Features Explore Pricing This repository Search Sign in or Sign up

spacehuhn / esp8266_deauther

Watch 55 Star 397 Fork 98

Code Issues 13 Pull requests 0 Projects 0 Pulse Graphs

ESP8266 deauther

wifi arduino attack esp8266 deauth-attack

119 commits 2 branches 2 releases 5 contributors MIT

Branch: master New pull request Find file Clone or download

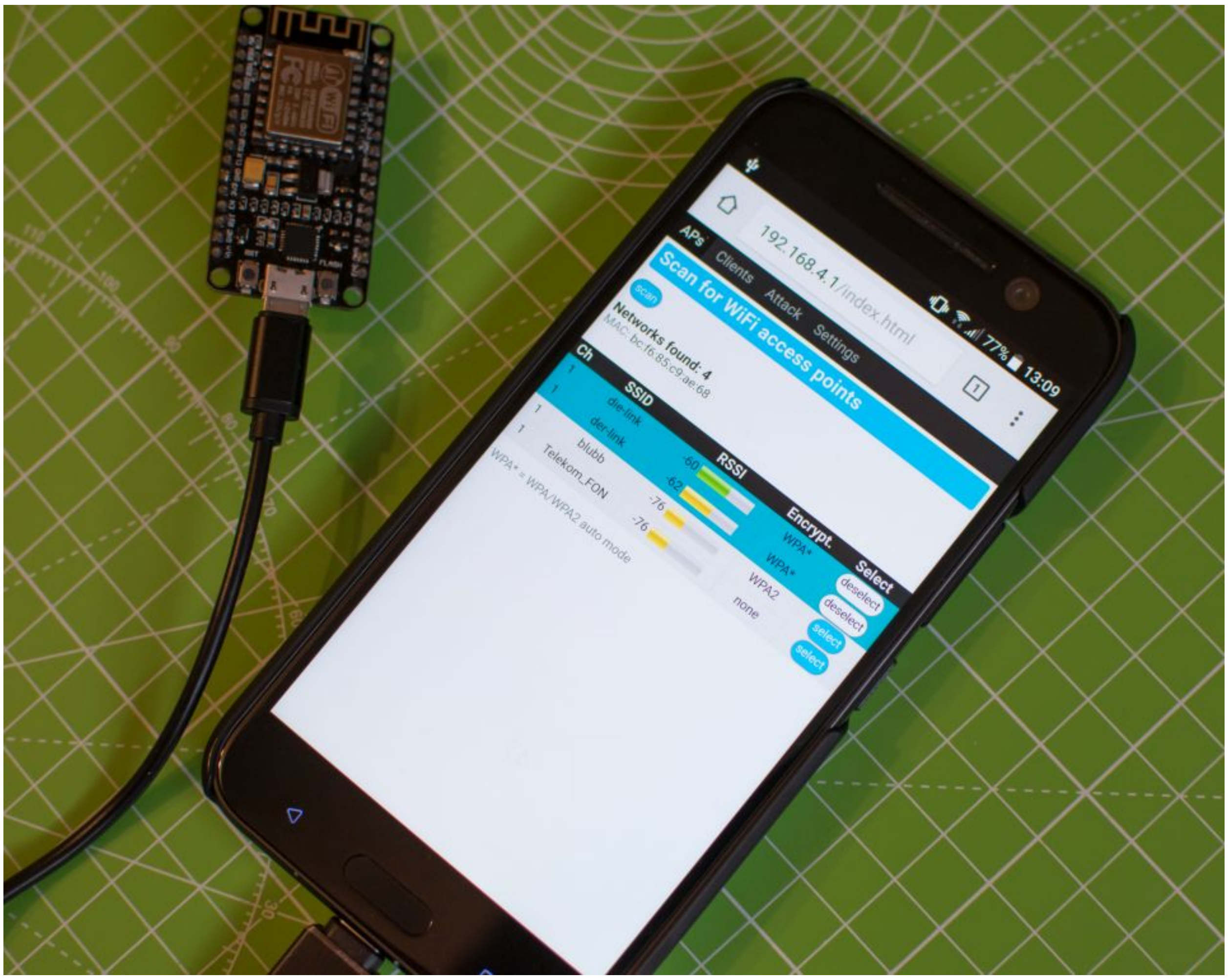
spacehuhn committed on **GitHub** Update issue_template.md Latest commit `7a8b5f3` 3 days ago

.github	Update issue_template.md	3 days ago
esp8266_deauther	marked position to comment out the mac vendor list	13 days ago
htmlfiles	Merge-fixes	13 days ago
screenshots	Readme changes & new images	25 days ago
sdk_fix	Added infos to install the SDK fix	22 days ago
.gitignore	Initial commit	3 months ago
LICENSE	Initial commit	3 months ago
README.md	Update README.md	12 days ago

README.md

ESP8266 Deauther

Build your own WiFi jammer with an ESP8266.



Contents

Introduction

What it is

Basically it's a device which performs a deauth attack.
You select the clients you want to disconnect from their network and start the attack. As long as the attack is running, the selected devices are unable to connect to their network.

How it works

The 802.11 WiFi protocol contains a so called deauthentication frame. It is used to disconnect clients safely from a wireless network.

Because these packets are unencrypted, you just need the mac address of the WiFi router and of the client device which you want to disconnect from the network. You don't need to be in the network or know the password, it's enough to be in its range.

What an ESP8266 is

The ESP8266 is a very cheap micro controller with build in WiFi. It contains a powerfull 160 MHz processor and you can program it with the Arduino IDE. This makes it perfect for this project.

You can buy these chips for under $2 from China!

How to protect against it

With 802.11w-2009 WiFi got an update to encrypt management frames. So make sure your router is up to date and has management frame protection enabled. But be sure that your client device supports it too, both ends need to have it enabled!

The only problem is that most devices don't use it. I tested it with different WiFi networks and devices, it worked every time! It seems that even newer devices which support frame protection don't use it by default.

Disclaimer

Use it only for testing purposes on your own devices!

Please check the legal regulations in your country before using it. Jamming transmitters are illegal in most countries and this device can fall into the same category (even if it's technically not the same).

My intention with this project is to draw attention to this issue. This attack shows how vulnerable the 802.11 WiFi standard is and that it has to be fixed. **A solution is already there, why don't we use it?**

Installation

The only thing you will need is a computer and an ESP8266.

I recommend you to buy a USB breakout/developer board, because they have 4Mb flash and are very simple to use. It doesn't matter which board you use, as long as it has an ESP8266 on it.

You have 2 choices here. Uploading the bin files is easier but not as good for debugging, so keep that in mind in case you want to open an new issue.

Uploading the bin files

Note: the 512kb version won't have the full MAC vendor list.

0 Download the current release from here

1 Upload using the ESP8266 flash tool of your choice. I recommend using the nodemcu-flasher. If this doesn't work you can also use the official esptool from espressif.

That's all! :)

Make sure you select the right com-port, the right upload size of your ESP8266 and the right bin file.

Compiling the source with Arduino

0 Download the source code of this project.

1 Install Arduino and open it.

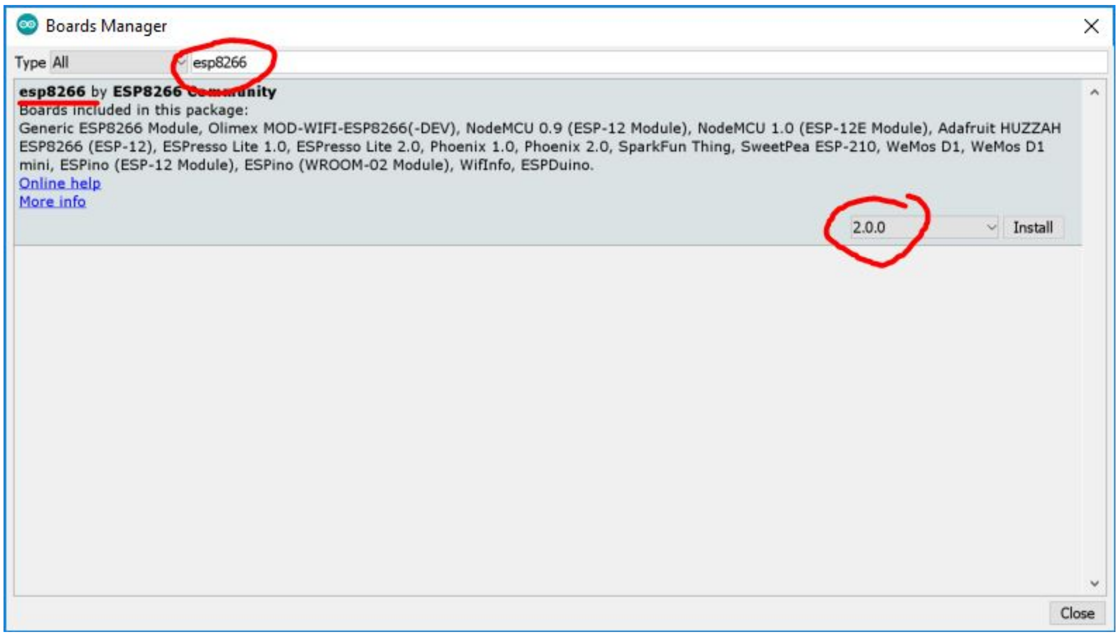
2 Go to `File > Preferences`

3 Add `http://arduino.esp8266.com/stable/package_esp8266com_index.json` to the Additional Boards Manager URLs. (source: https://github.com/esp8266/Arduino)

4 Go to `Tools > Board > Boards Manager`

5 Type in `esp8266`

6 Select version `2.0.0` and click on `Install` (**must be version 2.0.0!**)



Boards Manager

Type All esp8266

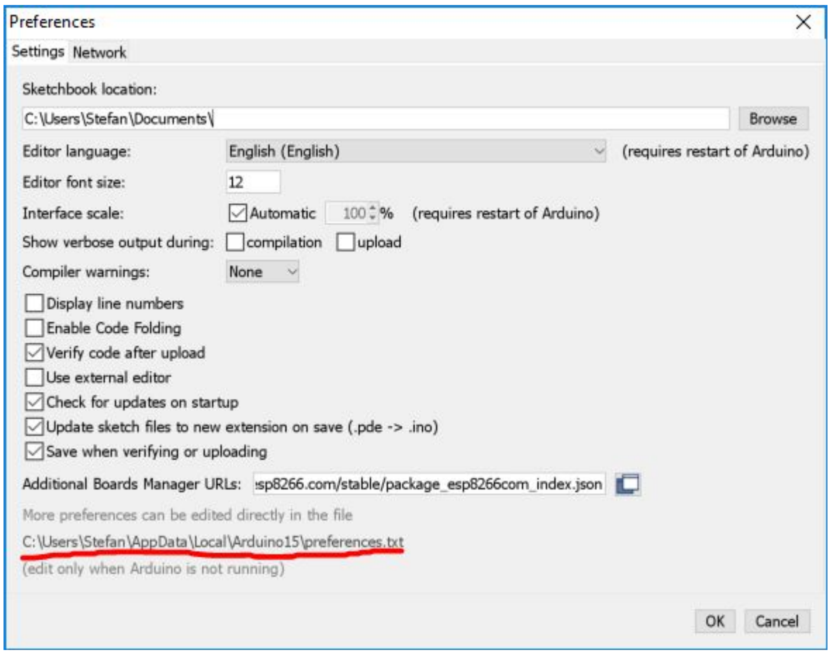
esp8266 by **ESP8266 Community**
Boards included in this package:
Generic ESP8266 Module, Olimex MOD-WIFI-ESP8266(-DEV), NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Adafruit HUZZAH ESP8266 (ESP-12), ESPresso Lite 1.0, ESPresso Lite 2.0, Phoenix 1.0, Phoenix 2.0, SparkFun Thing, SweetPea ESP-210, WeMos D1, WeMos D1 mini, ESPino (ESP-12 Module), ESPino (WROOM-02 Module), WifInfo, ESPDuino.
Online help
More info

2.0.0 Install

Close

7 Go to `File > Preferences`

8 Open the folder path under `More preferences can be edited directly in the file`



Preferences

Settings Network

Sketchbook location:
C:\Users\Stefan\Documents\ Browse

Editor language: English (English) (requires restart of Arduino)

Editor font size: 12

Interface scale: ☑ Automatic 100 % (requires restart of Arduino)

Show verbose output during: ☐ compilation ☐ upload

Compiler warnings: None

☐ Display line numbers
☐ Enable Code Folding
☑ Verify code after upload
☐ Use external editor
☑ Check for updates on startup
☑ Update sketch files to new extension on save (.pde -> .ino)
☑ Save when verifying or uploading

Additional Boards Manager URLs: sp8266.com/stable/package_esp8266com_index.json

More preferences can be edited directly in the file
C:\Users\Stefan\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

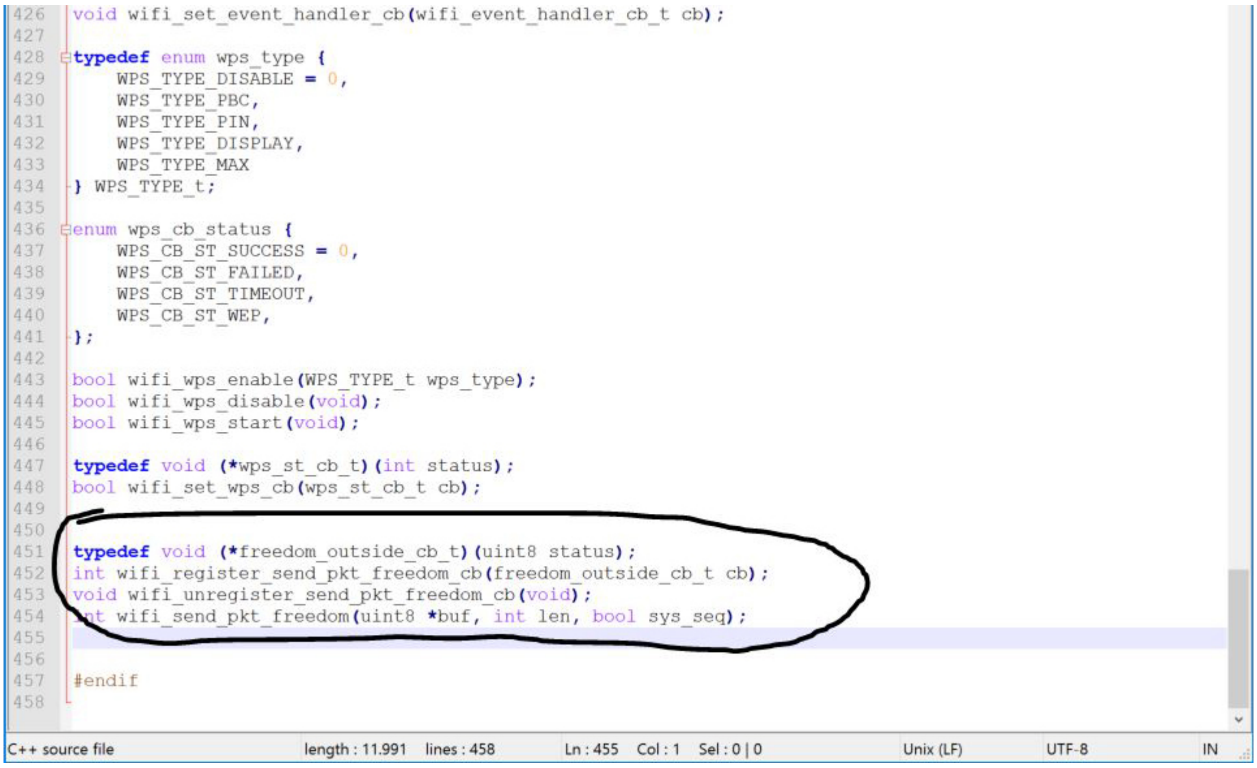
OK Cancel

9 Go to `packages > esp8266 > hardware > esp8266 > 2.0.0 > tools > sdk > include`

10 Open `user_interface.h` with a text editor

11 Scroll down and before `#endif` add following lines:

```
typedef void (*freedom_outside_cb_t)(uint8 status);
int wifi_register_send_pkt_freedom_cb(freedom_outside_cb_t cb);
void wifi_unregister_send_pkt_freedom_cb(void);
int wifi_send_pkt_freedom(uint8 *buf, int len, bool sys_seq);
```



```
void wifi_set_event_handler_cb(wifi_event_handler_cb_t cb);

typedef enum wps_type {
    WPS_TYPE_DISABLE = 0,
    WPS_TYPE_PBC,
    WPS_TYPE_PIN,
    WPS_TYPE_DISPLAY,
    WPS_TYPE_MAX
} WPS_TYPE_t;

enum wps_cb_status {
    WPS_CB_ST_SUCCESS = 0,
    WPS_CB_ST_FAILED,
    WPS_CB_ST_TIMEOUT,
    WPS_CB_ST_WEP,
};

bool wifi_wps_enable(WPS_TYPE_t wps_type);
bool wifi_wps_disable(void);
bool wifi_wps_start(void);

typedef void (*wps_st_cb_t)(int status);
bool wifi_set_wps_cb(wps_st_cb_t cb);


typedef void (*freedom_outside_cb_t)(uint8 status);
int wifi_register_send_pkt_freedom_cb(freedom_outside_cb_t cb);
void wifi_unregister_send_pkt_freedom_cb(void);
int wifi_send_pkt_freedom(uint8 *buf, int len, bool sys_seq);



#endif
```

don't forget to save!

12 Go to the SDK_fix folder of this project

13 Copy ESP8266WiFi.cpp and ESP8266WiFi.h

14 Past these files here `packages > esp8266 > hardware > esp8266 > 2.0.0 > libraries > ESP8266WiFi > src`

15 Open `esp8266_deauther > esp8266_deauther.ino` in Arduino

16 Select your ESP8266 board at `Tools > Board` and the right port at `Tools > Port`
If no port shows up you may have to reinstall the drivers.

17 Upload!

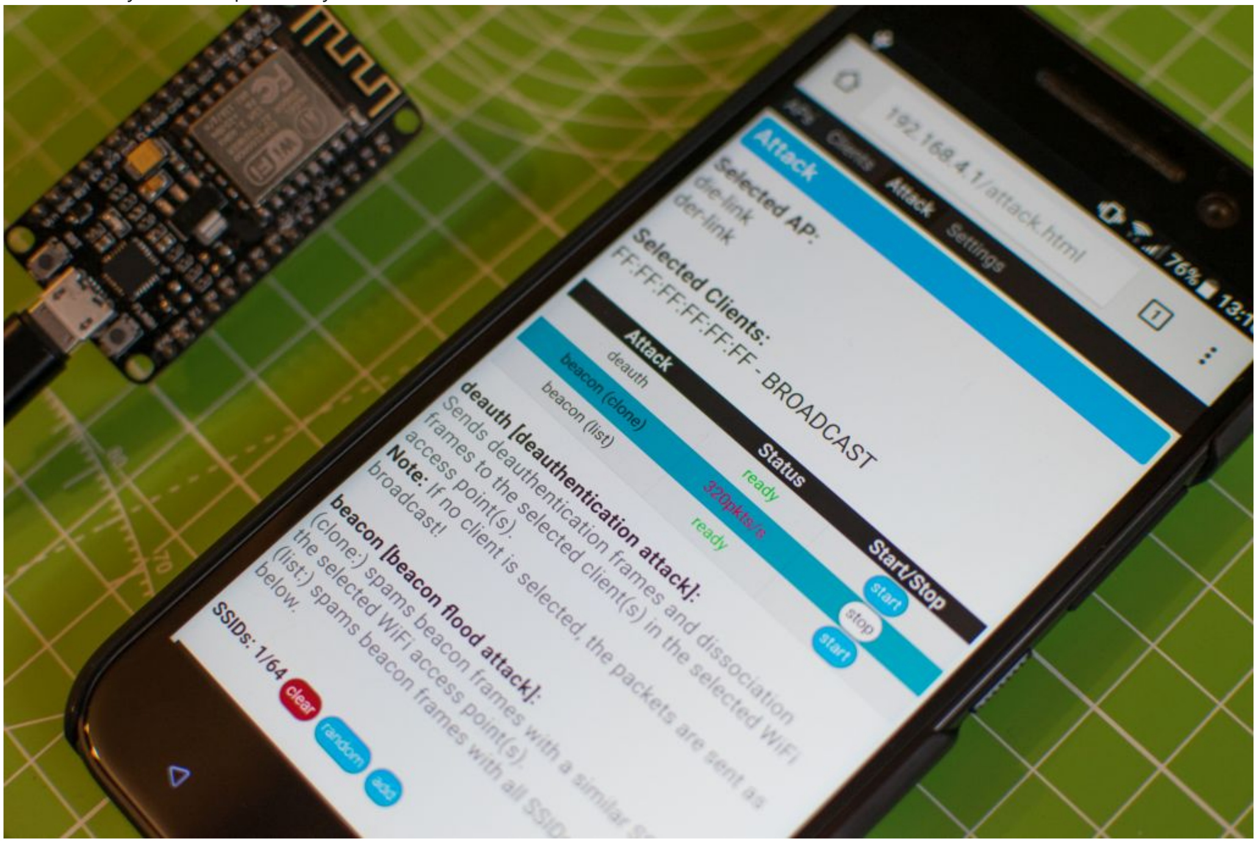
Note: If you use a 512kb version of the ESP8266, you need to comment out a part of the mac vendor list in data.h.

Your ESP8266 Deauther is now ready!

How to use it

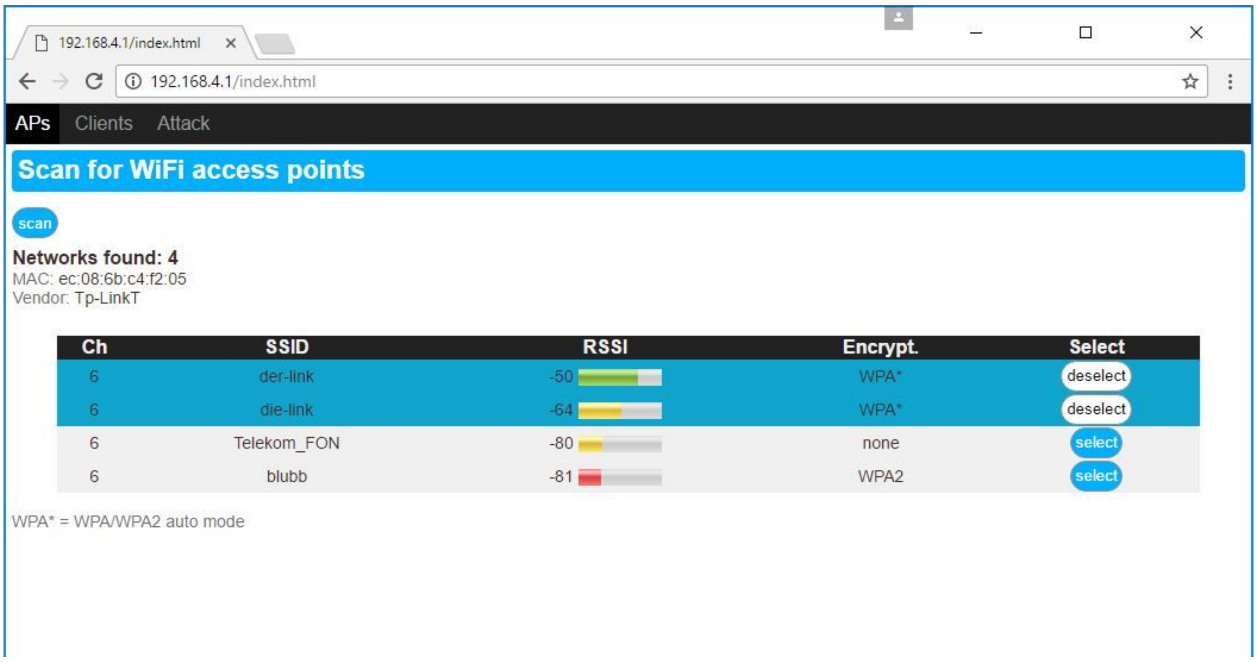
First start your ESP8266 by giving it power.

You can use your smartphone if you have a USB OTG cable.



Scan for WiFi networks and connect to `pwned`. The password is `deauther`.
Once connected, you can open up your browser and go to `192.168.4.1`.

You can now scan for networks...



Scan for WiFi access points

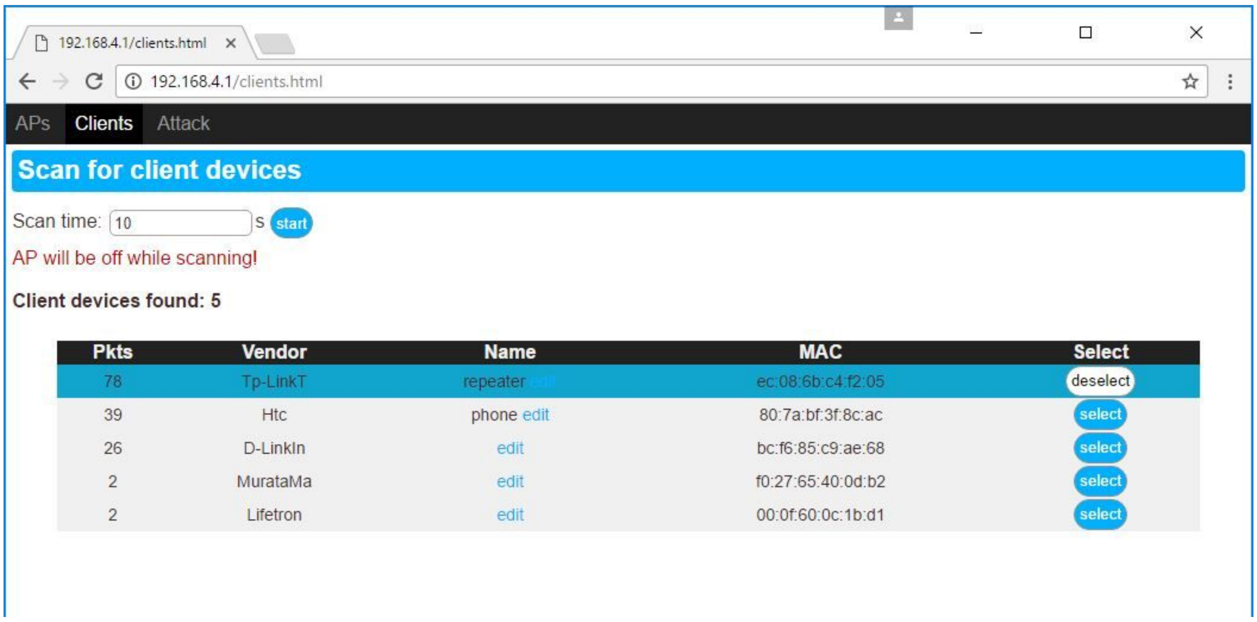
scan

Networks found: 4
MAC: ec:08:6b:c4:f2:05
Vendor: Tp-LinkT

Ch	SSID	RSSI	Encrypt.	Select
6	der-link	-50	WPA*	deselect
6	die-link	-64	WPA*	deselect
6	Telekom_FON	-80	none	select
6	blubb	-81	WPA2	select

WPA* = WPA/WPA2 auto mode

scan for client devices...



Scan for client devices

Scan time: 10 s

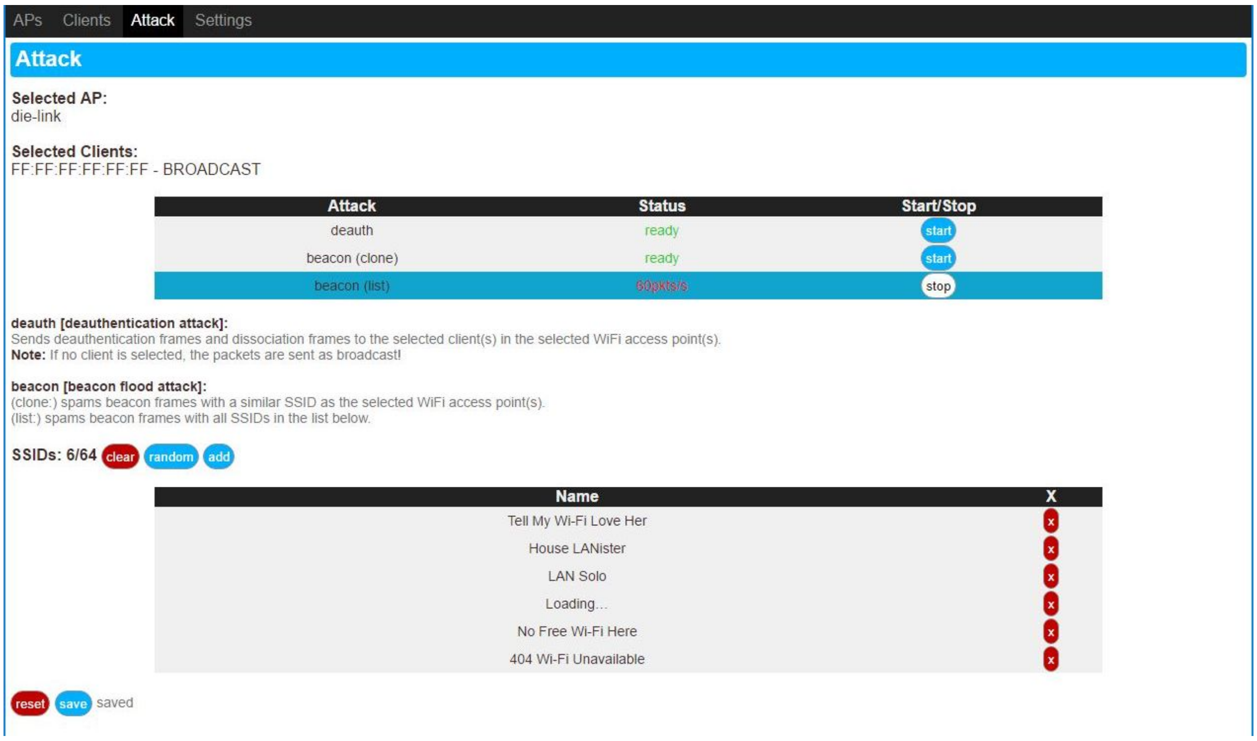
AP will be off while scanning!

Client devices found: 5

Pkts	Vendor	Name	MAC	Select
78	Tp-LinkT	repeater edit	ec:08:6b:c4:f2:05	deselect
39	Htc	phone edit	80:7a:bf:3f:8c:ac	select
26	D-LinkIn	edit	bc:f6:85:c9:ae:68	select
2	MurataMa	edit	f0:27:65:40:0d:b2	select
2	Lifetron	edit	00:0f:60:0c:1b:d1	select

Note: While scanning the ESP8266 will shut down its access point, so you may have to go to your settings and reconnect to the WiFi network manually.

...and start different attacks.



Attack

Selected AP:
die-link

Selected Clients:
FF:FF:FF:FF:FF:FF - BROADCAST

Attack	Status	Start/Stop
deauth	ready	start
beacon (clone)	ready	start
beacon (list)	60pkts/s	stop

deauth [deauthentication attack]:
Sends deauthentication frames and dissociation frames to the selected client(s) in the selected WiFi access point(s).
Note: If no client is selected, the packets are sent as broadcast!

beacon [beacon flood attack]:
(clone:) spams beacon frames with a similar SSID as the selected WiFi access point(s).
(list:) spams beacon frames with all SSIDs in the list below.

SSIDs: 6/64

Name	X
Tell My Wi-Fi Love Her	x
House LANister	x
LAN Solo	x
Loading...	x
No Free Wi-Fi Here	x
404 Wi-Fi Unavailable	x

saved

Happy hacking :)

FAQ

Could it auto-deauth all APs in the range?

Yes, but I will not implement this 'feature' for ethical and legal reasons.

Can it sniff handshakes?

The ESP8266 has a promiscuous mode in which you can sniff packets, but handshake packets are dropped and there is no other way to get them with the functions provided by the SDK.
Maybe someone will find a way around this barrier but I wasn't able to.

espcomm_sync failed/espcomm_open when uploading

The ESP upload tool can't communicate with the chip, make sure the right port is selected!
You can also try out different USB ports and cables.
If this doesn't solve it you may have to install USB drivers.
Which drivers you need depends on the board, most boards use a cp2102, cp2104 or ch340.

AP scan doesn't work

There is a reported issue on this: https://github.com/spacehuhn/esp8266_deauther/issues/5
Try out switching the browser or open the website with another device.

Deauth attack won't work

If you see 0 pkts/s on the website you have made a mistake. Check if you have followed the the installation steps correctly and that the right SDK installed, it must be version 2.0.0!
If it can send packets but your target don't loose its connection then the WiFi router uses 802.11w and it's protected against such attacks or they communicate via 5 GHz WiFi, which the ESP8266 doesn't support.

###If you have other questions or problems with the ESP8266 you can also check out the official community forum.

License

This project is licensed under the MIT License - see the license file file for details

Sources and additional links

deauth attack: https://en.wikipedia.org/wiki/Wi-Fi_deauthentication_attack

deauth frame: https://mrncciew.com/2014/10/11/802-11-mgmt-deauth-disassociation-frames/

ESP8266:

- https://de.wikipedia.org/wiki/ESP8266
- https://espressif.com/en/products/hardware/esp8266ex/overview

packet injection with ESP8266:

- http://hackaday.com/2016/01/14/inject-packets-with-an-esp8266/
- http://bbs.espressif.com/viewtopic.php?f=7&t=1357&p=10205&hilit=wifi_pkt_freedom#p10205
- https://github.com/pulkin/esp8266-injection-example

802.11w-2009: https://en.wikipedia.org/wiki/IEEE_802.11w-2009

wifi_send_pkt_freedom function limitations: http://esp32.com/viewtopic.php?f=13&t=586&p=2648&hilit=wifi_send_pkt_freedom#p2648

esp32 esp_wifi_internal function limitations: http://esp32.com/viewtopic.php?f=13&t=586&p=2648&hilit=wifi_send_pkt_freedom#p2648

Videos: