WiFi Chipset Fingerprinting

Günther Lackner · Mario Lamberger · Udo Payer · Peter Teufl

Institut für Angewandte Informationsverarbeitung und
Kommunikationstechnologie, IAIK,
Technische Universität Graz, A-8010 Graz, Austria
{guenther.lackner, mario.lamberger, udo.payer,
peter.teufl}@iaik.tugraz.at

Zusammenfassung

Die vorliegende Arbeit untersucht Aspekte der Media Access Control (MAC) bei WiFi Geräten. Nach einer kurzen Zusammenfassung von Problemen und Attacken wird eine Versuchsanordnung beschrieben, die es ermöglicht, die Chipsätze von 802.11-Geräten anhand der Verzögerung von Acknowledge-Paketen mit einem auf Self Organizing Maps (SOMs) basierenden Algorithmus zu klassifizieren.

1 WIFI MAC-Adressen Spoofing

Eine MAC (Media Access Control)-Adresse sollte laut ihrer Definition ein Hardware-Netzwerkinterface global eindeutig identifizierbar machen. Dies gilt sowohl für LAN als auch für WLAN Geräte. Die Verteilung der MAC-Adressen unterliegt dem so genannten **OUI** (Organizationally Unique Identifier) Schema [1], welches von der **IEEE** verwaltet wird. Eine Standard-MAC-Adresse besteht aus 48 Bits [2], wobei die Bits 45 - 24 den oben genannten OUI repräsentieren. Dieser muss von einem Gerätehersteller bei der IEEE beantragt und lizensiert werden [3]. Die restlichen Bits einer MAC-Adresse dienen dann als Nummerierung der vom OUI Besitzer hergestellten Geräte. Jede MAC-Adresse darf nur einmal vergeben werden. Ein OUI erlaubt die Vergabe von ca. 2^{24} verschiedenen Adressen. Der Vollständigkeit halber sei hier noch angemerkt, dass es neben dem recht teuren OUI auch den so genannten **IAB** [4] gibt. Dieser wird ebenfalls von der IEEE als Lizenz vergeben und ermöglicht es kleinen Geräteherstellern kostengünstig MAC-Adressen zu vergeben. Pro IAB können jedoch nur 2^{12} verschiedene Adressen vergeben werden.

Das für uns relevante Faktum ist, dass durch die Vergabe der MAC-Adressen versucht wurde, die Netzwerkgeräte global einfach und eindeutig identifizierbar zu machen. Auf der Grundlage dieses Konzepts ließen sich einfach effektive Zugangskontroll- und Beschränkungsmethoden für Netzwerke realisieren. Alle diesbezüglich verwirklichten Ansätze setzen jedoch die Eindeutigkeit einer MAC-Adresse und deren Resistenz gegen Nachahmung voraus. Es ist jedoch sehr einfach die MAC-Adresse in Netzwerkpaketen zu verändern und somit die Identität eines anderen Netzwerkgerätes vorzutäuschen. Auf Grund dieser Tatsache wurden alle auf MAC-Adressen basierenden Identifikations- und Authentifizierungsmechanismen wertlos. Unser Ziel ist es nun, Eigenschaften von Netz-

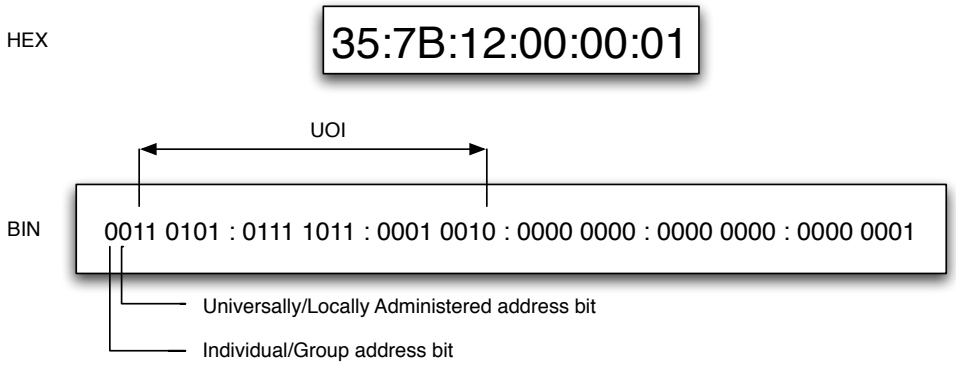

Abbildung 1: Aufbau von MAC-Adressen

werkgeräten zu finden welche eine zumindest lokal eindeutige Identifizierung zulassen.

2 Bekannte, auf MAC-Spoofing basierende Attacken

Es existieren viele mögliche Attacken gegen auf 802.11 basierende Netze, welche auf dem Fälschen von MAC-Adressen beruhen. Viele davon gehen in die Richtung von **Denial of Service** (DoS) Attacken aber auch Replay und Injection Attacken, welche zum Mitsniffen von Anmeldevorgängen oder zum Erzeugen von *Traffic* dienen fallen in diesen Bereich. Im Nachfolgenden werden kurz drei mögliche Attacken erklärt. Eine genauere Abhandlung zu diesem Thema würde jedoch den Rahmen dieses Artikels sprengen.

2.1 Denial of Service Attacke durch Deauthentifizierung

Jeder Client muss sich bei einem Access Point (AP) authentifizieren, wenn er Zugang zum Netzwerk haben will. Dieser Vorgang erfolgt durch den Austausch von bestimmten Management-Frames. Nach diesem Vorgang ist der Client authentifiziert, der AP kennt nun die MAC-Adresse des Clients und gestattet auf Grund dieser die Kommunikation mit dem Netzwerk. Unglücklicherweise gibt es jedoch in 802.11 Protokollen keinen Authentifizierungsmechanismus für diese Management-Frames [6]. Folglich kann ein Angreifer beliebige Management-Frames mit gefälschten MAC-Adressen versenden [5], unter anderem natürlich auch jene, welche für den An- und Abmeldevorgang eines Clients am AP verwendet werden. Alleine durch das Absetzen von so genannten *Disassoziation-Frames* mit der gefälschten AP-Absenderadresse ist es nun möglich einen Client vom AP zu trennen da dieser nach dem Erhalt eines solchen Frames jegliche Kommunikation mit dem AP einstellt. Denselben Effekt erzielt man durch Absetzen eines Disassoziation-Frames mit der Client-Absenderadresse an den AP, siehe [5]. In diesem Fall erfährt der Client nicht einmal, dass er abgemeldet wurde, da der AP nun alle Nachrichten an den Client ohne Meldung verwirft. Erst wenn der Client wieder etwas senden will bemerkt er, dass er vom AP getrennt ist und startet einen neuen Anmeldevorgang [6]. Dies kann von einem Angreifer zum Sniffen von Anmeldevorgängen missbraucht werden.

2.2 Denial of Service durch Ausnützung des Energiesparmodus

Der 802.11 Standard ermöglicht es den Client-Geräten, in einen Energiesparmodus einzutreten [5]. Während dieses Zustandes ist der Client weder in der Lage zu Senden noch

zu Empfangen. Durch das Verschicken einer Nachricht an den AP kann der Client den bevorstehenden Eintritt in den Energiesparmodus mitteilen, woraufhin der AP alle für diesen Client bestimmten Frames in einem Buffer zwischenspeichert. Von Zeit zu Zeit erwacht der Client und überprüft die so genannte *TIM (Traffic Indication Map)*[1] des APs, ob dieser Frames für ihn gebuffert hat. Durch Absenden eines *PS-Poll-Frames*[2] erbittet der Client nun die Übermittlung der gespeicherten Frames, welche nach der Übertragung aus dem AP Buffer gelöscht werden [7].

Befindet sich nun ein Client im Energiesparmodus so kann ein Angreifer durch Fälschen eines PS-Poll-Frames die gebufferten Nachrichten des Clients abrufen, welche danach sofort gelöscht werden. Auf diese Art ist es möglich den schlafenden Client zu blockieren [7].

2.3 Der falsche Accesspoint

Ein Angreifer ist in der Lage durch Fälschen einer MAC-Adresse die Identität eines Access Points (AP) nachzuahmen. Dies kann zum Beispiel in der Nähe eines öffentlichen Hot Spots geschehen. Ein Client kann sich nun versehentlich bei diesem *falschen AP* anmelden, da dieser den wahren AP zum Beispiel durch höhere Signalstärken verdeckt [6]. Nun geht die gesamte Kommunikation des Clients durch den falschen AP welcher durch das Umleiten auf gefälschte Internet Portale die Login Daten des Clients stehlen könnte. Auf diese Art können auch SSH und HTTPS Verbindungen angegriffen werden, siehe [6].

3 Maßnahmen gegen 802.11 MAC-Spoofing

Alle bisher entwickelten Ansätze zur Erkennung gespoofter MAC-Adressen sind nur bedingt wirksam, da sie keine einzelnen Geräte sondern nur bestimmte Geräteklassen von einander Unterscheiden können.

3.1 Sinnhaftigkeit von MAC-Adressen

Dieser Ansatz überprüft die MAC-Adressen aller verbundenen WLAN Geräte auf Plausibilität. Viele Programme welche automatisiert MAC-Adressen spoofen, erstellen die gefälschten Adressen per Zufall aus dem gesamten möglichen Wertebereich von 2^{45} Adressen. Da es aber nur eine endliche Zahl von OUIs gibt (ca. 10.000) würde nur ein geringer Teil dieser zufällig generierten Adressen einem realen OUI entsprechen. Durch den Vergleich mit einer Liste aller lizenzierten OUIs könnte man recht einfach den Versuch von MAC-Spoofing detektieren sofern er auf zufällig generierten MAC-Adressen beruht [5]. Diese Maßnahme kann man jedoch sehr einfach umgangen werden, indem man die gespooften MAC-Adressen nur mit gültigen OUIs erstellt. Diese Methode kann aus diesem Grund auch als wirkungslos angesehen werden [5].

3.2 WLAN Sequence Number Analyse

Der MAC-Layer von 802.11 ist deutlich komplexer als der von älteren IEEE 802 Protokollen. Die Planung der 802.11 Protokolle wurde durch Faktoren wie Verfügbarkeit und Zuverlässigkeit bei der kabellosen Übertragung und die notwendige Fähigkeit des Roamings

[1] Die TIM wird vom AP in so genannten Beacon Frames als Broadcast übertragen
[2] Power Safe Poll-Frame

zwischen verschiedenen Zugangspunkten beeinflusst [5]. Es wurden unter anderem 2 Bytes für eine Sequenz Kontrolle zur Verfügung gestellt, 4 Bits für eine Fragment Number und 12 Bits für eine Sequence Number. Sollte es bei einer Übertragung notwendig sein Pakete zu fragmentieren, so werden diese Fragmente mit konstanter Sequence Number und aufsteigender Fragment Number gesendet [6].

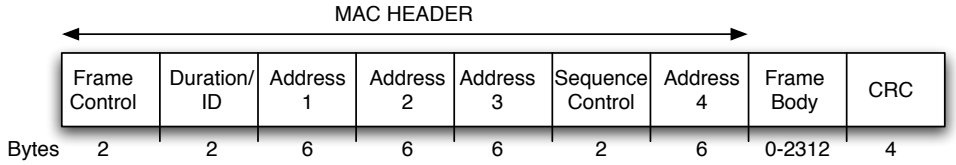

Abbildung 2: MAC-Header

4 802.11 Media Access Control System

Ähnlich wie bei anderen Netzwerk-Standards findet die *Media Access Control* auch bei 802.11 am MAC-Layer (Layer 2) statt. Auf Grund der Tatsache, dass WiFi Geräte nicht *full duplex*[3] fähig sind, können sie auch keine Kollisionen im Übertragungsmedium erkennen [8]. Aus diesem Grund ist ein Kollisionserkennungsverfahren notwendig. Das in 802.11 Implementierte nennt sich **DCF** (Distributed Coordination Function) und ist vom Prinzip her ein so genanntes **CSMA/CA** (Carrier Sense Multiple Access with Colission Avoidance) [9]. Will ein Client eine Nachricht senden, so muss er zuerst überprüfen ob das Medium frei ist. Sollte dies nicht der Fall sein so muss er seine Übertragung auf einen späteren Zeitpunkt verschieben. Die Kontrolle über das Übertragungsmedium wird durch den Einsatz einer ganzen Reihe von Management Nachrichten gewahrt [8]. Diese erlauben es einer zentralen Stelle, zum Beispiel einem Access Point (AP), die Zugriffe der Clients auf das Medium in gewissem Maße zu beeinflussen. Zu diesen Nachrichten gehört neben CTS (Clear to Send), RTS (Request to Send) und einigen mehr auch das ACK (Acknowledge). Dieses ACK wird zur Bestätigung des Erhalts einer Unicast Nachricht an den AP verschickt. Management Nachrichten sind kleine Pakete welche über das gemeinsame Medium übertragen den Clients den Zustand des Mediums mitteilen [8]. Die für unsere Messungen interessante Eigenschaft dieses Systems ist die Tatsache, dass jedes Unicast Datenpaket vom Empfänger mit einem ACK bestätigt wird. Fr Eine detailliertere Erklärung des 802.11 Media Access Control Systems sei auf [8] und [9] verwiesen.

5 Die Acknowledge Verzögerung

Wie bereits im letzten Absatz erwähnt, wird vom Empfänger eines 802.11-Datenpakets nach dem korrekten Berechnen der Checksumme ein ACK an den Sender abgesetzt[4] [8]. Management Frames, Multi- und Broadcast Pakete werden nicht bestätigt. Ein ACK Paket hat eine Länge von 14 Bytes [10].

Die ersten 2 Bytes dienen der *Frame Control*. Byte 0 gibt den Paket Typ an,`d4` steht für ACK. Die Bytes 1 bis 3 werden bei ACK Paketen nicht benötigt, ihre Werte sind für uns

[3] Fähigkeit gleichzeitig zu senden und zu empfangen
[4] Sofern die Übertragung fehlerfrei war

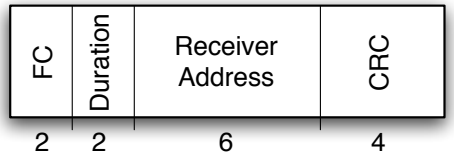

Abbildung 3: ACK Paket

daher nicht von Interesse. Auf die Frame Control Parameter folgt die 6 Byte Receiver Adresse MAC-Adresse. Und am Ende gibt es noch 4 Byte für eine CRC Checksumme [8].

Den zeitlichen Verlauf der Kommunikation zwischen Client und Access Point kann man sich wie in Abbildung 4 dargestellt vorstellen.

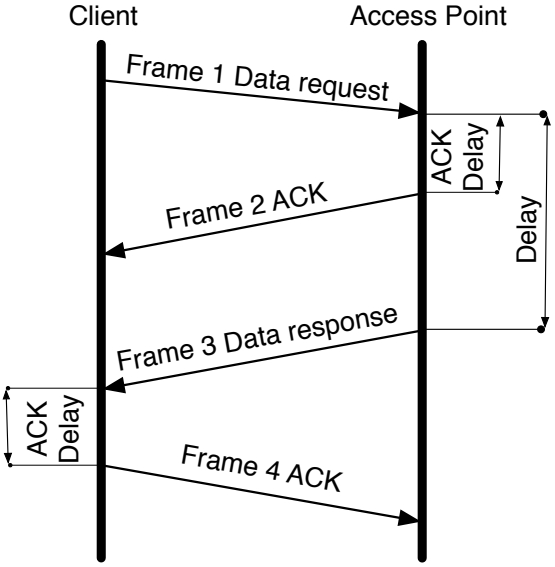

Abbildung 4: ACK Delay

Frame 1 könnte ein DNS Request oder etwas ähnliches sein. Der Access Point empfängt das Paket und überprüft nun die Integrität desselben durch Berechnung der CRC Checksumme. Dieser Vorgang erfolgt in Hardware und ist somit vom Netzwerk-Stack des WiFi Gerätes, welcher in Software implementiert ist, unabhängig und sollte somit in konstanter Zeit erfolgen. Direkt danach wird ein ACK mit dem Client als Empfänger abgesetzt [10]. Nun wird der DNS Request an das Netzwerk weitergeleitet. Sobald die Antwort vom DNS Server über das Netzwerk zum Access Point zurückgekommen ist, wird diese als 802.11 Datenpaket an den Client versendet. Dieser überprüft wiederum die CRC Checksumme in Hardware und generiert ein ACK Paket an den Access Point. Erst danach wird die Payload über die Firmware des Geräts an den IP-Stack des Client Betriebssystems weitergegeben. Das ACK Delay ist somit vom OS und der CPU Auslastung des Clients unabhängig [10]. Dieser Vorgang erfolgt in relativ konstanter Zeit. In den Abbildungen 6 ist die zeitliche Verteilung der ACK Verzögerung für drei verschiedene WiFi Chipsätze dargestellt. Das Clustering um die Bereiche von 110 ms, bzw. 220 ms steht vermutlich in direktem Zusammenhang mit der Beacon-Rate von 102,4 ms. Der Vorgang der ACK Generierung im

Chipsatz benötig laut Hersteller nur in etwa 50 μs. Die Fingerprints entstehen durch die Streuung des ACK Delays um das Beacon-Raster. (Wir danken Herrn Ernst Ahlers von Heise für seine Anregungen.)

5.1 Messaufbau

Um die ACK Verzögerung messen zu können ist es notwendig die Kommunikation zwischen einem WiFi *Client* und dem *Access Point* mitzuhören. Als Access Point haben wir ein Gerät der CISCO AIRONET 350er Serie verwendet. Der AP war über einen *Fast Ethernet Switch* mit dem *Netzwerk (LAN)* verbunden in dem unser *Server* stand. Die *Sonde (Probe)* war ein Apple PowerPook G4[5] , welches im Monitor Mode betrieben wurde. Das *Packet Capturing* erfolgte mit dem WLAN Sniffer *KISMAC* [13] und dem Netzwerkanalyse-Tool *ETHEREAL* [12]. Der ETHEREAL-Output wurde mit einem *PYTHON Skript* gefiltert und in eine Datenbank geschrieben.

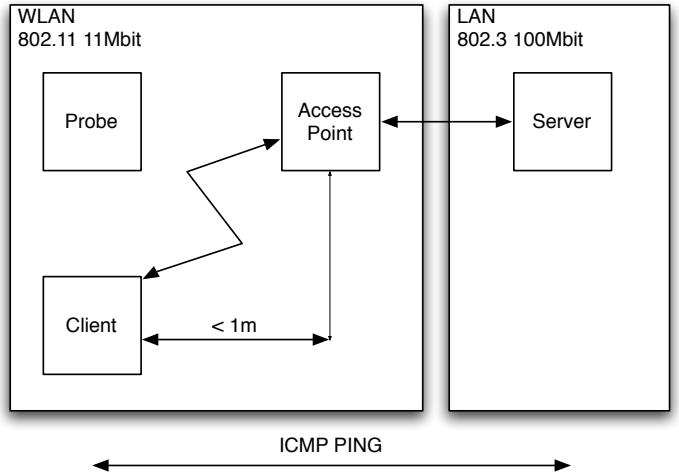

Abbildung 5: Messaufbau

Zur Erzeugung von Paketen und dazugehörigem ACK haben wir ICMP PING verwendet. Vom Server ausgehend wurde an den Client ein PING REQUEST gesendet welcher den Empfang dieses Pakets mit einem ACK und einem PING REPLY beantwortet hat. Das PYTHON Skript hat nun das PING REQUEST und das dazugehörige ACK aus dem gesamten gesnifften Traffic gefiltert. Durch Subtraktion der beiden von ETHEREAL erzeugten Timestamps konnten wir die Verzögerung zwischen dem Auftreten des PING REQUEST-Pakets und dem Auftreten des ACK-Paketes berechnen. Diese Verzögerung ist außer von der Zeit zur Berechnung der Checksumme auch noch von anderen Faktoren, wie der Übertragungsgeschwindigkeit im Medium abhängig. Da die zeitliche Auflösung von ETHEREAL jedoch nur im Mikrosekundenbereich liegt können diese ruhig vernachlässigt werden, da sie keinen messbaren Einfluss auf unser Experiment haben oder im Betrag konstant auftreten.

Dieses Experiment wurde wiederholt bis wir pro verwendetem Chipsatz mehrere tausend Testwerte zur Verfügung hatten. Diese Testwerte wurden vom PYTHON Skript in eine

[5] Airport Extreme, MAC OSX 10.4

Datenbank geschrieben und nach Beendigung der Messung für das Importieren in MATLAB vorbereitet, wo alle weiteren Analysen stattfanden.

6 Klassifikationsmethoden

Um festzustellen, ob die Delay-Zeiten der einzelnen ACK-Pakete für die Klassifizierung des verwendeten Chipsets verwendet werden können, wurden die gesammelten Daten folgendermaßen aufbereitet: Für jeweils 50 ACK Pakete wurde ein Histogramm erstellt, dass auf der X-Achse die Delay Werte von 1-300 ms und auf der Y-Achse die relative Häufigkeit zum jeweiligen Zeitpunkt dargestellt. Nach einer ersten Analyse der verwendeten Chipsets wurde erkannt, dass sich diese Aufbereitung der Daten dazu eignet die verschiedenen Chipsets zu trennen und somit klassifizieren zu können. Die Histogramme der verwendeten Chipsets sind in Abbildung 6 dargestellt.
Für die Klassifikation der Daten wurden Self Organizing Maps (SOMs) aus dem Bereich des unüberwachten Lernens herangezogen. Der Grund für die Wahl von SOMs ist, dass bereits bei einer früheren Arbeit [14] sehr gute Ergebnisse mit einem selbstentwickelten, auf SOMs basierenden Klassifikationsalgorithmus erzielt wurden. Bei diesem Algorithmus wurde der SOM-Algorithmus erweitert, so dass überwachtes Lernen durchführbar ist. Weiters wurde es durch die Verwendung von mehreren SOMs, die in einer baumartigen Struktur angeordnet sind, möglich die Klassifizierungsgenauigkeit zu erhöhen. Die Anzahl der benötigten SOMs wird dabei automatisch vom Algorithmus bestimmt.

6.1 Self Organizing Maps

Self Organizing Maps (SOMs) [15] gehören zu der Gruppe der Neuronalen Netze (siehe [16]) und können dazu verwendet werden nichtlineare Abhängigkeiten von hochdimensionalen Daten in einem niedrig-dimensionalen Raum darzustellen. SOMs sind aus dem Bereich des unüberwachten Lernens und benötigen somit beim Training keine markierten Daten. Dadurch besteht die Möglichkeit, Zusammenhänge und Abhängigkeiten in unbekannten Daten zu erkennen. SOMs können aber auch so erweitert werden, dass überwachtes Lernen möglich ist. Dies und andere Erweiterungen wurden bereits in einer früherer Arbeit entwickelt.

6.1.1 HITC - Hit Based Classification

Der von uns entwickelte Algorithmus basiert auf der Best Matching Unit (BMU). Eine BMU kann für jeden Datensatz gefunden werden und ist jene Unit, die diesem Datensatz am nächsten liegt. Nun können einer trainierten SOM markierte Daten übergeben werden und für jeden dieser Datensätze die BMU bestimmt werden. Mit dieser Information kann für jede SOM Unit die Anzahl der Treffer (Hits) der verschiedenen Klassen bestimmt werden. Im Falle von Klassen, die über eine unterschiedliche Anzahl von Trainingsdaten verfügen, muss die Trefferanzahl einer Klasse pro Unit mit der Gesamtanzahl der Daten in dieser Klasse normiert werden.
Unbekannte Daten können jetzt anhand dieser Trefferinformationen klassifiziert werden. Für jeden unbekannten Datensatz kann eine BMU auf der SOM gefunden werden. Der Datensatz wird zu jener Klasse zugeordnet, die diese BMU beim Training am häufigsten getroffen hat. Abbildung 7 zeigt eine SOM, die die Hitinformationen der Chipsets Broadcom (rot), Linksys (grün) und Intel BG2200 (blau) darstellt.

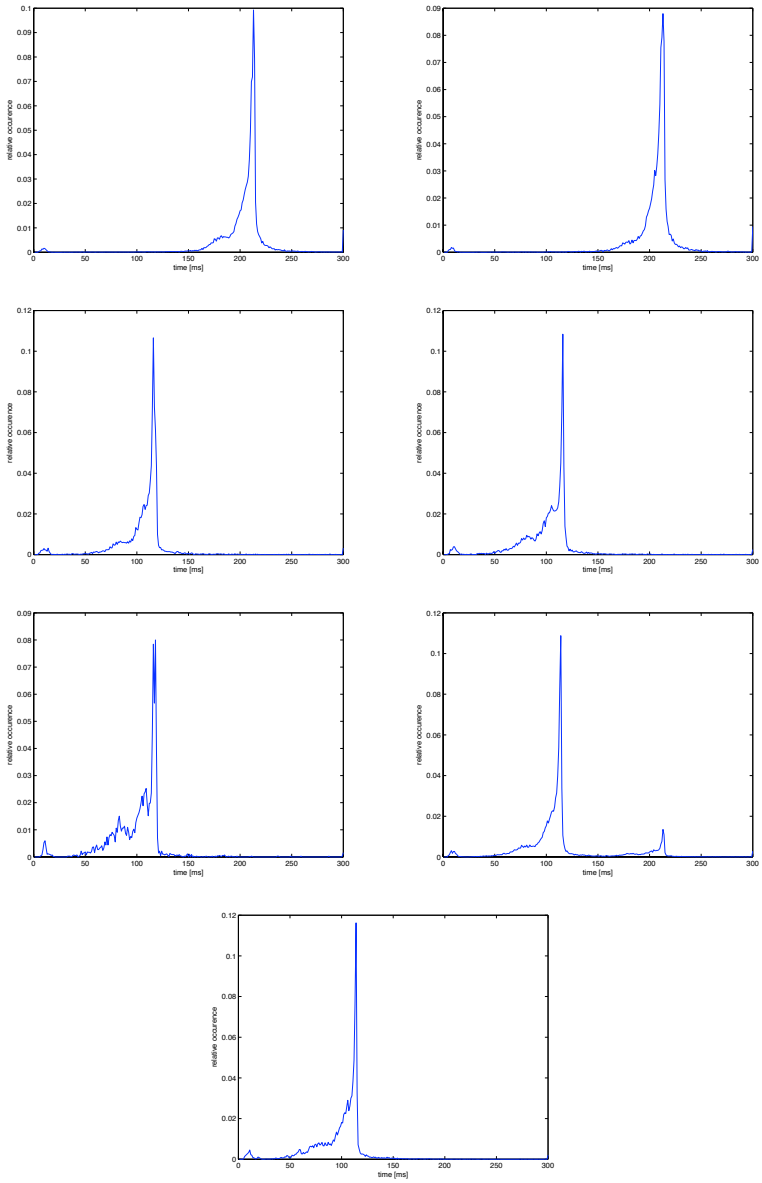

Abbildung 6: Chipset Orinoco, Prism2, Broadcom 94306 (Linksys), BG2200, BG2100, Broadcom 43xx (Acer NB), Broadcom 43xx (Ibook)

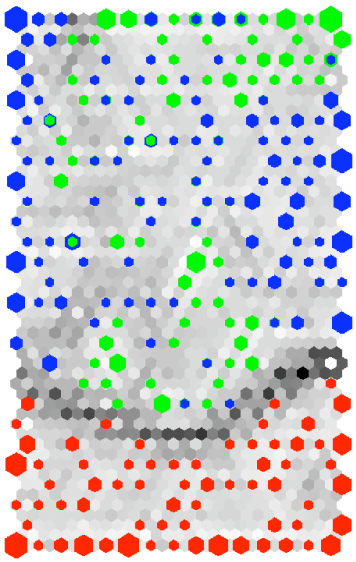

Abbildung 7: SOM mit Hits von drei Klassen - Rot (Broadcom), Grün (Linksys) und Blau (Intel BG2200)

6.1.2 Som Tree

Bei vielen Klassifikationsaufgaben reicht eine einzelne SOM oft nicht aus, um die nötige Klassifikationsgenauigkeit zu erreichen. Der verwendete SOM Algorithmus wurde so gestaltet, dass automatisch die Anzahl der benötigten SOMs festgestellt wird, diese trainiert und in einer baumartigen Struktur abgespeichert werden. Dabei werden folgende Schritte durchgeführt:

1. Trainieren einer SOM mit den Eingangsdaten
2. Überprüfen, ob die Trainingsdaten innerhalb einer adaptiven Fehlerrate mit HITC klassifiziert werden können.
3. Wenn alle Klassen innerhalb einer bestimmten Fehlerrate klassifiziert werden können, gehe zu Schritt 5.
4. Die Fehlerrate wird solange erhöht, bis mindestens zwei Gruppen [6] trennbar sind. Für jede dieser Gruppe werden die Daten zusammengefasst und rekursiv an Schritt 1 übergeben.
5. Abbruch des Algorithmus

Der beschriebene Algorithmus wurde mit Matlab und der Erweiterung SOMTOOLBOX[11], die Self Organizing Maps in Matlab zur Verfgung stellt, implementiert.

7 Ergebnisse

Für das Sammeln der Trainingsdaten wurden die dafür notwendigen ACK-Pakete mit Hilfe von ICMP Pings erzeugt. Diese Pakete wurden für jeden der verwendeten Chipsets gesammelt und in Matlab importiert. Die Daten wurden dann in Trainings -und

[6] Eine Gruppe besteht aus verschiedenen Klassen, die nicht innerhalb einer gewissen Fehlerrate voneinander trennbar sind

Testdaten aufgeteilt, wobei davon 60% der Daten für das Training der SOMs verwendet wurden. Für die Testdaten wurden drei unterschiedliche Aufteilungen in Verbindungsgrößen verwendet - 200, 500 und 1000 Pakete pro Verbindung. Als Verbindung wird in diesem Paper jener Verkehr bezeichnet, der von einer MAC Adresse ausgeht bzw. dorthin übertragen wird. Die Qualität der Klassifikationsergebnisse, die bei den drei unterschiedlichen Verbindungsgrößen erreicht wird, gibt Auskunft darüber, wieviele Pakete von einer MAC Adresse gesammelt werden müssen, um eine verlässliche Aussage über den verwendeten Chipsatz treffen zu können. Diese Erkenntnisse können für die Entwicklung eines Detektors verwendet werden, der entweder passiv oder aktiv in einem WLAN Netzwerk eingesetzt wird, um Rechner mit gefälschten MAC Adressen zu erkennen.

Für das Trainieren der SOMs wurden die vorhanden Trainingsdaten in Histogramme aufgeteilt, die mit jeweils 50 Paketen erstellt wurden. Das Training erfolgte mit dem oben beschriebenen Algorithmus. Für die Größe der verwendeten SOMs wurde die Standard Einstellung der SOMTOOLBOX verwendet.

Tabelle 1 zeigt die verwendeten Chipsets und die Anzahl der Trainings- und Testdatensätze.

Chipset	**Trainingsdaten** in Paketen	**Testdatensätze** zu 200/500/1000 Paketen
1: Orinioco (RoamAbout WLAN Karte)	36200	121/49/25
2: Prism II	22950	77/31/16
3: Broadcom 94306 (Linksys)	11750	40/16/8
4: Intel BG2200	18050	61/25/13
5: Intel BG2100	7050	24/10/5
6: Broadcom 43xx (Acer NB)	45350	152/61/31
7: Broadcom 43xx (Ibook)	11900	40/16/8

Tabelle 1: Test/Trainingsdaten 200/500/1000 Paketen pro Verbindung

Nach dem Training der SOMs wurde die Performance anhand der Testdaten evaluiert. Die Tabellen 2, 3, 4 zeigen die Evaluierungsergebnisse, welche in den Zeilen und Spalten gegenübergestellt werden. Bei Chipset 1 bedeutet dies, dass Chipset 1 zu 75,2% als Chipset 1 und zu 24,8% als Chipset 2 klassifiziert wird. Die Summe der Zeilen muss nicht 100% ergeben, da es möglich ist dass SOM Units getroffen werden, die während dem Training nicht getroffen wurden. Eine Verringerung der Anzahl der nicht klassifizierten Verbindungen kann erreicht werden, indem mehr Daten zum Trainieren der SOMs verwendet werden.

Aus den Ergebnissen können folgende Schlussfolgerungen getroffen werden:

- Wie erwartet, steigt die Performance mit der Größe der Verbindungen. Es zeigt sich, dass bei 1000 Paketen recht gute Ergebnisse erzielt werden können. Diese Erkenntnisse werden für die Entwicklung eines Scanners verwendet, der entweder im aktiven Modus - der Scanner sendet selbst Verkehr aus, um Daten zu bekommen - oder im passiven Modus - der Scanner sendet selbst keine Pakete aus und horcht nur auf den bestehenden Verkehr - verwendet.

- Die Chipsets 1 und 2 sowie 6 und 7 können vom Klassifikator nicht korrekt unterschieden werden. Dies war zu erwarten, da es sich dabei um die selben Chipsets handelt, die in unterschiedlichen WLAN Karten eingebaut sind.
- Die Ergebnisse in 4 sind vielversprechend und zeigen, dass die verwendeten Features durchaus für die Unterscheidung von WLAN Chipsets geeignet sind.

	1	**2**	**3**	**4**	**5**	**6**	**7**
1	**75,2%**	24,8%	0,0%	0,0%	0,0%	0,0%	0,0%
2	57,1%	**42,9%**	0,0%	0,0%	0,0%	0,0%	0,0%
3	0,0%	0,0%	**90.0%**	5,0%	5,5%	0,0%	0,0%
4	0,0%	0,0%	11,5%	**86,9%**	0,0%	0,0%	0,0%
5	0,0%	0,0%	29,1%	4,2%	**66,7%**	0,0%	0,0%
6	0,0%	0,0%	0,0%	0,0%	0,0%	**78,3%**	21,1%
7	0,0%	0,0%	0,0%	0,0%	0,0%	65,0%	**35,0%**

Tabelle 2: Ergebnisse mit 200 Paketen

	1	**2**	**3**	**4**	**5**	**6**	**7**
1	**55,1%**	44,9%	0,0%	0,0%	0,0%	0,0%	0,0%
2	61,3%	**38,7%**	0,0%	0,0%	0,0%	0,0%	0,0%
3	0,0%	0,0%	**87,5%**	0,0%	12,5%	0,0%	0,0%
4	0,0%	0,0%	0,0%	**96,0%**	0,0%	0,0%	0,0%
5	0,0%	0,0%	10,0%	0,0%	**90,0%**	0,0%	0,0%
6	0,0%	0,0%	0,0%	0,0%	0,0%	**82,0%**	18,0%
7	0,0%	0,0%	0,0%	0,0%	0,0%	62,5%	**37,5%**

Tabelle 3: Ergebnisse mit 500 Paketen

	1	**2**	**3**	**4**	**5**	**6**	**7**
1	**84,0%**	16,0%	0,0%	0,0%	0,0%	0,0%	0,0%
2	43,8%	**56,3%**	0,0%	0,0%	0,0%	0,0%	0,0%
3	0,0%	0,0%	**100,0%**	0,0%	0,0%	0,0%	0,0%
4	0,0%	0,0%	0,0%	**92,3%**	0,0%	0,0%	0,0%
5	0,0%	0,0%	0,0%	0,0%	**100,0%**	0,0%	0,0%
6	0,0%	0,0%	0,0%	0,0%	0,0%	**84,0%**	16,1%
7	0,0%	0,0%	0,0%	0,0%	0,0%	62,5%	**37,5%**

Tabelle 4: Ergebnisse mit 1000 Paketen

Im Echteinsatz werden Daten anhand der MAC-Adresse aufgeteilt, in Histogramme umgewandelt und anschließend klassifiziert. Das Gesamtergebnis entspricht der Klasse, der die Mehrheit der Pakete angehört.

8 Ausblick

Es wäre denkbar basierend auf unserem Ansatz einen Sensor zur Detektierung von gespooften MAC-Adressen zu entwickeln. Dieser könnte auf kleinen Geräten wie Handhelds leicht implementiert werden. Diese könnten das Medium belauschen und die Zusammegehörigkeit von Chipsätzen und MAC-Adressen überwachen. Angreifer mit dem gleichen Chipsatz wie das Opfer können jedoch durch diesen Ansatz nicht identifiziert werden. Es wäre jedoch vorstellbar das durch weitere Nachforschungen auf diesem Gebiet neben den unterschiedlichen Chipsätzen auch verschiedene Firmware Versionen klassifiziert werden könnten.

Im Moment arbeiten wir an der Implementierung eines Prototypen basierend auf Embedded Linux und iPAQ Hardware. Weiters ist der Bau eines Access Point Prototype mit integrierter MAC-Spoofing Detection geplant.

Literatur

[1] IEEE OUI License Page, IEEE OUI and Company id Assignments, (2006). http://standards.ieee.org/regauth/oui/index.shtml

[2] IEEE Tutorial, Standard Group MAC Addresses A Tutorial Guide, (2005). http://standards.ieee.org/regauth/groupmac/tutorial.html

[3] IEEE OUI License Page, IEEE Complete List of all OUIs, (2006). http://standards.ieee.org/regauth/oui/oui.txt

[4] Request Form for an Individual Address Block, (2006). http://standards.ieee.org/regauth/oui/pilot-ind.html

[5] Joshua Wright, Detecting Wireless LAN MAC Address Spoofing, (2003). http://home.jwu.edu/jwright/

[6] Fanglu Guo and Tzi-cker Chiueh, Sequence Number-Based MAC Address Spoof Detection, (2005), Technical Report, http://www.ecsl.cs.sunysb.edu/tr/TR182.pdf

[7] Pierre Devevey, IEEE 802.11 Power Saving Mode presentation in infrastructure network, (2004). http://sealab.disi.unige.it/Krakatoa/devevey/PSmode.pdf

[8] Pablo Brenner, A Technical Tutorial on the IEEE 802.11 Protocol, BreezeCOM Wireless Communications, (1997). http://www.cs.ucsb.edu/ cs290i_mc/papers/80211.pdf

[9] Timucin Ozugur, Optimal MAC-Layer Fairness in 802.11 Networks, ALCATEL Research andInnovation Center, USA, Copyright IEEE (2002). http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=996942

[10] Christian Hoene, Measuring Round Trip Times to Determine the Distance between WLAN Nodes, NETWORKING 2005: 768-779. http://www.tkn.tu-berlin.de/publications/papers/hoene_paper2.pdf

[11] SOM Toolbox 2.0 http://www.cis.hut.fi/projects/somtoolbox/

[12] Ethereal: A network protocol analyzer. http://www.ethereal.com/

[13] KisMAC: Passive stumbler for MacOS X http://kismac.de

[14] U. Payer and M. Lamberger and P. Teufl, Traffic Classification using Self-Organizing Maps, INC 2005 Fifth International Networking Conference Workshops, 05 - 07 July 2005, Samos Island, Greece.

[15] Teuvo Kohonen. *Self-organizing maps*, volume 30 of *Springer Series in Information Sciences*. Springer-Verlag, Berlin, third edition, 2001.

[16] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern classification*. Wiley-Interscience, New York, second edition, 2001.