Beispiel Konfiguration für einen Freeradius Server
802.1X EAP/TTLS

(Version 1.1.1)
zusammengetragen von Ralf Paffrath DFN-Verein

Vorraussetzung:

Für das DFN-Roaming wird ein aktueller freeradius benötigt. Freeradius muss mit OpenSSL-Support compiliert werden. OpenSSL 0.9.7d empfiehlt sich hierbei.

Konfiguratuion des freeradius:

Einträge in der radiusd.conf, die wichtig sind:

```
proxy_requests = yes
$INCLUDE ${confdir}/proxy.conf
$INCLUDE ${confdir}/eap.conf
```

In eap.conf sollte folgende Einträge Beachtung finden:

```
 eap {
# Invoke the default supported EAP type when
# EAP-Identity response is received.
#
# The incoming EAP messages DO NOT specify which EAP
# type they will be using, so it MUST be set here.
#
# For now, only one default EAP type may be used at a
# time.
#
default_eap_type = ttls
# Default expiry time to clean the EAP list, It is
# maintained to correlate the EAP-Response for each
# EAP-request sent.
timer_expire = 60

# Supported EAP-types
```

```
# We do NOT recommend using EAP-MD5 authentication
# for wireless connections. It is insecure, and does
# not provide for dynamic WEP keys.
#
md5 {
}

# Cisco LEAP
#
# Cisco LEAP uses the MS-CHAP algorithm (but not
# the MS-CHAP attributes) to perform it's
# authentication.
#
# As a result, LEAP *requires* access to the plain-text
# User-Password, or the NT-Password attributes.
# 'System' authentication is impossible with LEAP.
#
leap {
}

## EAP-TLS is highly experimental EAP-Type at the
moment.
# Please give feedback on the mailing list.
tls {
private_key_password = *********
private_key_file = /pfad/zum/radius.key

# If Private key & Certificate are located in
# the same file, then private_key_file &
# certificate_file must contain the same file
```

```
# name.
certificate_file = /pfad/zum/radius.pem

# Trusted Root CA list
CA_file = /pfad/zur/CAcert.pem

dh_file = /var/radius/DH
random_file = /var/radius/random

#
# This can never exceed the size of a RADIUS
# packet (4096 bytes), and is preferably half
# that, to accomodate other attributes in
# RADIUS packet. On most APs the MAX packet
# length is configured between 1500 - 1600
# In these cases, fragment size should be
# 1024 or less.
#
fragment_size = 1024

# include_length is a flag which is
# by default set to yes If set to
# yes, Total Length of the message is
# included in EVERY packet we send.
# If set to no, Total Length of the
# message is included ONLY in the
# First packet of a fragment series.
#
include_length = yes
}
```

```
        ttls {

        default_eap_type = md5

        copy_request_to_tunnel = no

# Ist die Authentifizierung erfolgreich, so wird die innere Identität des Nutzer zurückgeliefert,
# siehe auch dazu die Bemerkungen zur users Datei weiter unten
        use_tunneled_reply = yes

        }



        }
```

Proxykonfiguration im freeradius:

```
    realm NULL {
 }
```

Alle Anfragen, die ohne Realm gestellt werden, werden lokal behandelt.

```
    realm <einrichtung>.de {
}
```

Alle Anfragen, die mit dem Realm "<einrichtung>de" kommen werden ebenfalls lokal behandelt. Sollte man nicht vergessen, um Loops zu vermeiden.

```
    #

    # This realm is for ALL OTHER requests.

    #



    realm DEFAULT {

    type = radius

    authhost = radius1.dfn.de:1812

    accthost = radius1.dfn.de:1813

    secret = ********
```

```
    #ldflag = round_robin[1]

    nostrip

    }

realm DEFAULT {

    type = radius

    authhost = radius2.dfn.de:1812

    accthost = radius2.dfn.de:1813

    secret = ********

    #ldflag = round_robin[1]

    nostrip

    }
```

Alle anderen Anfragen werden an den Radius des DFN weitervermittelt. nostrip ist hier zwingend, da der DFN-Radius den ursprünglichen Realm benötigt, um die Anfrage an den eigentlich zuständigen Radius weiterzuschicken.

Klienten definieren

Access Points, Radius Server und andere Radius Klienten werden in der clients.conf definiert. Als default ist nur localhost dort definiert. Access Points und Radius Klienten, die den Radius Server befragen dürfen, sollten dort eingetragen werden:

```
#########################
#  Access Points
#########################
client ap-ip-address {
        secret = shared secret0
        shortname = ap-name
        nastype = ...
}
#########################
#   Radius Klienten
#########################
client radius1.dfn.de {
        secret = shared secret1
        shortname = top-level-radius1
        nastype = other
}
client radius2.dfn.de {
        secret = shared secret2
        shortname = top-level-radius2
        nastype = other
}
```

Ein sogenannter loopback Klient zum testen macht auch Sinn:

```
client 127.0.0.1 {
        secret = shared secret looppback
```

[1] ldflag=round_robin sollte auskommentiert werden: Es hat sich gezeigt, dass der Mechanismus im freeradius bei der Verwendung von eap nicht funktioniert und zu Fehlern führen kann.

```
        shortname = loopback
}
```

User Authentifizierung, Accounting und Realm Verarbeitung

Für die Authentifizierung der Nutzer ist die Konfigurationsdatei *users* besonders wichtig. Sie kann u.a. auch statisch konfigurierte Nutzer-Einträge enthalten, insbesondere kann sie festlegen wie Anfragen an den Radius Server behandelt werden bzw. welche Module für die Authentifizierung der Nutzer entscheidend sind. Auch das Accounting der Nutzer lässt sich sehr schön in Form eines Wokaround in der *users* definieren. Die Reihenfolge, in der die Regeln in die *users* eingetragen werden ist signifikant. Folgende Beispielkonfiguration zeigt, welche Möglichkeiten zur Verfügung stehen, bitte beachten, nach jeder Zeile folgt ein expliziter Zeilenumbruch:

```
# Der Nutzer anonymus wird abgewiesen
DEFAULT         User-Name =~ "^[Aa][Nn][Oo][Nn][Yy][Mm][Oo][Uu][Ss]$"
                Auth-Type := Reject

# Der Nutzer anonymus@einrichtung.de wird durchgelassen und EAP wird aktiviert
DEFAULT         User-Name =~ "^[Aa][Nn][Oo][Nn][Yy][Mm][Oo][Uu][Ss]@.*$"
                Auth-Type := EAP

# Nutzer ohne Realm werden abgewiesen, sowohl außerhalb als auch innerhalb des Tunnels
DEFAULT         Realm == NULL, Auth-Type := Reject

# Die innere Identität wird im Access Accept für das Accounting zurückgeliefert
#  Tests haben gezeigt, dass nur dann die innere Identität zurückgeliefert wird, wenn in eap.conf
# use_tunneled_reply auf „yes“ gesetzt wird
DEFAULT         Realm == <einrichtung>.de, FreeRADIUS-Proxied-To == 127.0.0.1
                User-Name = `%{User-Name}`,
                Fall-Through = yes

# statisch eingetragener Nutzer
static-user     User-Password == streng-geheim

# Radius Server Monitoring Eintrag, zeigt das der Server aktiv ist und Nutzer authentifizieren kann
bittemeldedich  Huntgroup-Name == "monitoring", Auth-Type := Accept
                Reply-Message == "Hey Dude, I'm still doing my job!"

# Authentifiziert Nutzer gegen /etc/shadow
DEFAULT         Auth-Type := System
```

Eine EAP/TTLS Anfrage muss dann wie folgt aussehen:

- äußere Identität: anonymous@einrichtung.de
- innere Identität: nutzername@einrichtung.de

In beiden Fällen muss der komplette Realm angegeben werden. In der Accounting Datei reply-details steht dann, z.B. nicht mehr:

```
Packet-Type = Access-Accept
Thu Mar  28 12:32:05 2005
        Auth-Type := EAP
        MS-MPPE-Recv-Key = 0x8ad7ef024a0ca45fef9bc3ad46a61c6d800a380874974d883cdabd63575139045
        MS-MPPE-Send-Key = 0xda8896943cc7dbf16a8ffb40d91473ae81729d15848109acf84a5bf25763feafc7b
        EAP-Message = 0x05050004
        Message-Authenticator = 0x00000000000000000000000000000000
        User-Name = "anonymous"
```

sondern

```
Packet-Type = Access-Accept
Thu Mar  28 12:32:05 2005
        Auth-Type := EAP
        MS-MPPE-Recv-Key = 0x8ad7ef024a0ca45fead46a61c6d800a80874974d883cdabd63575139045
```

MS-MPPE-Send-Key = 0xda8896943cc7db16a8ffb49147ae81729d1848109acf84a5bf25763feafc7b
EAP-Message = 0x08050004
Message-Authenticator = 0x00000000000000000000000000000000
User-Name = "nutzer@einrichtung.de"

Testen der Konfiguration:

Um die Konfiguration zu Testen, startet man den Radius mit den Optionen -xA:

```
/usr/sbin/radiusd -xA
```

Außerdem benötigt man spätestens jetzt einen Partner im DFN, der die ganze Installation von "außerhalb" testen kann s. http://www.dfn.de/content/dienstleistungen/dfnroaming/roamingstandorte/