

**Elastyczna autoryzacja adresow MAC przy pomocy
FreeRADIUSa na AP600/2000 (wersja 0.1)**

Jakub Wartak <vnull@pcnet.com.pl>

11 wrzesien 2005

1. Wstep

Nie bede sie rozwodzil jak dziala protokol RADIUS i jakie sa korzysci wynikajace dzięki wdrozeniu go w sieciach, bo to nie jest tekst o tym. Tekst jest przeznaczony dla poczatkujacych/srednio-zaawansowanych administratorow.

O mozliwosciach sprzetu ORINOCO (zwlaszcza Access Pointow) wiele juz bylo pochlebstw. Ten krotki tekst bedzie opisem wdrozenia systemu autoryzacji wlasnie dla APEKow klasy Orinoco 600/2000 (Powinno tez dzialac z AP700/4000 czy nawet Tsunami). Skoncentruje sie na rozwiazaniach OpenSource. Istnieje kilka serwerow RADIUSowych dostepnych w internecie za darmo. Jednym z najbardziej modularnych i elastycznych pod wzgledem konfiguracji jest FreeRADIUS.

UWAGA#1: Tekst byl pisany w oparciu o dzialajace instalacje FreeRADIUSa, wiec moglem cos przeoczyć w tym opisie.

UWAGA#2: Wszystko opisane tutaj jest pod katem FreeBSD, aczkolwiek instalacja pod dowolnie innym systemem nie powinna sprawiac problemow.

UWAGA#3: OpenBSD na styczen/luty 2005 **NIE** nadawal sie do instalacji FreeRADIUSa (linker nie radzil sobie z linkowaniem bibliotek wspoldzielonych)

UWAGA#4: W tym artykule tworze nowa baze MySQLowa, w rzeczywistosci bylo jednak odwrotnie, baza zostala obudowana instalacją FreeRADIUSa.

2. Instalacja serwera RADIUS

Instalacja jest bardzo prosta, wystarczy posiadac aktualne drzewo portow w /usr/ports:

```
root# cd /usr/ports/net/freeradius
root# make config
```

Zaznaczamy wsparcie dla MySQLa (FreeRADIUS wspiera także PostgreSQL, Oracle oraz drivery innych baz przez ODBC – tak, istnieje mozliwosc autoryzacji z bazy MS-SQL). Zaznaczamy również moduły eksperymentalne (chodzi nam głównie o rlm_perl).

```
root# make
```

A po udanej kompilacji:

```
root# make install clean
```

I juz mozemy sie cieszyć zainstalowanym serwerem RADIUSowym, ale najtrudniejsze przed nami.

3. Przygotowanie bazy MySQL

Kazdy wie jaka udreka jest reczne modyfikowanie plikow konfiguracyjnych, lub tez pisanie skryptow majacych modyfikowac te pliki. Prawdziwym wybawieniem od tych meczarni sa relacyjne bazy danych oraz jezyk SQL.

UWAGA: ten artykul nie jest i nie bedzie przewodnikiem po instalacji MySQLa dlatego zakladan ze baza MySQL (lub podobna, np. PostgreSQL) dziala juz prawidlowo i mozna sie do niej bezproblemowo laczyć. Sugeruje również zapoznanie sie manuałem MySQLa, zwlaszcza z dzialem traktujacym o bezpieczenstwu bazy i danych w niej przechowywanych.

a) musimy stworzyc baze danych w ktorej bedziemy przechowywali nasze dane odnosnie klientow i accesspointow (tutaj nazywa sie ona siec):

```
root# mysqladmin -u root -p create siec
```

b) tworzymy uzytkownika w MySQLu i nadajemu mu prawa do czytania z bazy siec:

```
root# mysql -u root -p siec
mysql> grant select on siec.* to radius@localhost identified by 'haslo';
mysql> flush privileges;
mysql> exit;
```

c) sprawdzamy czy mozemy sie polaczyc z baza jako utworzony user:

```
# mysql -u radius -phaslo siec
mysql> exit;
```

Aby zintegrowac RADIUSa z baza danych musimy utworzyc odpowiednie tabele. Na potrzeby autoryzacji stworzylem dwie zamiast jednej. W jednej trzymam dane dotyczace accesspointow w drugiej dane dotyczace userow:

```
CREATE TABLE accesspoints (
    id INT(5) NOT NULL auto_increment,
    ip VARCHAR(16) NOT NULL UNIQUE default "",
    lan_macaddr VARCHAR(32) NOT NULL UNIQUE default "",
    wap_macaddr VARCHAR(32) NOT NULL UNIQUE default "",
    PRIMARY KEY(id)
);
```

```
CREATE TABLE users (
    id INT(5) NOT NULL auto_increment,
    ip VARCHAR(16) NOT NULL UNIQUE default "",
    macaddr VARCHAR(32) NOT NULL UNIQUE default "",
    PRIMARY KEY(id)
);
```

d) powywszy fragment zapisujemy w pliku siec.sql, a nastepnie tworzymy table przez wczytanie tego pliku przez polecenie mysql:

```
root# mysql -u root -p siec < siec.sql
```

Jesli polecenie mysql nie zglosi bledu powinniśmy stac sie szczesliwymi posiadaczami tabel accesspoints i users w bazie siec :)

e) przystepujemy do dodania danych do tych tablek, mozna to zrobic przez oprogramowanie w stylu phpMyAdmin badz tez recznie:

dodanie accesspointa:

```
mysql> INSERT INTO accesspoints (ip,lan_macaddr,wap_macaddr) VALUES ( 'a.b.c.d',
'11:22:33:44:55:66', '11:22:33:44:55:66' );
```

dodanie usera:

```
mysql> INSERT INTO users (ip,macaddr) VALUES ( 'a.b.c.d', '11:22:33:44:55:66');
```

Małe wyjaśnienie:

W tabeli accesspoints są dwa pola `lan_macaddr` oraz `wap_macaddr`, gdyż część Access Pointów które będą łączyły się do Orinoco mają dwa adresy MAC (LAN i Wireless) - czy jakos tak (FIXME-uzasadnienie).

f) Tworzymy tabelki do accountingu: A to już jako zadanie domowe ;)

4. Konfiguracja serwera RADIUS

Główne pliki zostały zainstalowane w `/usr/local/etc/raddb`. Najważniejszym z plików konfiguracyjnych jest `radiusd.conf`. Przykładowy plik ten może wyglądać następująco:

```
prefix = /usr/local
exec_prefix = ${prefix}
sysconfdir = ${prefix}/etc
localstatedir = ${prefix}/var
sbindir = ${exec_prefix}/sbin
logdir = /var/log/radius
raddbdir = ${sysconfdir}/raddb
radacctdir = ${logdir}/radacct
confdir = ${raddbdir}
run_dir = /var/run/radius
log_file = ${logdir}/radius.log
libdir = ${exec_prefix}/lib
pidfile = ${run_dir}/radiusd.pid
```

```
user = radius
group = radius
```

```
max_request_time = 70
delete_blocked_requests = no
cleanup_delay = 5
max_requests = 2560
port = 1645
```

```
listen {
    ipaddr = naszAdresIP
    port = 1645
    type = auth
}
```

```
hostname_lookups = no
allow_core_dumps = no
regular_expressions = yes
extended_expressions = yes
log_stripped_names = no
log_auth = yes
log_auth_badpass = yes
log_auth_goodpass = yes
usercollide = no
lower_user = yes
lower_pass = no
```

```
nospace_user = before
nospace_pass = no
checkrad = ${sbindir}/checkrad

security {
    max_attributes = 200
    reject_delay = 0
    status_server = no
}

proxy_requests = no
snmp = no

$INCLUDE ${confdir}/clients.conf

thread pool {
    start_servers = 8
    max_servers = 64
    min_spare_servers = 8
    max_spare_servers = 32
    max_requests_per_server = 300
}

modules {
    exec bad-login {
        wait = no
        program = "/usr/local/etc/raddb/bad-login.sh %{User-Name}"
        input-pairs = request

        output-pairs = reply
        packet-type = Access-Reject
        shell-escape = yes
    }

    acct_unique {
        key = "User-Name, Acct-Session-Id, NAS-IP-Address, Client-IP-Address, NAS-Port"
    }

    perl macauth {
        func_authenticate = authenticate
        func_authorize = authorize
        func_detach = detach
        module = /usr/local/etc/raddb/macauth.pl
    }

    always handled {
        rcode = handled
    }
}

authorize {
    macauth
```

```

#   daily
}

authenticate {
    Auth-Type PCNET-MAC {
        macauth
    }
}

preacct {
    acct_unique
}

accounting {
}

session {
}

post-auth {
    Post-Auth-Type REJECT {
        bad-login
    }
}

```

Powyzszy plik nie konfiguruje obslugi accountingu polaczonego bezposredniej do bazy SQLowej (nie moge dawac gotowych rozwiazan ;)), odpala skrypt `/usr/local/etc/raddb/bad-login.sh` z jednym parametrem , ktorym jest adres MAC, w przypadku adresu MAC ktorego nie bylo w bazie. Caly rdzen autoryzacji i autentykacji jest przerzucony na plik perla `/usr/local/etc/raddb/macauth.pl`. Modul `rlm_perl` jest posrednikiem miedzy rdzeniem serwera a aplikacjami napisanymi w jezyku skryptowym Perl. Umozliwia to elastyczne zwiekszenie funkcjonalnosc serwera bez pisania modulow w jezyku C. Mozna korzystac z bardzo bogatego w pomocnicze moduly repozytorium Perla jakim jest CPAN. My wykorzystamy glownie modul DBI oraz jego driver do bazy MySQL. Da to nam znacznie wieksze pole manewru niz surowe zapytania SQL. Warto zauwazyc ze skrypt nie jest odpalany za kazdym razem, lecz inicjalizowany raz, po czym po kazdym zapytaniu do serwera zgodnie z konfiguracja serwer odwołuje sie do funkcji skryptu. A tutaj mamy przykladzik:

```

#!/usr/bin/perl -W

#
# macauth.pl v0x5 (c) Jakub Wartak 2004 ( vnull@pcnet.com.pl )
#
# 1.3.2004 - pierwsza wersja ( na podstawie starego skryptu z Gnu-RADIUSa ), samo MAC-AUTH
# 9.3.2004 - pierwsza (nieudana) integracja z FreeRADIUS => nieobsluje wiecej niz 1 modulu perlowego
#      jednoczesnie => rewrite na rozlaczone PPPoE i MAC-AUTH :(
# 12.3.2004 - udany rewrite ( rozlaczone PPPoE i MAC-AUTH )
#

use strict;
use vars qw(%RAD_REQUEST %RAD_REPLY %RAD_CHECK);
use Data::Dumper;
use lib qw( /usr/local/pcnet );
use PCNET;

```

```

use constant RLM_MODULE_REJECT=> 0;# immediately reject the request */
use constant RLM_MODULE_FAIL=> 1;# module failed, don't reply */
use constant RLM_MODULE_OK=> 2;# the module is OK, continue */
use constant RLM_MODULE_HANDLED=> 3;# the module handled the request, so stop.*/
use constant RLM_MODULE_INVALID=> 4;# the module considers the request invalid.*/
use constant RLM_MODULE_USERLOCK=> 5;# reject the request (user is locked out) */
use constant RLM_MODULE_NOTFOUND=> 6;# user not found */
use constant RLM_MODULE_NOOP=> 7;# module succeeded without doing anything */
use constant RLM_MODULE_UPDATED=> 8;# OK (pairs modified) */
use constant RLM_MODULE_NUMCODES=> 9;# How many return codes there are */

```

```

my $db;
# undef => nodebug
# 1  => debug(funkcjonowanie w radiusd dalej normalnie)
# 2  => console debug
my $debug = 1;

```

```

sub mylog {
    if($debug eq "2") {
        print "MACAUTH: @_n";
    } else {
        &radiusd::radlog(1, "MACAUTH:: @_");
    }
}

```

```

sub authorize
{
    my $login = $RAD_REQUEST{'User-Name'};

    if(defined($login) && $login ne "") {
        $RAD_CHECK{'Auth-Type'} = "PCNET-MAC";
        return RLM_MODULE_OK;
    } else {
        mylog("login not def?");
        return RLM_MODULE_NOOP;
    }
}

```

```

sub authenticate
{
# if( $debug ) {
#     mylog("Dumping");
#     for(keys %RAD_CHECK) {
#         mylog("$_ = $RAD_CHECK{$_}");
#     }
#     mylog("End of Dump");

    return &MAC_authenticate;
}

```

```

sub MAC_authenticate
{
    my $org;

```

```

my $nas_ip;

if($debug eq "2") {
    $org = "112233445566";
    $nas_ip = "127.0.0.1";
} else {
    $org = $RAD_REQUEST{'User-Name'};
    $nas_ip = $RAD_REQUEST{'NAS-IP-Address'};
}

$org =~ s/^//g;
my $mac = $org;

mylog("NAS_IP=$nas_ip; MAC2VERIFY=$mac") if ($debug);

my $result = $db->Array("SELECT macaddr FROM users WHERE macaddr='$mac' AND block_radius='N'");
if( ! $result ) {
    # nie ma takiego usera... moze jednak jest taki APek?
    my $res2 = $db->Array("SELECT name FROM aps WHERE lan_mac='$mac' OR wlan_mac='$mac'");
    if( ! $res2 ) {
        # nie ma takiego APeka
        mylog("$mac not found in aps and in users") if ($debug);
        return RLM_MODULE_NOTFOUND;
    } else {
        # jest taki APek
        mylog("$mac FOUND in aps") if($debug);
        return RLM_MODULE_OK;
    }
} else {
    # jest taki user
    mylog("$mac FOUND in users") if ($debug);
    return RLM_MODULE_OK;
}
}

sub detach
{
    $db = PCNET->Connect('radius');
    $db->Verbose;
}

sub postauth {
    mylog("Dumping");
    for(keys %RAD_CHECK) {
        mylog("_ = $RAD_CHECK{$_}");
    }
    mylog("End of Dump");
}

&detach;
if($debug eq 2) {

    &authorize;
}

```

```
    &authenticate;
}
```

Jest to jedynie przykład mozliwej implementacji (i dosyc starej, bo z poczatku roku 2004). Po pierwsze ten skrypt korzysta z modulu *PCNET.pm*, który implementuje glowna logike laczenia sie do bazy i pare innych drobnych rzeczy. Po drugie, skrypt nie jest pisany z naciskiem na wydajnosc (nie bylo takiej potrzeby). Po trzecie zastosowanie perla daje bardzo duza elastycznosc w procesie dodawania nowej funkcjonalnosci. Teraz go poprawnie zainstalujemy:

```
root# cd /usr/local/etc/raddb
root# chmod 750 macauth.pl
root# chown root:radius macauth.pl
```

Kolejnym waznym plikiem jest *clients.conf*. Jego glównym zadaniem jest przyznanie poszczegolnym Access Pointom dostepu do uslug oferowanych przez serwer FreeRADIUS. Format jest bardzo prosty:

```
# ponizszy wpis umozliwi nam dostep do odpytywania serwera RADIUS lokalnie w celach testowych
client 127.0.0.1 {
    secret = test
    shortname = tester
    nastype = other
}

# przykladowy AP
client 10.1.1.100 {
    secret = haselkoDostepowe
    shortname = LudzkaNazwaAP
    nastype = other
}
```

Oczywiscie jezeli posiadamy baze Apekow można sobie bardzo ulatwic zycie generujac powyzszy plik dynamicznie jakimis skryptem.

Teraz przyszla pora na przykladowy *bad-login.sh*:

```
#!/usr/local/bin/bash

# by Jakub Wartak , 10.4.2004

who2mail="ktos1@blebleble.pl, ktos2@blebleble.pl"
user=$1

stopka="--
Ten list zostal wygenerowany automatycznie. Nie odpowiadaj na niego.
PCNET-MACAUTH RADIUS SYSTEM na $HOSTNAME";

MAILRC="/dev/null"
DNS=`/usr/local/bin/resolveip -s $NAS_IP_ADDRESS 2> /dev/null`
data=`date`

#####
# MAC #
#####
```

```
( cat << EOF
Proba polaczenia z nieznanego adresu MAC: $user
Adres Accesspointa: $NAS_IP_ADDRESS ( DNS=$DNS )
Data: $data

$stopka
EOF
) | mail -s "Nieznany adres MAC" $who2mail > /dev/null 2>&1

#logger -p local2.notice -t RADIUS-macauth "Auth error for: $user"

# wsadzamy info o zlej probie autoryzacji do bazy
echo "INSERT INTO radmac_fails (time,macaddr,apip ) VALUES (NOW(), '$user', '$NAS_IP_ADDRESS' ); " |
mysql -s -uradius -phaselko -h 127.0.0.1 siec

exit 4
```

Jak widac, powyższy skrypt wsadza złe “rekordy” do specjalnej tabeli *radmac_fails*. Jest ona bardzo prosta, a w warunkach produkcyjnych czyta z niej skrypt PHP oraz generowane sa raporty co pewien czas (scislej mowiac co tydzien).

A teraz instalacja:

```
root# chmod 750 bad-login.sh
root# chown 0:radius bad-login.sh
```

5. RADIUSa konfiguracji ciag dalszy i pierwsze testy

Teraz musimy zadbać o to, aby nasz serwer uruchamiał się przy każdym podnoszeniu systemu, we FreeBSD jest to bardzo proste, dodajemy poniższą linijkę do */etc/rc.conf*:

```
radiusd_enable="YES"
```

Uruchamiamy próbnie radiusd:

```
root# radiusd -X
```

Jeśli wszystko poszło OK to program powinien czekać na zapytania. Pierwsze testy można przeprowadzić, np. tak:

```
root# radtest 00:0f:3d:03:80:59 00:0f:3d:03:80:59 127.0.0.1:1645 1 test
Sending Access-Request of id 159 to 127.0.0.1:1645
  User-Name = "00:0f:3d:03:80:59"
  User-Password = "00:0f:3d:03:80:59"
  NAS-IP-Address = localhost
  NAS-Port = 1
rad_recv: Access-Accept packet from host 127.0.0.1:1645, id=159, length=20
```

Jak widzimy powyższe zapytanie o MAC, który rzeczywiście jest bazie zwróciło nam poprawny wynik (Access-Accept). A teraz dla przykładu zapytanie o zły adres MAC:

```
root# radtest 00:0f:3d:03:80:5X 00:0f:3d:03:80:5X 127.0.0.1:1645 1 test
```

Sending Access-Request of id 205 to 127.0.0.1:1645

User-Name = "00:0f:3d:03:80:5X"

User-Password = "00:0f:3d:03:80:5X"

NAS-IP-Address = localhost

NAS-Port = 1

rad_recv: Access-Reject packet from host 127.0.0.1:1645, id=205, length=20

... po czym powinien dotrzec do nas e-mail, o nieudanej probie autoryzacji :)

6. Wlaczanie RADIUSa na AP

Nie chce mi sie, znaczy sie FIXME + screenshoty ;)

Adres IP

Port: 1645

Shared secret: taki jak w *clients.conf*

Format: AA:BB:CC:DD:EE:FF

7. Podsumowanie

FIXME too.