The Emperor Has No Cloak – WEP Cloaking Exposed

Vivek Ramachandran
( vivek.ramachandran@airtightnetworks.net )
Deepak Gupta
( deepak.gupta@airtightnetworks.net )

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 Chaffing.

- **Question**: Is it safe to use WEP again now that we have a cloak for it?
Important Note

Our presentation this afternoon concerns a technique that we refer to as "chaff insertion," which recently has been proposed as a way to prevent cracking WEP keys. To avoid any confusion, while our abstract may have mentioned "WEP Cloaking," and while we may mention "WEP Cloaking" during the course of our presentation, it should be noted that WEP Cloaking is the name one company is using to refer to its particular implementation of the chaff insertion technique. Our presentation is not intended as an analysis of or commentary on this company's particular implementation, but rather addresses the technique of chaff insertion in general and demonstrates our belief that the approach is easily defeated and does not provide any useful protection against cracking WEP keys.
Talk Outline

- Evolution of WEP Cracking
- What is WEP Chaffing? What are Chaff packets?
- WEP Chaffing example
- Techniques to counter different types of Chaff:
  - Random frames
  - Single key
  - Multiple keys
  - Random keys
  - ...
- Implementation problems with WEP Chaffing
- Final verdict on WEP Chaffing
- Q&A
Talk Outline

- Evolution of WEP Cracking
- What is WEP Chaffing? What are Chaff packets?
- WEP Chaffing example
- Techniques to counter different types of Chaff:
  - Random frames
  - Single key
  - Multiple keys
  - Random keys
  - ...
- Implementation problems with WEP Chaffing
- Final verdict on WEP Chaffing
- Q&A
WEP Cracking – what is it?

- WEP is a per packet encryption mechanism which uses a combination of a shared secret (WEP Key) and an Initialization Vector (IV) for generating the key stream using the RC4 encryption mechanism. This key stream is XOR’ed with the data and transmitted.

- WEP cracking involves trying to infer the WEP Key using various statistical attacks – FMS, KoreK.

- Lets now look at the historical evolution of WEP Cracking.
IEEE WG admitted that WEP cannot hold any water. Recommended users to upgrade to WPA, WPA2

2001 - The insecurity of 802.11, Mobicom, July 2001
N. Borisov, I. Goldberg and D. Wagner.


2002 - Using the Fluhrer, Mantin, and Shamir Attack to Break WEP
A. Stubblefield, J. Ioannidis, A. Rubin.

2004 – KoreK, improves on the above technique and reduces the complexity of WEP cracking. We now require only around 500,000 packets to break the WEP key.

2005 – Andreas Klein introduces more correlations between the RC4 key stream and the key.

2007 – PTW extend Andreas technique to further simplify WEP Cracking. Now with just around 60,000 – 90,000 packets it is possible to break the WEP key.
This hasn’t stopped people from using band-aids to stop leakage

128-bit key
Suppress weak IV generation
ARP filtering
The Latest Development

Is chaffing approach yet another band-aid which cannot hold water? OR Can chaffing approach indeed hide all WEP cracks?

WEP Chaff frame insertion
Talk Outline

- Evolution of WEP Cracking
- What is WEP Chaffing? What are Chaff packets?
- WEP Chaffing example
- Techniques to counter different types of Chaff:
  - Random frames
  - Single key
  - Multiple keys
  - Random keys
  - ...
- Implementation problems with WEP Chaffing
- Final verdict on WEP Chaffing
- Q&A
What is WEP Chaffing?

- WEP Chaffing is a technique of mixing spoofed WEP encrypted frames a.k.a. “Chaff” which are indistinguishable from the legitimate frames. This confuses the statistical analysis of WEP cracking tools.

- The current versions of WEP key cracking tools such as Aircrack-ng and AirSnort will either produce wrong results or won’t converge on the WEP key in presence of WEP Chaffing.
What are Chaff packets?

- Chaff packets are spoofed WEP encrypted packets which try to mislead the decision making process of cracking tools.

- In reality, 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.

- The WEP Chaffing process will craft the IV and the first two encrypted bytes of the Chaff packet to make it satisfy an FMS or Korek condition.
Talk Outline

- Evolution of WEP Cracking
- What is WEP Chaffing? What are Chaff packets?
- WEP Chaffing example
- Techniques to counter different types of Chaff:
  - Random frames
  - Single key
  - Multiple keys
  - Random keys
  - ...
- Implementation problems with WEP Chaffing
- Final verdict on WEP Chaffing
- Q&A
WEP Chaffing Example
(Demo)
Why does this work?

- Current generation of WEP cracking tools “trust” all data seen over the air

- WEP Chaffing exploits this trust and spoofs garbage data into the air

- This data is blindly used in the statistical analysis for calculating the WEP key – causing either a wrong / no result

- WEP crackers such as Aircrack-ng, Airsnort etc thus fail to crack the key in presence of Chaffing

- As, Aircrack-ng is the most reliable WEP cracker currently available which implements all the known statistical attacks till date. We decided to use Aircrack-ng 0.7 version as a benchmark to run our tests

  Let us now try and understand the key cracking process in Aircrack-ng (0.7 version)
AirCrack-ng Review – the Cracking Logic

- **Init:**
  - Preprocess the input packet trace & store a list unique IVs and first two encrypted bytes; 
    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 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
AirCrack-ng – Possible Attack Points

- **Init:**
  - Preprocess the input packet trace & store a list unique IVs and first two encrypted bytes; **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 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-ng

- **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-ng will exit with failure.
So, can AirCrack-ng be made ‘Smarter’?

- Aircrack-ng “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-ng to separate the “Chaff” a.k.a. Noise from the “Grain” a.k.a. Signal?

- Let’s now look at various techniques to beat WEP Chaffing
Talk Outline

- Evolution of WEP Cracking
- What is WEP Chaffing? What are Chaff packets?
- WEP Chaffing example
- Techniques to counter different types of Chaff:
  - Random frames
  - Single key
  - Multiple keys
  - Random keys
  - ...
- Implementation problems with WEP Chaffing
- Final verdict on WEP Chaffing
- Q&A
Chaff Insertion Approaches
Chaff Insertion Approaches

Necklace

Inject random frames

Naive

Sophisticated
Chaff Insertion Approaches

- Naive: Inject random frames
- Sophisticated: Off goes the necklace

Aircrack-ng Default
Chaff Insertion Approaches

- **Naive**
  - Inject random frames
- **Sophisticated**
  - Crown
  - Aircrack-ng Default
  - Weak IV frames of fixed size
Chaff Insertion Approaches

- Naive
  - Inject random frames
  - Missing Something?

- Sophisticated
  - Weak IV frames of fixed size

- Aircrack-ng Default
- Frame Size Filter

Diagram showing the progression from naive to sophisticated chaff insertion approaches.
Chaff Insertion Approaches

- Naive
  - Inject random frames
  - Weak IV frames of fixed size

- Robe
  - Chaff using single key

- Aircrack-ng Default
  - Frame Size Filter

- Sophisticated
Chaff Insertion Approaches

- Naive
  - Inject random frames
  - Weak IV frames of fixed size
- Sophisticated
  - Off goes your robe!! 😊
  - Chaff using single key
  - Aircrack-ng: Default, Frame Size Filter, Visual Inspection
Quick review - AirCrack-ng with a normal trace file

- 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-ng 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 in a "normal" trace of ~300,000 packets
Visual Inspection using Aircrack-ng

**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-ng 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-ng uses the user’s choice as the “guessed byte” for that byte of the key.
Guiding Aircrack-ng with Manual Inputs: Chaff with single key

Aircrack-ng 0.7

[00:00:02] Tested 0 keys (got 320881 IVs)

<table>
<thead>
<tr>
<th>KB</th>
<th>depth</th>
<th>byte (vote)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0/1</td>
<td>41 (1580)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>67 (69)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3F (13)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E4 (12)</td>
</tr>
<tr>
<td>1</td>
<td>0/1</td>
<td>1C (1158)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>37 (112)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>50 (55)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A2 (42)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0D (40)</td>
</tr>
</tbody>
</table>

Guessed keybyte for B=1 is 1C
Do you want to change it? Enter the new value: 37

Demo
Guiding Aircrack-ng with Manual Inputs: Analysis

- **Strengths**
  - Can crack the key in many cases
    - Single chaff key
    - Multiple chaff keys

- **Weaknesses**
  - May not work in presence of a chaffer with random keys
Chaff Insertion Approaches

- **Naive**
  - Inject random frames

- **Sophisticated**
  - I like that sword ;-)  
  - Aircrack-ng
    - Default
    - Weak IV frames of fixed size
    - Chaff using single key
    - Chaff using multiple keys
    - Visual Inspection

Chaff Insertion Approaches

Naive
- Inject random frames

Sophisticated
- Thanks!
- Aircrack-ng
- Weak IV frames of fixed size
- Chaff using single key
- Sequence Number analysis
- Chaff using multiple keys

Visual Inspection
- Aircrack-ng Default
- Frame Size Filter

Naive
- Sequence Number analysis
- Chaff using multiple keys

Sophisticated
- Thanks!
Example Illustrating Sequence Filter Implementation (using a subset of packets from the trace)

Sequence number is a part of the MAC header and is present in all Management and Data packets.

It is important to note the distinct pattern of sequence numbers for different sources.

This pattern can be used as a filter.

Most MAC spoofing detection algorithms already use Sequence#.

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.

Few lines of pseudo-code illustrating sequence filter!

For all packets of a given device in the trace:

```plaintext
    Prev_seq_num = First Sequence number seen for device;

    If (current_seq_num - prev_seq_num < threshold) {
        prev_seq_num = current_seq_num;
        consider packet for key cracking
    } else {
        Discard packet
    }
```
Chaff Insertion Approaches

- Naive
  - Inject random frames
  - Weak IV frames of fixed size

- Sophisticated
  - Shoes
  - Chaff using single key
  - Chaff using multiple keys
  - Chaff using random keys

Aircrack-ng
- Default
- Frame Size Filter
- Visual Inspection
- Sequence Number analysis
Chaff Insertion Approaches

- Naive
  - Inject random frames
  - Frame Size
  - Aircrack-ng Default

- Sophisticated
  - Poof!
  - Aircrack-ng
  - Weak IV frames of fixed size
  - Sequence Number analysis
  - Initialization Vector analysis
  - Visual Inspection
  - Chaff using single key
  - Chaff using multiple keys
  - Chaff using random keys
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 for different devices

- Legacy devices which WEP Chaffing desires to protect have Sequential IVs

- IV pattern can thus be used as a filter

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.

Few lines of pseudo-code illustrating IV filter!

For all packets of a given device in the trace:

```
Prev_wep_iv = First WEP IV seen for device;

If (current_wep_iv – prev_wep_iv < threshold)
{
    prev_wep_iv = current_wep_iv;
    consider packet for key cracking
}
else {
    Discard packet
}
```
Sequence Number and IV based Chaff filtering

(Demo)
Separating Chaff using Sequence No and IV: Analysis of this technique

- **Strengths**
  - 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
  - An independent chaff separator can be built

- **Weakness**
  - Reduced filtering efficiency when IVs are generated randomly. *(The good news is that most legacy WEP devices for which WEP Chaffing is recommended don’t seem to use random IVs)*
Chaff Insertion Approaches

- **Naive**
  - Inject random frames
  - Weak IV frames of fixed size
  - Chaff using single key
  - Aircrack-ng
  - Default

- **Sophisticated**
  - Chaff using multiple keys
  - Frames indistinguishable from AP Seq# & IV sequence
  - Chaff using random keys
  - Visual Inspection
  - Sequence Number analysis
  - Initialization Vector analysis

- **Pants 😊**
  - Chaff Insertion using random keys
  - Sequence Number analysis
  - Initialization Vector analysis
Chaff Insertion Approaches

Naive
- Inject random frames
- Weak IV frames of fixed size
- Aircrack-ng Default

Sophisticated
- Down they come!
- Aircrack-ng
- Frame Size Filter
- Aircrack-ng Visual Inspection
- Sequence Number analysis
- Initialization Vector analysis
- Active Frame replay
- Chaff using single key
- Chaff using multiple keys
- Chaff using random keys
- Frames indistinguishable from AP Seq# & IV sequence

Chaff using multiple keys

Chaff using random keys

Frames indistinguishable from AP Seq# & IV sequence
Chaff Separation 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)
  - The packet size is another parameter which can be used to identify the replayed packet
- 100% chaff separation is possible
Chaff Separating using Active Frame Replay
Successful Key Cracking

- The corrupted trace was filtered using Active replay technique and a new filtered trace was created.
- Aircrack-ng is able to crack the filtered trace after the application of packet replay filter
Separating Chaff using Active Frame Replay

**Strengths**
- Get a bonus packet for every packet we send
- 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

**Weakness**
- WEP cracker cannot be totally passive; Active frame injection required
- This has to be done “online” and at least one client needs to be associated with the network, whose source MAC we can forge and use for packet replays
Chaff Insertion Approaches

- **Naive**
  - Inject random frames
- **Sophisticated**
  - Shirt
  - Aircrack-ng
  - Default
    - Weak IV frames of fixed size
  - Aircrack-ng
    - Visual Inspection
    - Sequence Number analysis
    - Frame Size Filter
  - Chaff using multiple keys
  - Initialization Vector analysis
  - Chaff using single key
  - Sequence Number analysis
  - Frames indistinguishable from AP Seq# & IV sequence
  - Active Frame replay
  - Using Super secret magic

Using Super secret magic
Chaff Insertion Approaches

- **Naive**
  - Inject random frames
  - Weak IV frames of fixed size
- **Sophisticated**
  - Chaff using single key
  - Chaff using multiple keys
  - Chaff using random keys
  - Frames indistinguishable from AP Seq# & IV sequence
- **Using Super secret magic**
  - Active Frame replay
  - Replay & Fingerprinting Chaff

Aircrack-ng
- Default
- Frame Size Filter
- Visual Inspection
- Sequence Number analysis
- Initialization Vector analysis
- Visual Inspection
Replay & Fingerprinting Chaff

- Implementation of chaffing dictates that there will be an identifiable fingerprint in the Chaff.

- This is because the WIPS needs to identify its own Chaff packets from the real network data.

- Finding a usable fingerprint is a one-time job:
  - Check packet header fields for any abnormality.
  - Packet is fixed length?
  - Something appended, pre-pended to the packet?
  - ...many more possibilities.

- Once found, simply write a filter to weed out all the Chaff, then release the fingerprint to the community.
Chaff Insertion Approaches

Naive
- Inject random frames
  - Weak IV frames of fixed size

Sophisticated
- Anyone for a Full Monty??
  - Aircrack-ng
    - Default
    - Frame Size Filter
    - Aircrack-ng Visual Inspection
    - Sequence Number analysis
    - Initialization Vector analysis
    - Active Frame replay
    - Replay & Fingerprinting Chaff

Chaff using single key
Chaff using multiple keys
Chaff using random keys
Frames indistinguishable from AP Seq# & IV sequence

Using Super secret magic
## Overlapping Countermeasures

<table>
<thead>
<tr>
<th>Counter Types of Chaff</th>
<th>Aircrack-ng Default</th>
<th>Frame Size Filter</th>
<th>Aircrack-ng Visual Inspection</th>
<th>Sequence Number Analysis</th>
<th>IV Analysis</th>
<th>Active Frame Replay</th>
<th>Fingerprinting Chaff</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Frames</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Weak IV frames of fixed size</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chaff using single key</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chaff using multiple keys</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chaff using random keys</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Indistinguishable Seq# and IV in AP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Super Secret Magic potion</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Talk Outline

- Evolution of WEP Cracking
- What is WEP Chaffing? What are Chaff packets?
- WEP Chaffing example
- Techniques to counter different types of Chaff:
  - Random frames
  - Single key
  - Multiple keys
  - Random keys
  - ...
- Implementation problems with WEP Chaffing
- Final verdict on WEP Chaffing
- Q&A
Implementation problems with Chaffing

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

- Passive key cracking tools cannot be detected
  - Chaffing needs to be done 24x7x365

- Chaffing needs to be done on all channels on which WEP devices operate.
  - Imagine the load on the WIPS and the bandwidth wasted.

- Chaffing 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 Chaffing on each channel.
Implementation issues ...

- 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 Chaffing 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 be done continuously, as newer countermeasures are discovered.
Talk Outline

- Evolution of WEP Cracking
- What is WEP Chaffing? What are Chaff packets?
- WEP Chaffing example
- Techniques to counter different types of Chaff:
  - Random frames
  - Single key
  - Multiple keys
  - Random keys
  - …
- Implementation problems with WEP Chaffing
- Final verdict on WEP Chaffing
- Q&A
Even if Chaff frames were made indistinguishable by an oracle, WEP can still be cracked.

WEP has so many other vulnerabilities which can be easily exploited!
Final Verdict: WEP Chaffing was indeed too good to be true ...

- WEP Chaffing can at best slow down a Cracker by a couple of minutes but cannot stop him from breaking the key.

- Though our talk only includes Aircrack-ng, the chaff separation techniques we have outlined can be easily added to the functionality of any WEP cracking tool, without much additional work.

- Chaffing is another attempt of providing security through obscurity.

- Chaffing cannot provide a robust protection against WEP key cracking. **WEP was broken .... it is broken ....it will remain broken. PERIOD**.
Open Challenge!!

If you believe you have a WEP Chaffing implementation which works very differently and is unbeatable, then we request you send it to us and we will break it within 72 hours.

Demo Setup:

We will provide you an AP and clients – you can bring the WEP Chaffer to protect them.
Talk Outline

- Evolution of WEP Cracking
- What is WEP Chaffing? What are Chaff packets?
- WEP Chaffing example
- Techniques to counter different types of Chaff:
  - Random frames
  - Single key
  - Multiple keys
  - Random keys
  - ...
- Implementation problems with WEP Chaffing
- Final verdict on WEP Chaffing
- Q&A
Questions?
References (1)

- Vendor aims to ‘cloak’ WEP
- 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
- 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
- Airsnort: WEP Cracker
- Pcap2air: Packet replay tool
  [http://www.802.11mercenary.net/pcap2air/](http://www.802.11mercenary.net/pcap2air/)
- Chop-Chop: Packet decoder using WEP ICV flaw
- Intercepting Mobile Communications: The Insecurity of 802.11 – N. Borisov
- Your 802.11 Wireless Network has No Clothes – William Arbaugh
- Detecting Detectors: Layer 2 Wireless Intrusion Analysis – Joshua Wright
- Detecting WLAN MAC address spoofing – Joshua Wright
- WPA/WPA2 the replacement for WEP
- AirDefense Perpetuates Flawed Protocols – Joshua Wright