

Wellenbrecher

DaNiel Ettle
FH-Regensburg
Informatikstudent

23.01.2002

1 Einstimmung

W-Lan ist inzwischen ein weitverbreitetes Medium und wird immer wieder in Nachrichten als die Technik gepriesen, die es ermöglicht, immer und ueberall online zu sein.

Viele Firmen, Hochschulen und Privatpersonen verwenden die Technologie, ohne sich Gedanken ueber die Sicherheit zu machen.

Wie die Geschichte zeigt, wird Sicherheit immer klein geschrieben, bis es irgendwo hackt(ed).

2 Grundlagen und Voraussetzungen

2.1 Verantwortung

Dieser Artikel soll zeigen, wie man im W-LAN Netze eindringt und wie man sich gegen solche Hacks schuetzen kann.

Jeder, der die Moeglichkeit besitzt, einzudringen, sollte wissen, was er tut und verantwortungsbewusst handeln.

Das Veraendern oder Loeschen von Daten sowie das Weitergeben von Daten an Dritte ist zwar in einigen Laendern – wie Argentinien – noch nicht strafbar, hier in Deutschland allerdings muss man mit hohen Strafen rechnen.

Vor allem sollte davon abgesehen werden, anderen Menschen Schaden zuzufuegen.

2.2 Hinweise

Am Rande sollte allerdings erwaeht werden, dass das Mitsniffen und Abhoeren von W-LAN Verbindungen sowie das „versehentliche Sichten von nicht oder nur unzuLaenglich gesicherten Daten“ nicht strafbar ist.

Auf was ich auch hinweisen will ist, dass das Ganze hier nicht aus meinen Wissen entstanden ist, sondern dass die Tools und das Know-How darueber schon seit langem existieren.

2.3 Grundlagen

Allgemeine Grundkenntnisse in UNIX, C++, TCP/IP und mathematisches Verstaendnis sollten vorhanden sein.

3 Mathe

3.1 Einfuehrung

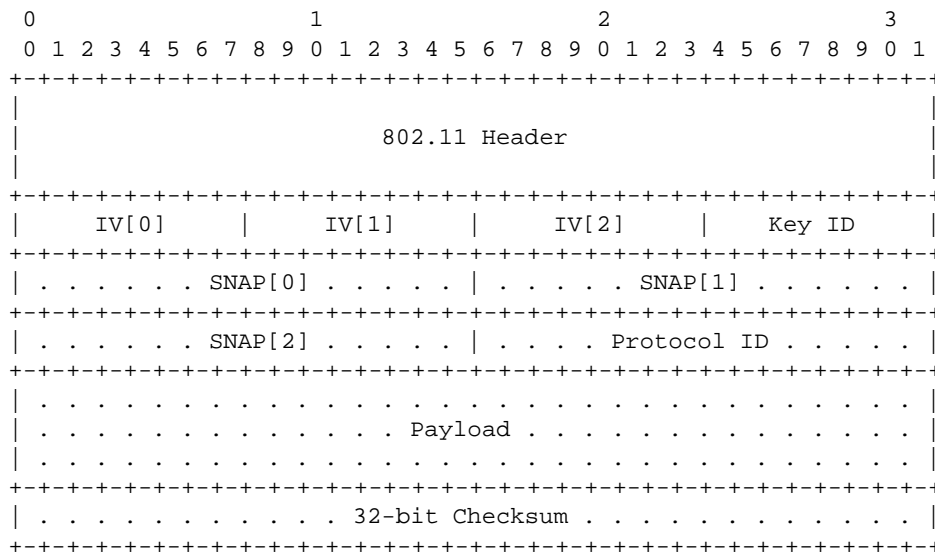
Hier ein kurzer Ueberblick ueber die 802.11b basierten WEP (Wireless Equivalent Protocol) Schwaechen und ein generelles „flame“ ueber RC4: Schliesslich reisen schon seit Jahren Cryptografen durchs Land und sagen, dass man um Himmels Willen nicht RC4 verwenden soll. Naja, IEEE hat's halt fuer W-Lan als Standard erklart (gut gemacht Jungs).

Die vorgestellten Angriffsmoeglichkeiten basieren auf zwei Konzepten, die in „Weaknesses in the Key Scheduling Algorithm of RC4“ (FMS) und „Using the Fluhrer, Mantin, and Shamir Attack to Break WEP“ (SIR) beschrieben sind. Sie erklaren eine Methode, die es erlaubt, ein optimiertes 'key-recovery' durchzufuehren.

3.2 Hintergrund

WEP basiert auf dem „RSA RC4 stream cipher“-Verfahren und benutzt einen 24-Bit Initialisierungsvektor (IV), welcher mit einem geheimen 40- oder 104-Bit langen („shared secret key“) verknuepft wird. Daraus resultiert dann der 64- oder 128-Bit Schluessel, der als „RC4 seed“ verwendet wird.

Die meisten Karten (eigentlich alle) benutzen fuer die Generierung des 24-Bit IV entweder einen Zaehler oder einen Pseudo-Zufallsgenerator (PRNG). Die Payload wird dann inklusive der 32-Bit CRC damit verschluesselt, wobei aber der IV plaintext, also unverschluesselt, ueber's Netz geht.



. - denotes encrypted portion of packet

Nachdem die Daten gesendet worden sind, verknuepft der Empfaenger den IV mit seinem geheimen Schluessel und entschluesst somit die Nutzdaten. Wenn die Pruefsumme stimmt, ist das Packet gueltig.

3.3 Aktuelle Cracking Methoden

In der Realitaet basieren die meisten Attacken auf Brute-Force Angriffen. Diese sind durch Methoden optimiert, wie zum Beispiel dem Verwenden einer Wortliste oder durch statistische Analysen des IV und der ersten paar Bytes des RC4-codes, die es ermoeglichen, einen „key scheduling algorithm“ (ksa) aufzustellen, mit dem man einzelne Bytes aus dem geheimen Schluessel extrahieren kann.

3.3.1 Brute Forcing

Brute-Force-Attacken haben sich als sehr effektive und effiziente Methode herausgestellt, ins WEP einzubrechen. Der Dank gebuehrt hier vor allem Tim Newsham. Die Methode, die er entwickelt hat, basiert darauf, die Nutzdaten zu entschluesseln, die von einem 802.11b gesniff worden sind, und mit dem CRC zu pruefen. Wenn die Pruefsumme stimmt, hat man den Schluessel. Wenn man den Schluessel hat, ist es dennoch wichtig, ihn mit anderen Packeten zu ueberpruefen, ob er auch/noch gueltig ist. In einigen Faellen passiert es naemlich, dass die Pruefsumme stimmt, aber der Schluessel dennoch, oder schon wieder, falsch ist.

Man braucht also mindestens 2 Packete fuer diese Art des Angriffes.

Tim Newsham's effektivere Cracking-Methode basiert auf der Schwaeche, dass der Schluesselgenerierungsalgorithmus auf einer Passwortliste beruht. Diese benutzen alle gaenigen 40-Bit-Karten und es ist ueberall die Gleiche. Wenn man sich diesen Vorteil zu Nutzen macht, kann man die 40-Bit-Verschluesselung auf 21-Bit runterbrechen, was mit der heutigen Rechenleistung in ca. 20-30 Sekunden moeglich ist. Die Passwortlisten-Attacke ist bei der 104-bit Verschluesselung aehnlich effektiv. Wenn man diese Optimierungen nicht nutzt, braucht man ca. 30 Tage um einen 40-Bit-Schlüssel zu berechnen.

Dieser Angriff fuehrt bei allen Angriffen zum gewuenschten Erfolg. Falls die Schluesellaenge allerdings auf 104-bit hochgesetzt wird, was momentan aber nur die teuren Karten koennen, so dauert das Entschluesseln wesentlich laenger. Aber Berechnungen ueber verteilte Systeme oder von einer grossen staatlichen Organisation, welche eigentlich gar nicht existiert, sollten das in aehnlich kurzer Zeit erledigen koennen.

Anzumerken ist hier noch, dass der Zeitaufwand nur so hoch ist, wenn man keine Optimierungsmethoden benutzt wie sie im folgenden beschrieben sind. Mit Optimierung bleiben auch hier die Berechnungszeiten unter einer Minute.

3.3.2 FMS Attack

Alle Open Source WEP Cracking Tools, die auf einer FMS Attacke basieren, benutzen einen sehr eingeschraenkten Bereich von Operationen, die in Kapitel 7.1 des Artikels „IV Precedes the Secret Key“ beschrieben wird. Herr Wagner hat dies ebenfalls bereits im Artikel „Wag95“ veroeffentlich, welcher nur das Uebereinstimmen von IVen ueberprueft:

$$(A + 3, N - 1, X)$$

Dies ist eine besondere Eigenschaft, die immer und ueberall funktioniert und nicht auf den vorausgegangenen Schluesseln basiert. Dies wird auch im FMS in Section A „Applying The Attack to WEP-like Cryptosystems“ vorgestellt und es wird vorgeschlagen, folgende Gleichung auf die S Permutationen gleich nach dem „key scheduling algorithmus“ (KSA) anzuwenden, um eine Schwaeche festzustellen:

$$X = SB + 3[1] < B + 3$$

$$X + SB + 3[X] = B + 3$$

Diese Gleichung enthuehlt wesentlich mehr IVen als die 256 Schluessel, die in allen Implementationen momentan verwendet werden. Dies wird ebenfalls in SIR, Section 4.1 „Choosing IVs“ beschrieben. Was allerdings fehlt, ist wie man einen Liste von gesniffen IVen verwendet, um eine moeglichst effektive Attacke in einer moeglichsten kurzen Zeit durchzufuehren.

Das Hauptproblem dieser Gleichung ist, dass man alle IVen, die man berechnet hat, mit jedem IV, den man mitgeloggt hat, byteweise vergleichen muss. Das verbraucht natuerlich sehr viele Ressourcen, wenn man das bei 2 Mio. Packeten durchfuehrt. Man hat also nicht nur einen extrem hohen Rechenaufwand, sondern man steht auch einer grossen Anzahl von Moeglichkeiten gegenueber.

3.4 Weitere Schwachstellen des KSA

Die Hauptschwachstelle im RC4, die noch nicht ausgenutzt wird ist, dass es Regelmäßigkeiten zwischen dem „pseudo random generation algorithm“ (PRGA) und dem erzeugten Bitstrom gibt. Diese Attacke ist vergleichbar mit der FMS Attacke, erfordert aber zusätzlichen Rechenaufwand, da man zusätzlich auch noch den PRGA emulieren muss, um festzustellen, ob eine IV ein Informationsbyte liefert. Die Anzahl der Bytes, die man mit dieser Methode attackieren kann, hängt direkt von der Anzahl der Bytes ab, die man bereits für den Schlüssel berechnet hat. Das macht eine genaue Voraussage des Zeitaufwands unmöglich. Um zu demonstrieren wie das geht, werde ich zuerst darstellen, wie man das erste Byte attackiert und das auf die anderen ausdehnen kann.

3.4.1 Angriff des ersten Bytes

Die „first byte attack“ beruht auf der Tatsache, dass es möglich ist, einen Teil des KSA zu simulieren, indem man die bekannten IVen benutzt. Die IVen teilt man dann durch die Elemente der Permutation von S, die sich nur um $1 - (e^{-X})$ unterscheiden, wobei X die Anzahl von S-Elementen ist, auf denen die Attacke basiert. Das folgende sollte die Attacke auf das erste byte des secret key (SK) besser veranschaulichen.

Definitions:

```
KSA(K)
Initialization:
  For i = 0 ... N - 1
    S[i] = i
  j = 0
Scrambling:
  For i = 0 ... N - 1
    j = j + S[i] + K[i mod l]
    Swap(S[i], S[j])
```

```
PRGA(K)
Initialization:
  i = 0
  j = 0
Generation Loop:
  i = i + 1
  j = j + S[i]
  Swap(S[i], S[j])
  Output z = S[S[i] + S[j]]
```

- Zu Demonstrationszwecken setzt man $N = 16$, normalerweise 256
- Ebenfalls sind alle Additionen und Subtraktionen noch durch modulo N zu nehmen und zu negativen Ergebnisse ist N zu addieren, so dass für die Ergebnisse immer gilt: $0 \leq x < N$.

Simulation:

```
let B = 0 - byte in K that we are attacking
let IV = B + 3, f, 7
let SK = 1, 2, 3, 4, 5
let K = IV . SK
let l = the amount of elements in K
assume that no S elements get swapped when i > B + 3

KSA - K = 3, f, 8, 1, 2, 3, 4, 5
Known Portion:
```

```

0 1 2 3 4 5 6 7 8 9 a b c d e f          j S[i] K
3      0                                i = 0, j = 0 + 0 + 3 = 3
0      1                                i = 1, j = 3 + 1 + f = 3
      d                                2      i = 2, j = 3 + 2 + 8 = d
Unknown Portion:
      f                                1      i = 3, j = d + 1 + 1 = f
- Note that S[B + 3] always contains information relating to SK[B], since
  SK[B] is used to calculate j.

PRGA - S = 3, 0, d, f, 4, 5, 6, 7, 8, 9, a, b, c, 2, e, 1
Unknown Portion:
0 1 2 3 4 5 6 7 8 9 a b c d e f          j S[i]          S[i] S[j]
3 0 d f 4 5 6 7 8 9 a b c 2 e 1      Unknown KSA Output
0 3                                i = 1, j = 0 + 0 = 0, z = S[0 + 3] = f

```

In diesem Fall stellt sich 'f' als erstes PRGA-Byte heraus, welches mit dem ersten Byte des snap headers xor'ed wird. Das erste Byte des snap-headers ist immer 0xaa, also teilen wir einfach das original 'f' durch simples xor'en mit dem ersten Byte der verschlüsselten Nutzdaten. Um das 'f' wieder in das erste Byte des SK zu stellen, welches wir fuer die Generierung benutzt haben, gehen wir einfach in Iterationsschritten durch den KSA, solange, bis wir auf den Punkt stossen, wo wir die Werte des j und des S[j] kennen, die wir benutzt haben, um durch 'f' zu teilen. Wenn wir erst einmal j und S[i] abgeleitet haben, koennen wir das ganze auch umgekehrt fuer SK[B] machen:

Definitions:

```

let S{-1}[Out] be the location of Out in the S permutation
let Out be z in the first iteration of the PRGA
assume values in Known Portion of KSA from where i = 2

SK[B] = S{-1}[Out] - j - S[i + 1]

```

Application:

```

SK[B] = S{-1}[f] - c - S[3] = f - d - 1 = 1

```

Mit dieser Methode bekommt man den geheimen Schluessel nach ca. $e^{-2} = 13\%$ der Zeit, manchmal sogar schon nach $e^{-3} = 5\%$, wenn man sich darauf verlaesst, dass 2 Elemente in der S-Permutationsfolge gleich bleiben. Wie man in der obigen KSA und PRGA Simulation sieht, verlassen wir uns nur darauf, dass die Elemente 0, 1 und 3 sich nicht fuer die Veranderung des Ausgabestroms verantwortlich zeigen, so dass die Wahrscheinlichkeit, dass unser Ausgabe-Byte 'f' ist, bei 5% liegt.

Je mehr verschiedene SK[B] Werte man sammelt, desto offensichtlicher wird der richtige SK[B]. Nachdem wir den wahrscheinlichsten Wert fuer das erste Byte berechnet haben, koennen wir diesen Algorithmus auch fuer das naechste Byte des geheimen Schluessels verwenden, und so weiter und so fort, solange, bis wir den kompletten geheimen Schluessel (SK) haben. In den meisten Implentierungen wird diese Methode zusaetzlich von Brute Force Attacken unterstuetzt, so dass die Stellen des berechneten Schluessels nicht unbedingt in der richtigen Reihenfolge stehen muessen.

3.5 Attackieren zusaetzlicher Ausgabe Bytes

Dieser Abschnitt demonstriert, wie man aus ein paar speziellen IVen Anhaltspunkte auf verschiedene Bytes im geheimen Schluessel bekommen kann und in manchen Faellen sogar groessere Chancen hat, als mit der „first byte“-Attacke. Zuerst wird gezeigt, was passiert wenn RC4 auf eine solche IV trifft und anschliessend eine Methode, wie man erkennt, dass diese Schwaeche aufgetreten ist.

3.5.1 RC4 und die Zweite-Byte-Schwaeche

In dieser Demonstration verwendet man einen schwachen IV, um das zweite Byte zu attackieren und zeigt damit, wie sicher/unsicher der Aufbau der S Permutation des Initial-Vektors ist, so dass geheime Schluessel-Informationen im zweiten Byte auftauchen. Diese Methode funktioniert auch auf beliebige andere Output Bytes und kann auf andere Bytes ausgedehnt werden, abhaengig davon welche Bytes des geheimen Schluessels man attackiert.

Was passiert also nun:

Simulation:

```

let B = 0           - byte in K that we are attacking
let IV = 4, c, c
let SK = 1, 2, 3, 4, 5
let K = IV . SK
assume that no S elements get swapped when i > B + 3

KSA - K = 4, c, c, 1, 2, 3, 4, 5
Known Portion:
  0 1 2 3 4 5 6 7 8 9 a b c d e f           j S[i] K
  4           0                           i = 0, j = 0 + 0 + 4 = 4
  1                                           i = 1, j = 4 + 1 + c = 1
  f                                           2 i = 2, j = 1 + 2 + c = f
Unknown Portion:
  3                                           i = 3, j = f + 3 + 1 = 3

PRGA - S = 4, 1, f, 3, 0, 5, 6, 7, 8, 9, a, b, c, d, e, 2
Unknown Portion:
  0 1 2 3 4 5 6 7 8 9 a b c d e f           j S[i]           S[i] S[j]
  4 1 f 3 0 5 6 7 8 9 a b c d e 2   Unknown KSA Output
  1                                           i = 1, j = 0 + 1 = 1, z = S[1 + 1] = f
  f 4                                           i = 2, j = 1 + f = 0, z = S[f + 4] = 3

```

Nachdem wir auch immer davon ausgehen koennen, dass das 2. Byte des snap headers 0xaa ist, koennen wir somit auch das zweite Byte des PRGA Ausgabestroms fuer den Originalschlüssel zurueckberechnen:

$$SK[B] = S^{-1}[3] - f - S[3] = 3 - f - 3 = 1$$

Wie man sieht, baut dieser spezielle IV den KSA und PRGA so auf, dass das zweite Ausgabe-Byte in fast jeder erdenklichen Situation Informationen ueber das erste Byte preis gibt. Zusaetzlich stellt es sich als richtig heraus, dass sich die Elemente 0, 1, 2 und 3 bei dem zweiten Byte nicht aendern, so dass man nur $e^{-4} = 2\%$ der Zeit braucht. Sozusagen als Zugabe koennen wir, falls das vorangegangene Ausgabe-Byte durch Folgende abgeleitet werden kann, sehen und ueberpruefen ob das aktuell ausgegebene Byte Einfluss auf das Verhalten des vorangegangenen Bytes hat und somit feststellen, dass es damit veraendert wurde.

Ein weiterer Vorteil ist, falls die gewuenschte Ausgabe und das erste Ausgabe-Byte auftaucht, dass es die benoetigten Berechnungen stark reduziert und wir nur noch $e^{-2} = 13\%$ berechnen muessen.

3.5.2 Auffinden von schwachen IVen im Ausgabe-Byte

Das Hauptproblem hier wiederum ist, durch Attackieren zusaetzlicher Ausgabe-Bytes festzustellen, ob ein IV einen Teil des geheimen Schluessels nun freigibt und an welcher Stelle des Ausgabe-Bytes dies auftritt, ebenso wie gross die Wahrscheinlichkeit dafuer ist, dass man dies in Betracht ziehen sollte.

Das Erkennen, ob ein IV diese Schwaeche aufweist und somit anfaellig fuer diese Art des Attacke ist, ist aehnlich zu der „first byte“-Attacke. Diese braucht allerdings einen

Schleifendurchlauf des PRGA bis zu $i = (A + 1)$, wobei A der Offset in dem PRGA Strom ist, von dem wir den Wert des Bytes wissen. Fuer jede Iteration der PRGA-Schleife, falls es ein Ergebnis fuer j oder $i \geq B + 3$ gibt – ansonsten muessen wir den IV verwerfen – verlassen wir uns auf Elemente, die sich am wahrscheinlichsten aendern. Dies kann man erreichen, indem man einfach den PRGA-Algorithmus aendert, so dass er ungefaehr wie folgt aussieht:

```

PRGA(K)
  Initialization:
    For i = 0 ... N - 1
      P[i] = 0
    P[B + 3] = 1
    i = 0
    j = 0
  Generation Loop:
    While i < A + 1
      i = i + 1
      j = j + S[i]
      If i or j >= B + 3 Then Fail
      Swap(S[i], S[j])
      Output z = S[S[i] + S[j]]
      P[i] = 1
      P[j] = 1
  Verification:
    If S[i] + S[j] = B + 3 Then Success
  Probability Analysis:
    j = 0
    For i = 0 ... N - 1
      If P[i] > 0 Then j = j + 1

```

Dieser Algorithmus funktioniert fast identisch wie die Gleichung, um herauszufinden ob ein IV anfaellig fuer die „first byte“-Attacke ist und kann somit erweitert werden, um IVen aufzuspüren, die einzelne Bytes des geheimen Schluessels an jedem Byte des PRGA-Stroms preiszugeben.

Danach kann man abwaegen, ob die Wahrscheinlichkeit hoch genug ist, sich dafuer zu entscheiden.

In Tests stellt es sich jedoch heraus, dass diese Methode voellig ungeeignet ist, hauptsachlich aufgrund der Tatsache, dass der Rechenaufwand, um IVen zu berechnen, die die gewuenschte Eigenschaft aufweisen, viel zu hoch ist. Da jeder IV auf jedes vorangegangene geheime Schluesselbyte ueberprueft werden muss, ist es effektiver, sich zuvor manuell eine Liste von anfaelligen IVen zu erstellen und somit den Aufwand waehrend des Schluesselrueckgewinnens zu reduzieren.

In vielen Faellen ist es praktikabler, viele IVen zu sammeln und nur die ersten paar Bytes fuer die Schluesselwiederherstellung zu benutzen. Vor allem in Faellen in denen man nur wenige Daten-Frames hat, kann dies den Aufwand der Berechnungen enorm reduzieren.

4 Implementierung

Dieser Abschnitt beschreibt die praktische Methode, einen Angriff auf das schwache erste Byte des IVs zu machen, ohne hinderliche Performance-Probleme. Es werden auch Optimierungsroutinen angesprochen, um Brute-Force-Attacken zu machen und ein paar Tricks, wo man „schummeln“ kann, um den Zeitaufwand des Berechnens stark zu reduzieren. Die

Optimierung sollte es ermöglichen, eine Schlüsselwiederherstellung mit ca. 500 000 bis 2 000 000 Paketen unter einer Minute Rechenzeit zu halten.

Ebenfalls wird in SIR eine Methode erwähnt, die es ermöglicht, mit einer nicht wesentlich unterscheidbaren Anzahl von Paketen eine Attacke zu fahren, die nicht voraussetzt, dass der WEP-Key aus ASCII-Zeichen besteht, und zudem unabhängig von dem verwendeten Schlüsselegenerator des Opfers ist.

4.1 Ausfiltern von schwachen IVen

Das Problem mit allen Angriffen auf WEP, die das schwache erste Byte benutzen, ist die Gleichung, die im FMS beschrieben wird und mit der man alle IVen fuer alle Schluesel, die man testet, ausprobieren muss. Es kommt oft genug vor, dass man ca. 2 000 000 gesammelte Pakete hat und Tausende von Schluesseln durchprobieren muss, bevor man den Richtigen findet. Diese Art und Weisse, den Schluesel zu kriegen, stellt sich als unpraktikabel heraus, da es sehr viel Speicher verbraucht und der Rechenaufwand immens hoch ist. Um diese Dilemmas zu umgehen, bedient man sich der Analyse von Mustern von schwachen IVen und wie sie sich auf die Schlueselbytes auswirken, auf die sie sich beziehen. Das ist das generelle Verfahren, das ich gefunden habe:

Definitions:

```
let x = iv[0]
let y = iv[1]
let z = iv[2]
let a = x + y
let b = (x + y) - z
```

Byte 0:

```
x = 3 and y = 255
a = 0 or 1 and b = 2
```

Byte 1:

```
x = 4 and y = 255
a = 0 or 2 and b = SK[0] + 5
```

Byte 2:

```
x = 5 and y = 255
a = 0 or 3 and b = SK[0] + SK[1] + 9
a = 1 and b = 1 or 6 + SK[0] or 5 + SK[0]
a = 2 and b = 6
```

Byte 3:

```
x = 6 and y = 255
a = 0 or 4 and b = SK[0] + SK[1] + SK[2] + 14
a = 1 and b = 0 or SK[0] + SK[1] + 10 or SK[0] + SK[1] + 9
a = 3 and b = 8
```

Byte 4:

```
x = 7 and y = 255
a = 0 or 5 and b = SK[0] + SK[1] + SK[2] + SK[3] + 20
a = 1 and b = 255 or SK[0] + SK[1] + SK[2] + 15 or
          SK[0] + SK[1] + SK[2] + 14
a = 2 and b = SK[0] + SK[1] + 11 or SK[0] + SK[1] + 9
a = 3 and b = SK[0] + 11
a = 4 and b = 10
```

Dieses Muster kann einfach auf eine Gleichung erweitert werden, die einen Bereich von unabhängigen geheimen Schlüsseln beinhaltet, die man besitzt. Als Ergebnis bekommt man Verteilungsmuster ähnlich dem unteren:

	Secret Key Byte												
	0	1	2	3	4	5	6	7	8	9	a	b	c
			+		+		+		+		+		+
0	8	16	16	16	16	16	16	16	16	16	16	16	16
1	8		16	16	16	16	16	16	16	16	16	16	16
2		16	8		16	16	16	16	16	16	16	16	16
a 3			16	8	16		16	16	16	16	16	16	16
4				16	8	16	16		16	16	16	16	16
v 5					16	8	16	16	16		16	16	16
a 6						16	8	16	16	16	16		16
l 7							16	8	16	16	16	16	16
u 8								16	8	16	16	16	16
e 9									16	8	16	16	16
s a										16	8	16	16
b											16	8	16
c												16	8
d													16

8 - 8-bit set of weak ivs

16 - 16-bit set of weak ivs

+ - 2 additional x and y dependent 8-bit weak ivs

Mit diesem Ergebnis koennen wir grob festlegen, wie viele der schwachen IVen fuer jedes einzelne Schluesselbyte existieren. Somit kann auch folgende Gleichung angenommen werden:

es sei $\text{MAX}(x, y)$ gleich $x > y ? x : y$,

$$\begin{aligned} & ((B \bmod 2 ? \text{MAX}(B - 2, 0) + 2 : B + 1) \times (2^{16})) + \\ & (((B \bmod 2 ? 0 : 2) + (B > 1 ? 1 : 0) + 1) \times (2^8)) \end{aligned}$$

Unser eigentliches Ziel ist es ja, einen Algorithmus zu finden, der es uns erlaubt, schwache IVen herauszufiltern, basierend auf dem geheimen Schluesselbyte, das sie attackieren koennen. So koennen wir unsere 2 000 000 Elemente umfassende Tabelle auf eine wesentlich kleinere hier runterbrechen, die somit auch einfacher zu durchsuchen ist. Diese berechnet man durch Anwenden eines einfachen Algorithmus, aehnlich dem unteren:

es sei l = die Anzahl der Elemente im SK

$i = 0$

For $B = 0 \dots l - 1$

If $((0 \leq a \text{ and } a < B) \text{ or}$

$(a = B \text{ and } b = (B + 1) * 2)) \text{ and}$

$(B \% 2 ? a \neq (B + 1) / 2 : 1)) \text{ or}$

$(a = B + 1 \text{ and } (B = 0 ? b = (B + 1) * 2 : 1)) \text{ or}$

$(x = B + 3 \text{ and } y = N - 1) \text{ or}$

$(B \neq 0 \text{ and } !(B \% 2) ? (x = 1 \text{ and } y = (B / 2) + 1) \text{ or}$

$(x = (B / 2) + 2 \text{ and } y = (N - 1) - x) : 0)$

Then ReportWeakIV

Die Ausgabe ist dann mit folgender vergleichbar:

Byte	# of IVs	Probability
0	768	0.00004578
1	131328	0.00782776
2	197376	0.01176453
3	197120	0.01174927
4	328703	0.01959223
5	328192	0.01956177
6	459520	0.02738953
7	459264	0.02737427
8	590592	0.03520203
9	590336	0.03518677
a	721664	0.04301453
b	721408	0.04299927
c	852736	0.05082703

Diese sollte sich nur leicht von der vorherigen Bestimmung der schwachen IVen unterscheiden, die vorher berechnet wurden, da sich einige IVen in dem Muster ueberlappen. Sortiert man diese IVen in eine Tabelle, kann man sehr leicht die Anzahl der IVen, die man fuer den Durchlauf eines Berechnungsdurchgangs benoetigt, auf ein Maximum von 852.736 reduzieren, oder nur 101.654, wenn man 2 000 000 Pakete mitgesniff hat. Das ganze reduziert den Aufwand, um einen Schluessel zu berechnen, auf $\frac{1}{20}$.

4.2 Schummeln

Wenn man nun dummerweise eine mitgesniffte Datei hat, die nicht genuegend statistisch verwertbare Ergebnisse liefert, um den geheimen Schluessel zu berechnen, benutzt man gewoehnlicherweise einen Brute-Force-Angriff basierend auf dem wahrscheinlichsten Byte des Schluessels. Bis zum jetzigen Zeitpunkt der Entwicklung vieler Programme wird mit einer festen Zahl „gemogelt“ bzw. die Breite des Suchfeldes angegeben, in dem nach einem Schluessel gesucht werden soll. Wie auch immer, mit $> 2\,000\,000$ Paketen und einer grossen Anzahl von schwachen IVen fuer jedes Byte steigt die Wahrscheinlichkeit, dass der korrekte Schluessel gefunden wird, mit der Anzahl der Durchlaeufer, die man fuer jedes Byte macht. Einen Schaetzwert fuer diese Wahrscheinlichkeit sieht man hier:

Byte	# of IVs	Probability
0	768	0.00004578
1	768	0.00004578
2	2304	0.00013732
3	1792	0.00010681
4	3072	0.00018311
5	2560	0.00015259
6	4096	0.00024414
7	3584	0.00021362
8	5120	0.00030518
9	4608	0.00027466
a	6144	0.00036621
b	5632	0.00033569
c	6656	0.00039673

Wenn man nun eine Brute-Force-Angriffe basierend auf 2 000 000 packeten macht, erhaelt man ausreichend richtige IVen, wie man sieht:

Byte	# of IVs	# of Correct Keys
0	92	5
1	92	5
2	275	14
3	214	11
4	366	18
5	305	15
6	488	24
7	427	21
8	610	30
9	549	27
a	732	36
b	671	33
c	793	39

Es ist hoechstwahrscheinlich, wenn man das zweite Byte erreicht hat, dass der Schlues- sel, der am wahrscheinlichsten aussieht, der Richtige ist. Das bedeutet, dass man so gut wie nie schummeln muss, oder zumindestens es sehr stark reduzieren kann, je mehr Bytes des richtigen Schluessels man bereits hat. Das reduziert natuerlich wiederum die Anzahl der benoetigten Brute-Force-Angriffe, was zu dem Ergebniss fuehrt, dass man eigentlich nur fuer die erste paar Bytes schummeln muss. In der Praxis laesst sich feststellen, dass man aufgrund der Eigenschaft der schwachen IVen weitaus weniger als 2 000 000 Packete braucht, um einen Schluesel zu berechnen. In einigen wenigen Fallen gelang es mir so- gar, ohne den Durchlauf der statistischen Auswertung der ersten paar Bytes den geheimen Schluesel innerhalb einer angemessenen Zeit zu berechnen.

5 praktische Anwendung

Hier stelle ich mal vor, welche Tools es gibt und wie man sie benutzt. Ebenfalls, warum es so einfach ist, einzubrechen und warum man ueberhaupt einbrechen sollte.

5.1 Betriebssystem und Hardware

Das Betriebssystem der Wahl sollte ein !=Mirc*soft Produkt sein, da man sonst zum einen damit beschaeftigt ist sein System abzusichern und zum anderen gibt es die vorgesehene Software fast ausschliesslich nur unter UNIX.

Ja, Free/Net-BSD oder Linux eignen sich am besten dafuer, wobei ich aber nur Tools unter FreeBSD und Linux ausprobiert habe.

Bei dem Kauf der Karte sollte man darauf achten, dass diese einen Monitor-Mode hat, Lucent, prismII und Cisco unterstuetzen das momentan. Andere Karten moegen zwar bil- liger/besser sein, sind fuers sniffen aber nicht zu gebrauchen.

’Monitor mode is a non-standard feature of the PRISM MAC firmware, intended for use only in testing and debugging environments. When a station is operating in monitor mode, it receives and passes to the host all frames wherein the PLCP header and beginning of the frame are successfully received, regardless of FCS errors or other protocol defects. Warning! Monitor mode is incompatible with operation as a standard-compliant STA. In particular, it poses significant security risks in a production network. Intersil may remove Monitor mode from future releases of the firmware without advance notice. It does not participate in the 802.11 net-work, it only listens. Use it only at your own risk.’

5.2 Warum hacken?

- Weil es einfach ist und Spass macht.

- Man lernt viel ueber die Technik und ueber's System.
- Bei Firmen, die immer meinen, das sie cool sind, Sicherheitsbewusstsein erhoehen.
- Anonymes Internet mal nutzen zu koennen.

5.3 Wie war das gleich wieder?

- Das Protokoll ist zu einfach aufgebaut.
- Der AccessPoint teilt sich mit dem Client einen gemeinsamen Schluessel (synchrones Verschluesseln).
- Die Pruefsumme baut auf den Inhalt der Nutzdaten auf (auch schlecht).
- Encryption und Decryption benutzen den gleichen Algorithmus.
- Man verlaesst sich auf 'Random * irgendwas = Random'.
- PRNG- und IV-Algorithmen sind schlecht.
- ...

Anschliessend ein paar Erklaerungen:

5.3.1 IV Kollision

Die Attacke basiert darauf, dass sich die IVs irgendwann man gleichen. Man snifft einfach mit und falls ein packet zu dem IV passt den man selber hat ist man sofort im selben Kanal mit dabei. Zudem ist es nicht einmal im Protokoll festgelegt, dass sich die IV fuer jedes Packet aendern muss. Alle mir bekannten Hersteller berechnen den IV zudem nicht mit einem Algorithmus, sondern stellen den Wert anfangs (Rechner einschalten, oder PCMCIA-Device reseten/initialieren) auf den Wert 0 und erhoehen diesen dann einfach fuer jedes Packet. Glaubt man nicht, ist aber so. Der IV ist ja 24bit lang. Das heist es gibt 2^{24} Moeglichkeiten. Das hoert sich erst mal viel an, aber wenn nun viel W-Lan Traffic vorhanden ist (irgendwelche Messen, Industriegebiet), kriegt man meistens in weniger als einer Stunde einen gueltigen IVs.

5.3.2 lineare Pruefsumme

Fuer die Integrietaetspruefung wird ein CRC32 benutzt. Das mag ja fuer zufaellig auftretende Fehler im Uebertragungsbereich dort ganz gut funktionieren, aber fuer einen Verschluesselung trifft das nicht ganz zu. Da CRC linear aufgebaut ist, kann man zum Beispiel folgendes damit machen:

- $\text{CRC}(X \oplus Y) = \text{CRC}(X) \oplus \text{CRC}(Y)$
- $\text{RC4}(k, X \oplus Y) = \text{RC4}(k, X) \oplus Y$
- $\text{RC4}(k, \text{CRC}(X \oplus Y)) = \text{RC4}(k, \text{CRC}(X)) \oplus \text{CRC}(Y)$

Was nichts anderes heisst, als dass man ein paar Bits in den Packeten einfach tauschen kann. Klingt jetzt nicht so spannend, aber das versetzt einen in die Lage, Daten in den Packeten zu veraendern, ohne dass es der Integrietaetscheck merkt. Beispielsweise: 1000,23 EUR == 2300,01 EUR. Zudem kann man das im aktiven Datenverkehr machen, was soviel heisst, dass der Nutzer da gar nichts mitbekommt. Man umgeht damit auch einfach irgendwelche Sicherheitskontrollen wie laestige Passwortabfragen oder Firewalls.

5.3.3 Umleitung

Vorausgesetzt, man weiss die IP des Clients. Was nicht weiter schwierig sein duerfte, da es sich meistens um interne Netze handelt, die dort gaengigen IPs sind ja nicht allzuviele. Dann dreht man einfach ein paar Bits um, so dass die IP Checksum nicht mehr stimmt und der Access-Point denkt dann, dass er ja ganz schlau ist und gemerkt hat, dass ein Fehler in der Uebertragung der IP-Adresse aufgetreten ist. Dann freut er sich, dass er das ja so toll gemerkt hat und schickt ab sofort alle Anfragen an unsere IP. Wir freuen uns auch und setzen erst einmal den Port auf 80 was so ziemlich jede Firewall ja unproblematisch durchlaesst. Somit sitzt man zwischen den AP und dem Client und hoert alles mit. Aehnlich einer MITM-Attack.

5.3.4 Shared Key Angriff

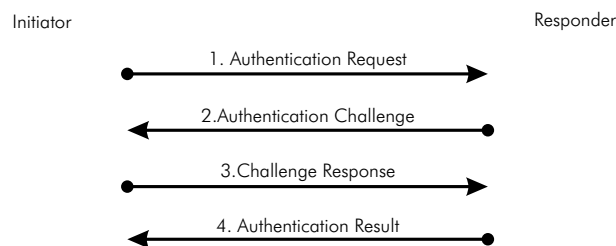


Abbildung 1: Shared Key Authentication

- Request
- Challengentext
- WEP(key, Challengentext)
- ACK/NACK

1. Man snifft die Msg 2 und 3 mit.
2. Damit weiss man den Challengentext P und den Ciphertext C
3. Damit hat man $RC4(K, IV) = C \oplus P$
4. Dann authentifiziert man sich nun selber ganz einfach so.
5. Den neuen Challengentext P' bekommt man ja und baut die Antwort einfach und mit $P' \oplus RC4(K, IV)$ zusammen.

5.3.5 PlainText Angriff

Das Ganze ist im Kapitel Mathe schon genau beschrieben und soll hier nur noch kurz verdeutlicht werden.

Das Problem dabei ist, dass man ein Stuecken von dem, was uebertragen wurde, im Klartext braucht, was aber im Grunde genommen nie ein Problem darstellt: Man versucht sich einfach mal mit einem DHCP-Request mit der Source-Adresse 0.0.0.0 und einen Request auf 255.255.255.255, da meldet sich dann schon einer. Damit hat man schon mal 24 Byte vom pseudo-random-stream (PRNG) mit $n=24$. Wenn wir soweit sind, kann's losgehen.

1. Nun erzeugen wir ein Datagramm mit der Laenge n-3, indem wir zum Beispiel einen ARP-Request machen.
2. CRC berechnen, aber nur mit den 3 Bytes
3. Das Ganze mit n Bytes des PRNG XOR'en
4. Dann ein letztes Byte anhaengen n+1

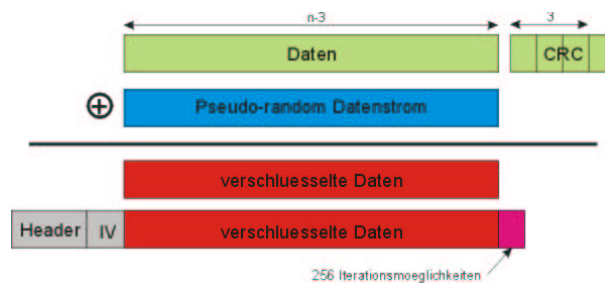


Abbildung 2: Stelle des zu iterierenden Bytes

5. Nun senden wir das Packet mal und warten ab.
6. Passiert nix, dann das Naechste schicken.
7. Wenn wir eine Antwort bekommen, dann wissen wir folgendes: Das n+1. Byte war korrekt, da der CRC stimmte, somit haben wir einen passenden Plain- und Ciphertext gefunden, welcher uns einen PRNG-Strom der Laenge n+1 gibt.



Abbildung 3: Austauschen der Bytes bei erfolgreichem CRC

5.4 Netze finden

Wo man am besten offene W-Lan Netze findet, habe ich hier mal aufgelistet:

- Messen sind am besten dafuer geeignet. Da dort sowieso immer alles drunter und drueber geht und alles moeglichst schnell zum Laufen gebracht werden muss, gibt es hier keine gesicherten Netze, oder wenn, dann nur selten. Kann aber sein, dass sich das in Zukunft aendert.
- Industriegebiete sind auch gut, da sich die meisten Firmen dort in Sicherheit waegen, weil sie ja eine Firewall haben und das W-Lan schon an der Aussenmauer aufhalten wird. Selbst wenn sich mit den Access-Points nichts anfangen laesst, weil diese

abgesichert sind, kann man mit dem einem oder anderen Laptop des Aussendienstmitarbeiters, die ja inzwischen fast alle W-Lan integriert haben, auch schon mal ein bisschen Spass haben.

- UNIs und Fachhochschulen. *ups*
- Wohngebieten mit wohlhabenden Leuten.
- Flughäfen egal in welchem Land. (Manche werben sogar dafür)
- Webseiten (free2air.org, wavehan.de, bawug.org, ...)
- gar nicht.



Abbildung 4: Karte der verfügbaren offenen W-LAN in der Bay Area

Manchmal muss man nicht einmal suchen. Mir haben ein paar Kids auf der CeBit erzählt, dass sie gar nichts dafür koennen, dass sie jetzt eine IP Adresse von einem W-Lan bekommen haben. Die Sender waren so stark, dass sie gar nichts dagegen machen koennen — und ich haette es ihnen dann auch fast noch geglaubt.

6 Tools

Ich stelle hier nun mal kurz ein paar Tools vor, gehe aber nicht genau drauf ein. Das wuerde den Rahmen sprengen. Ausserdem lernt man das durch Ausprobieren ohnehin schneller als durch Lesen.

Zudem kann ich nur Hinweise geben, auf was man bei der Installation achten muss. Die Distributionen sind dafür zu unterschiedlich, um eine Step-by-Step Anleitung geben zu koennen. Man muss sich schon auf das Nachbessern von ein paar config-files und einige Compilerlaufe gefasst machen.

*BSD hat hier mit ihrem ports-system wieder mal die Nase vorne, jedoch laufen nicht alle Tools auch auf BSD. Die Wichtigen und Gaengigen jedoch unterstuetzen beide Plattformen.

Sehr wichtig ist, dass man vorher Patches einspielt, um die Karten in den Monitor-mode zu setzen! Es gibt bereits ausreichend Anleitungen fuer diese Prozedur, so dass an dieser Stelle darauf verzichtet wird.

6.1 airsnort (der Klassiker)

6.1.1 Installation

Falls man keine gnome, bzw. gnome-libs auf seinem Rechner hat, wird's anstrengend. Falls ja, laeuft die Installation ohne grosse Probleme ab und man erreicht schnell sein Ziel.

6.1.2 Bedienung

Die Bedienung ist sehr einfach und man kommt schnell damit zurecht. In frueheren Versionen gab es ab und zu core dumps, wenn man laenger gesniff hat, was sehr aergerlich war. Seit Version 0.2 scheint das aber geloest zu sein und es laeuft anstandslos. Das Programm ist eines der ersten gewesen, die es ermoeeglichten, eine Key-Recovery durchzufuehren. Jeder der mal sniffen will, sollte sich das Tool mal installieren. Es lohnt sich.

6.2 Wellenreiter

6.2.1 Installation

Ich gehe hier von aktuellen Linux-Distributionen aus (SuSE 8.0, Redhat 7.2). Bei diesen muss man selber noch manuell folgendes selber compilieren. Es ist nicht schwer, dauert aber ein Weilchen, bis man soweit ist.

Zuerst einmal auf glib 2.x aufruesten, dann schauen, dass man gtk+ 2.x hat, wenn nicht, muss man sich das holen und zusaetzlich noch atk+ und pango. Bei pango unbedingt noch die freetype lib vorher installieren, sonst geht nix. Das `configure` laeuft normalerweise ohne zu meckern durch. Anschliessend geht pango und atk+ anstandslos. gtk+ kann noch ein wenig mosern, dass was fehlt, sollte aber alles bei einer „Grundinstallation“ schon vorhanden sein.

Fuer perl braucht man noch ein paar Modules, die es auf CPAN gibt. Net:Pcap faellt da auch drunter. Fuer die man aber vorher noch eine libpcap compilieren muss, was aber nirgends steht. Danach sollte ein `perl 'Makefile.PL'` mit folgendem `perl 'Wellenreiter.pl'` mit bisschen Config-Zeuch am laufen sein.

6.2.2 Das Programm an sich

... ist sehr schoen gemacht. Alles sehr komfortabel erreichbar und gut beschrieben.

Man kann seit der version 1.0 auch versteckte W-LANs damit finden. Beacon Frames spuert es seit 1.1 sehr zuverlaessig auf. Das GPS Feature ist zwar wirklich nett, aber erfordert zusaetzliche Hardware bzw. zusaetzliche Konfiguration. Ein professionelles Tool von fuer Profis.

6.3 BaseStation Configurator

Ein voellig neuer Ansatz ist es, gleich die Basestations anzugreifen. Hier habe ich bis jetzt nur wenige Tools gefunden, die auch nicht immer funktionieren. Das Ganze steckt noch in den Kinderschuhen, wie es aussieht.

Die Idee basiert darauf, dass man die die BaseStations entweder so zumuellt, dass sie keine weiteren Clients mehr aufnimmt und der Angreifer sich selber dann als Basestation ausgibt und somit jeglichen Verkehr mitloggt.

Diese Ansatz ist deswegen so interessant, weil er selbst dann noch funktioniert, wenn das Verschlüsselungsverfahren mal erhoeht wird. Da der Client mit der vermeintlichen Basestation ja den Schluessel austauscht, waegt sich dieser in Sicherheit. Das wir als Angreifer die Daten allerdings ohne Probleme wieder entschluesseln koennen, mit dem Schluessel den wir vorher bestimmt haben, nuetzt auch eine hoehere Verschluesselung nichts. Das Prinzip ist aehnlicher einer MITM-Attack. Andere Ansaetze sind die Basestation einfach so umzustimmen das sie die Verschluesselung ganz abdreht. Aeltere Stations schalten angeblich die Verschluesselung ab, bzw. runter, wenn viel Traffic auf der „Leitung“ ist.

Ansaetze gibt es genuegend in dieser Richtung, mal sehen was daraus wird.

6.4 Andere Programme

Neben diesen eben vorgestellten gibt es noch jede Menge anderer Tools, die ich hier aber nicht alle vorstellen kann, da mir die Zeit dazu fehlt. Der Einstieg sollte damit gelungen sein und weitere Vertiefung muss jeder fuer sich selber erarbeiten. Im Netz finden sich genuegend Adressen mit Programmen, die aehnliches koennen oder versprechen. Die vorgestellten Programme zaehlen momentan zu meinen Favoriten.

6.5 Wardriving

Angeblich hat Peter Shipley Wardriving, oder auch 'Drive-by-Hacking' genannt erfunden, herausfinden laesst sich das aber nicht mehr.

Man faehrt mit dem Auto, ausgeruestet mit einem Laptop und einer mehr oder weniger guten Antenne, durch die Gegend und sucht nach offenen W-Lans. Dadurch, dass man von Umwelteinflusse, wie laestigem Regen oder Kaelte, geschuetzt ist, kann man praktisch zu jeder Jahres- und Uhrzeit aktiv werden. Unschlagbarer Vorteil dabei ist, dass man so gut wie unbemerkt bleibt und nach zwei Stunden nicht nach Hause muss, weil der Akku leer ist. Man kann sich also ruhig Zeit lassen und muss nicht gegen die Laufzeit des Notebooks ankaempfen.

Auf diese Art und Weise sind auch viele der Karten entstanden, die zeigen wo offene W-Lans anzutreffen sind.

Am besten man sucht sich noch einen Freund, der mitfaehrt und den Signal- und Linklevel im Auge behaelt. Dadurch verringert sich nicht nur die Unfallgefahr.



Abbildung 5: Wardriving in der Bay Area, Ausruestung und Einsatz

6.5.1 Warwalking

Eine etwas neuere Art, die sich fast schon als nerd-sport bezeichnen lässt, ist 'Warwalking'. Mit dem Laptop im Rucksack kommt man heutzutage überall rein. Seien es Supermärkte oder grosse Einkaufszentren. Viele Grosshandelsgeschäfte stellen inzwischen von analogen auf digitale Preisschilder um. Früher wurden diese auf der freien 433MHz Frequenz gesendet, inzwischen gibt es aber auch einige, die auf W-Lan Technik basieren. Inwieweit sich das durchsetzt, bleibt abzuwarten.

Nike-town in SF ist auch ein nettes Ziel. Dort gibt es keine Kassen mehr, sondern die Verkäufer haben kleine, PDA-ähnliche Geräte bei sich, vergleichbar mit denen von UPS, nur nicht ganz so gross. Der Verkäufer scannt den Artikel ein, tippt die Kreditkartennummer ab und lässt den Kunden unterschreiben. Das Ganze geht dann, na ratet mal, via 802.11 an einen Zentralrechner. Der Rucksack wird zwar beim Rausgehen kontrolliert, aber man muss dem ebenso breiten wie grossen Typen ja nicht erzählen was alles drauf ist auf dem Laptop, den er findet.

Wenn man dann schon in der Nähe des Silicon Valleys ist, kann man dort auch gleich ein paar Firmen besuchen. Manche bieten an gewissen Tagen Besuchertouren an. Dort gibt's bestimmt eine Menge interessanter Dinge zu sehen.



Abbildung 6: Rucksack mit Laptop und externer WLAN Antenne

6.5.2 Warflying

In modernen Flugzeugen wird auch W-Lan angeboten und man kann sogar per Satellit surfen. Dieser Dienst ist momentan noch sehr teuer und es leisten sich nur Top-Manager dieses Angebot. Was wiederum das ganze interessant macht. Durch geschickt manipulierte Börsenkurse oder leicht veränderte Newsmeldungen könnte man durchaus daraus Profit schlagen. Inwiefern man in die Flugzeugsteuerung eingreifen kann, bzw. ob man sich am Flughafen direkt von den Economyclass in die Firstclass buchen kann, bleibt mal weiteren Tests vorbehalten.

7 Schutzmassnahmen

Was kann man tun damit man nun nicht gehacked wird.

- Don't do it
- IPsec benutzen
- Keine sicherheitsbedenklichen Bereiche ueber W-Lan zugaenglich machen (ftp, smtp, DB)
- Firmen oder private Inhalte nicht nutzen (POP3, Online-Banking)
- W-Lan Hidding (wurde mit Wellenreiter Ver 1.1 gebrochen)

7.1 Vorhandene Sicherheit nutzen

Ein paar Sachen sind bereits vorhanden, mit denen sich die Sicherheit erhoehen laesst. Es erschwert den Angriff, macht diesen aber nicht unmoeglich.

- Zuerst sollte man mal DHCP entweder ganz abdrehen oder nur an MAC-Adressen binden. Wer munter IP-Adressen unters Volk streut, sollte sich nicht wundern wenn er gehacked wird. Da man aber die MAC-Adresse von Netzwerkkarten ziemlich einfach aendern kann, ist das keine effektive bzw. ueberhaupt keine Schutzmassnahme gegen Hacking.
- Es gibt die Moeglichkeit sein W-Lan zu verstecken. Somit wird der Netzwerkname nicht provided und der Angreifer sieht das Netz somit nicht. Problem dabei ist aber, dass durch den Traffic der „in der Luft liegt“, dennoch festgestellt werden kann, dass sich ein W-Lan im Betrieb befindet. Das Tool Wellenreiter findet anhand dieser Eigenschaft auch versteckte Netzwerke. Somit ist dies auch keine Schutzmassnahme mehr gegen Hacking.
- W-Lan nur zu aktiven Buerzeiten aktivieren. Vom Stromnetz trennen waere die optimale Loesung und durch Zeitschaltuhren sicherlich einfach realisierbar. Eine einfachere Loesung ist es, dem Access-Point einfach bei Nichtbenutzen mit einer Firewall-Regel alles zu verbieten. Der Angreifer haengt dann am AP fest (normalerweise), aber er kann damit auch schon wertvolle Informationen erlangen ueber die Configuration und somit ein Eindringen tagsueber in wesentlich weniger Zeit schaffen.
- Begrenzung von W-Lan moeglichst auf interne Gebaedeteile. Wer Access-Points an Aussenmauern oder gar im Freien montiert, kann gleich ein TP-Dose an die Pforte schrauben.

7.2 Mal kurz etwas zu IPsec

IPsec ist momentan das einzige Moeglichkeit, W-Lan auf sichere Art und Weise nutzen zu koennen. Solange man aber keine IPv6 benutzt, ist das Einsetzen auesserst muehsam und unhandlich. Die IPv4 Implementierungen sind da nicht so toll. Zudem gibt es meines Wissens nach keine Implentierung, die auf allen Plattformen gut laeuft. Mein Wissensstand beruht aber auf Anfang 2002, vielleicht hat sich da inzwischen etwas getan.

Zudem bietet es nur Schutz bei Verbindungen, die von einem selbst ausgeloeset wurden. Gegen Eindringen in das System schuetzt man sich damit nicht zuverlaessig.

7.3 ...aber mit WEP2 ...

... wird alles gleich bleiben. Die Schluessellaengen haben sich zwar erhoehrt, dennoch ist das grundlegende Problem des Protokollaufbaus nicht geloest/verbessert. Zudem sind die Rechner auch deutlich schneller geworden und es gibt in absehbarer Zeit sicherlich auch

Notebooks mit knapp 2GHz. Damit aendert sich also nur wenig am Zeitaufwand fuer die Berechnung. WEP2 dient fuer die Industrie nur als Geldeinnahme-Quelle, ohne die Sicherheit zu erhoehen.

Hier eine kurze Zusammenfassung was geandert wurde:

- WEP2 vergroessert die IV Schluessellaenge auf 128bits. Verhindert aber nicht das Problem das der IV einfach hochgezahlt wird und es ist weiterhin erlaubt das der IV wiederbenutzt werden darf.
- Die Erlaubnis den IV wiederzubenutzen, kombiniert mit einer gefakten MAC Adresse, erlaubt es dem Angreifer weiterhin die Authentifizierungen zu umgehen oder zu faelschen.
- Der Klartext Angriff, bei welchem der Angreifer einen Teil der Nutzdaten erraet bzw. einfach bekannt sind, und diese mit dem IV Schluessel und dem CRC32 check die Verschlusselung brechen kann, funktioniert mit WEP2 auf die gleiche Weise wie mit WEP1
- Die verpflichtende Einfuehrung von KerberosV Unterstuetzung macht den WEP2 anfaellig gegen dictionary-attacks. Es verwenden einfach zu viele Angestellte noch zu einfache Passwoerter.
- Der Verbdinaungsauffbau und Abbau ist weiterhin unsicher
- Beacon Messages benoetigen weiterhin keinerlei Authentifizierung, was es Clients weiterhin erlaubt einfach Access-Points als roaming stelle zu benutzen.

7.4 Was nun? Was tun?

Um es mit Peter Lustig's Worten zu sagen: **Abschalten**, solange keine sichere Loesung auf Hardware-Ebene verfuegbar ist. IPsec ist zwar eine Loesung die eine sichere Kommunikation ueber W-Lan erlaubt, eigentlich die Einzige, aber dies erfordert zum einen einen sehr hohen administrative Aufwand und ist zudem auch nicht gut in heterogenen Netzwerken einsetzbar. Ein weiterer Punkt ist immer noch der User, der IPsec aus Versehen abschalten kann oder die Software verstellt und somit das Ganze aushebelt.

Abgesehen davon, dass nichts sicher ist, sollte man sich nicht noch Gefahren aussetzen, die von vornherein absehbar sind und zu grossen Schaeden fuehren koennen. Falls man trotzdem c00l sein will und W-Lan haben muss, sollte man wenigstens keine kritischen Daten in den Aether jubeln.

8 CTRL+D

Aufpassen was man tut. Das sollte man eigentlich immer, wenn man sich in solches Umfeld begibt. Dennoch ist hier mehr Vorsicht geboten, als sonst. Stecker ziehen hilft bei W-LAN nicht viel.

Vor Ueberwachungskameras sollte man sich auch in Acht nehmen, Konflikte mit der ausfuehrenden Rechtskraft sollte man generell vermeiden. Im absoluten Halteverbot parken um irgendwo rumzusniffen ist nicht sonderlich ratsam, wird aber mal schnell uebersehen, wenn man einmal ein Netz gefunden hat.

Aber wer mal gern anonymes Internet nutzen will, sollte die Gelegenheit nutzen, solange sich einem die Gelegenheit bietet . . .

Grundsatz: Wissen nutzen – nicht missbrauchen.

. . . dann mal viel Spass am Geraet.

9 Danksagung

Der Dank gilt allen, die sich bereits vor mir mit diesem Thema beschaeftigt haben und Infos darueber veroeffentlicht haben. Hier ist zum einem LISA THALHEIM und zum anderen GLENN FLEISHMAN zu erwaechnen.

Die Jungs und Maedls, die an Treiber und Toolz geschrieben haben und diese weiterhin pflegen.

Herrn Professor POHL fuer seine gute Vorlesung im Bereich „Codes und Kryptographie“ der mein Interesse noch mehr geweckt hat, als ohnehin eigentlich eh schon war.

Meinem persoentlichen Lektor der mich beim Durchlesen und Ueberarbeiten unterstuetzt hat und meine 10^{23} Rechtschreibfehler korrigiert hat.

Zuletzt auch noch etwas Kommerz: TOSHIBA REGENSBURG fuer den wirklich starken Support im Bereich Hardware, ohne den so ein Student, wie ich, das Ganze nie haette testen koennen.