Instalacja i konfiguracja serwera FreeRADIUS v. 2

Maja Górecka-Wolniewicz, UCI, UMK (mgw@umk.pl)
dokument przygotowany w ramach projektu B-R eduroam-PIONIER
Wersja 1.0 – 2008-03-25

Spis treści

Projekt FreeRADIUS

Jest to jedna z bardziej popularnych implementacji niekomercyjnych, powszechnie wykorzystywana w projekcie eduroam i zalecana w polskim projekcie eduroam.
Główna strona projektu znajduje się pod adresem http://www.freeradius.org. Strona http://wiki.freeradius.org zawiera wiele wartościowych informacji na temat instalacji i konfiguracji oprogramowania.

Aktualna wersja oprogramowania to 2.0.3.

Oprogramowanie FreeRADIUS cechują:

- duże możliwości, wielofunkcyjność,
- modularność,
- elastyczność w konfiguracji.

FreeRADIUS w wersji 2 znacząco różni się od oprogramowania w wersji 1. Zmienił się istotnie sposób konfiguracji serwera. Nie jest możliwe przeniesienie konfiguracji wersji 1 do wersji 2 bez dostosowania plików konfiguracyjnych. Najważniejszą nowością w wersji 2 jest możliwość

definiowania serwerów wirtualnych, dzięki czemu można w sposób bardziej naturalny dostosować konfigurację serwera. Drugą ważną nową własnością jest język unlang – narzędzie pozwalające na łatwą konfigurację w zależności od zawartości pakietu Radius. Sposób korzystania z nowej funkcjonalności oprogramowanie zostanie przedstawiony szczegółowo w rozdziale „Konfiguracja serwera FreeRADIUS".

Wymagania instalacyjne pakietu FreeRADIUS

Oprogramowanie nie ma specjalnych wymagań instalacyjnych. Typowo serwer obsługuje określone domeny (czyli realizuje uwierzytelnienie oraz autoryzację użytkowników z tych domen), wówczas niezbędne jest korzystanie z bezpiecznych połączeń, a więc pakietem wymaganym jest OpenSSL. W sytuacji, gdy serwer pełni wyłącznie funkcję proxy, połączenia SSL nie są realizowane. Zazwyczaj proces uwierzytelniania, autoryzacji i rozliczania współpracuje wybraną bazą danych, w zależności od potrzeb, konieczna jest instalacja takich pakietów, jak:

- OpenLDAP - jeśli używamy usługi LDAP do uwierzytelniania/autoryzacji (i nie korzystamy z komercyjnego oprogramowania LDAP),
- MySQL, Postgres - oprogramowanie bazodanowe.

Jeżeli zamierzamy korzystać ze skryptów w języku Perl lub Python, należy zadbać, by pakiety te były zainstalowane i dostępne w czasie instalacji oprogramowania.

Instalacja oprogramowania FreeRADIUS

Można zainstalować FreeRADIUS-a korzystając z gotowych pakietów instalacyjnych, które są dostępne ze strony http://www.freeradius.org/download.html, m.in. dla systemów: Cygwin, Debian, Fedora, Mandriva, Suse, Ubuntu, FreeBSD, NetBSD, OpenBSD a także Mac OSX i Windows.

Jeśli decydujemy się na samodzielną kompilację, na początek należy rozwinąć źródła oprogramowania pobranego ze wspomnianej strony.
Następnie wykonujemy polecenia:

```
cd freeradius-server-2.0.3
./configure -prefix=/opt/freeradius
```

Gdy chcemy użyć własnej instalacji LDAP:

```
--with-rlm-ldap-lib=/opt/ldap/lib
--with-rlm-ldap-include=/opt/ldap/include
```

Gdy chcemy użyć własnej instalacji OpenSSL:

```
--with-ssl-lib=/opt/ssl/lib
--with-ssl-include=/opt/ssl/include
```

Uwaga:
Jeśli używamy własnej instalacji OpenSSL-a, aby zagwarantować, że programy i biblioteki są dolinkowane do właściwych bibliotek OpenSSL najlepiej, przed wywołaniem configure ustawić zmienne środowiskowe CPPFLAGS i LDFLAGS, np.:

```
CPPFLAGS=-I/opt/ldap/include -I/opt/ssl/include
LDFLAGS=-L/opt/ldap/lib -L/opt/ssl/lib -Xlinker \
       -R/opt/ldap/lib:/opt/ssl/lib
```

Inne przykładowe ustawienia na etapie kompilacji:

```
--with-experimental-modules
--without-rlm-perl --without-rlm-sql
```

Kompilacja:

```
make
```

Instalacja (na ogół z uprawnieniami root):

```
make install
```

Konfiguracja oprogramowania FreeRADIUS

Nasze założenia konfiguracyjne są następujące:

- zmierzamy do uruchomienia serwera FreeRADIUS instytucji (poziom 3 w hierarchii eduroam), co oznacza, że jest wymagana możliwość kontaktowania się z serwerem ogólnopolskim, ew. z serwerami poziomu 4;
- realizujemy uwierzytelnianie i autoryzację użytkowników lokalnych (danej instytucji, w zdefiniowanym przez instytucję zakresie), zgodnie z przyjętymi na jej terenie zasadami uwierzytelniania (baza LDAP, baza MySQL, pliki Unix itp.).

Pliki konfiguracyjne FreeRADIUS-a znajdują się w katalogu `${katalog_instalacji}/etc/raddb` instalacji pakietu FreeRADIUS. Najistotniejsze pliki konfiguracyjne to:

- `radiusd.conf`
- `clients.conf`
- `proxy.conf`
- `eap.conf`
- `users`
- `huntgroups`
- `sites-enabled/*`

Plik clients.conf

Plik ten służy do specyfikacji adresów lub nazw urządzeń (Access Pointy, Switche 802.1x) oraz serwerów RADIUS, z którymi konfigurowany serwer RADIUS będzie się komunikował.
Postać definicji jest następująca:

```
client <nazwa | adres-IP | podsieć> {
<atrybut> = <wartość>
}
```

Na przykład:

```
client 192.168.1.120 {
 secret = tajny_klucz_RADIUS-AP
 shorname = campus1dev120
}
```

W atrybutach dla danego klienta można dodać

```
nastype = <słownikowa_wartość>
```

nastype mówi, jakiej metody użyć w skrypcie checkrad (program udostępniany w dystrybucji FreeRADIUS) do sprawdzenia jednoczesnego zalogowania użytkownika.

Uwagi:

- należy dbać o unikatowość kluczy;
- klucz tajny powinien mieć m.in. 16 znaków;
- zaleca się różnicowanie kluczy wspólnych, nie powinno definiować się podsieci AP-ów z jednym hasłem wspólnym;
- należy pamiętać, że klientami są również serwery RADIUS, wszystkie te, z którymi dany serwer potencjalnie może się kontaktować (są to tzw. serwery proxy), należy zdefiniować w pliku `clients.conf` i ustalić wspólny klucz do komunikacji;
- w testach przydaje się klient loopback (127.0.0.1).

Plik huntgroups

Służy do definicji grup urządzeń NAS.
Oto przykładowe definicje grup:

```
aptest NAS-IP-Address == 192.168.1.200
aptest NAS-IP-Address == 192.168.1.201
aptest NAS-IP-Address == 192.168.1.201, NAS-Port-Id == 0-3
             User-Name = user1,
             User-Name = user2
```

Grupę urządzeń można również definiować na podstawie adresu MAC urządzenia. Jest to szczególnie przydatne, gdy chcemy w specyficzny sposób obsługiwać użytkowników korzystających z określonego Access Pointa danego kontrolera (wówczas `NAS-IP-Address` jest adresem kontrolera).

```
aptest Called-Station-Id == ”00aabbccddee”
```

Z definicji zastosowanych w pliku `huntgroups` korzystają pliki używane w czasie autoryzacji, uwierzytelnienia oraz w fazie „po uwierzytelnieniu” (post-auth), czy „po przekierowaniu” (post-proxy). Można w nich definiować kryterium dopasowania typu `Huntgroup-Name == ”aptest”`

Plik proxy.conf

Za pomocą tego pliku odbywa się konfiguracja domen (ang. *realms*) obsługiwanych przez serwer. Również w tym pliku, na początku, w bloku `proxy server {...}` jest ustalany sposób obsługi zleceń proxy.
Począwszy od wersji 2 FreeRADIUS-a definiujmy oddzielnie serwery obsługujące domeny – w ramach bloku `home_server {...}` i oddzielnie domeny w blokach `realm {...}`. Poza tym w pliku `proxy.conf` można zdefiniować sposób rozłożenia obciążenia oraz zachowanie w przypadku niedostępności serwera – służy do tego blok `server_pool {...}`.

W przypadku serwerów poziomu jednostki, w tym pliku zamieszczamy definicje serwerów poziomu PL, np.:

```
home_server radius1pl {
        type = auth
        ipaddr = 158.75.1.25
        port = 1812
        secret = haslo_nasz_serwer_pl1
        response_window = 20
        zombie_period = 40
        status_check = status-server
        check_interval = 30
        num_answers_to_alive = 3
}
home_server radius2pl {
        type = auth
        ipaddr = 150.254.173.30
        port = 1812
        secret = haslo_nasz_serwer_pl2
        response_window = 20
        zombie_period = 40
        status_check = status-server
        check_interval = 30
        num_answers_to_alive = 3
}
```

`type` – `auth` oznacza uwierzytelnienie, można definiować serwer do komunikacji w celu realizacji procesu rozliczeniowego (`type=acct`)

`secret` – ustalenie hasła wspólnego do komunikacji z danym serwerem

`ipaddr` – adres IP serwera

`port` – port serwera

Pozostałe parametry mają charakter administracyjny, służą do kontroli działania serwera w komunikacji proxy.

Jeśli domena jest obsługiwana lokalnie, nie musimy deklarować specjalnego bloku `home_server`.

W sekcji `home_server_pool`:

```
home_server_pool poland {
        type = fail-over
        home_server = radius1pl
        home_server = radius2pl
}
```

ustalamy, że pula `poland` korzysta z serwera `radius1pl`, a jeśli ten nie odpowiada z serwera `radius2pl`.

`type` – `fail-over` jest definicją reakcji na niedostępność serwera

Deklaracja obsługi lokalnych domen ma postać:

`strip / nostrip` – określa, czy należy usuwać część domenową z nazwy użytkownika (`strip`), czy

```
realm a.b {
 [strip lub nostrip]
}
```

nie (`nostrip`).

W projekcie eduroam jest zlecane stosowanie `nostrip` – do serwerów proxy należy przekazywać pełne nazwy sieciowe użytkowników.

Deklaracja wskazująca, że domena c.d jest obsługiwana przez zdalny serwer ma postać:

```
realm c.d {
 type = radius
 authpool = poland
 nostrip
}
```

Pliki „users”

Są to pliki konfiguracyjne jest używane przez moduł `files`. Zastosowania modułu `files` to:

- autoryzacja użytkownika (authorize), domyślnie plik users;
- uwierzytelnienie użytkownika (authenticate), domyślnie plik auth_users;
- obsługa użytkownika przed rozpoczęciem procesu rozliczania (preacct), domyślnie plik acct_users;
- obsługa użytkowników przed realizacją funkcji proxy (pre-proxy) domyślnie plik preproxy_users;
- obsługa użytkowników po realizacji funkcji proxy (post-proxy), domyślnie plik postproxy_users;
- obsługa użytkowników po zakończeniu uwierzytelnienia (post-auth), domyślnie plik postauth_users.

Domyślne nazwy plików konfiguracji są zamieszczone w sekcji `files` pliku `radiusd.conf`:

```
usersfile = ${confdir}/users
acctusersfile = ${confdir}/acct_users
preproxy_usersfile = ${confdir}/preproxy_users
```

W plikach tych umieszcza się zestaw dyrektyw konfiguracyjnych używanych przez moduł files w celu podjęcia decyzji o sposobie autoryzacji i uwierzytelnienia użytkownika. Format pliku pliku jest następujący:

```
nazwa <lista_sprawdzanych_elementów (check items)> [, ]\
  <lista_elem_konfiguracyjnych>
TAB <lista_elementów_odpowiedzi (reply items)>
```

`nazwa` – nazwa użytkownika lub słowo DEFAULT (dowolny użytkownik)

Uwagi:

- wszystkie elementy sprawdzane MUSZĄ być umieszczone w jednym wierszu, za nazwą użytkownika;
- separatorem elementów sprawdzanych jest przecinek;
- w pierwszym wierszu, w ramach listy elementów sprawdzanych MOŻNA wskazać element konfiguracyjny, np. Auth-Type i/lub Autz-Type
- elementy odpowiedzi MOGĄ być umieszczane w wielu wierszach, w tym przypadku poprzedni wiersz (zawierający element odpowiedzi) musi kończyć się przecinkiem;
- przetwarzanie pliku jest realizowane sekwencyjnie od początku pliku;
- po dopasowaniu użytkownika i elementów sprawdzanych ustalane są elementy odpowiedzi (wg deklaracji);
- domyślnie po dopasowaniu przetwarzanie kończy się;
- element specjalny: `Fall-Through = Yes` oznacza "przetwarzaj dalej".

Dozwolone postaci definicji atrybut – wartość:

1. forma `atrybut = wartość`

   NIE MOŻE wystąpić na liście sprawdzanych elementów

   MOŻE wystąpić jako element konfiguracyjny (Auth-Type) - ustala wówczas wartość atrybutu, tylko gdy nie była ustalona na liście odpowiedzi oznacza dodanie wartości danego atrybutu, ale tylko, gdy atrybut nie ma dotychczas wartości
2. forma `atrybut := wartość`

   na liście sprawdzanych elementów oznacza "zawsze dopasowany" zastępuje wartość atrybutu konfiguracyjnego zastępuje wartość atrybutu konfiguracyjnego

   jako element odpowiedzi zastępuje wartość atrybutu
3. forma `atrybut += wartość`

   na liście sprawdzanych elementów oznacza "zawsze dopasowany"

   na liście konfiguracji i odpowiedzi oznacza dodanie wartości danego atrybutu
4. forma `atrybut == wartość`

   na liście sprawdzanych elementów sprawdza, czy atrybut istnieje w zleceniu i czy wartość jest równa podanej

   NIE MOŻE wystąpić na liście odpowiedzi
5. forma `atrybut != wartość`

   na liście sprawdzanych elementów sprawdza, czy atrybut istnieje w zleceniu i wartość jest różna od podanej

   NIE MOŻE wystąpić na liście odpowiedzi

6. forma `atrybut > wartość`

   na liście sprawdzanych elementów sprawdza, czy wartość atrybutu w zleceniu jest większa od podanej

   NIE MOŻE wystąpić na liście odpowiedzi

7. forma `atrybut >= wartość`

   na liście sprawdzanych elementów sprawdza, czy wartość atrybutu w zleceniu jest większa lub równa podanej

   NIE MOŻE wystąpić na liście odpowiedzi

8. forma `atrybut < wartość`

   na liście sprawdzanych elementów sprawdza, czy wartość atrybutu w zleceniu jest mniejsza od podanej

   NIE MOŻE wystąpić na liście odpowiedzi

9. forma `atrybut <= wartość`

   na liście sprawdzanych elementów sprawdza, czy wartość atrybutu w zleceniu jest mniejsza lub równa podanej

   NIE MOŻE wystąpić na liście odpowiedzi

10. forma `atrybut =~ wyrażenie`

    na liście sprawdzanych elementów sprawdza, czy wartość atrybutu dopasowuje się do wyrażenia (wartość znakowa!)

    NIE MOŻE wystąpić na liście odpowiedzi

11. forma atrybut !~ wyrażenie

    na liście sprawdzanych elementów sprawdza, czy wartość atrybutu nie dopasowuje się do wyrażenia (wartość znakowa!)

    NIE MOŻE wystąpić na liście odpowiedzi

12. forma `atrybut =* wartość`

    na liście sprawdzanych elementów sprawdza, czy zlecenie zawiera dany atrybut (wartość jest nieistotna)

    NIE MOŻE wystąpić na liście odpowiedzi

13. forma atrybut !* wartość

    na liście sprawdzanych elementów sprawdza, czy w zleceniu nie ma danego atrybutu (wartość jest nieistotna)

    NIE MOŻE wystąpić na liście odpowiedzi

Przykłady: (\ oznacza, że c.d. MUSI być w tej samym wierszu)

```
user1 User-Password == "user1pass"
user2 Auth-Type := Reject
      Reply-Message = "Zablokwane konto"
DEFAULT Realm == NULL, Auth-Type := Reject
DEFAULT Realm = "a.z", Client-IP-Address == 10.1.1.1, \
      Auth-Type := Reject
DEFAULT Real == "b.z" Hundgroup-Name =="XXX"
      Tunnel-Private-Group-Id := 40
      Tunnel-Medium-Type = 6
      Tunnel-Type = VLAN
DEFAULT Real == "c.z" FreeRadius-Proxied-To == 127.0.0.1, \
            Autz-Type := gosc, ldapgosc-Ldap-Group == "z_PL"
      Tunnel-Private-Group-Id := 40
      Tunnel-Medium-Type = 6
      Tunnel-Type = VLAN
```

Użytkownik `user1` (nazwa domeny pusta) zostanie zaakceptowany, jeśli poda hasło `user1pass`.

Użytkownik user2 (nazwa domeny pusta) zawsze zostanie odrzucony.

Dowolny użytkownik zostanie odrzucony, jeśli domena wynikająca z nazwy sieciowej jest pusta.

Dowolny użytkownik, który pochodzi z domeny "a.z" i próbuje uzyskać dostęp za pośrednictwem klienta o IP 10.1.1.1 zostanie odrzucony.

Dowolny użytkownik, który pochodzi z domeny "b.z" zostanie zaakceptowany, jeśli adres klienta należy do grupy XXX. Użytkownikowi zostanie przypisany VLAN 40.

Dowolny użytkownik, który pochodzi z domeny "c.z" i uwierzytelniania się poprzez EAP-TTLS lub EAP-PEAP (atrybut FreeRadius-Proxied-To==127.0.0.1) podlega autoryzacji za pomocą sekcji konfiguracji pod nazwą "gosc" i zostanie zaakceptowany tylko wówczas, gdy atrybut grupowy pobrany z bazy LDAP ma wartość "z_PL". Przyjęty użytkownik zostanie przypisany do VLAN-u 40.

Zalecenia dotyczące plików users:

1. w pliku users należy wyspecyfikować odrzucanie zleceń zawierających pustą domenę, przede wszystkim jest to potrzebne w odniesieniu do zleceń nadchodzących z serwerów krajowych:

```
DEFAULT Realm = NULL, Client-IP-Address == 158.75.1.25, \
            Auth-Type := Reject
DEFAULT Realm = NULL, Client-IP-Address == 150.254.173.30, \
            Auth-Type := Reject
```

2. za pomocą odpowiednich wpisów w pliku users unikamy zapętlenia serwera, np.

```
DEFAULT Realm = "pl",Client-IP-Address == 158.75.1.25, \
             Auth-Type := Reject
```

   zapobiega przesłaniu zlecenia dot. domeny pl do serwera krajowego, w przypadku, gdy właśnie stamtąd nadeszło zlecenie.

3. w pliku postauth_users można ustalić ustawienie specyficznych atrybutów w odpowiedzi, np.

```
test1@test.pl
        Login-LAT-Group := "42"
test2@test.pl
        Tunnel-Medium-Type := 6,
        Tunnel-Private-Group-Id := 41,
        Tunnel-Type := VLAN
```

W wersji 2 oprogramowania FreeRADIUS dużo wygodniejszą metodą jest zastosowanie języka unlang w sekcji post-auth.

Plik radiusd.conf

W pliku tym, na początku konfiguracji, są ustalane ścieżki wiodące, lokalizacja konfiguracji, logów itp.

Dostosowania mogą wymagać bloki listen {...}. Domyślnie wymienione są dwa bloki listen: jeden dla pakietów związanych z uwierzytelnianiem (type=auth), drugi dla pakietów rozliczeniowych (type=acct), wskazujące nasłuchiwanie na wszystkich interfejsach danego serwera (ipaddr=*), na domyślnych portach (port=0, co oznacza port 1812 w przypadku pakietów uwierzytelniania i port 1813 w przypadku pakietów rozliczeniowych port 1814). Jeżeli serwer ma nasłuchiwać na konkretnym interfejsie, należy w odpowiednim bloku listen wpisać właściwy adres IP w przypisaniu ipaddr i, jeśli to potrzebne, podać niedomyślny port. W bloku listen można również określić grupę

klientów, która może korzystać z danego serwera (jest to nowa możliwość w v.2). W tym celu w bloku `listen` dodajemy przypisanie:

`clients = nazwa_grupy_klientów`
i w pliku `clients.conf` umieszczamy deklarację grupy w postaci:

```
nazwa_grupy_klientów clients  {
 client klient1_IP {
       secret = test1
 }
 client klient2_IP {
       secret = test2
 }
}
```

Domyślnie w `radiusd.conf` są zakomentowane przypisania `user = ...` i `group = ...`
Zaleca się uruchamianie serwera z prawami użytkownika innego niż root.
Np. ustawienia

`user = radius`

`group = radius`

sprawiają, że serwer będzie pracował z uprawnieniami użytkownika i grupy `radius`. Niestety, gdy korzystamy z FreeRADIUS-a zintegrowanego z Active Directory, niezbędne jest działanie serwera z prawami root – wymaga tego demon winbindd.

Największą sekcją pliku `radiusd.conf` jest `modules`. Są w niej zdefiniowane moduły używane przez serwer. Postać definicji modułu jest następująca:

```
nazwa [ wystąpienie ] {
 element_konfiguracji = wartość
}
```

Przykładowe moduły, zdefiniowane w tej sekcji to: `pap, chap, pam, unix, mschap, ldap, sql, realm, preprocess, checkval, files, detail, exec, perl, eap, attr_rewrite, attr_filter`.

Moduł ldap

Moduł ten pozwala zdefiniować sposób komunikowania się z bazą LDAP w celu autoryzacji i/lub uwierzytelnienia. Jeśli jest potrzeba wskazania kilku wystąpień modułów `ldap`, bo serwer ma się kontaktować z różnymi serwerami, na przykład, w zależności od obsługiwanej domeny, kolejne definicje rozróżniamy nazwami wystąpienia modułu:

```
ldap AD { }
ldap OLStaff { }
```

Zdefiniowane nazwy modułu mogą być następnie używane w sekcjach `authorize`, `authenticate`.
Jeśli w pliku `users`, używamy sprawdzenia grupy LDAP, to w przypadku stosowania wielu definicji modułów `ldap`, w elemencie sprawdzenia należy podać konkretne wystąpienie modułu, np.:

```
AD-Ldap-Group == "XXX"
```
albo
```
OLStaff-Ldap-Group == "XXX"
```

Zaleca się korzystanie z bezpiecznych połączeń z serwerami LDAP (`start_tls=yes`) oraz nie używanie anonimowych przeszukiwań.
Jeśli dysponujemy serwerem repliki, to należy zadeklarować jego użycie dla danego poddrzewa poprzez utworzenie drugiego wystąpienia modułu, np.

```
ldap OLStaff1 { tu_deklaracje_serwera_głównego }
ldap OLStaff2 { tu_deklaracje_serwera_repliki }
```

W bloku definicji wystąpienia modułu LDAP można określić parametry `groupmembership_filter` oraz `groupmembership_attribute`.

Parametr `groupmembership_attribute` podaje typ atrybutu LDAP stosowanego do sprawdzania, czy użytkownik należy do określonej grupy. Najpopularniejszym rozwiązaniem jest stosowanie atrybutu `radiusGroupName` ze schematu RADIUS-LDAPv3.schema, załączonego w dystrybucji FreeRADIUS. Parametr ten decyduje o sposobie przeszukiwania bazy. Jeśli nie określono `groupmembership_filter`, to stosowany jest domyślny filtr:

```
(|(&(objectclass=groupOfNames)(member=%{Ldap-UserDn}))
        (&(objectclass=groupOfUniqueNames)(member=%{Ldap-UserDn})))
```

Moduł ms_chap

Moduł ten jest niezbędny, gdy dostarczamy metodę PEAP (uwierzytelnianie MS-CHAP i MS-CHAPv2). Konfiguracja w sekcji `mschap` powinna być następująca:

```
authtype = MS-CHAP
use-mppe = yes
require_encription = yes
require_strong = yes
```

Moduł attr_rewrite

Moduł ten można stosować w procesie autoryzacji oraz rozliczania. Umożliwia dokonanie zmian w pakietach.
Wystąpienia modułu są wyróżniane nazwą, np.

```
attr_rewrite copy.user-name { }
attr_rewrite strip.user-name { }
```

Działanie modułu polega na modyfikacji wskazanych atrybutów w przypadku spełnienia określonych kryteriów:

- w ten sposób wskazujemy, że zmiany dotyczą określonego atrybutu:
  `attribute = ..`
- określamy, czego dotyczy przeszukiwanie: pakietu, odpowiedzi, konfiguracji
  `searchin = packet | reply | config`
- określamy sposób dopasowania (na podstawie wyrażenia regularnego)
  `searchfor = ...`
- dopasowany napis jest zastępowany zawartością wskazaną w module jako `replacewith`
- można dopisać nowy atrybut (`new_attribute=yes`)
- można napis zastępujący dopisać do oryginalnego (`append=yes`)
- można zadeklarować maksymalną liczbę dopasowań (`max_matches`)
- można ignorować wielkość liter (`ignore_case=yes`)

Np. definicja:

```
attr_rewrite checkuser {
         attribute = User-Name
         searchin = packet
         searchfor = "(^\\\\\\)"
         replacewith = ""
         max_matches = 10
        }
```

ustala, że wywołanie checkuser spowoduje usunięcie z wartości atrybutu User-Name wiodących znaków \.
Moduł taki jest następnie wskazywany w odpowiedniej sekcji, np. authorize, authenticate czy accounting.

Moduł attr_filter

Moduł ten jest przeznaczony do filtrowania atrybutów. Może zostać wywołany m.in. po odbiorze odpowiedzi z serwera proxy, w sekcji post-proxy.

W bloku attr_filter w pliku radiusd.conf, element konfiguracji attrsfile wskazuje plik z deklaracją działania filtru (domyślnie plik o nazwie attrs). Domyślne działanie oznacza usunięcie z odpowiedzi wszystkich atrybutów oprócz tych, dla których określono działanie w pliku z filtrem. Domyślnie w konfiguracji jest zdefiniowanych kilka wystąpień tego modułu, przeznaczonych do użycia na różnych etapach pracy serwera (po uwierzytelnieniu, przed przekazaniem do innego serwera itp.). Celem takiego filtra mogłoby być wycięcie VLAN-ów ustawionych na obcym serwerze i ustawienie VLAN-u lokalnego. Funkcjonalność modułu attr_filer począwszy od wersji 2 oprogramowania można z powodzeniem zastąpić odpowiednimi instrukcjami języka unlang.

Moduły perl i python

Moduły te pozwalają wskazać skrypt przygotowany w danym język, który może służyć np. do filtrowania pakietu, realizacji uwierzytelnienia, autoryzacji itp. Definicja, np. w przypadku modułu perl polega na dodaniu sekcji perl o postaci:

```
perl {
  module = "/opt/FreeRADIUS/perl/vlans.pl"
  func_post_auth = "post_auth"
  func_post_proxy = "post_proxy"
 }
```

Skrypt /opt/FreeRADIUS/perl/vlans.pl realizuje wszystkie potrzebne zadania.

W tym przypadku elementy konfiguracyjne func_post_auth, func_post_proxy podają nazwy funkcji Perl zdefiniowanych dla odpowiednich sekcji.

Język unlang na ogół pozwala w prostszy i efektywniejszy sposób (bez uruchamiania zewnętrznych programów) dostosować postać pakietów do potrzeb.

Sekcja instantiate

Sekcja ta ustala kolejność ładowania modułów – moduły wymienione w niej są ładowane przed tymi, które są zdefiniowane w sekcjach: authorize, authenticate, preacct, accounting, session, post-auth, pre-proxy i post-proxy.

Sekcja authorize

Sekcja ustala sposób autoryzacji użytkowników. Przykładowa postać to:

```
authorize {
 preprocess
 checkuser
 Autz-Type pracownik {
   ldapstaff
 }
 Autz-Type student {
   ldapstud
 }
 eap
 files
}
```

Zdefiniowano typy autoryzacji Autz-Type pracownik i student, powiązane z wystąpieniami modułu ldap.

Autoryzacja odbywa się na podstawie typu określonego przez moduł eap oraz na podstawie modułu files.

Sekcja authenticate

Sekcja ustala sposób przebiegu uwierzytelnienia. Przykładowa postać to:

```
authenticate {
 eap
 Auth-Type pracownik {
   ldapstaff
 }
 Auth-Type student {
   ldapstud
 }
 Auth-Type MS-CHAP {
  mschap
 }
}
```

Zdefiniowano typy uwierzytelniania Auth-Type pracownik i student, powiązane z wystąpieniami modułu ldap oraz z modułem mschap. Uwierzytelnienie odbywa się na podstawie typu określonego przez moduł eap.

Sekcja post-auth

W sekcji tej domyślnie jest jedynie wywoływany moduł attr_filter w celu skasowania większości atrybutów w odpowiedzi Access-Reject. Można w niej uaktywnić reply_log, aby logować wszystkie odpowiedzi związane z uwierzytelnieniem, można tu też dodać wywołanie modułu perl w celu usunięcia ustawionych wcześniej VLAN-ów w przypadku, gdy Access-Accept z naszego serwera jest kierowany poza sieć danego dostawcy.

Plik postauth_users, używany w fazie post-auth może zawierać wpisy związane z modyfikacją pakietu Access-Accept:

```
test@test.pl Huntgroup == "stolowka"
       Tunnel-Private-Group-Id := 48
       Tunnel-Medium-Type = 6
       Tunnel-Type = VLAN
```

W tym przypadku użytkownik test@test.pl będzie miał ustawiony VLAN 48.

Według podobnej zasady ustala się zawartość sekcji pre-proxy i post-proxy, aby ustalić zachowanie na etapie obsługi pakietu przed i po przekierowaniu.

Katalogi sites-available/ i sites-enabled

Na końcu głównego pliku konfiguracyjnego FreeRADIUS-a – radiusd.conf znajduje się dyrektywa INCLUDE włączająca pliki definiujące serwery wirtualne. Katalog podany w dyrektywie jest przeszukiwany w celu włączenia wszystkich plików o nazwie zgodnej z wyrażeniem regularnym:

```
/[a-zA-Z0-9_.]+/
```

Dyrektywa domyślnie ma postać:

```
$INCLUDE sites-enabled/
```

Zakłada się, że w katalogu `etc/raddb/sites-available` instalacji FreeRADIUS znajdują się definicje dotyczące obsługi serwerów wirtualnych, natomiast w katalogu `etc/raddb/sites-enabled` są utworzone dowiązania do tych plików. Potrzebna struktura jest tworzona w procesie instalacji pakietu.

Serwer wirtualny nie mający nazwy jest przyjmowany za domyślny – po instalacji definicja ta znajduje się w pliku `default`. W ramach serwerów wirtualnych są definiowane sekcje `authorize{...}, authenticate{...}, pre-proxy{...}, post-proxy{...}, post-auth{...}`.

Informacje na temat sposobu korzystania z serwerów wirtualnych znajdują się w części „Serwery wirtualne".

Plik eap.conf

Plik tej jest włączany do konfiguracji w ramach pliku `radiusd.conf`. Zawiera definicję modułu `eap`.

Ustala `default_eap_type` (`peap, ttls, tls`), domyślny typ EAP-a, definiuje wspierane typy EAP, w postaci zagnieżdżonych sekcji `md5, leap, peap, tls, ttls, gtc`. Nie jest rekomendowane stosowanie uwierzytelniania EAP-MD5 i LEAP, dlatego te sekcje są puste. W projekcie eduroam zaleca się stosowanie metod EAP-TLS, EAP-TTLS i EAP-PEAP. W sekcji `peap` należy umieścić następujące dyrektywy:

```
peap {
 default_eap_type = mschapv2
 copy_request_to_tunnel = yes
 use_tunneled_reply = yes
}
```

Parametr `copy_request_to_tunnel = yes` powoduje, że atrybuty nie zdefiniowane w tunelu, a obecne poza tunelem są kopiowane.

Parametr `use_tunneled_reply = yes` – powoduje, że atrybuty wysyłane w odpowiedzi nie wiążą się z użytkownikiem widzianym na zewnątrz tunelu *(outer identity)*, lecz tym, który został określony wewnątrz tunelu *(inner identity).*

Podobną postać ma sekcja `ttls`:

```
ttls {
 default_eap_type = md5
 copy_request_to_tunnel = yes
 use_tunneled_reply = yes
}
```

Zarówno w sekcji `peap`, jaki i `ttls` można umieścić dyrektywę wskazującą, jaki serwer wirtualny ma obsługiwać zlecenia wewnątrz tunelu. Służy do tego wiersz:

`virtual_server = "nazwa"`

Sekcja `tls` definiuje zasady bezpiecznej komunikacji. Najistotniejsze elementy to:

`private_key_file = ścieżka_do_pliku_z_kluczem_prywatnym_serwera`

`private_key_password = hasło_do_klucza_prywatnego`

`certificate_file =ścieżka_do_pliku_z_certyfikatem_serwera`

`CA_file = ścieżka_do_pliku_z_certyfikatem_urzędu`

W ramach sekcji `tls` można ustalić, czy mają być sprawdzane:

- lista odwołanych certyfikatów (`check_crl=yes/no`),
- pole wystawcy certyfikatu w certyfikacie użytkownika (`check_cert_issuer="DN_wystawcy"`),
- nazwa CN w certyfikacie użytkownika (`check_cert_cn="%{User-Name}"`).

Aby mogły być używane metody kryptograficzne należy utworzyć plik DH, co jest realizowane poprzez polecenie:

`openssl dhparam -out certs/dh 1024`

Plik certs/dh należy wskazać w wierszu dh_file sekcji tls. Poza tym konieczne jest wskazanie pliku random w wierszu random_file, najlepiej użyć po prostu /dev/random.

Certyfikat dla serwera FreeRADIUS

W pliku eap.conf wskazujemy certyfikat serwera, klucz prywatny serwera oraz certyfikat urzędu certyfikacyjnego, który poświadczył certyfikat serwera.

W zleceniu certyfikacji związanym z kluczem publicznym serwera FreeRADIUS w miejscu commonName należy podać pełną kwalifikowaną nazwę serwera.

Urząd wystawiający certyfikat MUSI w certyfikacie serwera dodać rozszerzenie TLS Web Server Authentication. Korzystając z plecenia openssl, można to zrobić następująco:

```
openssl ca -out s.crt -in s.csr -extensions xpserver_ext \
            -exfiles xpextension
```

Plik xpextensions ma postać:

```
[xpserver_ext]
extendedKeyUsage = 1.3.6.1.5.5.7.3.1
```

Certyfikaty klientów i korzystanie z listy unieważnionych certyfikatów

Jeżeli używamy uwierzytelniania EAP-TLS klient musi się dysponować certyfikatem zawierającym rozszerzenie TLS Web Client Authentication. Używając poleceń openssl można zrobić to następująco:

```
openssl ca -out c.crt -in c.csr -extensions xpclient_ext \
            -exfiles xpextension
```

Plik xpextensions w tym przypadku ma postać:

```
[xpclient_ext]
extendedKeyUsage = 1.3.6.1.5.5.7.3.2
```

Plik c.csr zawiera zlecenie certyfikacji klucza publicznego klienta. Zazwyczaj zlecenia takie mają w polu commonName nazwę użytkownika w domenie, np. commonName = user@a.test.pl.

Jeżeli w pliku eap.conf wskazano korzystanie z list unieważnionych certyfikatów (CRL), serwer sprawdza, czy nazwa umieszczona w polu Subject nie znajduje się na liście CRL.

Certyfikat kliencki w postaci PEM, umieszczony w pliku c.crt dodajemy do listy odwołanych certyfikatów używając polecenia:

```
openssl ca -revoke c.crt
```

Odnowienie listy CRL następuje po wykonaniu polecenia:

```
openssl ca -gencrl -out crl.pem
```

Następnie plik crl.pem umieszczamy w katalogu wskazanym w konfiguracji w pliku eap.conf, w bloku tls, w wierszu certdir i wykonujemy polecenie:

```
c_rehash katalog_wskazany_w_certdir
```

Serwery wirtualne

Zastosowanie wirtualnych serwerów może być bardzo różnorodne. Podstawową zasadą tej funkcjonalności jest możliwość stosowania różnych polityk obsługi zleceń, w zależności od określonej specyfiki zlecenia. Serwery wirtualne pozwalają określić sposób postępowania na podstawie adresu IP serwera, adresu IP klienta, puli, do jakiej należy serwer, czy na podstawie faktu, że mamy do czynienia z wewnętrznym tunelem TLS.

Serwer FreeRADIUS może nasłuchiwać na wielu adresach IP i stosować odrębną politykę do pakietów przychodzących na dany adres. W sekcji `listen {..}` w `radius.conf` można podać nazwę serwera wirtualnego, który ma być powiązany z danym adresem IP. Można również wskazać jacy klienci mają być obsługiwanie przez dany serwer wirtualny. Jest również możliwe zdefiniowanie puli serwerów (`home_server_pool`) i wskazanie serwera wirtualnego obsługującego daną pulę. Bardzo przydatną funkcjonalnością jest zastosowanie dedykowanego serwera wirtualnego do obsługi wewnętrznego tunelu TLS. W tym celu w pliku `eap.conf`, w sekcjach `peap` i `ttls` definiujemy serwer w wierszu `virtual_server`. Jeśli np. metody EAP-TTLS i EAP-TLS uwierzytelniają użytkowników w oparciu o bazę LDAP, to tylko w ramach danego serwera wirtualnego należy wskazać odwołania do modułów `ldap`. Ponadto, w sekcji `post-auth` serwera wirtualnego, można w sposób bardzo wygodny zrealizować zamianę identyfikatora użytkownika zewnętrznego na nazwę pochodzącą z wewnątrz tunelu, lub ustaloną na podstawie pobranych w czasie uwierzytelniania atrybutów dodatkowych – szczegóły znajdują się w kolejnej części opracowania.

Nazwa użytkownika w pakiecie Access-Accept

W przypadku uwierzytelnienia przy użyciu metody EAP-TLS, po poprawnym uwierzytelnieniu standardowo jest odsyłany pakiet `Access-Accept` zawierający w polu `User-Name` nazwę, która pojawiła się w zleceniach `Access-Request` w polu `User-Name`. Po uwierzytelnieniu EAP-TTLS w pakiecie `Access-Accept` domyślnie pojawi się nazwa użytkownika występująca jako tożsamość zewnętrzna, w szczególności, w przypadku standardowego ustawienia programu klienckiego SecureW2 taka nazwa zawiera napis postaci `anonymus@domena`.

W celu umożliwienia realizacji funkcji monitorowania i rozliczania w usłudze eduroam, proponuje się korzystanie z atrybutu `Chargeable-User-Identity` do przekazywania informacji o użytkownika. W polu tym powinien zostać umieszczony unikatowy identyfikator użytkownika, umożliwiający jednoznaczne ustalenie użytkownika przez administratora serwera, w którym nastąpiło uwierzytelnienie. Identyfikator taki nie powinien przenosić takiej informacji o użytkowniku, która umożliwiłaby osobom postronnym identyfikację osoby.

Poniższej pokazano sposób ustawiania atrybutu `Chargeable-User-Identity`. Dokonujemy tego w sekcji `post-auth` serwera wirtualnego obsługującego tunel wewnętrzny. Ważne jest, że zgodnie z RFC 4372 atrybut ten może zostać ustawiony w odpowiedzi `Access-Accept` tylko wówczas, gdy taki właśnie atrybut wystąpił w zleceniu `Access-Request`.

Sprawdzamy więc, czy atrybut `Chargeable-User-Identity` wystąpił. Jeśli tak, gdy mamy do czynienia z domeną `a.test.pl` i jest określona wartość atrybutu `My1-User-Identity`, to przypisujemy atrybutowi `Chargeable-User-Identity` wartość tego atrybutu. Podobnie postępujemy dla domeny `b.test.pl`, przy czym korzystamy z innego atrybutu, zdefiniowanego jako `My2-User-Identity`.

```
post-auth {
 if (Chargeable-User-Identity ) {
  if ( Realm=="a.test.pl") {
   if (reply:My1-User-Identity) {
    update reply {
     Chargeable-User-Identity:="%{reply:My1-User-Identity}@%{Realm}"
    }
   }
  }
  if ( Realm=="b.test.pl") ) {
   if (reply:My2-User-Identity) {
    update reply {
     Chargeable-User-Identity:="%{reply:My2-User-Identity}@%{Realm}"
    }
   }
  }
 }
}
```

Powyższy przykład pokazuje jednocześnie zastosowanie wewnętrznych atrybutów FreeRADIUS – tu mają one nazwy `My1-User-Identity` i `My2-User-Identity`. Atrybuty te definiuje się w pliku `dictionary`, przykład znajduje się na końcu tego pliku. Atrybuty wewnętrzne muszą mieć numery od 3000 do 4000, np.

```
ATTRIBUTE       My1-User-Identity     3000    string
ATTRIBUTE       My2-User-Identity     3001    string
```

Jeżeli atrybutom takim ma być przypisywana wartość pochodząca z wpisu uwierzytelnionej osoby w bazie LDAP, to należy w pliku `ldap.attrmap` dopisać, np.

```
replyItem       My1-User-Identity    stafflid
replyItem       My2-User-Identity    studid
```

Oznacza to, że atrybut `My1-User-Identity` uzyska wartość atrybutu `staffid`, a atrybut `My2-User-Identity` uzyska wartość atrybutu `studid` w bazie LDAP.

Ustalenie VLAN-ów

W pakiecie `Access-Accept` mogą zostać wysłane atrybuty służące do ustalenia docelowego numeru VLAN, w którym użytkownik otrzyma adres. Można to zrealizować za pomocą odpowiednio przygotowanego pliku `postauth_users`. Prostszą metodą w wersji 2 FreeRADIUS-a jest użycie języka unlang w sekcji `post-auth`.

Poniższy kod:

```
if (("%{Packet-Src-IP-Address}" != "158.75.1.25") &&
     ("%{Packet-Src-IP-Address}" != "150.254.173.30") ) {
   if (Realm=="a.test.pl") {
    update reply {
     Tunnel-Private-Group-Id:=140
     Tunnel-Medium-Type:=6
     Tunnel-Type:=VLAN
    }
   }
   elsif (Realm=="b.test.pl") {
    update reply {
     Tunnel-Private-Group-Id:=141
     Tunnel-Medium-Type:=6
     Tunnel-Type:=VLAN
    }
   }
 }
```

Uwaga: operator := zamienia dotychczasowe ustawienia dla atrybutu, operator = dodaje nową wartość atrybutu o danym typie.

Numery VLAN są przypisywane tylko wówczas, gdy pakiet nie pochodzi z serwera poziomu PL.

Serwer RADIUS NIE MOŻE przekazywać na zewnątrz ustawień VLAN. Należy zwrócić uwagę, by nigdy nie były odsyłane atrybuty dotyczące VLAN-ów, gdy odpowiedź serwera jest przekazywana zwrotnie poza naszą sieć. Najlepiej w sekcji `post-proxy` umieścić kod:

```
if (Realm == "polska") {
  if ( Tunnel-Private-Group-Id ) {
    update reply {
      Tunnel-Private-Group-Id-=%{Tunnel-Private-Group-Id}
    }
   }
   if ( "Tunnel-Type" ) {
     update reply {
       Tunnel-Type-="%{Tunnel-Type}"
     }
    }
    if ( "Tunnel-Medium-Type" ) {
      update reply {
        Tunnel-Medium-Type-="%{Tunnel-Medium-Type}"
      }
    }
}
```

(atrybut `Realm` musi zawierać nazwę określającą obsługę przez serwer poziomu PL, można to zrealizować w pliku `users` poprzez dodanie do każdego zlecenia nie obsługiwanego lokalnie atrybutu `Proxy-To-Realm:=polska`).

Schemat RADIUS-LDAPv3 (fragment dystrybucji FreeRADIUS) definiuje klasę `radiusProfile`. W oparciu o ten schemat FreeRADIUS pozwala na dodanie we wpisie użytkownika w bazie LDAP informacji o VLAN-ie za pomocą atrybutów: `RadiusTunnelType`, `radiusTunnelMediumType`, `radiusTunnelPrivateGroupID`.

Obsługa rozliczania

Jeżeli urządzenia dostępowe typu Access Point mają włączoną funkcjonalność rozliczania i wskazują serwer FreeRADIUS jako miejsce obsługi rozliczania, to serwer poza pakietami Access-Request i Access-Challange odbiera pakiety rozliczeniowe, Accounting-Request. W domyślnym ustawieniu pakiety rozliczeniowe są logowane według zasad określonych w module detail. Można zdefiniować w sekcji preacct (obsługa wstępna przed rozliczeniem) korzystanie z modułu files. Wówczas przed przystąpieniem do rozliczania zostanie przeanalizowany plik „users" – domyślnie jest to acct_users, jeśli np. w pliku będzie wpis wskazujący, że pakiet ma zostać przekierowany do innego serwera, to takie przekierowanie nastąpi.

W eduroam obowiązuje zasada przechowywania informacji rozliczeniowej dotyczącej nielokalnych użytkowników na serwerze lokalnym, tak by administrator sieci, z której użytkownik korzysta mógł posiadać wszelkie informacje rozliczeniowe.

Popularnym rozwiązaniem dotyczącym obsługi rozliczania (ang. *accounting*) jest wykorzystanie bazy SQL. Domyślny plik konfiguracyjny to `sql.conf`. Zawiera on deklaracje dotyczące modułu `sql`. Są to:

deklaracja drivera, serwera, użytkownika i hasła do współpracy z bazą, nazwy bazy oraz definicje sposobu realizacji zapytań rozliczeniowych: `accounting_start_query`, `accounting_update_query`, `accounting_stop_query`.

Należy zainicjować bazę danych, utworzyć tablicę np. o nazwie `accounting`.

W tablicy umieszczamy pola, których dotyczą definicje zapytań `accounting_start_query`, `accounting_update_query`, `accounting_stop_query`.

W sekcji accounting dopisujemy wywołanie modułu sql.

Uruchomienie serwera FreeRADIUS

Jeśli konfiguracja znajduje się w domyślnej lokalizacji (etc/raddb), to startujemy serwer następująco:

```
radiusd &
```

Jeśli konfiguracja jest w innym katalogu, to musimy dodać opcję -d:

```
radiusd -d katalog_konfiguracji
```

Polecenie:

```
radiusd -X >& /tmp/freeradius&
```

uruchamia serwer w trybie debugowania, przy czym wyjście debugowania jest przekazywane do pliku /tmp/freeradius.

Jeśli nasz serwer nie pracuje w trybie debugowania, to informacje dotyczące pracy serwera będą dostępne w pliku $logdir/radius.log.

Integracja FreeRADIUS-a z Active Directory

Założenia są następujące:

- korzystamy z Active Directory (Windows Server 2000 / 2003) jako bazy użytkowników;
- użytkownicy mają zdefiniowany tryb 'allow access' (zakładka Dial-In we własnościach konta);
- rozszerzamy schemat poprzez dopisanie klasy radiusProfile oraz atrybutów dialupAccess, radiusGroupName, radiusTunnel*;
- rozbudowujemy wpis użytkownika w celu dodania nowych atrybutów.

Uwaga: możliwość zastosowania atrybutów klasy radiusProfile przy opisie kont użytkowników ustala własność klasy radiusProfile w ramach zakładki Relationship - jako 'possible superior' należy dodać klasę obiektów user.

W celu aktualizacji schematu Active Directory muszą być spełnione warunki:

- administrator lub użytkownik modyfikujący musi być w grupie (Member Of) Schema Admins;
- należy wykonać polecenie schmgmt.dll: Run... regsvr32 schmgmt.dll;
- otworzyć konsolę (mmc) i dodać przystawkę "Active Directory Schema";
- aby była możliwość zapisu aktualizacji niezbędna jest zmiana w rejestrze:

  dojść do

  ```
  HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\ Services\NTDS\Parameters
  Edit, New, DWORD Value,
     Value Name: Schema Update Allowed
     Data Type: REG_DWORD, Base: Binary, Value Data: 1
  ```

Przygotowanie Active Directory:
Konta użytkowników danego kontrolera domeny znajdują się w gałęzi cn=Users.

Korzeń drzewa to dc=a,dc=b,dc=c, jeśli Active Directory jest kontrolerem domeny a.b.c

Program ldifde pozwala importować i eksportować dane, domyślny tryb to eksport. Przykłady:

```
ldifde -d 'cn=users,dc=ad,dc=umk,dc=pl' -f u.ldif
```

w pliku u.ldif zostanie zapisany zrzut poddrzewa cn=users

```
ldifde -f user.ldif -i
```

import danych wg pliku user.ldif (wsad w postaci standardu LDIF, DN entry, definicja operacji (changetype: add/modify/ delete), atrybuty.

Dokument “FreeRADIUS Tutorial for AD Integration”, Ch. Schwartz
http://homepages.lu/charlesschwartz/radius/freeRadius_AD_tutorial.pdf
szczegółowo opisuje kolejne działania konfiguracyjne.

Definiujemy moduł ldap, np.:

```
ldap AD {
...
identity = DN_użytkownika z prawem_odczytu
password = "hasło_użytkownika"
start_tls = no
access_attr = "dialupAccess"
...
}
```

W sekcji `authorize` umieszczany odpowiedni `Autz-Type`, podobnie w sekcji `authenticate` umieszczamy `Auth-Type`.

W przypadku współpracy z Active Directory stosowanym typem uwierzytelnienia jest PEAP. Po stronie serwera Linux należy zdefiniować sposób dostępu do domeny Windows.

Korzystamy w tym celu z serwera Samba i narzędzi:

- `winbindd`: demon umożliwiający połączenie ze środowiskiem Windows,
- `ntlm_auth`: program współpracujący z `winbindd` do realizacji zlecenia NTLM.

```
AD.X.Y {
kdc = 192.168.3.33
default_domain = AD.X.Y
}
```

W pliku `/etc/krb5.conf` należy dodać domenę Windows.

Po stronie Samby, w pliku `smb.conf` umieszczamy w sekcji `global`:

```
[global]
workgroup = AD.UMK.PL
security = ads
...
password server = 192.168.3.33  // serwer AD
realm = AD.UMK.PL
```

Restartujemy serwer `smbd`.

W celu rejestracji serwera w domenie Windows wykonujemy:

```
net join -U administrator
```

Następnie uruchamiamy demona `winbindd`.

Wykonujemy próbę uwierzytelnienia przez NTLM:

```
ntlm_auth -request-nt-key -domain=AD.X.Y -username=xxx
```

Dostosowujemy konfigurację serwera FreeRADIUS:

- w sekcji `mschap` musi być wskazany `authtype = MS-CHAP`,
- odkomentowujemy definicje `with_ntdomain_hack = yes`
- definiujemy:

```
ntlm_auth = "/usr/bin/ntlm_auth -request-nt-key
             --domain=AD.UMK.PL -username=%{Stripped-User-Name}
             --challenge=%{mschap:Challenge}
             --nt-response=%{mschap:NT-Response}”
```

- w pliku `eap.conf` ustawiamy:
  `default_eap_type = peap`

w sekcji `tls` konieczna jest deklaracja certyfikatów (PEAP korzysta z zaszyfrowanego tunelu);

należy odkomentować sekcję `peap`

```
peap {
default_eap_type = mschapv2
}
```