Teil II:
Ausarbeitung zum WEP-Protokoll

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

24. November 2016

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 (sBox) verwendet. Diese wird zu beginn mit Werten aus 0 bis $n-1$ aufsteigend gefüllt. Die sBox wird aus dem geheimen Schlüssel berechnet und später zur Berechnung der Stromchiffre verwendet. In jedem Schritt der For-Schleife werden zwei Einträge der sBox vertauscht.
Nachdem die key scheduling phase eine zufällige sBox 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 erstelt in jedem Schritt ein Byte, mit einem Wert zwischen 0 bis $n-1$, als Ausgabe.