Practical attacks against WEP and WPA

Martin Beck, TU-Dresden, Germany
<hirte@aircrack-ng.org>

Erik Tews, TU-Darmstadt, Germany
<e_tews@cdc.informatik.tu-darmstadt.de>

November 8, 2008

In this paper, we describe two attacks on IEEE 802.11 based wireless LANs[2]. The first attack is an improved key recovery attack on WEP, which reduces the average number of packets an attacker has to intercept to recover the secret key. The second attack is (according to our knowledge) the first practical attack on WPA secured wireless networks, besides launching a dictionary attack when a weak pre shared key (PSK) is used. The attack works if the network is using TKIP to encrypt the traffic. An attacker, who has about 12-15 minutes access to the network is then able to decrypt an ARP request or response and send 7 packets with custom content to network.

1 Introduction

IEEE 802.11[2] is a standard family for wireless networks. Such networks can be found in home, office, and enterprise environments and are quite popular today. If sensitive informations are transmitted over a wireless network, privacy and integrity is a concern and must be taken care of.

The first version of the IEEE 802.11 standard supported a basic mechanism for protecting such networks named Wired Equivalent Privacy (WEP). WEP requires all clients and access points in the network to share up to four different secret symmetric keys, which is clearly not optimal for a larger installation where users change frequently. Most installations just use a single secret key named *root key*. WEP has some major design flaws and was completely broken in 2001[4, 13] by Fluhrer, Mantin, and Shamir. They showed that an attacker can recover the secret key of the network with an average consumer laptop in 1-2 hours. More advanced attacks were published in the last years making it possible to recover the secret key of the network in less than 60 seconds[15].

To fix the problems of WEP, a new standard named Wi-Fi Protected Access (WPA) was released in 2003, now part of the IEEE 802.11 specifications[2].

The structure of this paper is as follows: In Section 2, we give an introduction to the technical details of WEP and WPA and introduce the notation used in the rest of this paper. In Section 3, we give an overview over a selected number of attacks on WEP. In Section 4, we present a new attack on WEP, which reduces the number of packets an attacker needs to intercept to recover the secret root key compared to previous attacks. In Section 5, we present a new attack on WPA, which allows an attacker, who has about 12-15 minutes access to a WPA protected network to send 7 packets to the network with chosen payload and decrypt a single ARP[11] packet. According to our knowledge, this is the first practical attack on WPA protected networks, besides launching a dictionary attack against a weakly chosen pre shared key.

2 Notation

Numbers are always written in the decimal notation, for example 12 is the number twelve. The signs $+$ and $\cdot$ are addition and multiplication. $(\mathbb{Z}/n\mathbb{Z})^+$ is the additive group of the numbers 0 to $n-1$, where all additions are done $\bmod\ n$. When operations are done in $(\mathbb{Z}/n\mathbb{Z})^+$, we write $a+b$ as a short form of $a+b \bmod\ n$. For arrays, we use the $[\cdot]$ notation as used in many programming languages like *C* or *Java*. The first element in the array S is $\mathsf{S}[0]$. Permutations are written as arrays too. If S is a permutation, S^{-1} is the inverse permutation. For example, if $\mathsf{S}[i] = j$ holds, then $\mathsf{S}^{-1}[j] = i$ holds. When two arrays A and B are concatenated to a new array C, we write $\mathsf{C} = \mathsf{A}||\mathsf{B}$. $\mathbb{F}_2$ is the finite field with just the two elements 0 and 1. $\mathbb{F}_2[X]$ is the ring of polynomials over $\mathbb{F}_2$. When specifying estimations for a success probability or similar things, we use the $\approx$ sign to note that a formula or a value is only a good approximation, but not absolutely accurate.

Because WEP is mostly based on the RC4 stream cipher[12], we need to introduce a notation for analyzing the RC4 stream cipher. RC4 consists of two algorithms, the RC4-KSA, which transforms a key of length 1 to 256 bytes into an initial permutation S of the numbers 0 to 255. The internal state of RC4 consists of this permutation S and two numbers i and j used as pointers to elements of S. The RC4-PRGA generates a single byte of keystream from such a state and then updates the state.

To analyze the cipher, we will write S_k and j_k for the state of S and j, after exactly k rounds of the loop starting in line 5 in Listing 1 have been executed. To make the paper more readable, we write n for the constant value 256. Accordingly, we write S_{k+n} and j_{k+n} for the state of S and j, after the state was initialized by the algorithm in Listing 1 and exactly k bytes of output have been produced by the algorithm in Listing 2. When a key K is used for RC4 and a keystream X of arbitrary length is produced, we write $\mathsf{X} = \mathrm{RC4}(\mathsf{K})$. Please note that an attacker who knows the first k bytes of an RC4 key K also knows S_k and j_k.

In a WEP protected network, all stations usually share a single symmetric key Rk named root key. A single packet can easily be lost in an IEEE 802.11 network due to a transmission error, so WEP needs to encrypt all packets independently. Because RC4 does not support an initialization vector by itself, WEP generates a per packet key

Listing 1: RC4-KSA

```
1  for i ← 0 to 255 do
2    S[i] ← i
3  end
4  j ← 0
5  for i ← 0 to 255 do
6    j ← j+S[i]+K[i mod len(K)] mod 256
7    swap(S, i, j)
8  end
9  i ← 0
10 j ← 0
```

Listing 2: RC4-PRGA

```
1  i ← i + 1 mod 256
2  j ← j + S[i] mod 256
3  swap(S, i, j)
4  return S[ S[i] + S[j] mod 256 ]
```

for every packet. A three byte initialization vector IV is chosen and prepended to the root key Rk which results in the per packet key $\mathsf{K} = \mathsf{IV}||\mathsf{Rk}$. A keystream $\mathsf{X} = \mathrm{RC4}(\mathsf{K})$ is generated from K. To protect the integrity of the transmitted data, a 32 bit long CRC32 checksum named ICV is appended to the data. The resulting plaintext is then encrypted by XORing the plaintext (including the CRC32 checksum) with the generated keystream. The ciphertext together with the corresponding unencrypted initialization vector IV is then send over the air.

WEP originally only specified a 40 bit secret key Rk, but most vendors implemented an additional mode where Rk had a length of 104 bits. The length of the corresponding per packet keys K where 64 or 128 bit, and these variants were mostly marketed as 64 or 128 bit WEP. We restrict ourselves to the 104 bit variant, but our attacks can easily be adopted for networks with different key lengths with only minor modifications.

3 Previous attacks on WEP

A number of attacks on WEP have been published in the past.

3.1 The FMS attack

Fluhrer, Mantin and Shamir published[4, 13] the first key recovery attack on WEP in 2001. Their attack is based on the following ideas: An attacker who listens passively to the traffic of a WEP protected network can record a lot of encrypted packets including the initialization vectors used for these packets. Because the first bytes of the plaintext

of most packets are easily predictable, the attacker is able to recover the first bytes of the keystreams used to encrypt these packets. The initialization vector is transmitted unprotected with the packets, so the attacker initially also knows the first 3 bytes of the per packet key for all packets. All following bytes of the per packet key are the same for all packets, but are initially unknown to the attacker.

Lets assume that an attacker knows the first l bytes of an RC4 key used to generate a keystream X. He can therefore simulate the first l steps of the RC4-KSA and knows S_l and j_l. In the next step of the RC4-KSA, $j_{l+1} = j_l + \mathsf{K}[l] + \mathsf{S}_l[l]$ and $\mathsf{S}_l[l]$ is swapped with $\mathsf{S}_l[j_{l+1}]$. If the attacker could reveal $\mathsf{S}_{l+1}[l]$, he could easily recover $\mathsf{K}[l]$ by calculating the difference $\mathsf{S}_l^{-1}[\mathsf{S}_{l+1}[l]] - j_l - \mathsf{S}_l[l]$. Fluhrer, Mantin, and Shamir used the following trick to reveal this value:

Assume that the following conditions hold after the first l steps of the RC4-KSA:

1. $\mathsf{S}_l[1] < l$
2. $\mathsf{S}_l[1] + \mathsf{S}_l[\mathsf{S}_l[1]] = l$
3. $\mathsf{S}_l^{-1}[\mathsf{X}[0]] \neq 1$
4. $\mathsf{S}_l^{-1}[\mathsf{X}[0]] \neq \mathsf{S}_l[1]$

In the next step of the RC4-KSA, a value $k = \mathsf{S}_l[j_{l+1}]$ will be swapped to $\mathsf{S}_{l+1}[l]$. If j changes randomly for the rest of the RC4-KSA, the values $\mathsf{S}[1], \mathsf{S}[\mathsf{S}[1]]$, and $\mathsf{S}[l]$ won't be altered with a probability of approximately $\left(\frac{1}{e}\right)^3$ during the remaining RC4-KSA.

When the first byte of output is produced by the RC4-PRGA, j will take the value $\mathsf{S}_n[1]$ and $\mathsf{S}_n[1]$ and $\mathsf{S}_n[j]$ are swapped. After the swap, $\mathsf{S}[1] + \mathsf{S}[\mathsf{S}[1]] = l$ still holds and the first bye of output of the RC4-PRGA $\mathsf{X}[0]$ will be $\mathsf{S}[l]$. If conditions 3 or 4 wouldn't hold, this would indicate that $\mathsf{S}[1]$ or $\mathsf{S}[\mathsf{S}[1]]$ has been altered. In a nutshell, if these four conditions hold, the fuction:

$$\mathcal{F}_{fms}(\mathsf{K}[0], \ldots, \mathsf{K}[l-1], \mathsf{X}[0]) = \mathsf{S}_l^{-1}[\mathsf{X}[0]] - j_l - \mathsf{S}_l[l] \tag{1}$$

will take the value of $\mathsf{K}[l]$ with a probability of about $\left(\frac{1}{e}\right)^3 \approx 5\%$. We will refer to such a set of conditions together with such a function as a correlation for RC4. Flurer, Mantin, and Shamir referred to these conditions (or at least to the first two conditions) as the resolved condition.

A full key recovery attack on WEP can be built using this correlation. An attacker captures packets from a WEP protected network and recovers the first byte of keystream used to encrypt these packets by guessing the first byte of plaintext. There are also various active techniques to generate traffic on a WEP protected network even without the key, which allow the recovery of more than the first 1000 bytes of keystream per packet[1]. He selects the packets where the resolved condition holds and calculates $\mathcal{F}_{fms}$ for these packets. Each result of $\mathcal{F}_{fms}$ can be seen as a *vote* for the value of $\mathsf{Rk}[0]$. After enough packets have been captured, the attacker makes a decision for the value of $\mathsf{Rk}[0]$ based on the number of votes geberated by $\mathcal{F}_{fms}$. If the decision was correct, the attacker knows the first $l = 4$ bytes of all per packet keys and can continue with $\mathsf{Rk}[1]$. Please note that all packets need to be reevaluated wether the resolved condition holds, because this check depends on the value of $\mathsf{Rk}[0]$. After all

bytes of Rk have been determined, the attacker checks the resulting key for correctness using a number of trial decryptions. If the key is correct, the attacker has succeeded. If the resulting key is incorrect, the attacker looks for a decision for $\mathsf{Rk}[i]$, were an alternative value for $\mathsf{Rk}[i]$ was also very likely. The attacker corrects the decision in the decision tree at depth i and continues the attack with the alternate decision.

Although the 5% success probability of $\mathcal{F}_{fms}$ looks impressive, the attack needs 4,000,000 to 6,000,000 packets to succeed with a success probability of at least 50%, depending on the exact environment and implementation[14, 13]. The reason for this is that the resolved condition holds only for a small amount of randomly chosen initialization vectors.

3.2 The KoreK attack

In 2004, a person under the pseudonym KoreK posted[9, 3] an implementation of an advanced WEP cracking tool in an internet forum. KoreK used 16 additional correlations between the first l bytes of an RC4 key, the first two bytes of the generated keystream, and the next keybyte $\mathsf{K}[l]$. Most of these correlations have been found by KoreK him self, a few had been discussed[5] in public before. KoreK assigned names like *A_u15* or *A_s13* to these attacks, the original FMS attack is called *A_s5_1* here.

Nearly all correlations found by KoreK use the approach that the first or second byte of the keystream reveals the value of j_{l+1} under some conditions, if 2-4 values in S have a special constellation and are not changed during the remaining RC4-KSA after step $l+1$. An interesting exception is the *A_neg* correlation, which doesn't vote for a certain value of $\mathsf{K}[l]$. Instead a value can be excluded from the list of possible candidates for $\mathsf{K}[l]$, which can be seen as a negative vote for $\mathsf{K}[l]$.

The overall attack structure is the same decision tree based approach as for the FMS attack. The number of captured packets is reduced to about 700,000 for 50% success probability[14]. Again, the exact numbers depend on the exact environment and the implementation and parameters used for the attack. One important factor is if the initialization vectors are generated by a PRNG algorithm or if they are generated sequentially by a counter.

3.3 The PTW attack

In 2007, a new generation of WEP attacks was published[15, 14] by Tews, Weinmann, and Pyshkin. Their attack introduced two new concepts:

1. All previous correlations used required 2-4 values in S not to change during the remaining RC4-KSA. They also had a lot of preconditions which need to hold to use the correlation. Therefore, only a small number of packets could be used to vote for a certain keybyte.

   In 2005, Klein showed[7] that $l - \mathsf{X}[l-1]$ takes the value of $\mathsf{S}[l]$ with a probability of $\frac{2}{n}$. If $\mathsf{S}_l[l]$ remains unchanged until $\mathsf{X}[l-1]$ has been produced, the function:

$$\mathcal{F}_{Klein}(\mathsf{K}[0], \ldots, \mathsf{K}[l-1], \mathsf{X}[l-1]) = \mathsf{S}_l^{-1}[l - \mathsf{X}[l-1]] - (\mathsf{S}_l[l] + j_l) \qquad (2)$$

takes the value of $\mathsf{K}[l]$ with a probability of $\frac{2}{n}$. This result is also known as the Jenkins correlation[6]. $\mathsf{S}_l[l]$ remains unchanged with a probability of approximately $\frac{1}{e}$. If $\mathsf{S}_l[l]$ is modified before $\mathsf{X}[l-1]$ is produced, $\mathcal{F}_{Klein}$ takes a more or less random value. In total, this results in the following probability for $\mathcal{F}_{Klein}$ taking the value of $\mathsf{K}[l]$:

$$\left(\left(\frac{1}{e}\right)\frac{2}{n}\right) + \left(\left(1-\frac{1}{e}\right)\frac{1}{n}\right) \approx \frac{1.37}{n} \tag{3}$$

This correlation makes no requirements on the internal state of RC4 or the keystream, so that every packet can be used.

2. The second new concept is a change in the attack structure. Until now, every key recovery attack had a decision tree based structure and some kind of best first search strategy was used to determine the key byte per byte.

   Assume than an attacker knows the first l bytes of an RC4 key and manages to recover $k = \mathsf{S}_{l+2}[l+1]$ instead of $\mathsf{S}_{l+1}[l]$. Now $\mathsf{S}_{l+1}^{-1}[k] - \mathsf{S}_{l+1}[l+1] - \mathsf{S}_l[l] - j_l = \mathsf{K}[l]+\mathsf{K}[l+1]$ holds and an attacker would have recovered the value of $\mathsf{K}[l]+\mathsf{K}[l+1]$. With a very high probability $\mathsf{S}_{l+1}^{-1}[k] = \mathsf{S}_l^{-1}[k]$ and $\mathsf{S}_{l+1}[l+1] = \mathsf{S}_l[l+1]$ holds and $\mathsf{S}_l^{-1}[k] - \mathsf{S}_{l+1}[l+1] - \mathsf{S}_l[l] - j_l$ takes the value of $\mathsf{K}[l] + \mathsf{K}[l+1]$.

   We will call such correlations between the first l bytes of an RC4 key, the generated keystream, and the next i bytes of the key a *multibyte correlation* and write σ_i for the sum $\sum_{k=0}^{i} \mathsf{Rk}[k]$. Tews, Weinmann, and Pyshkin modified $\mathcal{F}_{Klein}$ to vote for the sum of the next m keybytes for every $m \in \{1, \ldots, 13\}$. This results in the following functions:

$$\begin{aligned} &\mathcal{F}_{ptw_m}(\mathsf{K}[0], \ldots, \mathsf{K}[l-1], \mathsf{X}[l+m-2]) \\ =&\mathsf{S}_l^{-1}[l+m-1-\mathsf{X}[l+m-2]] - \left(\sum_{a=l}^{l+m-1} \mathsf{S}_l[a]\right) \end{aligned} \tag{4}$$

   which only depend on the first 3 bytes of the per packet key (IV) and vote for σ_i instead of $\mathsf{Rk}[i]$.

   The PTW attack now works as follows: First an attacker captures packets and recovers their keystreams as for the FMS and KoreK attack. The attacker knows the first $l = 3$ bytes of all per packets keys. He now evalues $\mathcal{F}_{ptw_m}$ for every packet and every $m \in \{1, \ldots, 13\}$ and gets votes for $\sigma_0 \ldots \sigma_{12}$. After all packets have been processed, the resulting *root key* is calculated using $\mathsf{Rk}[0] = \sigma_0$ and $\mathsf{Rk}[i] = \sigma_i - \sigma_{i-1}$. If the key is correct, an alternative decision is made for one of the values σ_i and the key is updated using just 12 single subtractions without the need to reevaluate all packets.

The attack needs just about 35,000 to 40,000 packets[14, 15] for 50% success probability, which can be collected in less than 60 seconds on a fast network. Only a few seconds of CPU time is needed to execute the attack.

Some modifications of the PTW attack have been proposed[16, 10] which reduce the number of packets needed or allow the usage of the PTW attack in some special cases where the recovery of full key streams is difficult.

3.4 The Chopchop attack

The chopchop attack[8, 14] allows an attacker to interactively decrypt the last m bytes of plaintext of an encrypted packet by sending $m \cdot 128$ packets in average to the network. The attack does not reveal the *root key* and is not based on any special properties of the RC4 stream cipher.

We can summarize the chopchop attack as follows: Before encryption, a four byte CRC32 checksum named ICV is appended to the data of the packet. The packet with the trailing checksum P can be represented as an element of the polynomial ring $\mathbb{F}_2[X]$. If the checksum is correct, $P \bmod P_{CRC} = P_{ONE}$ holds, where P_{ONE} is a known polynomial and P_{CRC} is a known polynomial too, which is irreducible. We can write P as $QX^8 + R$. Here R is the last byte of P and Q are all remaining bytes. When the (encrypted) packet is truncated by one byte, Q will most probably have an incorrect checksum.

Assume that the attacker knows R. Adding $P_{ONE} + (X^8)^{-1}(P_{ONE} + R)$ to Q corrects the checksum again. If R was incorrect here, the resulting packet will have an incorrect checksum. This addition can also be done on the encrypted packet.

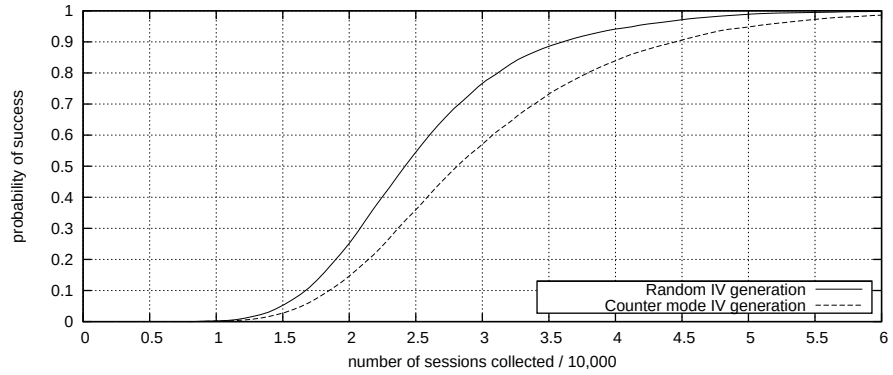
Most access points can be used to distinguish between encrypted packets with correct and incorrect checksum. For example if a client is not authenticated, and an access point receives a packet from this client, the access point will generate an error message. Packets with an incorrect checksum are silently discarded.

An attacker can use this to interactively decrypt packets. The attacker selects a captured packet for decryption. He truncates the packet by one byte, guesses R, corrects the checksum and sends the packet to the access point to find out if his guess for R was correct. If the guess for R was correct, the attacker now knows the last byte of plaintext and can continue with the second last byte. If the guess war incorrect, he makes another different guess for R. After at most 256 guesses and in average 128 guesses, he has guessed the correct value of R.

4 An improved attack on WEP

Unfortunately, after the release of the PTW attack, only little attention was drawn towards the old KoreK attack. Compared to the PTW attack, the KoreK attack has the advantage that it only needs the first two bytes of the keystreams of all captured packets. Usually, the recovery of the first two bytes of keystream is much easier than recovering the first 15 or 31 bytes. A pleasant exception is the work done by Vaudenay and Vuagnoux[16], who showed that the correlation used in the FMS attack can also be rewritten to vote for σ_i instead of $\mathsf{Rk}[i]$. This correlation is one of the 17 correlations used in the KoreK attack.

Figure 1: Success rate of the new WEP attack



To improve the performance of the PTW attack, we started rewriting all correlations used by KoreK to vote for σ_i instead of $\mathsf{Rk}[i]$. Surprisingly, we where able to successfully modify almost all correlations used by KoreK, with a few exceptions:

The correlations *A_4_s13*, *A_4_u5_1*, and *A_4_u5_2* in the original KoreK attack can only be used to vote for $\mathsf{Rk}[1]$ when $\mathsf{Rk}[0]$ is known. Using these correlations for $\mathsf{Rk}[2]$, $\mathsf{Rk}[3]$ or any other keybyte besides $\mathsf{Rk}[1]$ has not been implemented by KoreK. The modification of these correlations results in new correlations which vote only for σ_1, even with $\mathsf{Rk}[0]$ or σ_0 being unknown.

KoreK assigned labels with comments to some correlations. The correlation *A_u5_3* is the only correlation labeled with the comment *no good*. When we tried to modify *A_u5_3* to vote for σ_i, the resulting correlation did not produce any useful results.

The correlation *A_neg* was used by KoreK to exclude values from being $\mathsf{Rk}[i]$. The modification of this correlation results in a new correlation which can exclude values from being σ_i with a high probability. To implement this additional feature, a negative weight is assigned to this correlation.

Another interesting extension of the PTW attack was suggested by [16] and [10] independently. First they showed that it is possible to get four times more votes for σ_{13} than for all other values of σ_i. This makes it much easier for an attacker to decide on the value of σ_{12} than all other values of σ_i. Secondly, they found out, that the correlation used in the PTW attack can easily be modified to vote for the value of $\sigma_{12} + \sigma_i$, even when the value of σ_{12} is unknown at this moment. After the attacker has decided on the value of σ_{12}, he can get additional votes for each σ_i, by subtracting the value of σ_{12} from these votes. To use these additional correlations, an attacker needs the keystream bytes $\mathsf{X}[15]$ to $\mathsf{X}[30]$, which can sometimes be recovered too.

Using all these ideas, we modified an implementation of the PTW attack[1] resulting

[1] deleted obvious reference for blind reviewing, the final paper will contain a reference to this implementation

in a new WEP cracking tool, which clearly needs fewer packets than previous implementations of the PTW attack. We decided to use the same key ranking strategy as used for the original PTW attack. We limited the number of keys the implementation tests before failing to 2^{20}. The same limit has been used by previous publications about WEP attacks, so that it should be easier to compare our attack to previous attacks.

Figure 1 shows the success rate of our implementation. For a 50% success rate, the attack only needs about 24,200 packets, compared to 32,700 for the VX attack[16] and 35,000 to 40,000 for various implementations of the PTW attack [15, 14].

5 Breaking WPA

Our second contribution is an attack on WPA[2]. WPA standardizes two modes how payload can be protected during transmission, *Temporal Key Integrity Protocol* (TKIP) and *(AES-)CCMP*. For this paper, we will concentrate on TKIP. TKIP is a slightly modified version of WEP. TKIP implements a more sophisticated key mixing function for mixing a session key with an initialization vector for each packet. This prevents all currently known related key attacks because every byte of the per packet key depends on every byte of the session key and the initialization vector. Additionally, a 64 bit Message Integrity Check (MIC) named MICHAEL[2] is included in every packet to prevent attacks on the weak CRC32 integrity protection mechanism known from WEP. To prevent simple replay attacks, a sequence counter (TSC) is used which allows packets only to arrive in order at the receiver.

TKIP was designed so that legacy hardware only supporting WEP should be firmware or driver upgradeable to TKIP. Therefore, the RC4 stream cipher is still used and the ICV is still included in every packet.

We will now show that it is still possible to decrypt traffic in a chopchop like manner and to send packets with a custom content: Assume that the following conditions are met: The network being attacked is using TKIP for client to access point communication. The IPv4 protocol is used with an IP range where most bytes of the addresses are known to the attacker (for example 192.168.0.X). A long re-keying interval is used for TKIP, for example 3600 seconds. The network supports the IEEE 802.11e Quality of Service features[2] which allow 8 different channels (named TID - traffic identifier) for different data flows and a station is currently connected to the network.

These assumptions are quite realistic for most networks currently deployed in the wild. To attack such a network, an attacker first captures traffic, until he has found an encrypted ARP request[11] or response. Such packets can easily be detected because of the characteristic length. Additionally, the source and destination ethernet address is not protected by WEP and TKIP and requests are always sent to the broadcast address of the network. Most of the plaintext of this packet is known to the attacker, except the last byte of the source and destination IP addresses, the 8 byte MICHAEL MIC and the 4 byte ICV checksum. MIC and ICV form the last 12 bytes of the plaintext.

An attacker can now launch a modified chochop attack as against a WEP network to decrypt the unknown plaintext bytes. TKIP mainly contains two countermeasures against chopchop like attacks:

- If a packet with an incorrect ICV value is received by a client, a transmission error is assumed and the resulting packet is silently discarded. If the ICV value is correct, but the MIC verification fails, an attack is assumed and the access point is notified by sending a *MIC failure report frame*. If more than 2 MIC verification failures occur in less than 60 seconds, the communication is shut down, and all keys are renegotiated after a 60 second penalty period.

- When a packet has been received correctly, the TSC counter for the channel it was received on is updated. If a packet with a lower value than the current counter is received (the packet is received out of order), the packet is discarded.

Nevertheless, it is still possible to execute a chopchop attack. An attacker needs to execute the attack on a different QoS channel than the packet was originally received on. Usually, there will be a channel with no or low traffic where the TSC counter is still lower. If the guess for the last byte during the chopchop attack was incorrect, the packet is still dropped silently. If the guess was correct, a *MIC failure report frame* is sent by the client, but the TSC counter is not increased. The attacker needs to wait for at least 60 seconds after triggering a *MIC failure report frame* to prevent the client from engaging countermeasures. Within a little bit more than 12 minutes, the attacker can decrypt the last 12 bytes of plaintext (MIC and ICV). To determine the remaining unknown bytes (exact sender and receiver IP addresses), the attacker can guess the values and verify them against the decrypted ICV.

After the MIC and the plaintext of the packet is known, an attacker can simply reverse the MICHAEL algorithm and recover the MIC key used to protect packets being send from the access point to the client. The MICHAEL algorithm is not designed to be a one-way function and reversing the algorithm is as efficient as calculating the algorithm forward.

At this point, the attacker has recovered the MIC key and knows a keystream for access point to client communication. He is now able to send a custom packet to the client on every QoS channel, where the TSC counter is still lower than the value used for the captured packet. In most networks in the wild, all traffic is just transmitted on channel 0, so that the attacker is now able to send 7 custom packets to the client. After the attack has been successfully executed, an attacker can recover an additional keystream within 4-5 minutes, because he just needs to decrypt the 4 byte ICV using chopchop. The ip address bytes can be guessed, the MIC can then be calculated using the known MIC key and then be verified against the ICV.

To cause damage, the attacker could for example send messages triggering IDS systems which work on the IP layer. Alternatively, traffic could be rerouted using fake ARP responses. The attacker could try to establish a bidirectional channel to the client, if the client is connected to the internet using a firewall blocking incoming traffic, but allowing outgoing traffic. The responses of the client cannot be read over the air by the attacker, but could be routed back over the internet.

We created a proof of concept implementation of this attack[2] to verify that the attack actually works. We managed to attack hardware from various vendors, confirming that this attack is really applicable against real world networks.

[2]deleted obvious reference, the final paper will contain a reference to the attack implementation

Even if the network does not support the IEEE 802.11e QoS features, the attack still seems to be possible. Here, the attacker needs to prevent the client from receiving the data packet he chooses for the chopchop attack, and must disconnect the client from the access point for the time of the attack, so that the TSC counter is not increased by the packet or following packets. After the attacker has successfully executed the chopchop attack, he can send a single data packet to the client. However, we did not implement this attack mode.

If an attacker would manage to recover a keystream still valid for a QoS channel and the MIC key for both directions (our attack only recovery a keystream and the MIC key for access point to client communication), he would be able to use them to recover additional keystreams and could send a unlimited number of packets with custom plaintext.

5.1 Countermeasures

To prevent this attack, we suggest using a very short rekeying time, for example 120 seconds or less. In 120 seconds, the attacker can only decrypt parts of the ICV value at the end of a packet. Alternatively disabling the sending of *MIC failure report frame* frames on the clients would also prevent the attack. The best solution would be disabling TKIP and using a CCMP only network.

6 Conclusion

WEP is known to be insecure since 2001, however we think that key recovery attacks against WEP are still of interest. On the one hand, WEP is still used in the wild and on the other, some companies are selling hardware using modified versions of the WEP protocol, they claim to be secure. Secondly, the TKIP protocol used by WPA is not much different from WEP, so that attacks on WEP can affect the security of networks using TKIP, as seen in the paper.

Our attack on TKIP in Section 5 shows that even WPA with a strong password is not 100% secure and can be attacked in a real world scenario. Although this attack is not a complete key recovery attack, we suggest that vendors should implement countermeasures against this attack. Because the problem can be fixed in a high level part of the protocol, we think that updates can easily be developed and deployed with new drivers.

References

[1] Andrea Bittau, Mark Handley, and Joshua Lackey. The final nail in WEP's coffin. In *IEEE Symposium on Security and Privacy*, pages 386–400. IEEE Computer Society, 2006.

[2] IEEE-SA Standards Board. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.* Communications Magazine, IEEE, 2007.

[3] Rafik Chaabouni. Break wep faster with statistical analysis. Technical report, EPFL, LASEC, June 2006.

[4] Scott R. Fluhrer, Itsik Mantin, and Adi Shamir. Weaknesses in the key scheduling algorithm of RC4. In Serge Vaudenay and Amr M. Youssef, editors, *Selected Areas in Cryptography 2001*, volume 2259 of *Lecture Notes in Computer Science*, pages 1–24. Springer, 2001.

[5] David Hulton. Practical exploitation of RC4 weakness in WEP environments, 2002. presented at HiverCon 2002.

[6] Robert J. Jenkins. Isaac and rc4. http://burtleburtle.net/bob/rand/isaac.html, 1996.

[7] A. Klein. Attacks on the RC4 stream cipher. *Designs, Codes and Cryptography*, 48(3):269–286, 2008.

[8] KoreK. chopchop (experimental WEP attacks). http://www.netstumbler.org/showthread.php?t=12489, 2004.

[9] KoreK. Next generation of WEP attacks? http://www.netstumbler.org/showpost.php?p=93942&postcount=35, 2004.

[10] Yuko Ozasa, Yoshiaki Fujikawa, Toshihiro Ohigashi, Hidenori Kuwakado, and Masakatu Morii. A study on the Tews, Weinmann, Pyshkin attack against WEP. In *IEICE Tech. Rep.*, volume 107 of *ISEC2007-47*, pages 17–21, Hokkaido, July 2007. Thu, Jul 19, 2007 - Fri, Jul 20 : Future University-Hakodate (ISEC, SITE, IPSJ-CSEC).

[11] D. C. Plummer. RFC 826: Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware, November 1982.

[12] David Sterndark. Rc4 algorithm revealed. Usenet posting, Message-ID: <sternCvKL4B.Hyy@netcom.com>, Sep 1994.

[13] Adam Stubblefield, John Ioannidis, and Aviel D. Rubin. A key recovery attack on the 802.11b wired equivalent privacy protocol (WEP). *ACM Transactions on Information and System Security*, 7(2):319–332, May 2004.

[14] Erik Tews. Attacks on the wep protocol. Cryptology ePrint Archive, Report 2007/471, 2007. http://eprint.iacr.org/.

[15] Erik Tews, Ralf-Philipp Weinmann, and Andrei Pyshkin. Breaking 104 bit wep in less than 60 seconds. In Sehun Kim, Moti Yung, and Hyung-Woo Lee, editors, *WISA*, volume 4867 of *Lecture Notes in Computer Science*, pages 188–202. Springer, 2007.

[16] Serge Vaudenay and Martin Vuagnoux. Passive-only key recovery attacks on RC4. In *Selected Areas in Cryptography 2007*, Lecture Notes in Computer Science. Springer, 2007.