

The Emperor Has No Cloak – WEP Cloaking Exposed

Deepak Gupta, Vivek Ramchandran

Security Research Team (Amit, Gopi, Pravin)

AirTight Networks

www.airtightnetworks.net

Background

NETWORKWORLD

Vendor aims to 'Cloak' WEP, April 2007

- " .. [Cloaking] is designed to protect a widely used but flawed wireless LAN encryption protocol .."
- " .. Cloaking module creates dummy data traffic ... attacker can't tell the difference between product frames from the WLAN and spoofed frames generated .. "

- ◆ Claim: WEP key cracking can be prevented using "WEP Cloaking".
- ◆ Question: Is it safe to use WEP again now that we have a cloak for it?

Talk Outline

- ◆ Understanding the concept of WEP Cloaking
- ◆ How does WEP Cloaking prevent current cracking tools from getting the WEP key?
- ◆ Performance of Cracking tools in the presence of chaffing
- ◆ Is it so easy to protect WEP? Can it stop the next TJX from happening? ... Or is it too good to be true?
- ◆ 3 Different Techniques to counter WEP cloaking
- ◆ The final Verdict on WEP Cloaking
- ◆ Demo + Tools release
- ◆ Q&A

Where are we ... ?

- ◆ **Understanding the concept of WEP Cloaking**
- ◆ How does WEP Cloaking prevent current cracking tools from getting the WEP key?
- ◆ Performance of Cracking tools in the presence of chaffing
- ◆ Is it so easy to protect WEP? Can it stop the next TJX from happening? ... Or is it too good to be true?
- ◆ 3 Different Techniques to counter WEP cloaking
- ◆ The final Verdict on WEP Cloaking
- ◆ Demo + Tools release
- ◆ Q&A

What is WEP Cloaking?

- ◆ WEP cloaking is a technique of mixing “chaff” frames with legitimate WEP encrypted frames to prevent WEP cracking tools from cracking the secret WEP key
 - Chaff frames are injected by a WEP cloaking tool (typically a WIPS sensor).
 - Chaff frames are carefully spoofed so as to make them indistinguishable from the legitimate WEP encrypted traffic.
 - Chaff frames are encrypted using different WEP key(s) or may contain a random payload.
- ◆ The current release of WEP key cracking tools such as AirCrack and AirSnort either produce wrong results or don't converge to the WEP key in the presence of “chaff” frames.

What are Chaff packets?

- ◆ Chaff packets are WEP encrypted packets which try to mislead the decision making process of cracking tools.
- ◆ Not all WEP encrypted packets qualify as Chaff; Only those which satisfy any one of the FMS or Korek conditions can cause a bias in the cracking logic.
- ◆ Chaffer can craft the IV and the first two encrypted bytes of the Chaff packet to make it satisfy an FMS or Korek condition.

The spectrum of possible Chaff packets

- ◆ Chaffer can generate Chaff packets with one key to force crackers to converge to that key.
- ◆ Chaffer can generate Chaff packets with random keys to force a large bias towards many possibilities for each byte.
- ◆ Chaffer can generate Chaff packets using the authorized network key but randomizing the first two encrypted bytes. This has the effect of both masking legitimate IVs as well as randomizing the bias.

Before we beginlet us choose a benchmarking tool among crackers

- ◆ Since the time when the FMS + Korek statistical attacks on WEP were made public, more than a dozen tools were released under Open Source.
- ◆ Not all tools exhaustively implement these statistical attacks on WEP.
- ◆ We decided to compare various tools before deciding which one to use for our experiments.

Comparison of Wep Cracking tools : Infocus article on securityfocus.

www.zeallsoft.com			128 bit Cracking Time in Seconds						
Data Packets	Weak IVs	Unique IVs	aircrack	aircrack (4)	AirSnort	WepLab	WepLab (95)	WEPCrack	dwepcrack
23457438	8560	16775533	Failed	245	92	Failed	244	Failed	Error
21016149	1807	16775167	Failed	249	41	Failed	247	Failed	Failed
19584364	9340	16275925	Failed	230	114	Failed	229	Failed	Failed
15690079	8694	12860342	Failed	184	90	Failed	179	Failed	Error
15628308	5505	12361369	Failed	176	70	Failed	174	Failed	Failed
11743639	8473	11743639	Failed	154	69	Failed	153	Failed	Error
11739339	3037	11693841	Failed	150	Failed	Failed	151	Failed	Failed
7829104	1001	5031233	Failed	74	Failed	Failed	77	Failed	Error
7799213	5225	7779299	Failed	87	37	Failed	101	Failed	Failed
4175159	1554	4069824	52	51	Failed	Failed	54	Failed	Failed
3914568	767	3914568	Failed	Failed	Failed	Failed	Failed	Failed	Error
3914553	3958	3914553	48	49	Failed	Failed	56	Failed	Error
3884657	1490	3864743	48	46	Failed	Failed	52	Failed	Failed
978652	986	978652	Failed	Failed	Failed	Failed	11	Failed	Error
978633	371	978633	Failed	12	Failed	Failed	13	Failed	Error
977219	264	974902	Failed	9	Failed	Failed	13	Failed	Failed
684992	143	684992	8	8	Failed	Failed	11	Failed	Error
683605	238	681288	Failed	18	Failed	Failed	13	Failed	Failed
587184	117	587184	Failed	27	Failed	Failed	Long	Failed	Error
489293	103	489293	8	7	Failed	5	5	Failed	Error
489286	115	489286	15	16116	Failed	Failed	Long	Failed	Error
391465	78	391465	5	13	Failed	Failed	Long	Failed	Error
391433	78	391433	Failed	6	Failed	Failed	6	Failed	Error
293596	65	293596	Failed	5	Failed	Failed	Long	Failed	Error
293579	65	293579	Failed	Failed	Failed	Failed	Failed	Failed	Error

Table 1. 128 bit WEP Cracking Times (in seconds).

Source: <http://www.securityfocus.com/infocus/1814>

Conclusions drawn from the article and independently verified by us

- ◆ Among all, Aircrack is the most reliable and popular tool for WEP cracking. It is able to crack keys with approximately 250,000 samples for 40bit and 500,000 samples for 104bit WEP key.
- ◆ It's cracking logic exhaustively implements all known attacks on WEP (FMS + Korek) and also allows fine tuning of various parameters while cracking the key (fudge factor, brute forcing, disabling attacks etc).
- ◆ The code is well written (though not documented) and is actively maintained thus providing a reliable platform to add additional logic to the cracking process.

Due to all the above reasons we have chosen Aircrack as a benchmarking tool for our experiments.

Where are we ... ?

- ◆ Understanding the concept of WEP Cloaking
- ◆ **How does WEP Cloaking prevent current cracking tools from getting the WEP key?**
- ◆ Performance of Cracking tools in the presence of chaffing
- ◆ Is it so easy to protect WEP? Can it stop the next TJX from happening? ... Or is it too good to be true?
- ◆ 3 Different Techniques to counter WEP cloaking
- ◆ The final Verdict on WEP Cloaking
- ◆ Demo + Tools release
- ◆ Q&A

Can Cloaking Prevent WEP Cracking?

Before we can answer this question, let's:

1. Review the operation of AirCrack (popular WEP key cracking tool)
2. Understand different ways 'WEP Cloaking' can possibly confuse AirCrack
3. Think about how AirCrack can become 'smarter' and avoid getting fooled by a WEP Cloaker

AirCrack Review

- History of AirCrack
 - ◆ Original Version
 - Collection of tools to perform packet capture, injection and WEP/WPA cracking
 - Written by Christophe Devine
 - <http://www.wirelessdefence.org/Contents/AircrackMain.htm>
 - ◆ Next gen Aircrack: Aircrack-ng
 - Improved cracking
 - <http://www.aircrack-ng.org>
- Aircrack-ng 0.7 is used in our current work
 - ◆ For simplicity, we call it AirCrack in the rest of the discussion

AirCrack Review (WEP Cracking Basics)

- AirCrack is based on exploiting weaknesses in the RC4 implementation of WEP [FMS paper, Korek attack]
 - ◆ Aircrack works by testing each IV in the trace file to check if it reveals any information about the WEP key.
 - ◆ The check consists of running the IV and the first 2 encrypted bytes of the packet through various Korek conditions.
 - ◆ If the IV is weak i.e. satisfies a Korek condition, then it is used to guess a possible value for a byte of the WEP key.
 - ◆ After all the IVs in the trace are used, Aircrack checks if it can crack the WEP key.

Weak IVsin more detail

- ◆ IV is 3 byte prefix of WEP key; The IV is transmitted in clear text with the encrypted packet.
- ◆ The term “Weak IV” was coined by FMS in their classical paper “Weaknesses in the Key Scheduling Algorithm of RC4” . Soon after the discovery of more such statistical attacks by Korek, the Weak IV definition was broadened to include the Korek conditions as well.
- ◆ In the light of the above, an IV is called weak if it satisfies any of the FMS + Korek conditions. Any of these conditions if met would help in guessing a byte of the WEP key.
- ◆ A weak IV leaks out information about a specific key byte with probability ranging from 5%-13%.
- ◆ In 24-bit IV space approximately 16,000 are key independent weak IVs. (.095% of entire IV space)
- ◆ In 24-bit IV space approximately 50,000 are key dependent weak IVs (.35% of entire IV space). This value has been calculated empirically through experimentation.

AirCrack Review – the Cracking Logic

- ◆ Init:
 - Preprocess the input packet trace & generate a list of packets with unique IVs; ignore duplicates

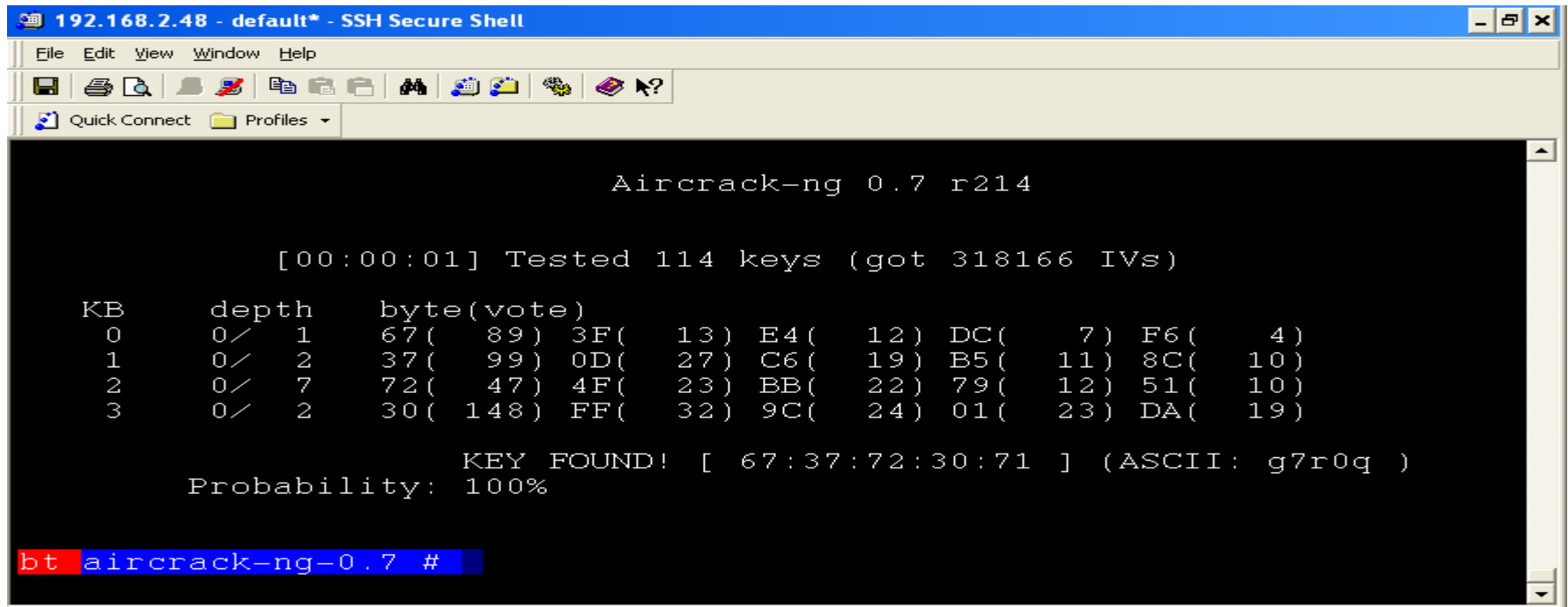
- ◆ Iteration:
 - To crack the B^{th} byte of the key assume the first $B-1$ bytes of the secret key have already been cracked. Start with $B=0$.

 - To crack B^{th} byte of the secret key
 - ◆ Simulate the first $B+3$ steps of the KSA
 - ◆ Find the next weak IV (matching any Korek condition) which leaks information about byte B of the secret WEP key; For the above IV
 - Compute a probable value for key byte B based on which Korek condition matched
 - Award a score (vote) for the above guess

 - ◆ After all unique IVs are processed,
 - Calculate weighted score for each possibility
 - The most probable value of secret byte B is the value with the highest score
 - ◆ Use the fudge factor to determine number of possibilities to use for bruteforce for each byte. By default fudge factor is 5 for 40 bit key and 2 for 104 bit key

- ◆ Crack the last key byte using brute force; Verify against 32 IVs from the list of IVs if the key is right

Example of Operation of AirCrack



```
192.168.2.48 - default* - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
AirCrack-ng 0.7 r214
[00:00:01] Tested 114 keys (got 318166 IVs)
KB      depth  byte(vote)
0       0/ 1    67( 89) 3F( 13) E4( 12) DC( 7) F6( 4)
1       0/ 2    37( 99) 0D( 27) C6( 19) B5( 11) 8C( 10)
2       0/ 7    72( 47) 4F( 23) BB( 22) 79( 12) 51( 10)
3       0/ 2    30( 148) FF( 32) 9C( 24) 01( 23) DA( 19)
KEY FOUND! [ 67:37:72:30:71 ] (ASCII: g7r0q )
Probability: 100%
bt airCrack-ng-0.7 #
```

- ◆ Traffic Characteristics
 - TCP based file download using WEP key "g7r0q"
 - Approx. 1 GB of the traffic collected in a packet trace
 - About 300,000 Unique IVs present in the trace
 - 6,958 Weak IVs were used to crack the key.
- ◆ AirCrack is able to crack the WEP Key using the above trace
- ◆ Note that the maximum vote (bias) is in the range of 47 to 148

Our observation

- ***Max vote for any cracked key byte is typically less than 250 on a "normal" (unchaffed) trace***

AirCrack – Possible Attack Points

◆ Init:

- Preprocess the input packet trace & generate a list of packets with unique IVs; **ignore duplicates**

◆ Iteration:

- To crack the Bth byte of the key assume the first B-1 bytes of the secret key have already been cracked. Start with B=0.

• To crack byte B of the secret key

- ◆ Simulate the first B+3 steps of RC4 KSA
- ◆ **Find the next weak IV** (matching any Korek condition) which leaks information about byte B of the secret WEP key; For the above IV
 - Compute a probable value for key byte B based on which Korek condition matched
 - Award a score (vote) for the above guess
- ◆ After all unique IVs are processed,
 - Calculate weighted score for each possibility
 - The most probable value of secret byte B is = value with the highest score
- ◆ Use the **fudge factor** to determine number of possibilities to use for brute force for each byte. By default fudge factor is 5 for 40 bit key and 2 for 104 bit key.

- ◆ **Crack the last key byte using brute force; Verify against 32 IVs from the list of IVs if the key is right**

Attacking AirCrack

- ◆ Attack point (1) "Eliminate legit IVs": If chaffer's packet containing weak IV is seen before a legit packet with the same IV, legit packets with weak IVs will be ignored by AirCrack.
- ◆ Attack point (2) "Influence voting logic": Chaffer can inject packets with weak IVs matching Korek conditions and in turn influence the voting logic.
- ◆ Attack point (3) "Beat Fudging": Maximum fudge factor allowed is 32. Hence the Chaffer can easily create a bias such that the legit key byte is never included for brute forcing.
- ◆ Attack point (4) "Beat the verification step": After a key is cracked, it is verified against a selected subset of IVs and first two encrypted bytes in the packet. If this set contains chaff, Aircrack will exit with failure.

Attack point (1) Eliminate Legit IVs

- ◆ Without Chaffing:

IV reuse is not allowed as the same IV will provide the same guess always. Hence reusing it is of no help.

- ◆ With Chaffing:

If chaffer's packet containing weak IV is seen before legit packet with the same weak IV, legit IVs will be ignored by AirCrack

Simple remedy

- ◆ Do not drop duplicate IVs
 - Aircrack implements a function called `uniqueiv_check()` to check if the IV has already been seen. Simple removing this functionality from the code would do the job.

Attack point (2) Influence Voting Logic

- ◆ Without chaffing there is 5%-13% chance (depending on which Korek conditions are satisfied) that a Weak IV packet will lead to a correct guess.

- ◆ With chaffing
 1. Bias will change towards a different value (byte B of the key used for crafting chaff frames) if chaff frames are encrypted using one key; or
 2. Bias will change towards multiple values if chaff frames are encrypted using multiple keys; or
 3. high votes will be assigned to all possible values if random keys are used for encrypting chaff frames

Attack point (3) Beat Fudging

- ◆ Without Chaffing:

Aircrack takes the vote of the most possible byte at any stage and divides it by the fudge factor. All bytes with votes greater than this value will be included for brute forcing for that byte.

- ◆ With Chaffing:

The largest fudge factor allowed is 32. The attacker can easily craft a large number of weak IV packets to create a huge vote, whose quotient, when divided by the fudge factor will still be greater than the real network key byte's vote. Also modifying Aircrack to allow a very large factor opens up too many possibilities making brute force impractical.

Attack point (4) Beat the Verification Step

- ◆ Without Chaffing:

After a possible key is generated, Aircrack picks 32 IVs and the corresponding first 2 encrypted bytes from its repository and verifies if it can decrypt them successfully. If decryption of more than 2 IVs in this list fails then the key is discarded. The 32 IVs are chosen by picking every 5th IV from the start of the list.

- ◆ With Chaffing:

If this set of 32 IVs contains more than two Chaff packets, Aircrack will exit with discard the key even if it is the correct one.

Where are we ... ?

- ◆ Understanding the concept of WEP Cloaking
- ◆ How does WEP Cloaking prevent current cracking tools from getting the WEP key?
- ◆ **Performance of Cracking tools in the presence of chaffing**
- ◆ Is it so easy to protect WEP? Can it stop the next TJX from happening?... Or is it too good to be true?
- ◆ 3 Different Techniques to counter WEP cloaking
- ◆ The final Verdict on WEP Cloaking
- ◆ Demo + Tools release
- ◆ Q&A

Example (1): Chaff frames encrypted using one WEP key

- ◆ Legitimate traffic
 - A Dlink DWL 650 client card connected to a Dlink DWL G730 AP
 - Traffic: TCP based file download, ICMP ping
 - About 300,000 unique IVs present in the trace
 - 48-bit WEP key for legitimate traffic: g7r0q
- ◆ Chaff traffic characteristics in the trace
 - About 2500 unique Chaff frames were present in the trace. There were multiple occurrences of the same Chaff packet in the trace.
 - 48-bit WEP key for the chaff frames: ABCDE

Example (1): Chaff frames encrypted using one WEP key

```
192.168.2.48 - default* - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles

Aircrack-ng 0.7

[00:00:03] Tested 0 keys (got 320881 IVs)

KB      depth  byte(vote)
0       0/ 1    41(1580) 67( 69) 3F( 13) E4( 12) F6( 9) DC( 7)
1       0/ 1    42(1171) 5D( 104) 76( 55) B1( 43) C8( 42) 33( 40)
2       0/ 1    43(1915) 6A( 28) 8D( 27) D6( 25) 94( 20) 6C( 15)
3       0/ 1    44(1863) 7A( 130) E6( 45) 49( 40) 73( 23) 8D( 23)
4       0/ 1    45(1614) A7( 75) 50( 34) AB( 32) D9( 32) D5( 28)

Attack failed. Possible reasons:

* Out of luck: you must capture more IVs. Usually, 104-bit WEP
  can be cracked with about one million IVs, sometimes more.

* If all votes seem equal, or if there are many negative votes,
  then the capture file is corrupted, or the key is not static.

* A false positive prevented the key from being found. Try to
  disable each korek attack (-k 1 .. 17), raise the fudge factor
  (-f)

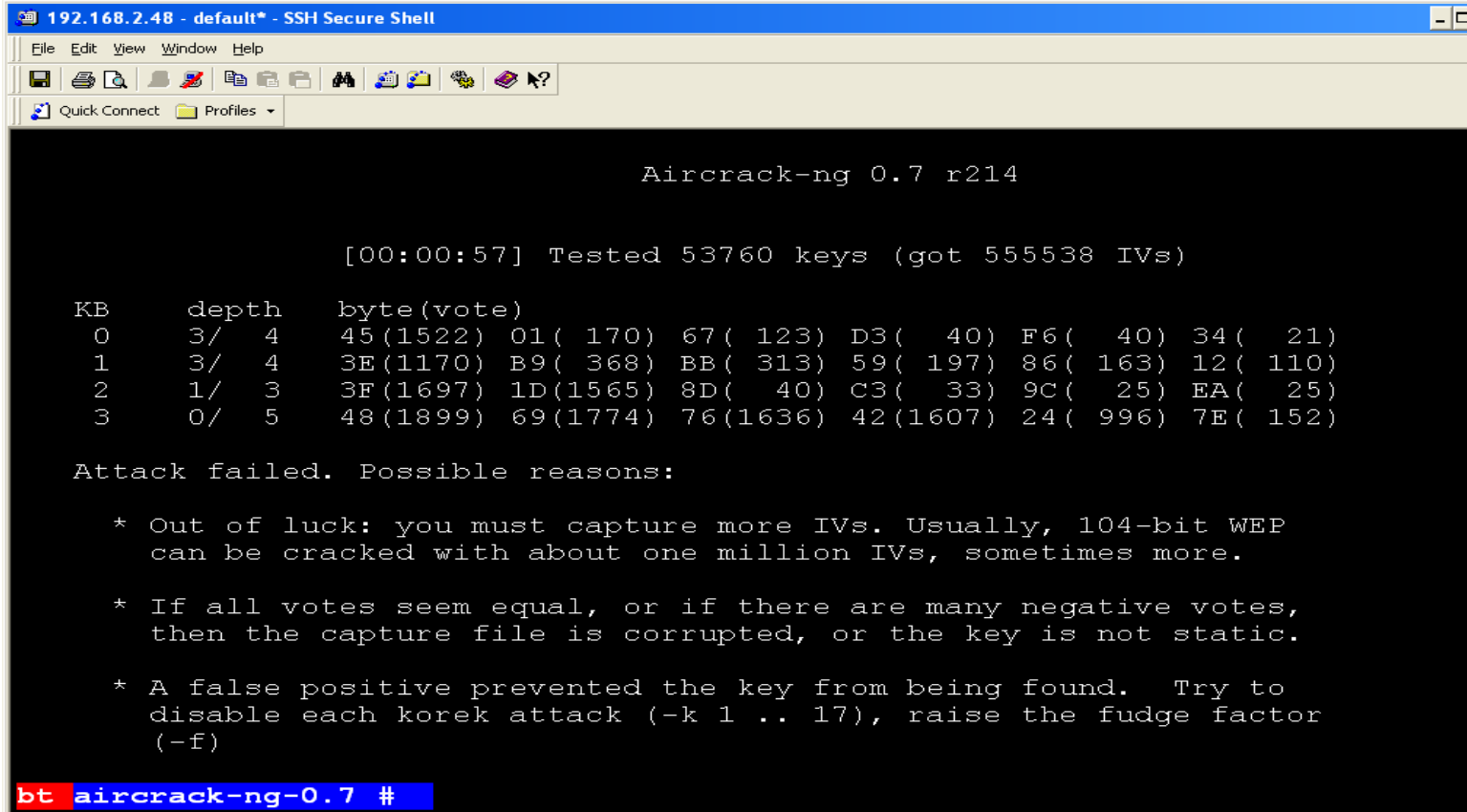
bt aircrack-ng-0.7 #
```

- ◆ In the above example, AirCrack shows a very strong bias to the chaffer's key "ABCDE" (41,42,43,44,45)
- ◆ As the volume of Chaff is low, the verification step fails using the Chaffers key, because of which Aircrack exits

Example (2): Chaff frames encrypted using multiple (four) WEP keys

- ◆ Legitimate traffic
 - A Dlink DWL 650 client card connected to a Dlink DWL G730 AP
 - Traffic: TCP based file download, ICMP ping
 - About 500,000 unique IVs present in the trace
 - 48-bit WEP key for legitimate traffic: g7r0q
- ◆ Chaff traffic characteristics in the trace
 - 4 sets of chaff frames were present in the trace (each set with Weak IVs corresponding to one of the four keys)
 - About 10000 unique weak IV Chaff packets were present
 - 48-bit WEP keys for chaff frames: ABCDE, 187F2, ED9B8, C1D30

Example (2): Chaffer using multiple (four) WEP keys



```
192.168.2.48 - default* - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles

Aircrack-ng 0.7 r214

[00:00:57] Tested 53760 keys (got 555538 IVs)

KB    depth  byte(vote)
0     3/ 4    45(1522) 01( 170) 67( 123) D3(  40) F6(  40) 34(  21)
1     3/ 4    3E(1170) B9( 368) BB( 313) 59( 197) 86( 163) 12( 110)
2     1/ 3    3F(1697) 1D(1565) 8D(  40) C3(  33) 9C(  25) EA(  25)
3     0/ 5    48(1899) 69(1774) 76(1636) 42(1607) 24( 996) 7E( 152)

Attack failed. Possible reasons:

* Out of luck: you must capture more IVs. Usually, 104-bit WEP
  can be cracked with about one million IVs, sometimes more.

* If all votes seem equal, or if there are many negative votes,
  then the capture file is corrupted, or the key is not static.

* A false positive prevented the key from being found. Try to
  disable each korek attack (-k 1 .. 17), raise the fudge factor
  (-f)

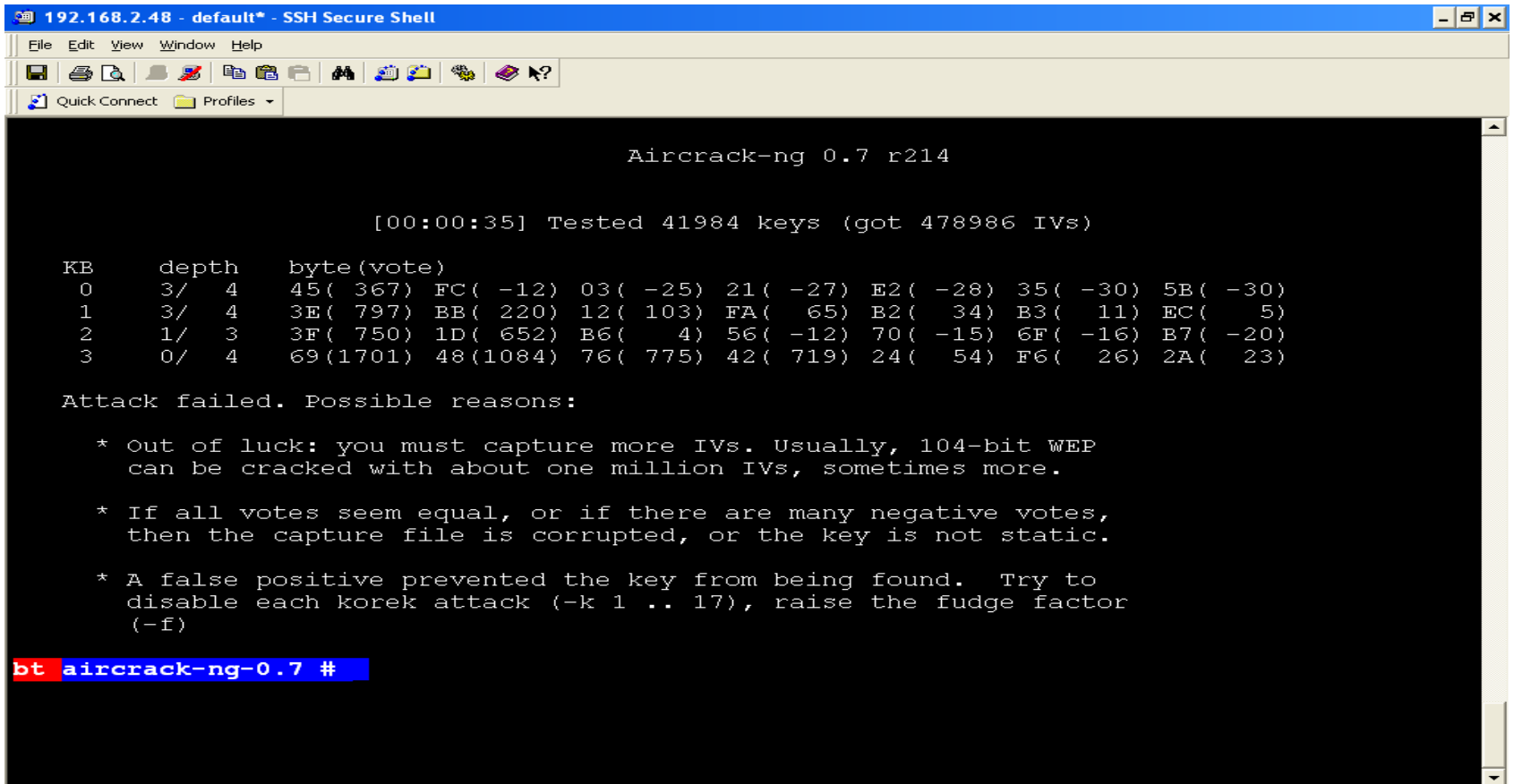
bt aircrack-ng-0.7 #
```

- ◆ In the above example, AirCrack fails to crack the WEP key in presence of a chaffer with four keys.
- ◆ A closer look shows a strong bias towards many values.

Example (3): Chaff frames encrypted using random WEP keys

- ◆ Legitimate traffic
 - A Dlink DWL 650 client card connected to a Dlink DWL G730 AP
 - Traffic: TCP based file download, ICMP ping
 - About 500,000 unique IVs present in the trace
 - 48-bit WEP key for legitimate traffic: g7r0q
- ◆ Chaff traffic characteristics in the trace
 - About 38000 unique weak IV Chaff packets were present in the trace
 - The Chaff packets contain random data
 - Simulates the scenario of using random WEP keys

Example (3): Chaffer using Random Keys



```
192.168.2.48 - default* - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
Aircrack-ng 0.7 r214

[00:00:35] Tested 41984 keys (got 478986 IVs)

KB    depth  byte(vote)
0     3/ 4    45( 367) FC( -12) 03( -25) 21( -27) E2( -28) 35( -30) 5B( -30)
1     3/ 4    3E( 797) BB( 220) 12( 103) FA(  65) B2(  34) B3(  11) EC(   5)
2     1/ 3    3F( 750) 1D( 652) B6(   4) 56( -12) 70( -15) 6F( -16) B7( -20)
3     0/ 4    69(1701) 48(1084) 76( 775) 42( 719) 24(  54) F6(  26) 2A(  23)

Attack failed. Possible reasons:

* Out of luck: you must capture more IVs. Usually, 104-bit WEP
  can be cracked with about one million IVs, sometimes more.

* If all votes seem equal, or if there are many negative votes,
  then the capture file is corrupted, or the key is not static.

* A false positive prevented the key from being found. Try to
  disable each korek attack (-k 1 .. 17), raise the fudge factor
  (-f)

bt aircrack-ng-0.7 #
```

- ◆ In the above example, AirCrack has failed to crack the WEP key in presence of chaffer with random keys

Where are we ... ?

- ◆ Understanding the concept of WEP Cloaking
- ◆ How does WEP Cloaking prevent current cracking tools from getting the WEP key?
- ◆ Performance of Cracking tools in the presence of chaffing
- ◆ **Is it so easy to protect WEP? Can it stop the next TJX from happening?... Or is it too good to be true?**
- ◆ 3 Different Techniques to counter WEP cloaking
- ◆ The final Verdict on WEP Cloaking
- ◆ Demo + Tools release
- ◆ Q&A

Seems too good to be true ?

- ◆ Chaffer has to spend significantly high resources to win all the time. If chaffing stops even for a brief period, the attacker might crack the key.
- ◆ **Chaffer has to win always, Attacker has to win only once.**
- ◆ Increasing sophistication of attack on Cloaking is possible; attacker can go off-line; take a lot of time, try a gamut of techniques and possibilities to break the key.
- ◆ Increasing sophistication of chaffing is more difficult; it has to be done continuously;

Where are we ... ?

- ◆ Understanding the concept of WEP Cloaking
- ◆ How does WEP Cloaking prevent current cracking tools from getting the WEP key?
- ◆ Performance of Cracking tools in the presence of chaffing
- ◆ Is it so easy to protect WEP? Can it stop the next TJX from happening?... Or is it too good to be true?
- ◆ **3 Different Techniques to counter WEP cloaking**
- ◆ The final Verdict on WEP Cloaking
- ◆ Demo + Tools release
- ◆ Q&A

Can AirCrack be made 'Smarter'?

- ◆ Aircrack "trusts" what it sees. It does not and in its current form cannot differentiate between the Chaffer's WEP packets (noise) and the Authorized networks WEP packets (signal).
- ◆ Could we teach Aircrack to separate the "Chaff" a.k.a. Noise from the "Grain" a.k.a. Signal?
- ◆ We now present 3 simple yet powerful techniques to separate Chaff packets from a corrupted trace.

Three Simple Noise Reduction Techniques

1. Providing manual guidance to AirCrack
 1. Human eye is very good at spotting anomalies! Our eye can see through the cloak 😊
2. Separating chaff using (sequence number, IV) pattern filtering
 1. Few simple lines of code!
3. Separating chaff using Active Frame replays
 1. Some good guys have already written code for this! (Tools like pcap2air will require trivial modifications to be used with this technique.)

Technique 1:

- 1. Providing manual guidance to AirCrack**
 1. Human eye is very good at spotting anomalies! Our eye can see through the cloak 😊
2. Separating chaff using (sequence number, IV) pattern filtering
 1. Few simple lines of code!
3. Separating chaff using Active Frame replays
 1. Some good guys have already written code for this! (Tools like pcap2air will require trivial modifications to be used with this technique.)

Guiding AirCrack with Manual Inputs

Basic Idea

- ◆ Pattern of votes caused by chaff packets is visibly different than naturally occurring voting distribution
- ◆ At each step of byte cracking, anomalous voting pattern can be identified and the corresponding guess can be eliminated

Simple AirCrack Modification

- While cracking a key byte, compute votes and display on screen.
- Take user's input on which value to choose for that key byte
- User can visually inspect the votes and remove any "obviously wrong" guesses
- Aircrack uses the user's choice as the "guessed byte" for that byte of the key.

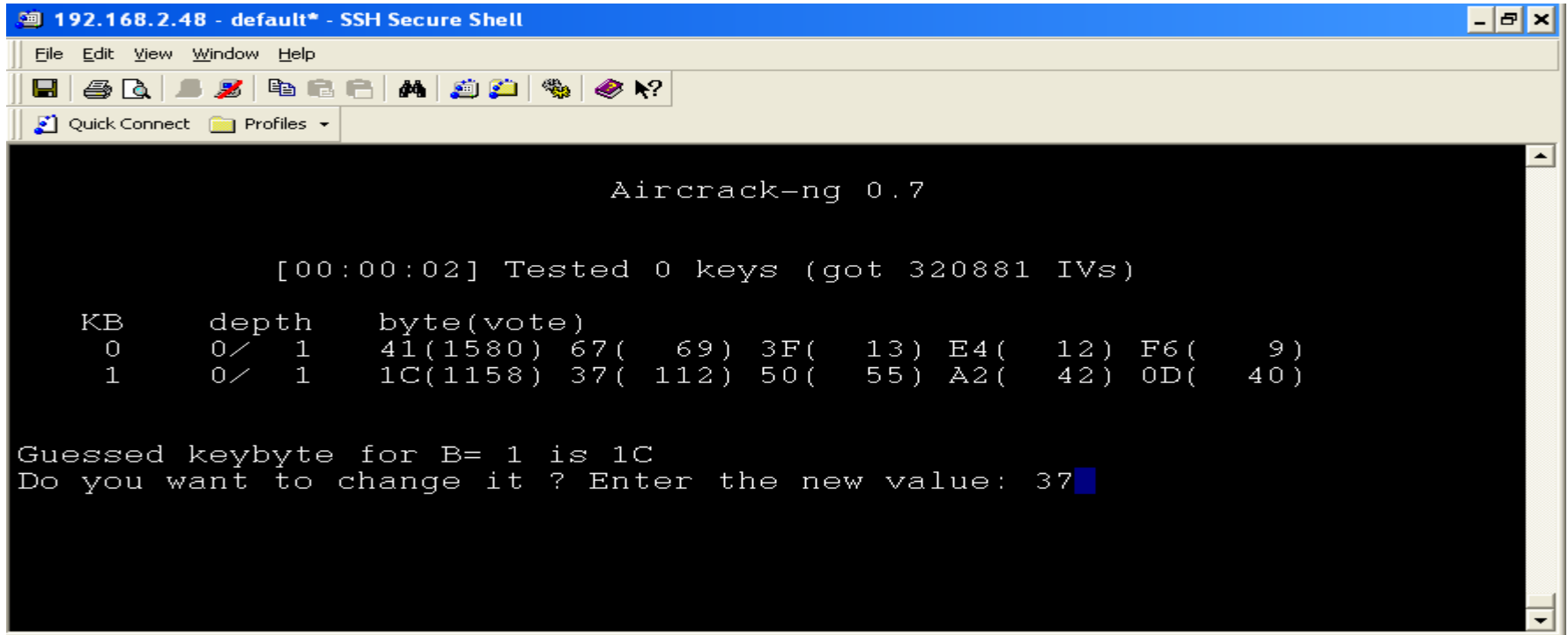
Choice of Manual Inputs

Observations

- ◆ We have observed that for any cracked key byte,
 - maximum vote is typically less than 250 on a “normal” (unchaffed) trace with around unique 350,000 to 500,000 IVs.
 - Maximum vote is not too much larger than other votes
- ◆ However, chaffing creates anomalous vote values
 - Unreasonably large values for votes (e.g., wrong key)
 - Multiple votes much larger than the rest (e.g., chaffing with multiple keys)
- ◆ Instruct (modified) AirCrack to ignore byte guesses corresponding to anomalous votes
- ◆ See example on next slide

Guiding AirCrack with Manual Inputs

Chaff with single key: User input for 2nd key byte

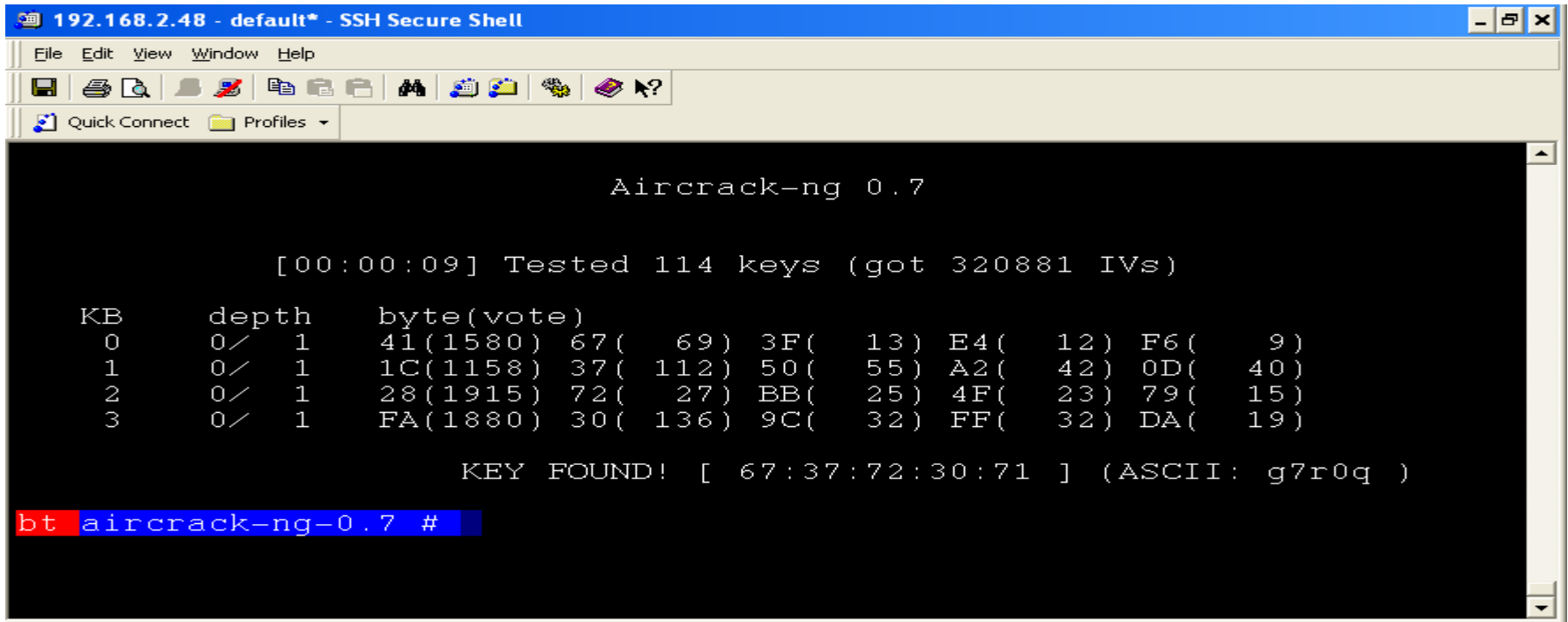


```
192.168.2.48 - default* - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
Aircrack-ng 0.7
[00:00:02] Tested 0 keys (got 320881 IVs)
KB      depth  byte(vote)
 0      0/ 1     41(1580) 67( 69) 3F( 13) E4( 12) F6(  9)
 1      0/ 1     1C(1158) 37( 112) 50( 55) A2( 42) 0D( 40)
Gussed keybyte for B= 1 is 1C
Do you want to change it ? Enter the new value: 37
```

- ◆ (Not shown) User has chosen 67 as the value to be used as first byte (as byte 41 has unreasonably large bias, 1580)
- ◆ In this step, the user is prompted to choose the value for next byte B=1
- ◆ The user chooses the byte 37 (instead of 1C) which has a "reasonable" vote of 112 as the choice to be used by AirCrack
- ◆ The process is continued for other bytes ...

Guiding AirCrack with Manual Inputs

Chaff with single key: AirCrack Succeeds!



```
192.168.2.48 - default* - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
Aircrack-ng 0.7
[00:00:09] Tested 114 keys (got 320881 IVs)
KB      depth  byte(vote)
0       0/ 1     41(1580) 67( 69) 3F( 13) E4( 12) F6( 9)
1       0/ 1     1C(1158) 37( 112) 50( 55) A2( 42) 0D( 40)
2       0/ 1     28(1915) 72( 27) BB( 25) 4F( 23) 79( 15)
3       0/ 1     FA(1880) 30( 136) 9C( 32) FF( 32) DA( 19)
KEY FOUND! [ 67:37:72:30:71 ] (ASCII: g7r0q )
bt aircrack-ng-0.7 #
```

- ◆ User has successfully “guided” the tool to crack the key in presence of chaff that leads to a wrong key
- ◆ It took only 4 iterations to crack the key correctly (2nd highest guess had to be used for each byte)
- ◆ Less than 5 minutes of extra time of the cracker is required!

Manual Input is required : Even using a fudge factor of 32 fails to crack the key.



```
2:192.168.2.48 - default* - SSH Secure Shell
Aircrack-ng 0.7 r214

[00:00:07] Tested 6144 keys (got 320881 IVs)

KB      depth  byte (vote)
0       1/ 2    67( 69) 3F( 13) E4( 12) F6( 9) DC( 7) 0D( 5) 38( 5) 5F( 5)
1       6/ 7    B5( 37) 8C( 36) 5E( 32) C6( 32) 3F( 29) 84( 29) DB( 29) 05( 26)

Attack failed. Possible reasons:

* Out of luck: you must capture more IVs. Usually, 104-bit WEP
  can be cracked with about one million IVs, sometimes more.

* If all votes seem equal, or if there are many negative votes,
  then the capture file is corrupted, or the key is not static.

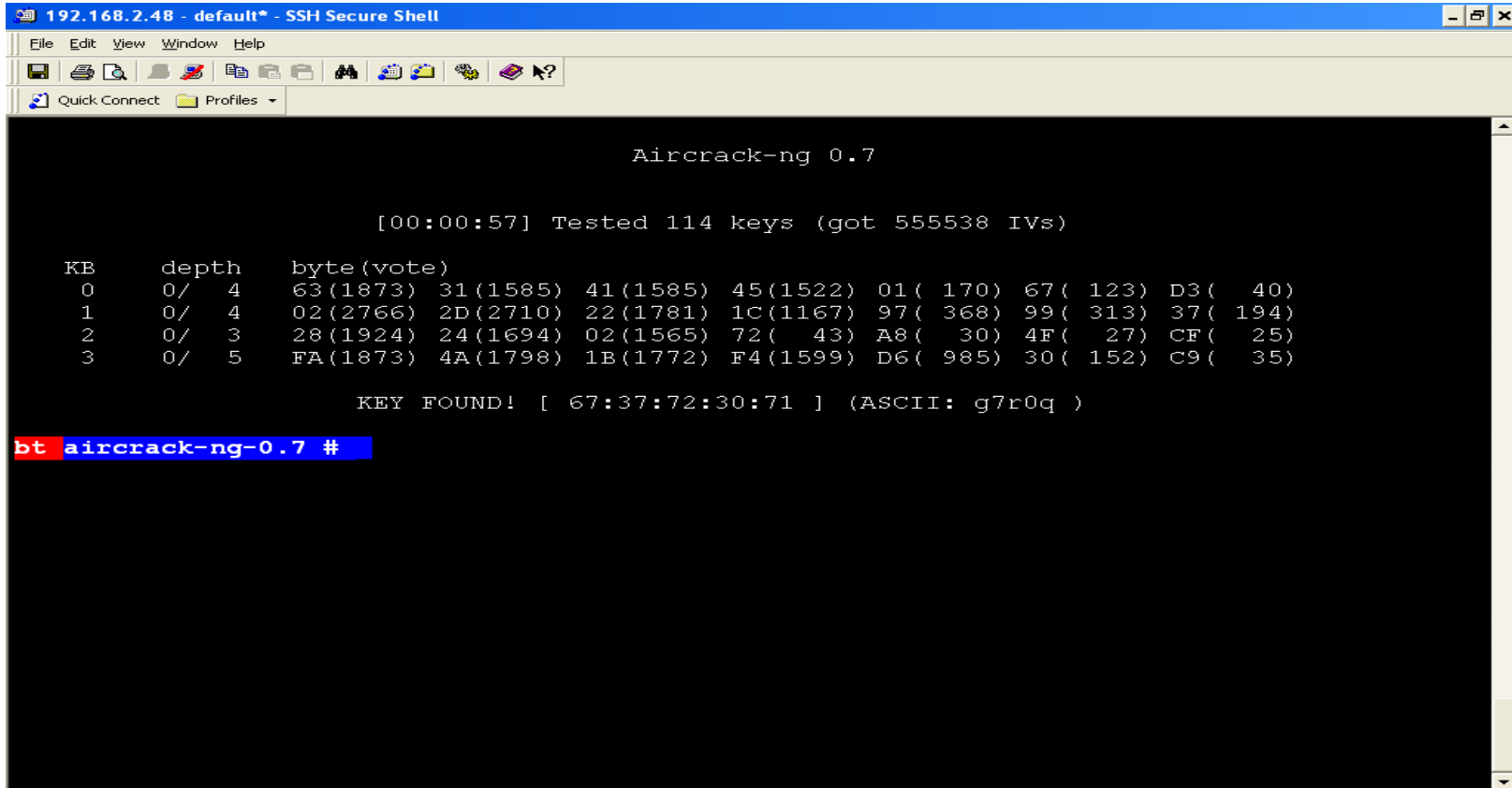
* A false positive prevented the key from being found. Try to
  disable each korek attack (-k 1 .. 17), raise the fudge factor
  (-f)

bt aircrack-ng-0.7 # aircrack-ng -n64 /mnt/hda2/home/amit/dump-100 -f32
```

- ◆ In the previous slide one might be tempted to believe that simply raising the fudge factor might have worked. But this is not that simple. Even with the highest allowed fudge factor of 32 – Aircrack was unable to crack the key.
- ◆ One can modify Aircrack to support a higher fudge factor, but this opens up too many possibilities making brute forcing impractical. Even this will anyway fail if the attacker creates a very large bias.

Guiding AirCrack with Manual Inputs

Chaff with four keys: AirCrack Succeeds!



```
192.168.2.48 - default* - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles

Aircrack-ng 0.7

[00:00:57] Tested 114 keys (got 555538 IVs)

KB      depth  byte (vote)
0       0/ 4    63(1873) 31(1585) 41(1585) 45(1522) 01( 170) 67( 123) D3(  40)
1       0/ 4    02(2766) 2D(2710) 22(1781) 1C(1167) 97( 368) 99( 313) 37( 194)
2       0/ 3    28(1924) 24(1694) 02(1565) 72(  43) A8(  30) 4F(  27) CF(  25)
3       0/ 5    FA(1873) 4A(1798) 1B(1772) F4(1599) D6( 985) 30( 152) C9(  35)

KEY FOUND! [ 67:37:72:30:71 ] (ASCII: g7r0q )

bt aircrack-ng-0.7 #
```

- ◆ (Not shown) 10 iterations were required to crack the key
- ◆ Less than 15 minutes of extra time of the cracker is required!

Technique 2:

1. Providing manual guidance to AirCrack
 1. Human eye is very good at spotting anomalies! Our eye can see through the cloak 😊
2. **Separating chaff using (sequence number, IV) pattern filtering**
 1. Few simple lines of code!
3. Separating chaff using Active Frame replays
 1. Some good guys have already written code for this! (Tools like pcap2air will require trivial modifications to be used with this technique.)

Separating Chaff Using Seq # & IV analysis

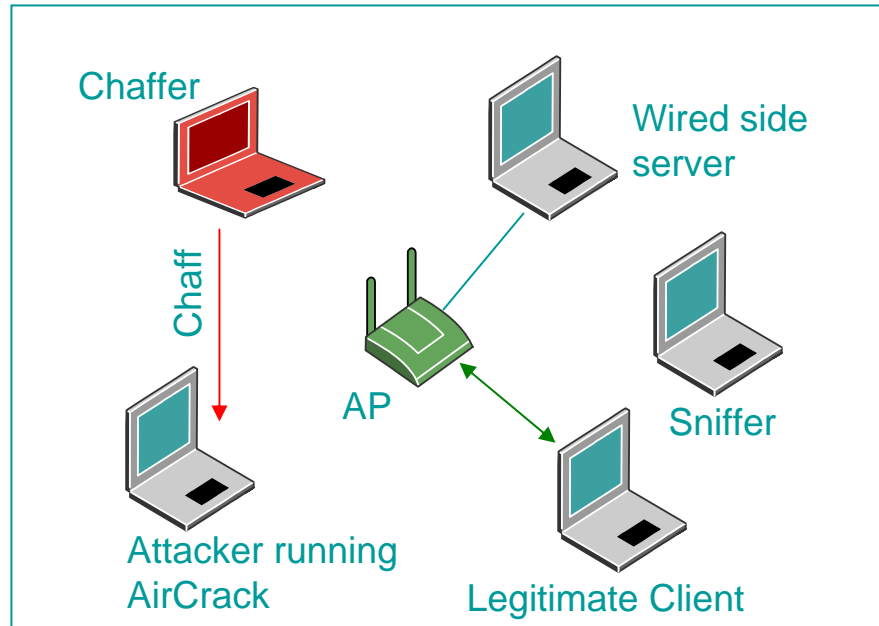
Basic Idea

- ◆ Chaff frames are spoofed frames. MAC spoofing detection is a well understood art now. Spoofed frames can be fingerprinted and separated from a mixed trace
- ◆ It is possible to build a
 - Sequence number filter using borrowed ideas from MAC spoofing detection based on sequence number anomalies [Joshua Wright]
 - IV progression filter based on observation that majority of WEP implementations, IVs in successive packets are sequential
 - The two filters can also be combined for better results

Just a few hours before we submitted this presentation, we came across Joshua Wright's blog in which he countered WEP Cloaking advocating the same technique (sequence number + IV based filtering). This submission will demonstrate the tool whose development Joshua predicted.

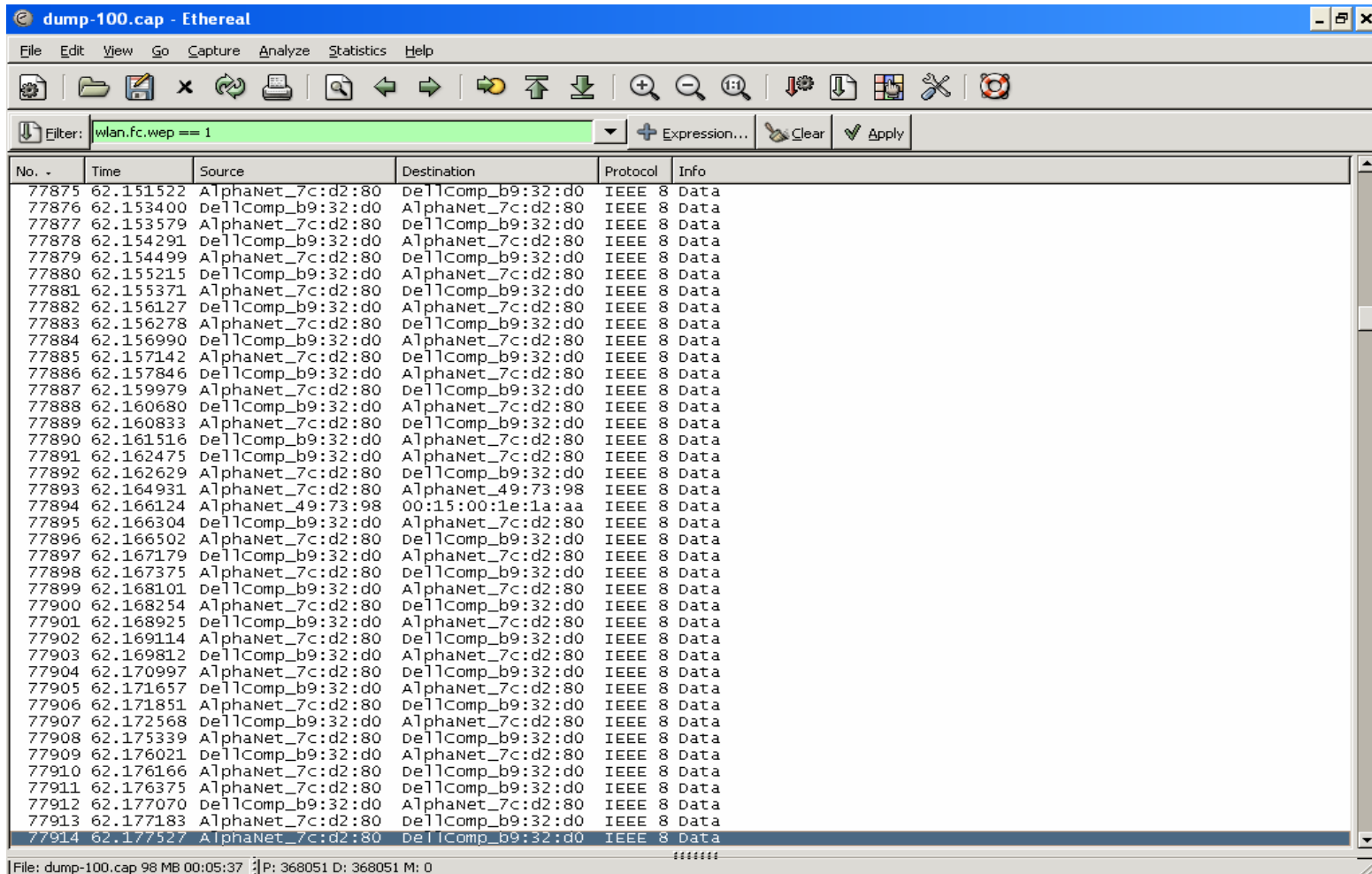
<http://edge.arubanetworks.com/blog/2007/04/airdefense-perpetuates-flawed-protocols>

Separating Chaff Using Seq # & IV analysis – (setup)



- ◆ Set up
 - A Dlink DWL 650 client card connected to a Dlink DWL G730 AP
 - Traffic: TCP based file download, ICMP ping
 - Sniffer to collect trace (Ethereal)

Example Packet Trace



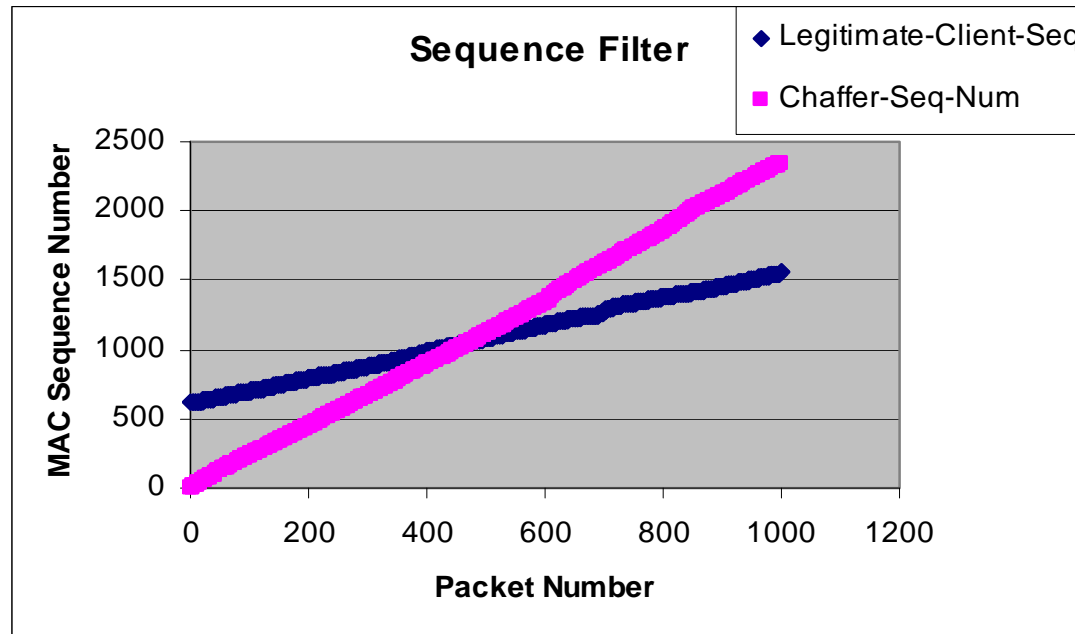
The screenshot shows the Wireshark interface with a packet capture named 'dump-100.cap'. The filter 'wlan.fc.wep == 1' is applied, resulting in a list of 27 packets. All packets are of type IEEE 802.11 Data. The source and destination MAC addresses alternate between AlphaNet_7c:d2:80 and DellComp_b9:32:d0. The time values range from 62.151522 to 62.177527 seconds. The status bar at the bottom indicates the file size is 98 MB and the capture time is 00:05:37.

No.	Time	Source	Destination	Protocol	Info
77875	62.151522	AlphaNet_7c:d2:80	DellComp_b9:32:d0	IEEE 8	Data
77876	62.153400	DellComp_b9:32:d0	AlphaNet_7c:d2:80	IEEE 8	Data
77877	62.153579	AlphaNet_7c:d2:80	DellComp_b9:32:d0	IEEE 8	Data
77878	62.154291	DellComp_b9:32:d0	AlphaNet_7c:d2:80	IEEE 8	Data
77879	62.154499	AlphaNet_7c:d2:80	DellComp_b9:32:d0	IEEE 8	Data
77880	62.155215	DellComp_b9:32:d0	AlphaNet_7c:d2:80	IEEE 8	Data
77881	62.155371	AlphaNet_7c:d2:80	DellComp_b9:32:d0	IEEE 8	Data
77882	62.156127	DellComp_b9:32:d0	AlphaNet_7c:d2:80	IEEE 8	Data
77883	62.156278	AlphaNet_7c:d2:80	DellComp_b9:32:d0	IEEE 8	Data
77884	62.156990	DellComp_b9:32:d0	AlphaNet_7c:d2:80	IEEE 8	Data
77885	62.157142	AlphaNet_7c:d2:80	DellComp_b9:32:d0	IEEE 8	Data
77886	62.157846	DellComp_b9:32:d0	AlphaNet_7c:d2:80	IEEE 8	Data
77887	62.159979	AlphaNet_7c:d2:80	DellComp_b9:32:d0	IEEE 8	Data
77888	62.160680	DellComp_b9:32:d0	AlphaNet_7c:d2:80	IEEE 8	Data
77889	62.160833	AlphaNet_7c:d2:80	DellComp_b9:32:d0	IEEE 8	Data
77890	62.161516	DellComp_b9:32:d0	AlphaNet_7c:d2:80	IEEE 8	Data
77891	62.162475	DellComp_b9:32:d0	AlphaNet_7c:d2:80	IEEE 8	Data
77892	62.162629	AlphaNet_7c:d2:80	DellComp_b9:32:d0	IEEE 8	Data
77893	62.164931	AlphaNet_7c:d2:80	AlphaNet_49:73:98	IEEE 8	Data
77894	62.166124	AlphaNet_49:73:98	00:15:00:1e:1a:aa	IEEE 8	Data
77895	62.166304	DellComp_b9:32:d0	AlphaNet_7c:d2:80	IEEE 8	Data
77896	62.166502	AlphaNet_7c:d2:80	DellComp_b9:32:d0	IEEE 8	Data
77897	62.167179	DellComp_b9:32:d0	AlphaNet_7c:d2:80	IEEE 8	Data
77898	62.167375	AlphaNet_7c:d2:80	DellComp_b9:32:d0	IEEE 8	Data
77899	62.168101	DellComp_b9:32:d0	AlphaNet_7c:d2:80	IEEE 8	Data
77900	62.168254	AlphaNet_7c:d2:80	DellComp_b9:32:d0	IEEE 8	Data
77901	62.168925	DellComp_b9:32:d0	AlphaNet_7c:d2:80	IEEE 8	Data
77902	62.169114	AlphaNet_7c:d2:80	DellComp_b9:32:d0	IEEE 8	Data
77903	62.169812	DellComp_b9:32:d0	AlphaNet_7c:d2:80	IEEE 8	Data
77904	62.170997	AlphaNet_7c:d2:80	DellComp_b9:32:d0	IEEE 8	Data
77905	62.171657	DellComp_b9:32:d0	AlphaNet_7c:d2:80	IEEE 8	Data
77906	62.171851	AlphaNet_7c:d2:80	DellComp_b9:32:d0	IEEE 8	Data
77907	62.172568	DellComp_b9:32:d0	AlphaNet_7c:d2:80	IEEE 8	Data
77908	62.175339	AlphaNet_7c:d2:80	DellComp_b9:32:d0	IEEE 8	Data
77909	62.176021	DellComp_b9:32:d0	AlphaNet_7c:d2:80	IEEE 8	Data
77910	62.176166	AlphaNet_7c:d2:80	DellComp_b9:32:d0	IEEE 8	Data
77911	62.176375	AlphaNet_7c:d2:80	DellComp_b9:32:d0	IEEE 8	Data
77912	62.177070	DellComp_b9:32:d0	AlphaNet_7c:d2:80	IEEE 8	Data
77913	62.177183	AlphaNet_7c:d2:80	DellComp_b9:32:d0	IEEE 8	Data
77914	62.177527	AlphaNet_7c:d2:80	DellComp_b9:32:d0	IEEE 8	Data

◆ Observation

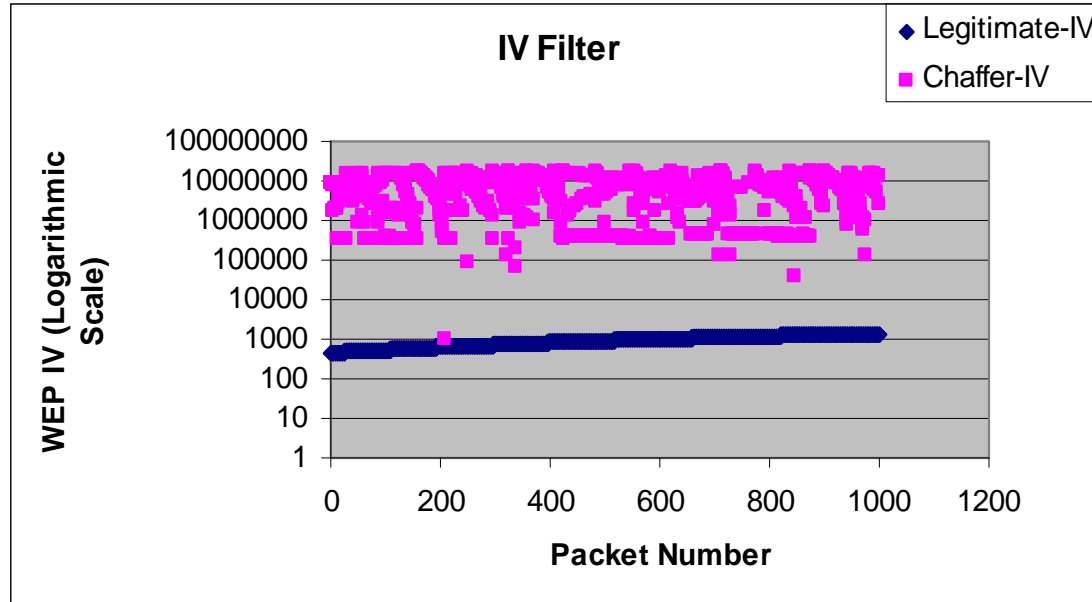
- Chaff appears indistinguishable from legitimate traffic
- But, we can filter chaff!

Seq# Vs Time graph



- ◆ Example Illustrating Sequence Filter Implementation (using a subset of packets from the trace)
- ◆ Note the distinct pattern of sequence numbers in general
- ◆ Sequence num range is short: 0 to 4096
 - Can cause problems to filtering at certain times

WEP IV vs Time Graph



- ◆ Example Illustrating WEP IV Filter Implementation (using a subset of packets from the trace)
- ◆ Note the distinct pattern in IV progression

Separating Chaff using Sequence No and IV : Advantages of this technique

◆ Advantages

- Works with all 3 kinds of chaff discussed – chaffer with single key, multiple keys and random keys
- Passive, off-line method
- Combination of sequence number and IV analyses creates a very robust filter
- Filtering need not be perfect! Even if some noise leaks out, AirCrack already has some tolerance to noise.
- An independent chaff separator can be built

Technique 3:

1. Providing manual guidance to AirCrack
 1. Human eye is very good at spotting anomalies! Our eye can see through the cloak 😊
2. Separating chaff using (sequence number, IV) pattern filtering
 1. Few simple lines of code!
3. **Separating chaff using Active Frame replays**
 1. Some good guys have already written code for this! (Tools like pcap2air will require trivial modifications to be used with this technique.)

Chaff Separating using Active Frame Replay

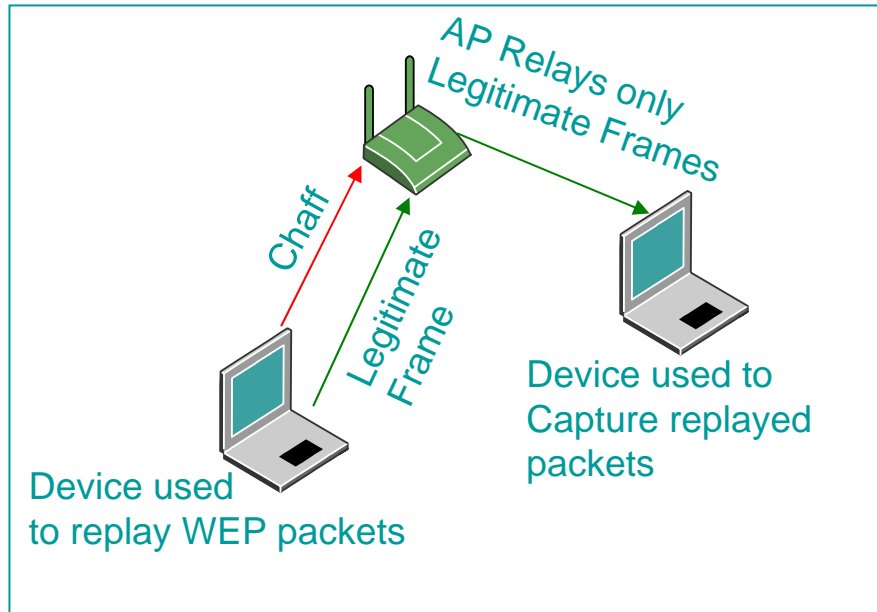
Basic Idea

- ◆ WEP has no replay protection
- ◆ Header of WEP frames can be modified
- ◆ Upon receiving a correctly encrypted WEP frame, the receiver will forward the frame as directed by its header
- ◆ Upon receiving an incorrectly encrypted WEP frame, the receiver will drop the packet
- ◆ Idea inspired from chopchop tool by Korek.

Building a practical Frame Re-player

- ◆ Pick a frame whose authenticity is to be verified
- ◆ Change destination address to ff:ff:ff:ff:ff:ff or a chosen Multicast address and transmit
 - ◆ If the AP relays the broadcast frame – the frame is authentic
 - ◆ If AP drops the frame – it is a chaff frame
- ◆ Replay packets can be identified by looking at the transmitter address (addr3) of packets transmitted by AP
 - Optionally, a signature can be added to identify the replay packets (e.g., specific multicast as destination)
- ◆ 100% chaff separation is possible

Chaff Separating using Active Frame Replay (Setup)

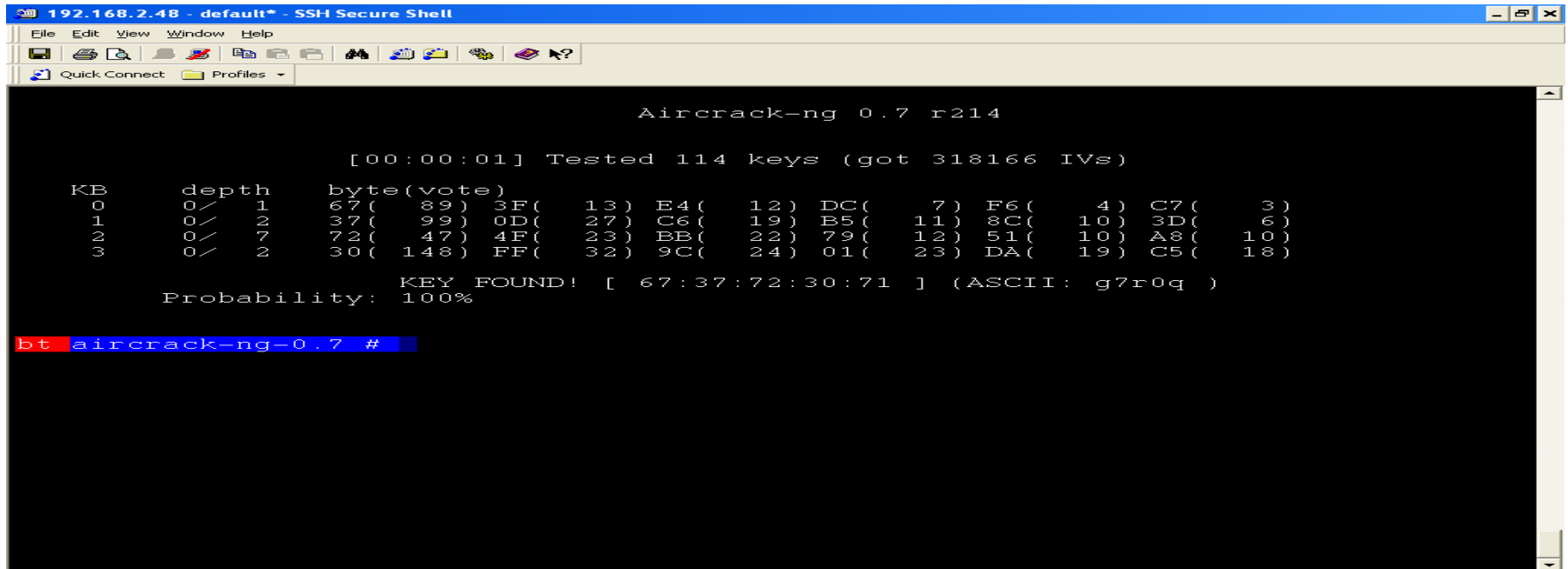


- ◆ Mixture of legitimate WEP frames and chaff was collected in a trace
- ◆ A device (laptop) was used to modify the WEP packets with broadcast destination address and transmit (replay) it to the AP
- ◆ Another device (laptop) was used to capture broadcast packets from the AP
- ◆ AP acts as a "filter" for chaff frames which are not encrypted with the legitimate key and drops it

Chaff Separation using Active Frame Replay (Experiment)

- ◆ In our experiment,
 - Mixed trace contained
 - ◆ 346584 Legitimate frames
 - ◆ 21467 chaff frames
 - Time between successive packet replays was 2 ms
 - ◆ Total time for replay was about 750 sec or less than 15 minutes
 - Passed the trace through a separate program, however, it can be done in real-time/online while capturing the trace using AirCrack

Chaff Separating using Active Frame Replay Successful Key Cracking



```
192.168.2.48 - default* - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles

AirCrack-ng 0.7 r214

[00:00:01] Tested 114 keys (got 318166 IVs)

KB      depth  byte(vote)
0       0/1     67( 89) 3F( 13) E4( 12) DC( 7) F6( 4) C7( 3)
1       0/2     37( 99) 0D( 27) C6( 19) B5( 11) 8C( 10) 3D( 6)
2       0/7     72( 47) 4F( 23) BB( 22) 79( 12) 51( 10) A8( 10)
3       0/2     30( 148) FF( 32) 9C( 24) 01( 23) DA( 19) C5( 18)

KEY FOUND! [ 67:37:72:30:71 ] (ASCII: g7r0q )
Probability: 100%

bt aircrack-ng-0.7 #
```

- ◆ The corrupted trace was filtered using Active replay technique and a new filtered trace was created.
- ◆ AirCrack able to crack the filtered trace after the application of packet replay filter

Separating Chaff using Active Frame Replay

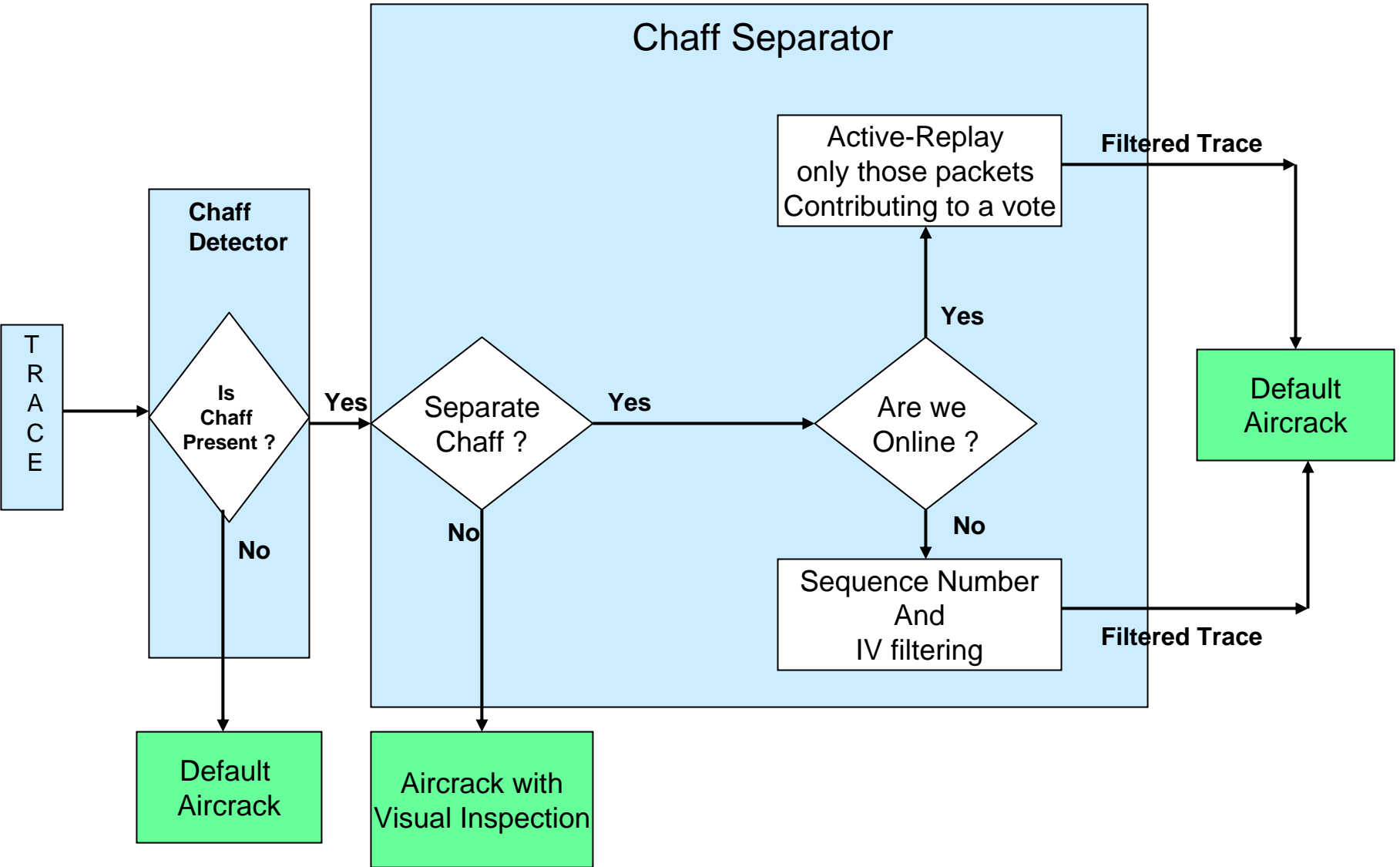
◆ Advantages

- Works with all 3 kinds of chaff discussed – chaffer with single key, multiple keys and random keys
- 100% accurate chaff separation
- Oblivious to the sequence number or IV progression of a device
- Can be done in real-time
- Frame replay tools already available in public domain

Lets now condense these 3 into one

1. Providing manual guidance to AirCrack
 1. Human eye is very good at spotting anomalies! Our Eye can see through the cloak 😊
2. Separating chaff using (sequence number, IV) pattern filtering
 1. Few simple lines of code!
3. Separating chaff using Active Frame replays
 1. Some good guys have already written code for this! (Tools like pcap2air will require trivial modifications to be used with this technique.)

Architecture of a Chaff Resistant Tool built atop Aircrack with these 3 methods



Where are we ... ?

- ◆ Understanding the concept of WEP Cloaking
- ◆ How does WEP Cloaking prevent current cracking tools from getting the WEP key?
- ◆ Performance of Cracking tools in the presence of chaffing
- ◆ Is it so easy to protect WEP ? Can it stop the next TJX from happening?... Or is it too good to be true?
- ◆ 3 Different Techniques to counter WEP cloaking
- ◆ **The final Verdict on WEP Cloaking**
- ◆ Demo + Tools release
- ◆ Q&A

Final Verdict: Implementation problems with Cloaking

Now, we mention several implementation issues which indicate that it may be highly impractical to have a scalable cloaking solution :

- Passive key cracking tools cannot be detected
 - ◆ Cloaking needs to be done 24x7
- Cloaking needs to be done on all channels on which WEP devices operate.
 - ◆ Imagine the load on the WIPS and the bandwidth wasted.
- Cloaking needs to be done for all APs and Clients connected to the authorized network.
- Achieving a reliable confusion for the attacker requires continual generation of chaff frames
 - ◆ Difficult (almost impossible) to achieve the above unless dedicated devices are installed for cloaking on each channel
- We have shown beyond doubt that WEP cloaking can be easily beaten with simple modifications to existing cracking logic.

Final Verdict: WEP Cloaking was indeed too good to be true ...

- ◆ Cloaking cannot provide a robust protection against WEP key cracking. **WEP was broken it is brokenit will remain broken. PERIOD .**
- ◆ WEP Cloaking can at best slow down a Cracker by a couple of minutes but cannot stop him from breaking the key.
- ◆ Though our demo only includes Aircrack, the chaff separation logic in the 3 techniques we have outlined can be easily added to the functionality of any WEP cracking tool, without much additional work.

Where are we ... ?

- ◆ Understanding the concept of WEP Cloaking
- ◆ How does WEP Cloaking prevent current cracking tools from getting the WEP key?
- ◆ Performance of Cracking tools in the presence of chaffing
- ◆ Is it so easy to protect WEP ? Can it stop the next TJX from happening?... Or is it too good to be true?
- ◆ 3 Different Techniques to counter WEP cloaking
- ◆ The final Verdict on WEP Cloaking
- ◆ **Demo + Tools release**
- ◆ Q&A

Demo : “Visual Inspection “ Using an offline Trace File

- We will use a Trace file containing a mix of legitimate traffic and Chaff.
- This trace file will be fed into Aircrack (unmodified). We will observe the key cannot be cracked.
- This trace file will be fed to Aircrack-modified with Visual Inspection functionality. We will observe the key can be cracked now with human guidance.

Where are we ... ?

- ◆ Understanding the concept of WEP Cloaking
- ◆ How does WEP Cloaking prevent current cracking tools from getting the WEP key?
- ◆ Performance of Cracking tools in the presence of chaffing
- ◆ Is it so easy to protect WEP ? Can it stop the next TJX from happening?... Or is it too good to be true?
- ◆ 3 Different Techniques to counter WEP cloaking
- ◆ The final Verdict on WEP Cloaking
- ◆ Demo + Tools release
- ◆ **Q&A**

Questions?

References (1)

- ◆ Vendor aims to 'cloak' WEP
<http://www.networkworld.com/news/2007/032907-air-defense-wep-wireless-devices.html?page=1>
- ◆ The TJX breach using Wireless
<http://www.emailthis.clickability.com/et/emailThis?clickMap=viewThis&etMailToID=2131419424>
- ◆ RC4 stream Cipher basics
<http://en.wikipedia.org/wiki/RC4>
- ◆ Wired Equivalent Privacy (WEP)
http://en.wikipedia.org/wiki/Wired_Equivalent_Privacy
- ◆ Weaknesses in the Key Scheduling Algorithm of RC4, Selected Areas in Cryptography, 2001 - Fluhrer, Mantin and Shamir
http://www.wisdom.weizmann.ac.il/~itsik/RC4/Papers/Rc4_ksa.ps
- ◆ Korek's post on Netstumbler
<http://www.netstumbler.org/showpost.php?p=89036>
- ◆ WEP Dead Again: Part 1 – Infocus, Securityfocus.com
<http://www.securityfocus.com/infocus/1814>
- ◆ WEP Dead Again: Part 2 – Infocus, Securityfocus.com
<http://www.securityfocus.com/infocus/1824>

References (2)

- ◆ Aircrack-ng : WEP Cracker
<http://www.aircrack-ng.org/>
- ◆ Airtsnort : WEP Cracker
<http://airsnort.shmoo.com/>
- ◆ Pcap2air : Packet replay tool
<http://www.802.11mercenary.net/pcap2air/>
- ◆ Chop-Chop : Packet decoder using WEP ICV flaw
<http://www.netstumbler.org/showthread.php?t=12489>
- ◆ Intercepting Mobile Communications: The Insecurity of 802.11 – N.Borisov
<http://www.isaac.cs.berkeley.edu/isaac/mobicom.pdf>
- ◆ Your 802.11 Wireless Network has No Clothes – William Arbaugh
<http://www.cs.umd.edu/~waa/wireless.pdf>
- ◆ Detecting Detectors: Layer 2 Wireless Intrusion Analysis – Joshua Wright
<http://home.jwu.edu/jwright/papers/l2-wlan-ids.pdf>
- ◆ Detecting WLAN MAC address spoofing – Joshua Wright
<http://home.jwu.edu/jwright/papers/wlan-mac-spoof.pdf>
- ◆ WPA/WPA2 the replacement for WEP
<http://en.wikipedia.org/wiki/WPA2>
- ◆ AirDefense Perpetuates Flawed Protocols – Joshua Wright
<http://edge.arubanetworks.com/blog/2007/04/airdefense-perpetuates-flawed-protocols>