Teil II:
Ausarbeitung zum WEP-Protokoll

Lukas Jung, Marc Narres-Schulz, Oliver Sänger, Tobias Zeimetz

30. November 2016

Inhaltsverzeichnis

1 Einführung

Bei der vorliegenden Arbeit handelt es sich um ein Protokoll über eine Teilaufgabe im „Hackerpraktikum“. Die erste Aufgabe bestand darin, sich in WEP und damit auch in RC4 einzuarbeiten. Hauptbestandteil für das Verständnis von WEP und RC4 waren die Artikel „Attacks on the RC4 stream cipher“, „Breaking 104 bit WEP in less than 60 seconds“ und „Intercepting Mobile Communications: The Insecurity of 802.11“.
Anschließend sollte eine Test-Umgebung und der Angriff nach Klein implementiert werden. Diese Testumgebung war eine programmierte Simulation der RC4-Stromchiffre. Die Testumgebung wurde in Python programmiert und lieferte als Ergebnis paare aus Initialisierungsvektoren (IV) und Stromchiffren. Dass Ziel bei dem Angriff nach Klein war es, den Hauptschlüssel zu berechnen und somit die Verschlüsselung zu brechen.
Die letzte Aufgabe dieses Abschnitts im Praktikum bestand darin, den Hauptschlüssel eines auf WEP konfigurierten Routers zu berechnen. Wie in der ersten Aufgabe bereits implementiert, eigenen sich Paare von IV und Schlüsselstrom sehr gut. Um das Auftreten von ARP-Paketen zu erhöhen sollte eine Technik namens „re-injection“ verwendet werden. Dazu sollten die Tools „aireplay-ng“ und „airodump-ng“ verwendet werden.
Wie bereits erwähnt bestand die letzte Aufgabe darin, aus den ARP-Paketen Paare von IV und und Schlüsselstrom zu berechnen und dadurch den Angriff von Klein auszuführen. Der Angriff lieferte als Ergebnis den Hauptschlüssel und somit wurde die Verschlüsselung von WEP gebrochen.
Das Protokoll besteht aus drei Abschnitten. Der erste Abschnitt dreht sich um die Verschlüsselung und wie diese Funktion. Dort wird genauer erläutert wie die RC4-Stromchiffre funktioniert, wie das WEP-Protokoll arbeitet und was unter den einzelnen Begrifflichkeiten zu verstehen ist. Anschließend wird im Detail auf den ausgeführten Angriff eingegangen. Dieser Abschnitt unterteilt sich zum einen in den Angriff nach Klein und anschließend folgt eine Verbesserung des Angriffs. **Der letzte Abschnitt.... muss noch ausformuliert werden...**

2 Grundlagen der Verschlüsselung

Zuerst folgt eine Einführung in die Verschlüsselung die verwendet wird. Dazu unterteilt sich dieses Kapitel in zwei Abschnitte; wie ist die Funktionsweise der RC4-Stromchiffre und des WEP-Protokolls. Die hier verwendete Literatur waren die Artikel „Attacks on the RC4 stream cipher“, „Breaking 104 bit WEP in less than 60 seconds“ und „Intercepting Mobile Communications: The Insecurity of 802.11“. Zuerst folgt eine Erklärung zum Verfahren der RC4-Stromchiffre und im Anschluss folgt eine Erklärung zum WEP-Protokoll.

2.1 RC4-Stromchiffre

Der Algorithmus zu RC4 wurde 1987 von Ron Rivest entwickelt und besteht aus zwei Teilen. Der erste Teil ist das „key scheduling“ und der zweite Teil ist eine „pseudo random generation“. Der RC4-Algorithmus bekommt zwei Übergabeparameter. Der erste Parameter ist ein zufällig gewählter Initialisierungsvektor (IV). Beim IV handelt es sich um ein Feld mit Einträgen aus $\mathbb{Z}_n$ und $n \in \mathbb{N}$. Beim zweiten Parameter k handelt es sich um den „main key“. Die Konkatenation von IV und „main key“ bilden den session key. Der session key wird meist als ein ein Feld K implementiert. Da in WEP der IV immer vorne an die verschlüsselte Nachricht angefügt wird, hat der session key folgende Form:

$$\text{session key} = \text{IV} \,||\, \text{main key}$$

Das Symbol || stellt den Operator für die Konkatenation dar. Die länge von k und die Größe des IVs sind Abhängig vom gewählten Modus. Bei einer 64-Bit-Verschlüsselung hat der IV eine Größe von 24 Bit und der main key ist 40 Bit groß. Wählt man die 128-Bit-Verschlüsselung so ergibt für den main key eine Größe von 104 Bit. Die Größe des IVs bleib unverändert bei 24 Bit. Somit ergibt sich die Größe l_K des session keys wie folgt:

$$| \text{K} | = | \text{IV} | + | \text{k} |$$

Die maximale Größe von l_K beträgt n, liegt jedoch typischerweise zwischen 5 und 128 Bit.
In der ersten Phase des RC4-Algorithmus, der key scheduling phase, wird eine initiale Permutation mit Hilfe des session keys K erzeugt. Die erste Phase lässt sich wie folgt darstellen:

```
def keyScheduling(sessionKey=bytearray(), n=256):
    # initialization
    sBox = []
    for i in range(n):
        sBox.append(i)

    sBox = bytearray(sBox)
    i,j = 0,0

    # generate a random permutation
    for i in range(n):
        j = (j + sBox[i] + sessionKey[i % len(sessionKey)]) % n
        sBox[i], sBox[j] = sBox[j], sBox[i]

    return sBox
```

Bei der key scheduling phase wird eine Substitutionsbox (S-Box) verwendet. Diese wird zu beginn mit Werten aus 0 bis $n-1$ aufsteigend gefüllt. Die S-Box wird aus dem geheimen Schlüssel berechnet und später zur Berechnung der Stromchiffre verwendet. In jedem Schritt der For-Schleife werden zwei Einträge der S-Box vertauscht um eine zufällige Permutation zu erzeugen.
Nachdem die key scheduling phase eine zufällige S-Box generiert hat, beginnt die zweite Phase des RC4-Algorithmus. In dieser Phase wird eine Zufallsfolge, auch Stromchiffre genannt, erstellt. Der Algorithmus dazu gestalltet sich wie folgt:

```
def pseudoRandomGenerator(sBox=bytearray(),n=256):
    # initialization
    i,j = 0,0

    # generate pseudo random sequence
    while True:
        i = (i + 1) % n
        j = (j + sBox[i]) % n
        sBox[i], sBox[j] = sBox[j], sBox[i]
        k = (sBox[i] + sBox[j]) % n

        yield bytes([sBox[k]])
```

Hierbei handelt es sich um den Hauptteil des RC4-Algorithmus. Der pseudo random generator startet mit dem permutierten Array $sBox$, dass in der key scheduling phase generiert wurde. Anschließend wird ein Index j mit Hilfe der zufällig permutierten S-Box gewählt und zwei Einträge ($sBox[i]$ und $sBox[j]$) werden wie bereits in der key scheduling phase vertauscht. Der nächste Schritt besteht darin einen Index k zu ermitteln, welcher aus den addierten Werten aus der S-Box bestehen. Durch den anschließenden Modulo-Operator wird sichergestellt, dass es sich um Werte aus dem Zahlenraum $/mathbbZ_n$ handelt. Der pseudo random generator erstellt somit in jedem Schritt ein Byte, mit einem Wert zwischen 0 bis $n-1$, als Ausgabe.

2.2 WEP

Wired Equivalent Privacy (WEP) ist das ehemalige Standard-Verschlüsselungsprotokoll für WLAN. Das Protokoll wurde dazu verwendet um die Vertraulichkeit und Integrität einer Nachricht sicherzustellen. Die Verschlüsselung unter WEP gestaltet sich wie folgt:

$$C = (M||c(M)) \oplus RC4(v,k)$$

Zuerst wird über eine Nachricht M eine Prüfsumme $c(M)$ gebildet. Anschließend wird die Nachricht M mit $c(M)$ konkateniert in der Form $(M||c(M))$. Der nächste Schritt besteht darin den RC4-Algorithmus aufzurufen und so viele zufällige Bytes zu produzieren wie der Term $(M||c(M))$ besitzt. Bei den Variablen die der RC4-Funktion übergeben werden, handelt es sich zum einen um den Initialisierungsvektor v und den main key k.
Der letzte Schritt besteht darin die zufällige Bytefolge $RC4(v, k)$ und den Term $(M||c(M)$ mittels XOR zu verrechnen. Dadurch entsteht eine Vernamchiffre C die nur bei Kenntnis von v und k (und letztendlich $RC4(v, k)$) entschlüsselt werden kann.
Über das Netzwerk werden durch den Router Nachrichten verschickt, welche die Form $v||C$ besitzen. Der Initialisierungsvektor wird dabei vorne and die Vernamchiffre konkateniert. Da die Länge des Initialisierungsvektors v bekannt ist, kann der Empfänger der Nachrichten diesen „abschneiden"und durch Kenntnis des main keys k die Stromchiffre $RC4(v, k)$ erzeugen. Anschließend kann durch $RC4(v, k) \oplus C$ die Chiffre C entschlüsselt werden.
Durch die Prüfsumme $c(M)$ ist der Empfänger außerdem in der Lage, die Integrität der eigentlichen Nachricht M zu gewährleisten.

3 Der Angriff

In diesem Abschnitt werden die verschiedenen Angriffe die wir in den Aufgaben 2 bis 4 durchführen sollten genauer dargelegt. Der erste Abschnitt handelt darüber wie Kleins Angriff auf den RC4-Algorithmus unter einer „simulierten Umgebung" getestet wurde. Anschließend wird darauf eingegangen wie man das Auftreten von ARP-Paketen mit Hilfe einer Technik namens „re-injection" erhöht. Danach werden aus diesen Paketen paare von IV und Schlüsselstrom extrahiert und der Angriff von Klein wird auf diese Daten angewendet. Abschließend wird erläutert wie man den Angriff deutlich effizienter machen kann.

3.1 Kleins Angriff in der simulierten Umgebung

Wie bereits erwähnt, bestand ein Teil der Aufgaben darin, den Angriff von Klein auf selbst generierte Paare von Initialisierungsvektoren und Stromchiffren anzuwenden. Dazu wurde ein Package RC4ërstellt, indem die Methoden für die key scheduling phase und den pseudo random generator enthalten waren. Außerdem beinhaltet das Package eine Methode mit dem Namen $fixed\_rc4(key, cipher\_length = 4096, n = 256)$.

```
def fixed_rc4(key, cipher_length=4096, n=256):
    # Initial permutation
    s_box = key_scheduling(key, n=n)
    cipher = bytearray()
    # Extract cipher_length stream key bytes
    for k in pseudo_random_generator(s_box):
        cipher + = k
        if len(cipher) >= cipher_length:
            break
    return cipher
```

Die Methode ruft den key scheduling Algorithmus auf und erzeugt somit eine permutierte S-Box. Neben dem Zahlenraum $\mathbb{Z}_{\aleph}$ der mittels n übergeben wird, wird auch der main key k und die Länge der Chiffre (Standardmäßig 4096 Byte) übergeben. Dadurch werden in der For-Schleife genau so viele zufällige Bytes generiert wie gefordert. Wurde die maximale Anzahl an zufälligen Bytes generiert bricht der Algorithmus ab und gib die zufällige Bytefolge aus.