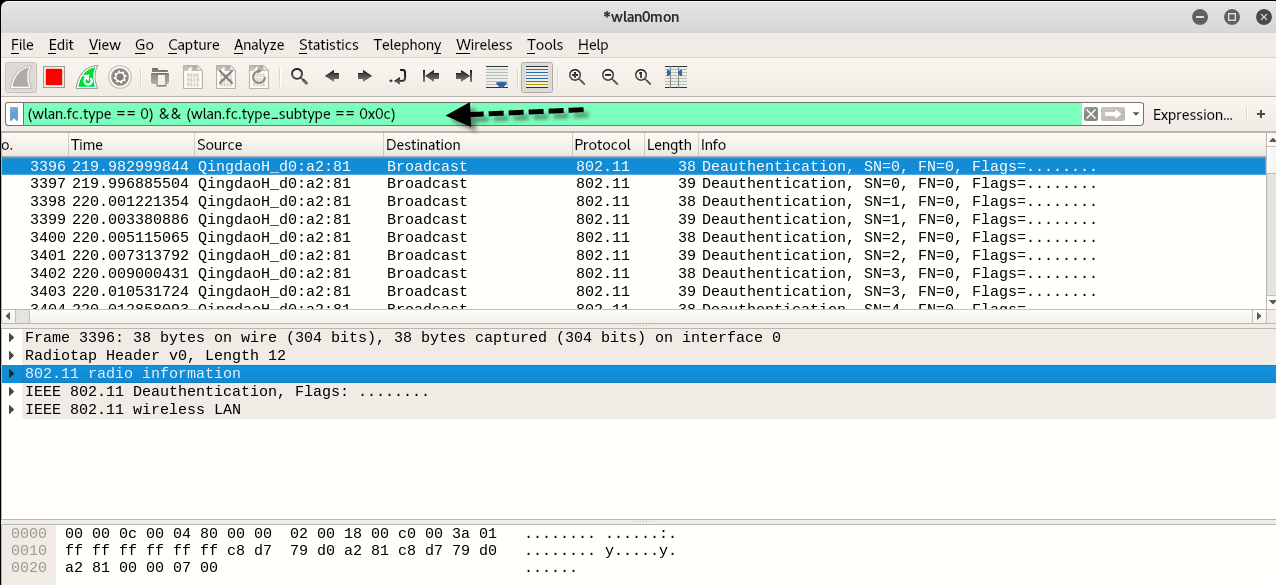
**For Filtering Deauthentication Frames, the filter is:**

(wlan.fc.type == 0) && (wlan.fc.type\_subtype == 0x0c)  
OR  
(wlan.fc.type eq 0) && (wlan.fc.type\_subtype eq 0x0c)  
OR  
(wlan.fc.type eq 0) && (wlan.fc.type\_subtype eq 12)

Here, **type** field of deauth frame have value **0** while **subtype** has the value **0x0c (12).**

The **Type** field is included in the frame **wlan.fc.type** control header and specifies the type of frame (data, management, or control) whereas The **Type/Subtype** field value is included as a convenience mechanism to uniquely identify the type and subtype combination that is included in the header of this frame. This field is commonly used in display filters.

For example, if the frame is a type management frame, the subtype field indicates the type of management frame (e.g., a beacon frame, authenticate request, or disassociate notice).



When assessing a wireless packet capture with Wireshark, it is common to apply display filters to look for or exclude certain frames based on the IEEE 802.11 frame **type** and frame **subtype** fields.

If you are trying to exclude frames from a capture, it is easy to identify the **Type** and **Subtype** fields by navigating the Packet Details window and using the values for your filter. If you are looking for a specific frame type, however, you have to remember either the Frame **Type** and **Subtype** values, or the Combined **Type/Subtype** value assigned by Wireshark.

Instead of expecting you to memorize the 35+ values for different frame types, we’ve listed out here for easy reference.

| **Frame Type/Subtype** | **Filter** |
| --- | --- |
| Management Frames | wlan.fc.type eq 0 |
| Control Frames | wlan.fc.type eq 1 |
| Data Frames | wlan.fc.type eq 2 |
| Association Request | wlan.fc.type\_subtype eq 0 |
| Association response | wlan.fc.type\_subtype eq 1 |
| Reassociation Request | wlan.fc.type\_subtype eq 2 |
| Reassociation Response | wlan.fc.type\_subtype eq 3 |
| Probe Request | wlan.fc.type\_subtype eq 4 |
| Probe Response | wlan.fc.type\_subtype eq 5 |
| Beacon | wlan.fc.type\_subtype eq 8 |
| Announcement Traffic Indication MAP (ATIM) | wlan.fc.type\_subtype eq 9 |
| Disassociate | wlan.fc.type\_subtype eq 10 |
| Authentication | wlan.fc.type\_subtype eq 11 |
| Deauthentication | wlan.fc.type\_subtype eq 12 |
| Action Frames | wlan.fc.type\_subtype eq 13 |
| Block Acknowledgement (ACK) Request | wlan.fc.type\_subtype eq 24 |
| Block ACK | wlan.fc.type\_subtype eq 25 |
| Power-Save Poll | wlan.fc.type\_subtype eq 26 |
| Request to Send | wlan.fc.type\_subtype eq 27 |
| Clear to Send | wlan.fc.type\_subtype eq 28 |
| ACK | wlan.fc.type\_subtype eq 29 |
| Contention Free Period End | wlan.fc.type\_subtype eq 30 |
| Contention Free Period End ACK | wlan.fc.type\_subtype eq 31 |
| Data + Contention Free ACK | wlan.fc.type\_subtype eq 33 |
| Data + Contention Free Poll | wlan.fc.type\_subtype eq 34 |
| Data + Contention Free ACK + Contention Free Poll | wlan.fc.type\_subtype eq 35 |
| NULL Data | wlan.fc.type\_subtype eq 36 |
| NULL Data + Contention Free ACK | wlan.fc.type\_subtype eq 37 |
| NULL Data + Contention Free Poll | wlan.fc.type\_subtype eq 38 |
| NULL Data + Contention Free ACK + Contention Free Poll | wlan.fc.type\_subtype eq 39 |
| QoS Data | wlan.fc.type\_subtype eq 40 |
| QoS Data + Contention Free ACK | wlan.fc.type\_subtype eq 41 |
| QoS Data + Contention Free Poll | wlan.fc.type\_subtype eq 42 |
| QoS Data + Contention Free ACK + Contention Free Poll | wlan.fc.type\_subtype eq 43 |
| NULL QoS Data | wlan.fc.type\_subtype eq 44 |
| NULL QoS Data + Contention Free Poll | wlan.fc.type\_subtype eq 46 |
| NULL QoS Data + Contention Free ACK + Contention Free Poll | wlan.fc.type\_subtype eq 47 |