Quality of Service in Reconfigurable,
Partial Wireless Networks

Nutzerbasierte Zugriffskontrolle für IEEE 802.11 basierte Netzwerke mittels OpenSource Software

Robert Hilbrich
Robert@Hilbri.ch

26. Januar 2008

Abstract

Mit zunehmender Bedeutung drahtloser Vernetzung steigt auch der Bedarf nach sicheren und effizient verwaltbaren Authentifikationsmechanismen. Bisher wurden meist Gerätemerkmale, wie MAC-Adressen oder in der Netzwerkkarte gespeicherte Netzwerkschlüssel, für die Authentifikation verwendet.

Einen wesentlich besseren Ansatz stellt die nutzerbasierte Authentifikation dar. Die Zugriffsverwaltung auf Benutzerebene verwendet standardisierte Protokolle, wie *RADIUS*, *EAP* und *PEAP*, und basiert auf einer generischen Architektur zur Netzwerkauthentifikation: dem Standard IEEE 802.1X.

Durch die Implementation des IEEE 802.1X Standards mittels OpenSource Software ist eine spätere Anpassung der Architektur an besondere Anforderungen oder Einsatzfelder ohne zusätzliche Lizenzkosten möglich.

Inhaltsverzeichnis

1 Einführung **5**

2 Allgemeiner Lösungsansatz **8**
2.1 AAA-Protokolle und das RADIUS Protokoll 8
2.2 EAP 11
2.3 IEEE 802.1X 12
2.4 EAP-TLS 15
2.5 Protected EAP (PEAP) 17
2.6 Fazit 18

3 Realisierung **20**
3.1 Architektur und OpenSource Komponenten 20
3.2 Konfiguration und Installation des RADIUS Servers 22
3.3 Konfiguration von *OpenWRT* auf dem *La Fonera*-Router 23
3.4 Konfiguration von *hostapd* 25

4 Anwenderperspektive **27**
4.1 Erfolgreicher Login 27
4.2 Nicht erfolgreicher Login 29
4.3 Verlassen des Empfangsbereichs 30

5 Zusammenfassung **31**

A Listings der *FreeRadius* Konfigurationsdateien **33**

B Listings der *OpenWRT*-Konfiguration **35**

C Listings der *hostapd*-Konfiguration **36**

Literatur **38**

Abbildungsverzeichnis

1 Einsatz von Frontend- und AAA-Protokollen 9
2 RADIUS Komponenten bei der Modem-Einwahl 10
3 Kombination aus RADIUS und EAP bei der Einwahl 12
4 Open und Closed-Ports bei IEEE 802.1X 13
5 IEEE 802.1X - Architektur 14
6 Kombination aus RADIUS, EAP und PEAP bei der Einwahl 18
7 IEEE 802.1X Umgebung mittels OpenSource Komponenten 20
8 *La Fonera*-Router 24
9 Dynamische Erstellung der HOSTAPD-Konfiguration 26

Listings

1 /etc/freeradius/clients.conf 33
2 /etc/freeradius/users.conf 33
3 /etc/freeradius/eap.conf 33
4 /etc/config/network 35
5 /etc/config/wireless 35
6 /lib/wifi/hostapd.sh 36
7 /etc/config/slam-wireless-config 37

1 Einführung

In Zeiten von *Pervasive Computing* und *Ubiquitous Computing* hat die Vernetzung der Teilnehmer durch drahtlose Netzwerke immens an Bedeutung gewonnen. Mit der neuen Popularität dieser Netzwerke steigt gleichzeitig aber auch der Bedarf an wirksamen Sicherungsmechanismen und Zugriffskontrollen für diese verteilten und vernetzten Ressourcen.

Die bisher häufig genutzten Mechanismen zur Umsetzung von Zugriffskontrollen für ein drahtloses Netzwerk basieren allerdings auf bestimmten Merkmalen von Geräten. So nutzt ein MAC-Filter im Access-Point[1] die weltweit eindeutige MAC-Adresse auf der Netzwerkkarte des Anwenders, um diesem den Zugriff auf das Netzwerk zu gestatten oder zu verwehren. Diese Entscheidung wird aber unabhängig davon getroffen, welche Person dieses Gerät verwendet.

Etwas flexibler ist die Möglichkeit, die Funknetzwerkkarte mit kryptographischen Schlüsseln auszustatten. Nur mit dem richtigen Schlüssel ist es nun einem Netzwerkgerät möglich, mit dem Access-Point zu kommunizieren und damit auf das Funknetzwerk zuzugreifen. Die erhöhte Flexibilität resultiert aus der Möglichkeit, unterschiedlich berechtigte Personengruppen mit unterschiedlichen Schlüsseln auszustatten.

In der Regel wird bei diesem Verfahren der gleiche Schlüssel zwischen allen Netzwerkbenutzern einer Berechtigungsgruppe verwendet. Dies führt jedoch bei einer hohen Gruppenanzahl oder einer hohen Fluktuation der Gruppenmitglieder zur einem beträchtlichen administrativen Aufwand. Sobald ein Teilnehmer das Zugriffsrecht entzogen werden soll, müssen alle anderen Teilnehmer einen neuen Schlüssel vereinbaren.

Die Nachteile dieser Herangehensweisen resultieren primär aus der Zuordnung der Zugriffskontrolle zu Netzwerkgeräten und anstelle von deren Benutzern. So kann zum Beispiel ein Laptop auch nicht flexibel von zwei Personen mit unterschiedlichen Zugriffsrech-

[1] Ein Access-Point ist die zentrale Infrastrukturkomponente in einem drahtlosen Netzwerk auf IEEE 802.11 Basis.

ten verwendet werden. Wird gar ein Gerät gestohlen, so erhält der Dieb auch gleichzeitig Zugriff auf das Netzwerk bis der Diebstahl bemerkt wird.

Eine wesentlich elegantere Lösung stellt die Zuordnung der Befugnisse zu den Benutzern anstelle von deren Geräten dar. Unabhängig vom konkreten Netzwerkgerät, z.B. PDA, Laptop oder SmartPhone, wird der Zugriff dann anwenderbezogen gewährt. Der Administrationsaufwand zur Pflege der Daten, welcher Benutzer welche Geräte verwendet, entfällt damit völlig.

Zudem wäre es ebenfalls möglich, eine SmartCard mit einem persönlichen Schlüssel zu verwenden oder die Authentifikation an biometrische Merkmale zu koppeln, um den Zugang weiter abzusichern.

Neben den Vorteilen, die diese Technik der Zugriffskontrolle für den Anwender und Administrator mit sich bringt, gibt es auch Anwendungsgebiete, die auf eine anwenderbezogene Authentifikation angewiesen sind. Eines dieser Gebiete ist die Umsetzung von *Quality of Service (QoS)* für drahtlose Netzwerke.

So könnte ein Internet Café unterschiedlich priorisierte Internetanbindungen für seine Kunden zur Verfügung stellen. Ein Kunde, der für eine hoch-priorisierte Anbindung bezahlt hat, soll diese unabhängig von seinem technischen Zugang, also per Laptop, PDA oder per stationärem PC, erhalten. Daher muß die Dienstgüteklasse direkt an die Authentifikation am Netzwerk gekoppelt sein, um den Aufwand auf Seiten des Betreibers zu minimieren.

In dieser Arbeit soll dazu eine standardisierte Netzwerkarchitektur vorgestellt und deren Umsetzung mit OpenSource Software gezeigt werden. Der besondere Fokus liegt hier auf OpenSource, denn dieses Lizenzkonzept gibt dem Betreiber dieser Netzwerkumgebung zusätzlich noch größtmögliche Flexibilität für Anpassungen an konkrete, kontextspezifische Anforderungen ohne zusätzliche Lizenzkosten.

Allgemeine Anforderungen für diese Form der Authentifikation können durch die Analyse des oben umrissenen Anwendungsfalls abgeleitet werden:

1. Ein Benutzer soll sich durch Eingabe eines Logins und eines Passwortes am Netzwerk anmelden können. Allein die Kombination aus Loginname und Passwort entscheidet dabei über die Zugriffsbefugnis.
2. Die Übertragung der Authentifizierungsinformationen (Login und Passwort) erfolgt verschlüsselt und kann nicht durch andere Netzwerkteilnehmer mitgelesen oder abgeleitet werden.
3. Alle Benutzerkonten sollen zentral verwaltet werden, um den Administrationsaufwand zu minimieren. Dabei ist der Einsatz von standardisierten Protokollen zu beachten, um bei unterschiedlichen Geräteherstellern die Interoperabilität zu gewährleisten.
4. Nach der erfolgreichen Authentifikation muß der weitere Netzwerkverkehr des Benutzers auf der Funkstrecke ebenfalls durch eine Verschlüsselung vor dem unerwünschten Abhören durch andere Funknetzwerkteilnehmer geschützt werden.
5. Zur Umsetzung von *Quality of Service* ist es ebenfalls erforderlich, den Benutzerkonten Attribute, wie z.B. die Serviceklasse, zuordnen zu können.

Dazu wird zunächst in Kapitel 2 der prinzipielle Lösungsansatz und die verwendete Protokollhierarchie eingeführt. In Kapitel 3 folgt dann die Beschreibung der Umsetzung des Lösungsansatzes mittels OpenSource Software.

Nachdem Kapitel 2 und 3 die technischen Grundlagen dargestellt haben, wechselt Kapitel 4 auf die Seite des Anwenders und beschreibt wie ein Benutzer mit einem OpenSource Betriebssystem, wie z.B. Ubuntu Linux, intuitiv und mit wenig Aufwand Zugriff auf das Netzwerk erhalten kann.

Kapitel 5 fasst noch einmal die wesentlichen Schritte und Komponenten zusammen und prüft die Umsetzung aller oben gestellten Anforderungen.

2 Allgemeiner Lösungsansatz

In diesem Abschnitt sollen die Grundlagen für die verwendeten Standards und Technologien vorgestellt werden. Das Ziel ist die Präsentation der prinzipiellen Herangehensweise und die Vorstellung der beteiligten Komponenten des Protokollstandards *IEEE 802.1X*, auf dem der vorgestellte Lösungsansatz basiert.

Im Folgenden werden dazu die wesentlichen Entwicklungsschritte und Bestandteile der Protokollumgebung präsentiert, damit der vorliegende Ansatz zur Realisierung von nutzerbasierter Zugriffskontrolle für drahtlose Netze richtig bewertet und sinnvoll eingesetzt werden kann.

2.1 AAA-Protokolle und das RADIUS Protokoll

Mit der Bedeutung von Netzwerken und dem Entstehen von verteilten, gemeinsam genutzten Resourcen, gewann auch deren Absicherung gegen unerwünschte Zugriffe an Wichtigkeit. Obwohl die verschiedenen Netzwerktechnologien sowohl topologisch, aber auch technologisch unterschiedlich sind, existiert eine allgemeine Unterteilung der für eine Netzwerkauthentifikation verwendeten Protokolle gemäß ihrer Aufgaben.

Der Ablauf der Authentifzierung eines Benutzers wird von zwei Klassen von Protokollen umgesetzt: *Frontend-Protokolle* und *Backend-Protokolle* (vgl. [BMB+05]). Die Frontend-Protokolle bezeichnen dabei alle Protokolle, die es dem Benutzer ermöglichen, mit dem Netzwerk zu kommunizieren, um seine Authentizität nachzuweisen.

Die Anzahl der verschiedenen Frontend-Protokolle führt jedoch zu einem hohen administrativen Aufwand, denn *jede* Gegenstelle eines dieser Frontend-Protokolle müßte die gesamten Authentifizierungsdaten zur Prüfung der Anmeldung bereit halten. Dies ist zum einen sehr wartungsintensiv stellt aber gleichzeitig auch ein hohes Sicherheitsrisiko dar. So genügt die Kompromittierung einer Gegenstelle, um den Zugang zu allen Authentifikationsdaten zu erhalten.

Eine Lösung bietet hier ein zentraler Authentifikationsserver, der mittels *eines* Backend-Protokolles die eigentliche Prüfung der Anmeldung durchführen kann. Neben den Authentifizierungsinformationen können auch Autorisierungs- und Accountingdaten auf dem zentralen Server gespeichert und über das Backend-Protokoll abgefragt werden. Aufgrund dieser drei Aufgabenbereiche (Authentifikation, Autorisation und Accounting) werden die Backend-Protokolle allgemein auch *AAA-Protokolle* genannt.

Ein Beispiel für die Anwendung beider Protokollklassen ist in Abbildung 1 dargestellt. Zwei Benutzer wollen hier Zugriff auf ein Netzwerk erhalten. Obwohl beide Benutzer dafür unterschiedliche Protokolle nutzen, wird nur ein Protokoll für die Kommunikation mit dem zentralen Authentifikationsserver (AAA-Server) verwendet. Bei erfolgreicher Authentifikation wird den Benutzern der Zugriff auf das Netzwerk gewährt.

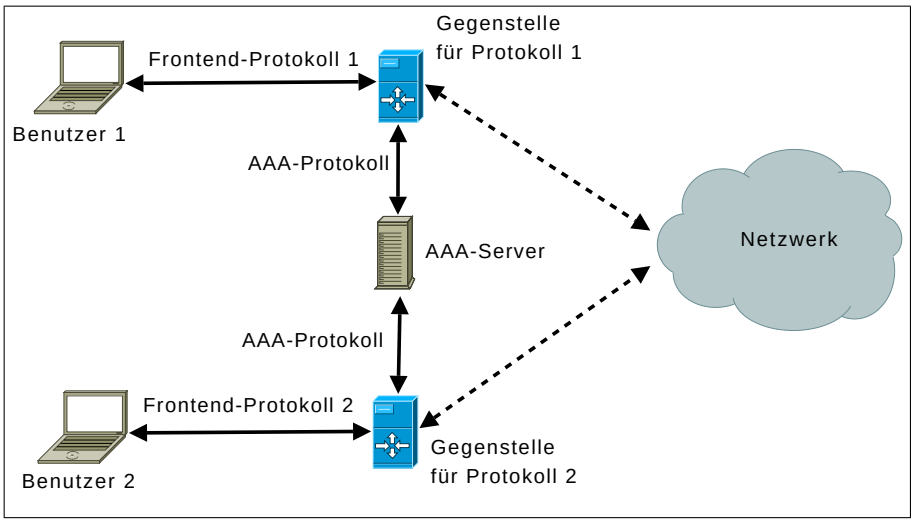


Abbildung 1: Einsatz von Frontend- und AAA-Protokollen

Die allgemeinen AAA-Konzepte wurden von der Internet Engineering Task Force (IETF) durch die Beschreibung einer generischen *AAA-Architektur* dokumentiert (vgl. [LGG+00]).

Eines der meist benutzten AAA-Protokolle heißt *RADIUS* und wurde ursprünglich für Einwählzugänge ins Internet mittels eines Modems konzipiert. Erstmals definiert wurde

der *Remote Authentication Dial In User Service (RADIUS)* von der IETF im Januar 1997 (vgl. [RRSW97]). Der aktuelle RADIUS Standard ist im RFC[2] 2865 ([RWRS00]) festgehalten.

Die Architektur des RADIUS-Protokolls entspricht einer klassischen Client-Server-Architektur. Dabei tritt die Protokoll-Gegenstelle des Benutzers, auch als *Network Access Server (NAS)* bezeichnet, nun als Client gegenüber dem RADIUS Server auf.

Im NAS werden dazu die Informationen aus dem zugangsabhängigen Frontend-Protokoll in das RADIUS-Protokoll übertragen und zur Prüfung an den RADIUS Server weitergeleitet. Der Rückweg einer Bestätigungs- oder Ablehnungsentscheidung wird dann analog zunächst per RADIUS Protokoll an den NAS übermittelt. Dieser wandelt die RADIUS Nachricht dann in eine Frontend-Protokoll spezifische Bestätigung oder Ablehnung um.

Ein Beispiel für die beteiligten RADIUS-Komponenten bei einer Modem-Einwahl mittels PPP ist in Abbildung 2 dargestellt.

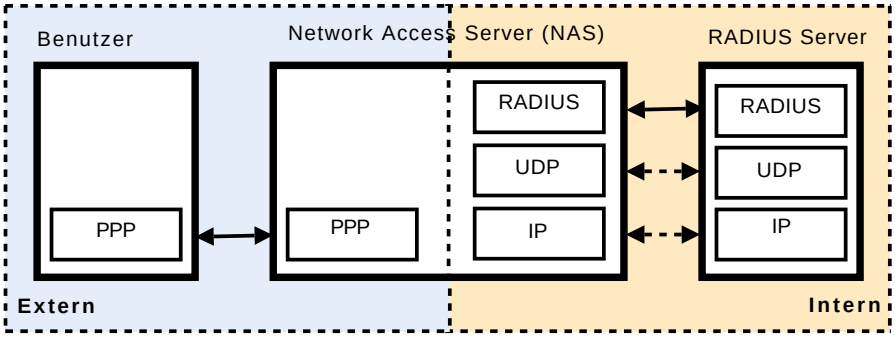


Abbildung 2: RADIUS Komponenten bei der Modem-Einwahl

[2]RFC steht für Request for Comments und bezeichnet ein Dokument, das einen (Internet-) Standard enthält.

2.2 EAP

Beim *Extensible Authentication Protocol (EAP)* handelt es sich um ein generisches Protokoll zur Authentifizierung ([ABV+04]) auf der Ebene der Netzzugangsschicht, also OSI Schicht [3] Zwei. Durch EAP wird ein Rahmenwerk bereitgestellt, innerhalb dessen ein konkretes Authentifzierungsprotokoll zwischen den Beteiligten ausgehandelt werden kann. Als konkrete Authentifizierungsprotokolle können neben der Prüfung der Identität durch eine Login und Passwortabfrage, auch andere Protokolle, wie z.B. eine *Generic-Token Card*, zum Einsatz kommen (vgl. [ABV+04]).

Die verschiedenen Mechanismen werden dabei in *EAP-Rahmen*[4] eingebettet und zwischen den Kommunikationspartnern ausgetauscht. Dabei ist kein Protokoll der Vermittlungsschicht, wie zum Beispiel das Internet Protokoll (IP), notwendig. EAP Rahmen können direkt innerhalb von Protokollen der Sicherungsschicht, also zum Beispiel innerhalb von *PPP*-Rahmen oder *IEEE 802.3*[5]-Rahmen, gesendet werden (vgl. [ABV+04]).

Analog zu den Überlegungen bei der Entwicklung des RADIUS Protokolls weiter oben, erfordert die Skalierbarkeit von EAP auf viele verschiedene Einwählsysteme die Verwendung eines zentralen Authentifikationsservers. Die Komplexität des Einwahlsystems wird damit reduziert auf die Fähigkeit EAP für die Kommunikation mit den einwählenden Benutzern und RADIUS für die Kommunikation mit einem zentralen Authentifikationsserver zu beherrschen.

Dieser Ansatz erfordert ein Umsetzen der Rahmen von EAP in RADIUS und umgekehrt. In [RWC00] wird dazu eine Erweitung zum RADIUS Protokoll definiert, die es erlaubt, EAP Rahmen direkt in RADIUS Rahmen einzubetten. Durch diese Erweite-

[3] Das Open Systems Interconnection Reference Model (OSI Modell) ist eine Designgrundlage von Kommunikationsprotokollen. (`http://www.iso.org`)

[4] Ein Rahmen bezeichnet ein Netzwerkpaket, das die Daten einer höheren OSI-Schicht kapselt und um weitere Informationen, wie zum Beispiel Prüfsummen, ergänzt.

[5] Das Institute of Electrical and Electronics Engineers (IEEE) ist ein weltweiter Berufsverband von Ingenieuren. Das *IEEE 802* ist ein Projekt des IEEE, das im Februar 1980 begann - daher die Bezeichnung *802* - und sich mit Standards im Bereich der lokalen Netze beschäftigt. Die Untergruppe 802.3 beschäftigt sich mit einem besonderen Standard lokaler Netze, dem *Ethernet*.

rung können sich die anmeldenden Benutzer direkt an einem Authentifikationsserver mittels EAP anmelden. Die verwendeten Frontend- und Backend-Protokolle sind dabei aus Benutzersicht völlig transparent. Abbildung 3 erweitert Abbildung 2 um EAP und soll die Möglichkeit zur direkten Authentifizierung am RADIUS Server illustrieren.

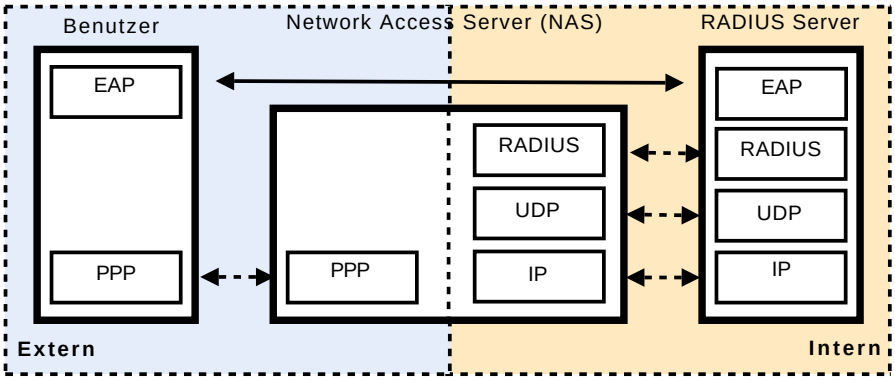


Abbildung 3: Kombination aus RADIUS und EAP bei der Einwahl

2.3 IEEE 802.1X

Die bisher beschriebenen Protokolle betrachten primär den Anwendungsfall einer Einwahl in ein Netzwerk, wie zum Beispiel die Einwahl in das Internet über einen Provider mit einem Modem. Der *IEEE 802.1X Standard* ([IEE01]) behandelt im Gegensatz dazu die Zugangskontrolle zu Ethernet-basierten Netzwerken[6] auf der OSI-Schicht Zwei, basiert aber ebenfalls auf EAP und RADIUS.

Obwohl drahtlose Netzwerke nicht Ethernet-basiert sind, sondern auf dem Standard IEEE 802.11 beruhen, sind die Anforderungen für diese Form der Zugriffskontrolle so allgemein gehalten, dass sich der IEEE 802.1X Standard auch hier zur Authentifikation verwenden läßt.

[6]Der Begriff Ethernet bezieht sich auf alle Netzwerke, die dem IEEE 802.3 Standard folgen.

Der IEEE 802.1X Standard realisiert eine *Port*-basierte Netzwerkzugriffskontrolle und erfordert daher die Existenz eines *Ports* im Einsatzumfeld. Ein Port ist als abstrakter Anschlußpunkt definiert, an dem ein Nutzer Zugriff auf das Netzwerk erhalten kann. In drahtgebundenen Netzwerken ist damit ein Anschluß eines Switching-Hubs[7] gemeint, in drahtlosen Netzen stellt der Access Point die Funktionalität eines Ports bereit.

Jeder Port ist laut Standard zunächst im *closed* bzw. *nicht-authorisierten* Zustand. Damit kann kein Zugriff auf das Netzwerk über diesen Port erfolgen, obwohl die physikalische Verbindung auf OSI-Schicht Eins hergestellt wurde. Erst nach einer erfolgreichen Authentifizierung des Nutzers, ändert sich der Zustand des Ports auf *open* bzw. *authorisiert*, so dass der Zugriff auf das Netzwerk durch den Port erfolgen kann.

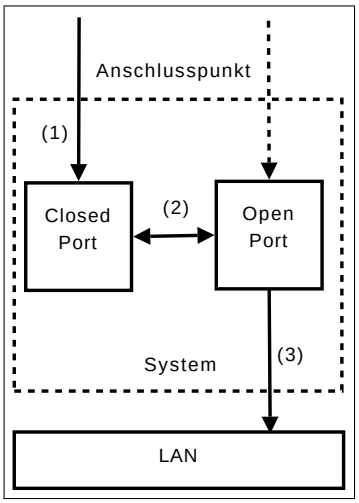


Abbildung 4: Open und Closed-Ports bei IEEE 802.1X

Der Ablauf ist ebenfalls in Abbildung 4 dargestellt. Im ersten Schritt erreicht die Verbindungsanfrage den Port zunächst im nicht-authorisierten Zustand und ermöglicht keinen Zugriff auf das LAN. Nach der erfolgreichen Authentifikation des Benutzers symbolisiert der zweite Schritt den internen Zustandswechsel des Ports. Durch den Wechsel in den authorisierten Zustand kann nun im dritten Schritt der Zugriff auf das LAN erfolgen.

[7] Ein Switching-Hub ist eine zentrale Netzwerkkomponente bei der Verkabelung von IEEE 802.3 Netzwerken in Stern-Topologie.

Für die Authentifzierung eines Benutzers an einem Port wird EAP verwendet. Allerdings werden die EAP-Nachrichten direkt in Ethernet-Rahmen eingebettet[8]. Damit die Komplexität des Ports möglichst gering ist, fungiert dieser lediglich als transparenter Proxy zwischen dem Benutzer und dem Authentifzierungsserver. Dazu müssen alle EAP Nachrichten aus den Ethernet-Rahmen entnommen und in RADIUS-Nachrichten eingebettet werden. Der Rückweg verläuft analog.

Aufgrund des ähnlichen Ablaufs ist die Architektur von IEEE 802.1X Umgebungen auch vergleichbar mit der Abbildung 3 aus dem Abschnitt *EAP*. Allerdings sind die Bezeichnungen für die beteiligten Komponenten unterschiedlich. So definiert der Standard die folgendenen drei Komponententypen (vgl. Abbildung 5): *Supplicant* (Englisch für Bittsteller), *Authenticator* und *Authentication Server*.

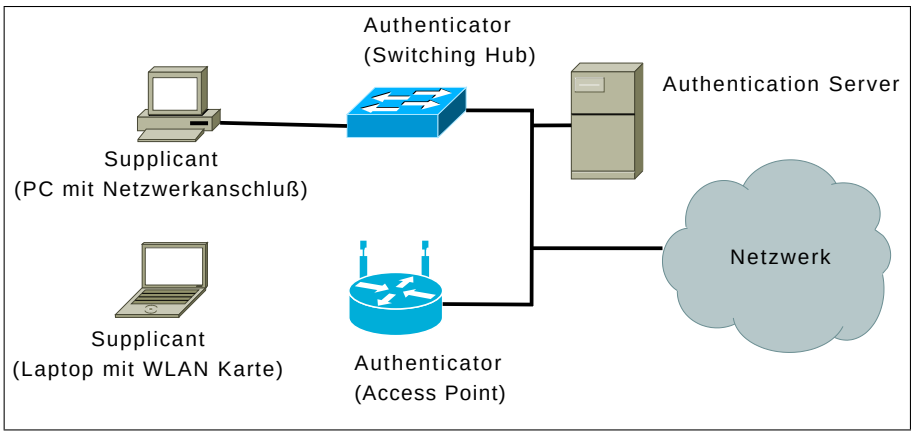


Abbildung 5: IEEE 802.1X - Architektur

Der IEEE 802.1X Standard definiert einen *Supplicant* als Entität auf der einen Seite einer Punkt-zu-Punkt-Verbindung zu einem Netzwerk, die durch den *Authenticator* auf der anderen Seite der Punkt-zu-Punkt-Verbindung authentifziert wird [IEE01]. Es handelt sich beim Supplicant also um den Benutzer aus Abbildung 3.

Die Rolle des *Authenticators* kann durch einen Access Point oder einen Switching-Hub übernommen werden. In Anlehnung an den Standard ist es die Aufgabe des Authenticators die Authentifikation des Supplicants zu vereinfachen. Der Authenticator übernimmt

[8] Für diese Einbettung gibt es einen speziellen Standard: EAP over LANs (*EAPoL*). Siehe dazu auch [Sch03]

daher die Rolle eines Proxies, der die Authentifikationsinformationen zwischen dem Supplicant und dem Authentication Server übermittelt. In Abbildung 3 wurde der Authenticator als Network Access Server (NAS) bezeichnet.

Der Authentication Server führt nun die typischen AAA-Aufgaben, also Authentifizierung, Autorisierung und Accounting, durch. Der Standard selbst läßt die konkrete Implementierung des AAA-Servers allerdings offen, so dass nebem RADIUS auch andere Protokolle, wie zum Beispiel *DIAMETER*, verwendet werden können.

2.4 EAP-TLS

Für drahtgebundene Netzwerke hat sich die IEEE 802.1X-Authentifzierung zum Quasi-Standard der Authentifizierung auf OSI-Schicht Zwei etabliert. Die Anwendung in drahtlosen Netzwerken beinhaltet aber Sicherheitslücken, die Erweiterungen zwingend notwendig machen.

Der kritische Punkt ist dabei die Übertragungsstrecke vom Supplicant zum Authenticator. Im Gegensatz zur Verbindung zwischen Authenticator und Authentication Server, die meist als vertrauenswürdig eingestuft wird, ist diese Strecke nicht vertrauenswürdig und öffentlich. Da EAP allerdings keine Mechanismen zur Verschlüsselung der Übertragung definiert, werden alle EAP-Nachrichten, also auch die Authentifizierungsdaten, im Klartext über diesen nicht vertrauenswürdigen Abschnitt übermittelt.

Bei drahtgebundenen Netzwerken spielt diese Unterscheidung eine untergeordnete Rolle, den zum Abhören der Kommunikation und zum Durchführen von *Man-in-the-Middle*-Angriffen ist hier ein physischer Zugang zu den Netzwerkkabeln notwendig. Meist läßt sich dieser aber leicht einschränken oder unterbinden bzw. ist mit großem Aufwand verbunden.

Im Gegensatz dazu, ist der Aufwand zum Abhören dieser Kommunikationsstrecke bei drahtlosen Netzwerken sehr gering. Drahtlose Netzwerke lassen sich in ihrer Reichweite

nur schwer beschränken, so dass sich der Bereich, in dem ein Abhören der Übertragung möglich ist, auch schwer abschätzen läßt.

Die Hersteller von Hardware für IEEE 802.11-Netzwerke geben typischerweise Übertragungsreichweiten in der Größenordnung von 500 m an. Erfahrungen zeigen, dass innerhalb von Gebäuden eine Entfernung von 80 bis 100 m eine realistische Empfangsreichweite darstellt (vgl. [BMB+05]). Daher ist es unbedingt notwendig, die nicht vertrauenswürdige Übertragungsstrecke zwischen Supplicant und Authenticator gegen unerwünschtes Abhören der Kommunikation abzusichern.

Eines der ersten Verfahren, das EAP um Authentifikation des Authentifikationsservers, Verschlüsselung der Kommunikation[9] und die dynamische Erzeugung von Schlüsselmaterial erweitert hat, ist *EAP-TLS* (vgl. [BMB+05] und [AS99]).

Bei diesem Verfahren verwendet der Authentifikationsdialog zur Absicherung das *Transport Layer Security*-Protokoll (TLS), welches in bzw. oberhalb der Transportschicht agiert. Das TLS-Protokoll wurde 1999 im RFC 2246 ([DA99]) in der Version 1.0 verabschiedet und unterstützt verschiedene kryptographische Algorithmen und Protokolle, u.a. auch die Authentifikation und den Schlüsseltausch (vgl. [Sch03]).

TLS soll laut Standard über ein zuverlässiges Transportprotokoll betrieben werden. In diesem Fall übernimmt EAP die für ein zuverlässiges Transportprotokoll notwendige Übertragungswiederholung bzw. Fragmentierung der Pakete (siehe auch [BMB+05]).

Der EAP-TLS-Standard erfüllt damit die Bedingung einer Authentifikation ohne das explizite Übertragen der Anmeldeinformationen durch Nutzung des TLS-Kryptographie-Frameworks. Es findet allerdings keine Verschlüsselung bei der Authentifikation selbst statt und damit sind die EAP-Identitäten der Clients und des Authentifikationsservers nicht gegen ein Abhören geschützt.

[9] Die Verschlüsselung erfolgt erst nach erfolgter Authentifzierung, also nach der Bekanntgabe der Identitäten der Kommunikationspartner im Klartext.

Zur Authentifizierung der Kommunikationspartner nutzt EAP-TLS sogenannte *X.509-Zertifikate*[10] um eine sichere Zugangskontrolle zu gewährleisten.

Für diese Arbeit ist der Aufbau einer kompletten Infrastruktur zur Verwaltung von Zertifikaten zu komplex und führt zu keiner qualitativen Verbesserung der Sicherheit, so dass die einfachere Methode zur Zugangskontrolle mit einer Kombination aus Login und Passwort bevorzugt wird.

2.5 Protected EAP (PEAP)

Unter dem Namen *Protected EAP (PEAP)* wurde in [PSS+04] von den Firmen Microsoft und Cisco ein Verfahren vorgestellt, das die oben angesprochenen Probleme von *EAP-TLS* löst, indem der Austausch *aller* EAP-Nachrichten *verschlüsselt* und *authentifiziert* abläuft.

Um dies zu realisieren beinhaltet das PEAP Protokoll zwei Phasen. So wird zunächst mittels TLS ein gesicherter Kommunikationskanal zwischen dem Client und dem PEAP-fähigen Server aufgebaut. Der Protokollablauf entspricht dem Ablauf von EAP-TLS, allerdings ist die Client-Authentizierung in dieser Phase *optional*, so dass auf ein Client-Zertifikat verzichtet werden kann (vgl. [BMB+05]).

In der zweiten Phase findet dann der Austausch der EAP-Nachrichten über diesen verschlüsselten und authentifizierten Kanal statt. Einem Angreifer ist es damit nicht mehr möglich die Identität des Benutzers, der sich gerade authentifiziert, zu ermitteln.

Die für eine IEEE 802.1X-PEAP-basierte Authentifikation benötigte Protokollhierarchie ist in Abbildung 6 dargestellt. Hier kann sich der Client entweder mittels PPP einwählen oder die EAP Nachrichten mittels *EAP over LAN (EAPoL)* direkt in die IEEE 802.3-Rahmen einbetten.

[10] Durch ein digitales Zertifikat können Nutzer eines asymmetrischen Kryptosystems den öffentlichen Schlüssel einer Identität, z. B. einer Person oder einem IT-System, zuordnen und seinen Geltungsbereich bestimmen.

Ein Network Access Server (NAS) fungiert ebenfalls als Proxy, der die EAP Nachrichten in RADIUS Mitteilungen eingebettet und an den PEAP Server weiterleitet. Die oberste Protokollschicht wird nun zusätzlich durch einen verschlüsselten PEAP-Tunnel geschützt.

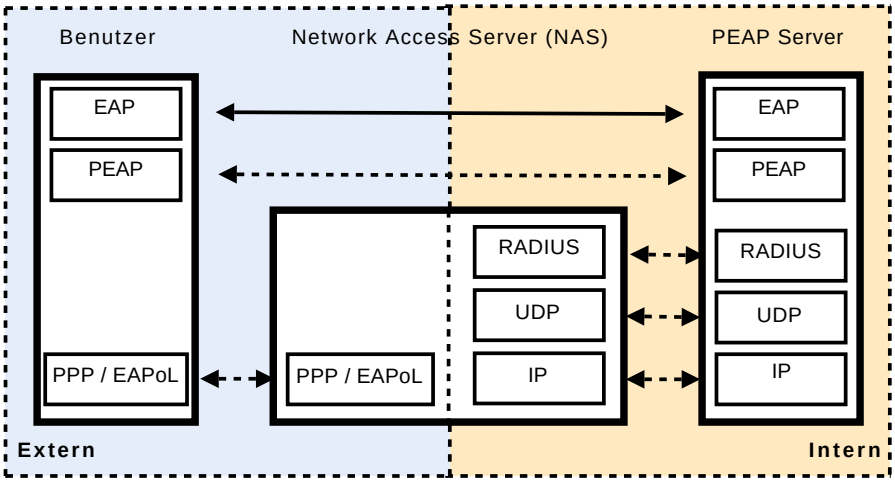


Abbildung 6: Kombination aus RADIUS, EAP und PEAP bei der Einwahl

Eine technische Alternative zu PEAP ist das *EAP Tunneled TLS Authentication Protocol (EAP-TTLS))*, welches die Unzulänglichkeiten von EAP-TLS, aber auch einige Schwächen von PEAP löst. Die Beschreibung der Alternative EAP-TTLS ist allerdings nicht mehr im Rahmen dieser Arbeit möglich, so dass auf die gängige Literatur, wie z.B. [BMB+05], verwiesen wird.

2.6 Fazit

Der vorgeschlagene Lösungsansatz besteht im Kern aus der Umsetzung einer *IEEE 802.1X Architektur*, um eine Nutzerauthentifikation auf Netzwerkschichtebene zu erreichen. Ein zentraler *RADIUS-Server* vereinfacht den Administrationsaufwand erheblich, da damit ein zentraler Authentifikationsserver mit einem standardisiertem Backend-Protokoll (RADIUS) zur Verfügung steht. Zur Frontend-Kommunikation wird das fle-

xible *EAP* eingesetzt, das verschiedene Formen der Anmeldung, wie z.B. SmartCards, unterstützt.

Da auf die Abhörsicherheit in drahtlosen Netzwerken ebenfalls großen Wert gelegt wird, erlaubt ein verschlüsselter *PEAP*-Tunnel die abhörsichere Authentifikation am RADIUS-Server. Mittels PEAP ist zudem die dynamische Schlüsselerstellung und Schlüsseländerung für eine verschlüsselte und sichere Netzwerkkommunikation nach erfolgter Authentifikation möglich[11].

[11] "[...] Die Authentifizierungsverfahren *EAP-TLS* und *PEAP-MS-CHAP v2* leiten wechselseitig zu ermittelnde Unicast-Verschlüsselungsschlüssel ab. Der Unicast-Verschlüsselungsschlüssel wird in regelmäßigen Abständen, entweder vom drahtlosen Zugriffspunkt, oder vom drahtlosen Windows-Client, geändert. Angriffe mit dem Ziel der Schlüsselaneignung können durch einen häufigen Schlüsselwechsel erschwert werden. [...]" (vgl. `http://www.microsoft.com/germany/technet/datenbank/articles/600207.mspx`)

3 Realisierung

Nachdem die prinzipielle Architektur und die verwendeten Protokolle eingeführt wurden, beschreibt dieses Kapitel die Umsetzung der Lösungsbestandteile mit OpenSource Softwarekomponenten.

3.1 Architektur und OpenSource Komponenten

Abbildung 7 enthält als Übersicht die geplante Umsetzung und die beteiligten Module.

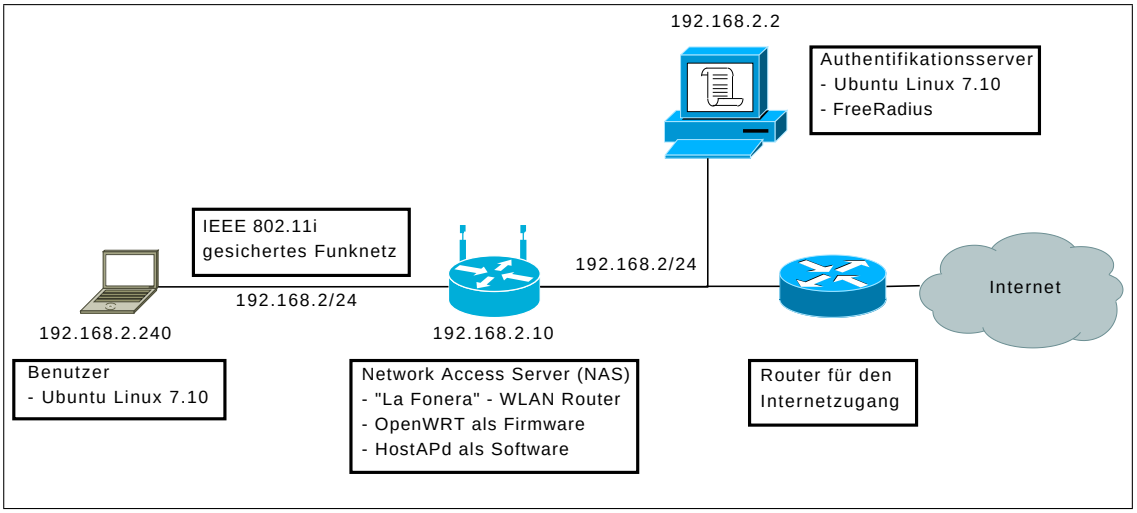


Abbildung 7: Architektur für die Implementation einer IEEE 802.1X Umgebung mittels OpenSource Komponenten

Der *Benutzer*, der sich am Netzwerk anmelden möchte, wird durch einen Laptop mit Ubuntu Linux 7.10[12] realisiert. Ubuntu Linux ist ein Vertreter der OpenSource Betriebssysteme auf Basis des Linux-Kerns und gestattet durch seinen sehr intuitiven Desktop einen einfachen Zugriff auf die Netzwerkanmeldung.

[12] Herunterzuladen unter `http://www.ubuntu.com`

Der *Network Access Server (NAS)*, sowie der *WLAN Access Point* werden durch einen *La Fonera*-Router[13] realisiert. Dabei handelt es sich um einen auf der MIPS-Architektur basierenden Hardware-Router, der sich mit einer OpenSource Firmware verwenden läßt. Diese OpenSource Firmware heißt OpenWRT[14] und stellt eine speziell angepasste Linux-Umgebung bereit.

Durch OpenWRT können nun auch weitere OpenSource-Komponenten aus dem Linux-Umfeld auf dem *La Fonera*-Router installiert und verwendet werden. Die wichtigste Komponente ist das Software-Paket *hostapd*[15]. Dieses Paket liefert zum einen die IEEE 802.11 basierte Access Point-Funktionalität, aber auch die Möglichkeit, eine IEEE 802.1X Einwahlumgebung bereitzustellen.

Der zentrale Authentifikationsserver wird ebenfalls mit dem Betriebssystem Ubuntu Linux 7.10 betrieben. Der Authentifikationsdienst und die Unterstützung des RADIUS-Protokolls wird von der freien Software FreeRadius[16] übernommen. Dieser RADIUS-Server bietet zusätzlich die Unterstützung für PEAP und EAP.

Die folgenden Abschnitte skizzieren kurz die Installation und Konfiguration der Komponenten. Für eine ausführliche Anleitung wird an dieser Stelle auf die Dokumentation der entsprechenden Programme verwiesen. Es wird weiterhin auf eine Beschreibung der Installation des Betriebssystems Ubuntu Linux 7.10 verzichtet, denn auch hier gibt es bereits sehr gute und ausführliche Beschreibungen.

Im Folgenden wird davon ausgegangen, dass Ubuntu Linux sowohl auf dem Laptop des Benutzers und als auch auf dem Authentifikationsserver installiert sind. Es wird ferner vorausgesetzt, dass die entsprechenden Netzwerkverbindungen bestehen und die in Abbildung 7 beschriebene IP-Adressstruktur verwendet wird.

[13] Siehe dazu `http://www.fon.com/de` für mehr Informationen.
[14] Mehr Informationen zu OpenWRT gibt es unter `http://openwrt.org/`.
[15] Dieses Paket findet sich unter `http://hostap.epitest.fi/`.
[16] Siehe `http://www.freeRadius.org` für mehr Informationen

3.2 Konfiguration und Installation des RADIUS Servers

Als erstes wird der zentrale Authentifikationsdienst installiert und konfiguriert. Obwohl Ubuntu Linux bereits über ein vorgefertigtes Softwarepaket für FreeRadius enthält, ist eine manuelle Übersetzung der Quellen notwendig, da verschiedene benötigte Module im vorgefertigten Paket nicht enthalten sind[17].

Nach dem erfolgreichen Kompilieren und Installieren von FreeRadius besteht der nächste Schritt in dessen Konfiguration. Diese befindet sich unter `/etc/freeradius` und kann nur vom Systembenutzer root bearbeitet werden. Die ersten Einträge betreffen die Konfigurationsdatei `clients.conf`, die in Listing 1 abgebildet ist.

In dieser Datei werden die RADIUS-Clients angegeben, die Anfragen an den RADIUS-Server stellen dürfen. Wir definieren hier das gesamte Netzwerk und `localhost` als zulässige Clients. Hier wird ebenfalls ein gemeinsamer Authentifikationsschlüssel für die RADIUS-Kommunikation festgelegt (`secret = radiussecret`).

Nun sind die Benutzerzugänge anzulegen. Für diese Umgebung werden vier Benutzer definiert (`slamuser1-4`), die zusätzlich noch ein Attribut `Class` zugewiesen bekommen. Im QoS-Umfeld könnte dieses Attribut für die Zuordnung der Dienstgüte verwendet werden. Der hinzugefügte Abschnitt der Datei `users` ist in Listing 2 dargestellt.

Als letzter Schritt der Konfiguration von FreeRadius muß noch EAP und PEAP aktiviert werden. Dies wird durch einfaches Entkommentieren der vorgesehenen Konfigurationsoptionen erreicht. Listing 3 enthält die korrekte `eap.conf` für das Scenario. In dieser Datei kann ebenfalls das Server-Zertifikat für die TLS Authentifizierung festgelegt werden. Für die vorliegende Arbeit ist dies nicht notwendig, so dass auf das *Default-Zertifikat* von FreeRadius zurückgegriffen werden kann.

[17] Eine gute Anleitung zum Kompilieren und Paketieren von FreeRadius gibt es unter `http://wiki.freeradius.org/Build#Building˙Debian˙packages`. Dort wird ebenfalls das Installieren der neu erstellten Pakete beschrieben.

Beim anschließenden Start des FreeRadius Servers ist es sinnvoll mittels

```
# freeradius -X
```

zunächst den Debug-Modus zu aktivieren. So lassen sich Konfigurationsfehler frühzeitig erkennen und Probleme diagnostizieren.

Für lokale Tests gibt es zudem das Programm `radtest`. Damit kann eine Authentifikation simuliert werden, um die Konfiguration oder einzelne Nutzerkonten zu überprüfen. Hier ein Beispiel für den Einsatz:

```
$ radtest slamuser1 slam1 192.168.1.2 1812 radiussecret
Sending Access-Request of id 54 to 192.168.1.2 port 1812
    User-Name = "slamuser1"
    User-Password = "slam1"
    NAS-IP-Address = 255.255.255.255
    NAS-Port = 1812
```

3.3 Konfiguration von *OpenWRT* auf dem *La Fonera*-Router

Nachdem der Authentifikationsserver installiert und konfiguriert ist, wird in diesem Abschnitt die Einrichtung des Network Access Servers auf einem *La Fonera* Access Point (siehe auch Abbildung 8) beschrieben.

Dieser Access Point basiert auf einer MIPS Architektur und besitzt eine Atheros WLAN Karte, die sehr gut von Linux unterstützt wird. Der *La Fonera* ist zudem mit 180 Mhz getaktet und verfügt über 16 MB RAM sowie 8 MB Flash-Speicher.

Weil diese Leistungsparameter für normale Desktop-Betriebssysteme viel zu gering sind, wird als Betriebssystem eine speziell angepasste und optimierte Linux Umgebung verwendet: OpenWRT.

Abbildung 8: *La Fonera*-Router

Über die Basisinstallation von OpenWRT auf den *La Fonera*-Routern gibt es sehr viele Anleitungen, so dass diese nicht Bestandteil der vorliegenden Arbeit ist. Für Informationen über die Installation wird daher auf die entsprechenden Webseiten verwiesen, z.B. `http://wiki.freifunk-hannover.de`. Dieser Abschnitt konzentriert sich auf die notwendige Netzwerk- und WLAN-Konfiguration innerhalb von OpenWRT.

Im ersten Schritt muß dazu das Netzwerk- und das WLAN-Interface in den *bridged*-Modus[18] geschaltet werden. Dadurch ist es möglich, *DHCP*[19] Anfragen des Benutzers direkt an einen DHCP Server weiterzuleiten, ohne zusätzliche Dienste, wie einen DHCP Relay Agent, auf dem Router zu installieren. Natürlich müssen ebenfalls alle relevanten IP Adressen konfiguriert werden. Listing 4 enthält dazu die relevante Datei `/etc/config/network`.

Der zweite Schritt beinhaltet die Konfiguration des WLAN-Interfaces. Hier wird u.a. der Funkkanal, der Name des Funknetzes und die Art der Verschlüsselung konfiguriert. Listing 5 enthält die dazu notwendigen Einstellungen.

[18] Der *Bridged*-Modus bezeichnet einen Modus zur Verbindung von zwei Netzwerken, bei der die Verbindung selbst für die Netzwerkteilnehmer in beiden Netzen völlig transparent ist.

[19] DHCP steht für *Dynamic Host Configuration Protocol* und beschreibt ein Netzwerkprotokoll mit dem ein Netzwerkclient seine Netzwerkkonfiguration dynamisch von einem DHCP beziehen kann. Die statische Konfiguration einer IP Adresse und eines Standard-Gateways entfällt damit.

3.4 Konfiguration von *hostapd*

Nachdem die grundlegenden Netzwerk-Einstellungen durchgeführt wurden, muß die Access-Point-Funktionalität für die Verwendung von IEEE 802.1X konfiguriert werden. Das Software-Paket, dass diese Funktionalität bereitstellt heißt HOSTAPD und wird über die Konfigurationsdatei `/var/run/hostapd-ath0.conf` gesteuert.

Bei OPENWRT wird diese Datei dynamisch beim Systemstart aus den Netzwerkinformationen der Dateien `/etc/config/network` und `/etc/config/wireless` generiert. Für einfache Umgebungen funktioniert dies sehr gut, aber die dynamische Einrichtung einer komplexen IEEE 802.1X Architektur ist bei OPENWRT zur Zeit noch nicht realisiert (siehe Listing 6, Zeile 30).

Obwohl einige der benötigten HOSTAPD-Parameter, wie zum Beispiel die IP Adresse des RADIUS Servers, in der Konfigurationsdatei `/etc/config/wireless` eingestellt werden können und auch in der Dokumentation zu OPENWRT enthalten sind, funktionierte die automatische Umsetzung in die HOSTAPD-Konfiguration nicht vollständig bzw. fehlerhaft. Eine automatische Konfiguration nach einem Systemstart ist dennoch wünschenswert, denn nur dann kann der Access Point nach einem Spannungsverlust ohne manuellen Eingriff neu gestartet werden.

Daher ist das OPENWRT Software Paket für HOSTAPD wie folgt modifiziert worden, um eine automatisch generierte Konfiguration zu erreichen. Die dynamische Erstellung der HOSTAPD-Konfigurationsdatei wird in OPENWRT durch das Shell-Skript `/lib/wifi/hostapd.sh` realisiert. Um die dynamische Konfiguration nach dem Systemstart zu erreichen wurde dieses Skript, wie in Listing 6 angegeben, modifziert.

Dabei wird nun die Konfigurationsdatei `/etc/config/slam-wireless-config` (siehe Listing 7) als Ausgangspunkt für alle Einstellungen in der HOSTAPD-Konfigurationsdatei `/var/run/hostapd-ath0.conf` verwendet. Dort wird dann insbesondere die IEEE 802.1X Authentifikation aktiviert und die Anbindung an den RADIUS Server konfiguriert. Einen Überblick über die beteiligten Konfigurationsskripte bei der HOSTAPD-Konfiguration gibt Abbildung 9.

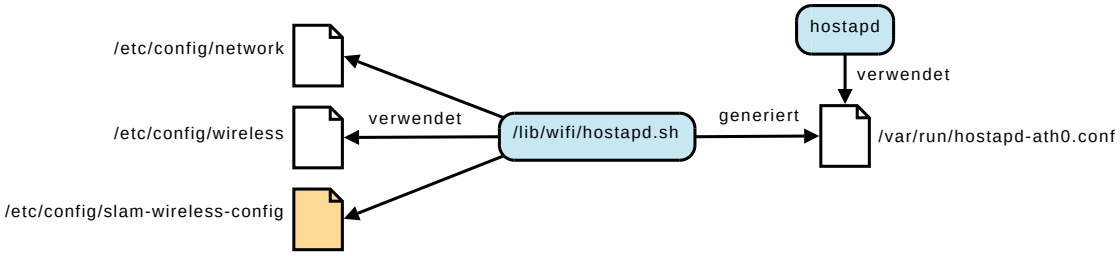


Abbildung 9: Kombination der verschiedenen Konfigurationsskripte zur dynamischen Erstellung einer HOSTAPD-Konfiguration

Zu Testzwecken ist es sinnvoll, das Programm HOSTAPD nicht im Hintergrund, sondern im Vordergrund zu starten. Aktiviert man zusätzlich noch den *Debug-Modus*, so läßt sich das Verhalten des Access Points sehr gut beobachten und diagnostizieren. Dazu wird HOSTAPD durch das Kommando:

```
# hostapd -ddd /var/run/hostapd-ath0.conf
```

aktiviert.

Mit dem abschließenden Start der Komponenten aus Abbildung 7 ist die IEEE 802.1X Umgebung einsatzbereit. Nun können sich die Benutzer drahtlos und sicher authentifzieren und nach erfolgreicher Authentifikation die Internetverbindung nutzen.

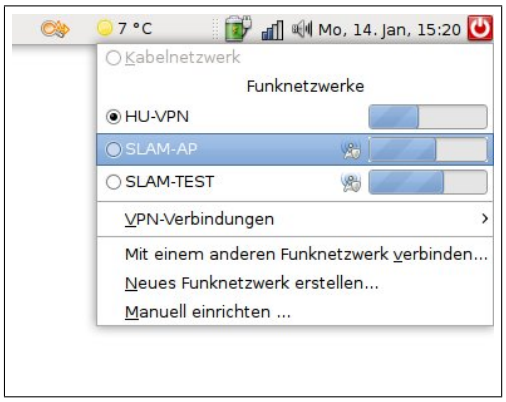
4 Anwenderperspektive

Nachdem der gesamte Backend-Bereich installiert und konfiguriert ist, widmet sich dieses Kapitel der Anwenderperspektive. Es wird gezeigt, wie intuitiv sich die Netzwerkanmeldung für den Benutzer darstellt und wie nahtlos sie sich in den Desktop integrieren läßt. In den folgenden Abschnitten werden dazu die drei primären Anwendungsfälle betrachtet.

4.1 Erfolgreicher Login

Der wichtigste Anwendungsfall für das vorliegende Scenario ist der erfolgreiche Login aus der Nutzerperspektive. Dieser läuft auf einem PC mit Ubuntu Linux 7.10 in mehreren Schritten ab:

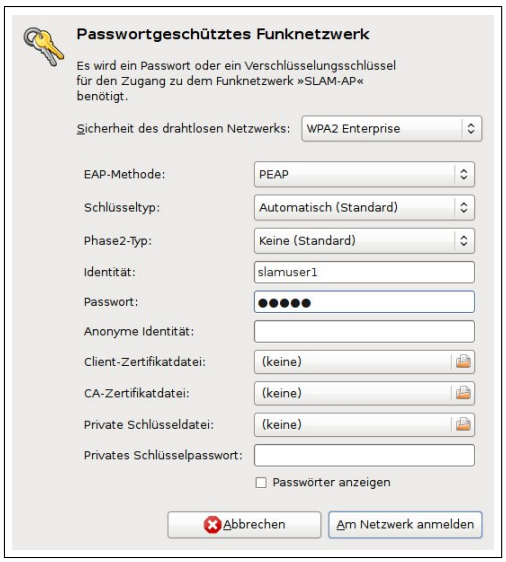
1. Auswählen des richtigen Netzwerkes im `NetworkManager`, in diesem Falle `SLAM-AP`.



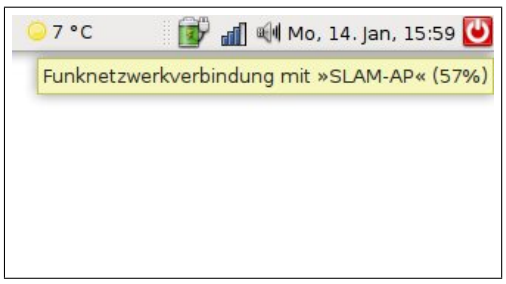
2. Eintragen der Anmeldeinformationen:
   `Identität: slamuser1`
   `Passwort: slam1`

   Die Default-Werte bleiben erhalten:
   `EAP-Methode: PEAP`
   `Schlüsseltyp: automatisch`
   `Phase-2-Typ: Keine`

   Angaben zu Benutzerzertifikaten sind nicht notwendig, da keine Zertifikate im RADIUS Server abgelegt wurden.



3. Die erfolgreiche Verbindung ist am Signalstärke-Applet erkennbar. Eine Informationsbox verrät zusätzlich noch den Namen des Funknetzwerks.



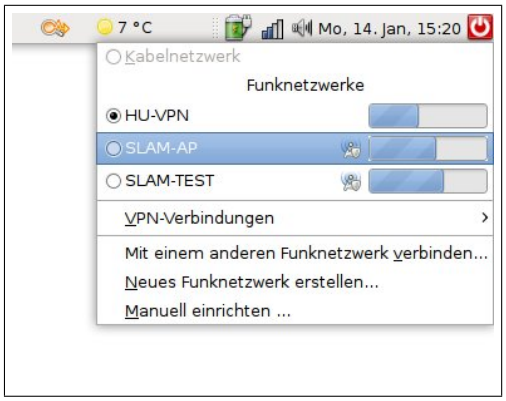
Sind die Anmeldeinformationen einmal erfolgreich verwendet worden, so werden diese abgespeichert und eine zukünftige Anmeldung besteht dann nur noch aus dem ersten Schritt, also dem Selektieren des entsprechenden Netzwerks. Die fehlenden Konfigurationseinstellungen werden dann automatisch eingetragen.

Obwohl die Anmeldung in intuitiven und einfachen Schritten abläuft, so dauert sie aufgrund des komplexen Protokollstapels im Hintergrund länger als klassische Anmeldeverfahren mit einem *Pre-Shared-Key*. So dauert es im vorliegenden Testszenario etwa 20 Sekunden, vom Absenden der Anmeldeinformationen bis zur erfolgreich hergestellten Netzverbindung.

Es wurde nicht getestet, wie sich die Last von vielen Benutzern auf die Anmeldezeit auswirkt. Hier muß noch mehr Forschung investiert werden, um sicherzustellen, dass der *La Fonera*-Router dieser Aufgabe gewachsen ist.

4.2 Nicht erfolgreicher Login

Ein nicht erfolgreicher Login beginnt zunächst analog:

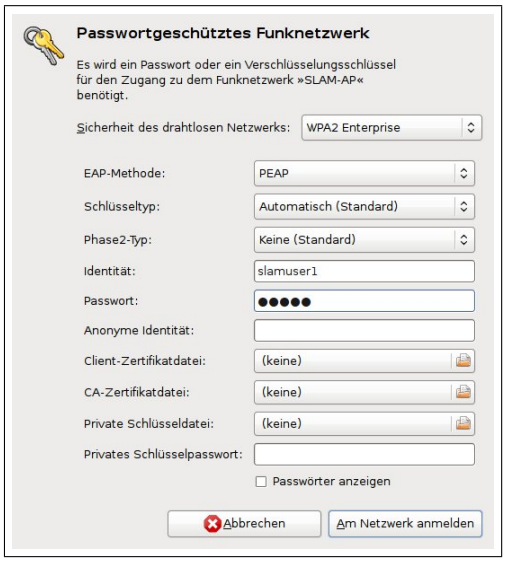


1. Es wird wieder das entsprechende Netzwerk im `NetworkManager` ausgewählt.

2. Man erhält wieder das Formular mit den Anmeldeinformationen. Wird nun ein falsches Passwort eingegeben, z.B.

   ```
   Identität: slamuser1
   Passwort:  slma
   ```

   schlägt die Prüfung des Passwortes fehl. In der Folge wird die gleiche Eingabemaske für die Anmeldeinformationen noch einmal präsentiert. Dieser Prozess kann so lange wiederholt werden, bis entweder das korrekte Passwort übermittelt wurde oder der Benutzer den gesamten Vorgang abgebrochen hat.



Obwohl ein Anwender mit fehlerhaften Login-Daten erfolgreich am Zugriff auf das Netzwerk gehindert wird, so gibt es an dieser Stelle dennoch Raum für Verbesserungen. So ist es aus Nutzersicht intuitiver, wenn der NetworkManager eine Meldung mit einem Hinweis auf die fehlerhafte Login-Passwort-Kombination anzeigt.

4.3 Verlassen des Empfangsbereichs

Das Verlassen des Empfangsbereichs eines Access-Points[20], verläuft ohne Hinweismeldungen. Sobald im NetworkManager ein Timeout überschritten wird, das den fehlenden Empfang von Paketen eines Access Points überprüft, wechselt dieser automatisch in ein anderes, vorkonfiguriertes Netzwerk. Sollte kein weiteres Netzwerk vorhanden sein, so wird die Funkschnittstelle deaktiviert.

[20] genauer gesagt das Verlassen des Abdeckungsbereichs einer ESSID im IEEE 802.11 Standard

5 Zusammenfassung

In dieser Arbeit wurde gezeigt wie eine Netzwerkzugriffskontrolle für drahtlose Netzwerke mit Benutzernamen und Passwörtern als Authentifikationsmerkmale realisiert werden kann. Neben flexibleren Nutzungsmöglichkeiten und verringertem Administrationsaufwand ist diese Art der Authentifikation vor allem im *Quality of Service* - Bereich zur Umsetzung von benutzerspezifischen Servicekategorien notwendig. Der Einsatz von OpenSource Software bietet zudem den Vorteil der leichten Anpassung an spezielle Anforderungen ohne zusätzliche Lizenzkosten.

Die beschriebene Technik basiert auf einem generischen und standardisierten Verfahren zur Netzwerkauthentifikation: dem IEEE 802.1X Standard. Dieser beinhaltet die drei Komponententypen: *Supplicant*, *Authenticator* und *Authentication Server*. Der Supplicant wurde durch einen Laptop mit dem OpenSource Betriebssystem Ubuntu Linux, der Authenticator durch einen *La Fonera* Access Point mit einer speziellen Linux-Umgebung und der Authentication Server durch einen PC mit der Software FreeRadius umgesetzt.

Der eigentliche Protokollkern dieser Umgebung besteht aus dem *Extensible Authentication Protocol (EAP)*, das für die Abwicklung der Authentifikation verantwortlich ist. Zusätzlich fungiert *RADIUS* als Transportprotokoll für die EAP Nachrichten zwischen dem Authenticator und dem Authentication Server. Das zusätzliche Tunnelprotokoll *PEAP* verschlüsselt die Kommunikation zwischen Supplicant und Authenticator durch einen TLS-Tunnel und verhindert so ein *Abhören* der Anmeldung.

Der vorgestellte Ansatz beschreibt durch die Nutzung dieser Protokolle einen sicheren Zugang zu einem drahtlosen Netzwerk und erlaubt zudem eine zentrale Administration aller Benutzerkonten. Nach der erfolgreichen Authentifikation wird die gesamte Kommunikation zudem gemäß dem IEEE 802.11i Standard verschlüsselt übertragen und entspricht damit den aktuellen Sicherheitsstandards für Funknetze.

Alle verwendeten Protokolle sind durch die entsprechenden *Request for Comments (RFCs)* standardisiert und veröffentlicht. Da keine proprietären Protokolle eingesetzt werden, ist die herstellerübergreifende Interoperabilität ebenfalls möglich.

Die Verwendung eines RADIUS Servers erlaubt zudem die Verwendung von weiteren, nutzerspezifischen Attributen. So kann ein Nutzeraccount mit wenig Aufwand um die Zuweisung einer Serviceklasse erweitert werden, so dass damit alle Anforderungen aus Kapitel 1 erfüllt sind.

Aus der Anwenderperspektive gestaltet sich der Zugang zum Netzwerk bei Verwendung Betriebssystems Ubuntu Linux sehr intuitiv und komfortabel. Allein die relativ lange Wartezeit von fast 20 Sekunden nach der Eingabe des Passwortes bis zum erfolgreichen Login bietet Raum für Verbesserungen. Eventuell könnte hier die Verwendung von *EAP-TTLS* gegenüber *PEAP* spürbare Verbesserungen durch eine Vereinfachung des Protokollstapels bringen.

A Listings der *FreeRadius* Konfigurationsdateien

Listing 1: /etc/freeradius/clients.conf

```
# clients.conf - client configuration directives

client 127.0.0.1 {
 secret    = radiussecret
 shortname = localhost
 nastype   = other
}
client 192.168.2.0/24 {
 secret    = radiussecret
 shortname = ap
}
```

Listing 2: /etc/freeradius/users.conf

```
"slamuser1"       User-Password == "slam1"
                  class = 1

"slamuser2"       User-Password == "slam2"
                  class = 2

"slamuser3"       User-Password == "slam3"
                  class = 3

"slamuser4"       User-Password == "slam4"
                  class = 4
```

Listing 3: /etc/freeradius/eap.conf

```
eap {
        default_eap_type = peap
        timer_expire     = 60
        ignore_unknown_eap_types = no
        cisco_accounting_username_bug = no
        md5 {}
        leap {}
        gtc {
                auth_type = PAP
        }
        tls {
                private_key_password = whatever
                private_key_file = ${raddbdir}/certs/cert-srv.pem
                certificate_file = ${raddbdir}/certs/cert-srv.pem
```

```
15                  CA_file = ${raddbdir}/certs/demoCA/cacert.pem
16                  dh_file = ${raddbdir}/certs/dh
17                  random_file = ${raddbdir}/certs/random
18          }
19          peap {
20                  default_eap_type = mschapv2
21          }
22          mschapv2 {}
23  }
```

B Listings der *OpenWRT*-Konfiguration

Listing 4: /etc/config/network

```
config interface loopback
        option ifname   lo
        option proto    static
        option ipaddr   127.0.0.1
        option netmask  255.0.0.0
config interface lan
        option ifname   eth0 ath0
        option type     bridge
        option proto    static
        option ipaddr   '192.168.2.10'
        option netmask  255.255.255.0
        option dns      '141.20.33.82'
        option gateway  '192.168.2.2'
```

Listing 5: /etc/config/wireless

```
config wifi-device  wifi0
        option type     atheros
        option channel  5
config wifi-iface
        option device   wifi0
        option network  lan
        option mode     ap
        option ssid     OpenWrt
        option encryption wpa2
        option hidden   0
```

C Listings der *hostapd*-Konfiguration

Listing 6: /lib/wifi/hostapd.sh

```
1  hostapd_setup_vif() {
2          local vif="$1"
3          local driver="$2"
4          local hostapd_cfg=
5          case "$enc" in
6                  wpa2*|WPA2*|*PSK2*|*psk2*)
7                          wpa=2
8                          crypto="CCMP"
9                  ;;
10                 *mixed*)
11                         wpa=3
12                         crypto="CCMP TKIP"
13                 ;;
14                 *)
15                         wpa=1
16                         crypto="TKIP"
17                 ;;
18         esac
19         case "$enc" in
20                 *tkip+aes|*TKIP+AES|*tkip+ccmp|*TKIP+CCMP) crypto="
                       CCMP TKIP";;
21                 *tkip|*TKIP) crypto="TKIP";;
22                 *aes|*AES|*ccmp|*CCMP) crypto="CCMP";;
23         esac
24         case "$enc" in
25                 *psk*|*PSK*)
26                         config_get psk "$vif" key
27                         append hostapd_cfg "wpa_passphrase=$psk" "$N"
28                 ;;
29                 *wpa*|*WPA*)
30                 # FIXME: add wpa+radius here
31                 ;;
32                 *)
33                         return 0;
34                 ;;
35         esac
36         config_get ifname "$vif" ifname
37         config_get bridge "$vif" bridge
38         config_get ssid "$vif" ssid
39         cat /etc/config/slam-wireless-config > /var/run/hostapd-
               $ifname.conf
40         hostapd -B /var/run/hostapd-$ifname.conf
```

```
41 }
```

Listing 7: /etc/config/slam-wireless-config

```
1  driver=madwifi
2  interface=ath0
3  bridge=br-lan
4  ssid=SLAM-AP
5  ieee8021x=1
6  auth_algs=1
7  eap_server=0
8  eapol_key_index_workaround=1
9  own_ip_addr=192.168.2.10
10 nas_identifier=ap
11 auth_server_addr=192.168.2.2
12 auth_server_port=1812
13 auth_server_shared_secret=radiussecret
14 acct_server_addr=192.168.2.2
15 acct_server_port=1813
16 acct_server_shared_secret=radiussecret
17 wpa=2
18 wpa_key_mgmt=WPA-EAP
19 wpa_pairwise=TKIP
20 wpa_group_rekey=300
21 wpa_gmk_rekey=640
22 ctrl_interface=/var/run/hostapd
23 ctrl_interface_group=0
```

Literatur

[ABV+04] Aboba, B. ; Blunk, L. ; Vollbrecht, J. ; H., J. C. ; Levkowetz, Ed.: *Extensible Authentication Protocol (EAP).* RFC 3748, 2004

[AS99] Aboba, B. ; Simon, D.: *PPP EAP TLS Authentication Protocol.* RFC 2719, 1999

[BMB+05] Bless, Roland ; Mink, Stefan ; Blass, Erik-Oliver ; Conrad, Michael ; Hof, Hans-Joachim ; Kutzner, Kendy ; Schöller, Marcus ; Springer (Hrsg.): *Sichere Netzwerkkommunikation.* Berlin : Springer, 2005. – ISBN 3540218459

[DA99] Dierks, T. ; Allen, C.: *The TLS Protocol Version 1.0.* United States : RFC 2246, 1999

[Gas02] Gast, Matthew S. ; Loukides, Mike (Hrsg.): *802.11 Wireless Networks: The Definitive Guide.* Sebastopol, CA, USA : O'Reilly & Associates, Inc., 2002. – ISBN 0596001835

[Has02] Hassell, Jonathan: *Radius.* Sebastopol, CA, USA : O'Reilly & Associates, Inc., 2002. – ISBN 0596003226

[IEE01] IEEE: *802.1X Port-Based Network Access Control.* `-http://www.ieee802.org/1/pages/802.1x.html˝`. Version: 2001

[LGG+00] Laat, C. de ; Gross, G. ; Gommans, L. ; Vollbrecht, J. ; Spence, D.: *Generic AAA Architecture.* United States : RFC 2903, 2000

[PSS+04] Palekar, Ashwin ; Simon, Dan ; Salowey, Joe ; Zhou, Hao ; Zorn, Glen ; Josefsson, S.: *Protected EAP Protocol (PEAP) Version 2.* `draft-josefsson-pppext-eap-tls-eap-10.txt`, 2004

[RRSW97] Rigney, C. ; Rubens, A. ; Simpson, W. ; Willens, S.: *Remote Authentication Dial In User Service (RADIUS).* United States : RFC 2058, 1997

[RWC00] Rigney, C. ; Willats, W. ; Calhoun, P.: *RADIUS Extensions.* United States : RFC 2869, 2000

[RWRS00] Rigney, C. ; Willens, S. ; Rubens, A. ; Simpson, W.: *Remote Authentication Dial In User Service (RADIUS).* United States : RFC 2865, 2000

[Sch03] Schäfer, Günter: *Netzsicherheit Algorithmische Grundlagen und Protokolle.* Dpunkt Verlag, 2003