Nazwa dokumentu:	Zastosowanie serwera RADIUS	Wersja dokumentu:	1.0a
Opiekun dokumentu:	TOBI		

Zastosowanie serwera RADIUS

Nazwa dokumentu:	Zastosowanie serwera RADIUS	Wersja dokumentu:	1.0a
Opiekun dokumentu:	TOBI		

Karta zmian dokumentu

LP	DATA	OPIS ZMIANY	KTO
1.	30.10.2005	Pierwsza wersja dokumentu	TOBI
2.	02.11.2005	1. Poprawki edytorskie 2. Rozpoczęcie rozdziału o PPPoE	TOBI
3.	08.11.2005	1. Rozbudowa rozdziału na temat PPPoE	TOBI

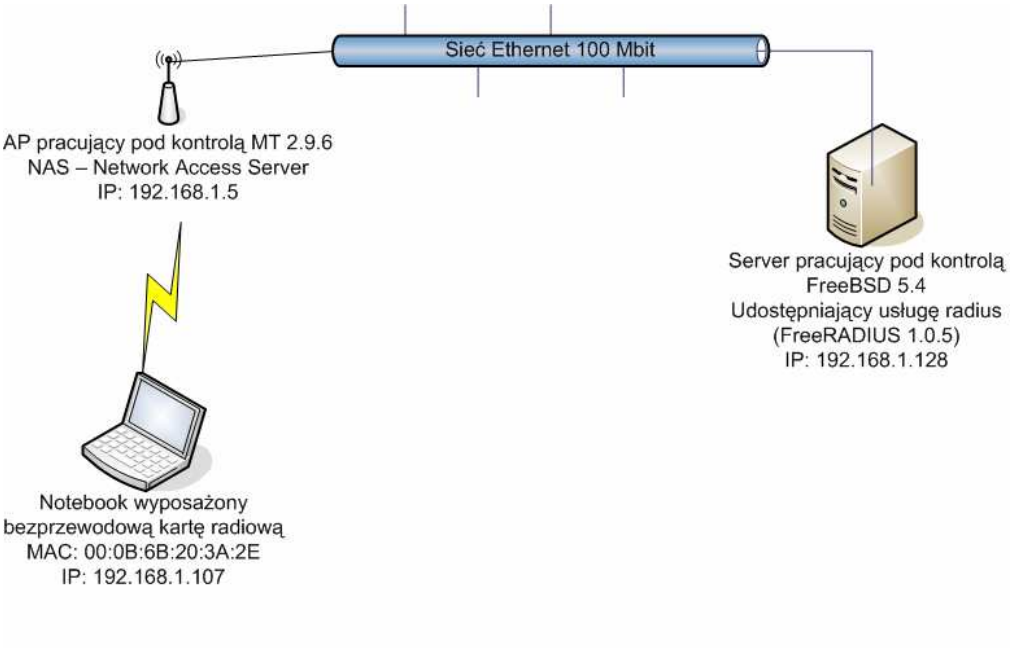
Nazwa dokumentu:	Zastosowanie serwera RADIUS	Wersja dokumentu:	1.0a
Opiekun dokumentu:	TOBI		

Wstęp

Niniejszy dokument stanowi instrukcję wdrożenia mechanizmu autoryzacji adresów MAC w serwerze Radius. Konfiguracja pracuje na wersjach:

- Mikrotik – v2.9.6
- FreeRADIUS - v1.0.5

Konfiguracja naszej laboratoryjnej sieci prezentuje się następująco:



Jako system operacyjny, pod kontrolą którego pracuje instalacja serwera Radius, wybrany został system FreeBSD – v5.4. Konfiguracja, jak i sama eksploatacja, nie różni się zasadniczo w porównaniu do innych systemów operacyjnych, dla których dostępny jest serwer FreeRADIUS.

Dokument zakłada, że użytkownik posiada elementarną wiedzę z zakresu konfiguracji systemu Mikrotik oraz posiada już zainstalowaną i uruchomioną wersję serwera FreeRADIUS.

Sprawdzamy to np. poleceniami:

```
bash-2.05b# ps auxc |grep radiusd
root  15869  0.0  1.5  4788  3376  ??  Is    8:52PM   0:00.00 radiusd
```

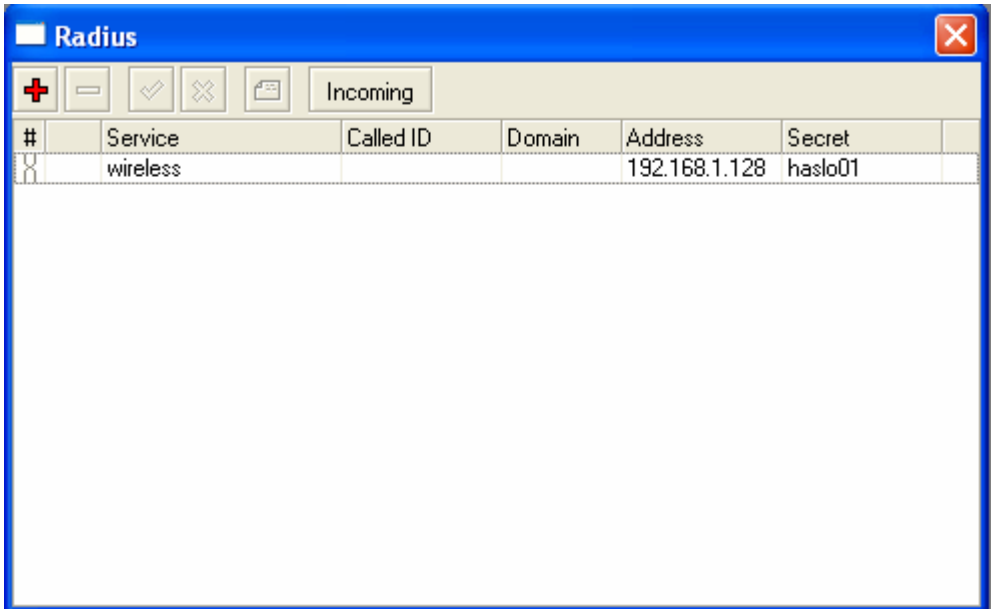
lub/oraz

```
bash-2.05b# tail /usr/local/var/log/radius/radius.log
Sun Oct 30 20:52:52 2005 : Info: Ready to process requests.
```

Konfiguracja Mikrotika 2.9.x do współpracy z RADIUSem

W celu uaktywnienia autoryzacji adresów MAC w serwerze Radius należy wykonać następujące czynności:

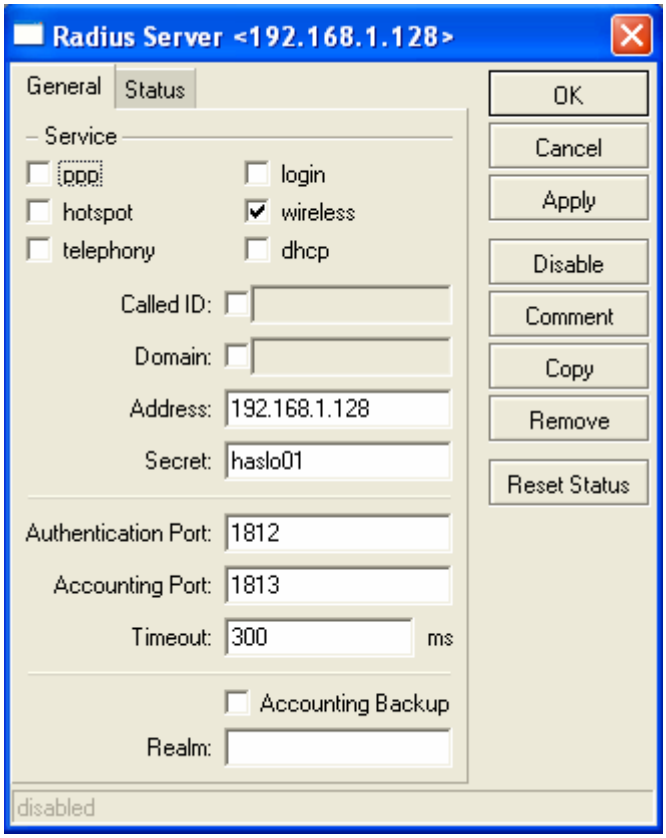
a. Zdefiniować serwer Radius jaki ma być wykorzystywany przez nasz system do autoryzacji



W następnym kroku po dodaniu wpisu za pomocą przycisku „+” pojawia się okienko pokazane w dalszej części tekstu. Przystępujemy do określenia właściwości serwera RADIUS, z jakiego mamy zamiar korzystać.

W pierwszej części *Service* zaznaczamy, jakie usługi mają być obsługiwane przez nasz serwer. Na potrzeby niniejszego rozdziału zaznaczamy jedynie *wireless* jednak jak widzimy usługa ta może dopomóc nam w takich aspektach jak *ppp,* obsługę *hotsopota*, konfigurację *dhcp* itd.

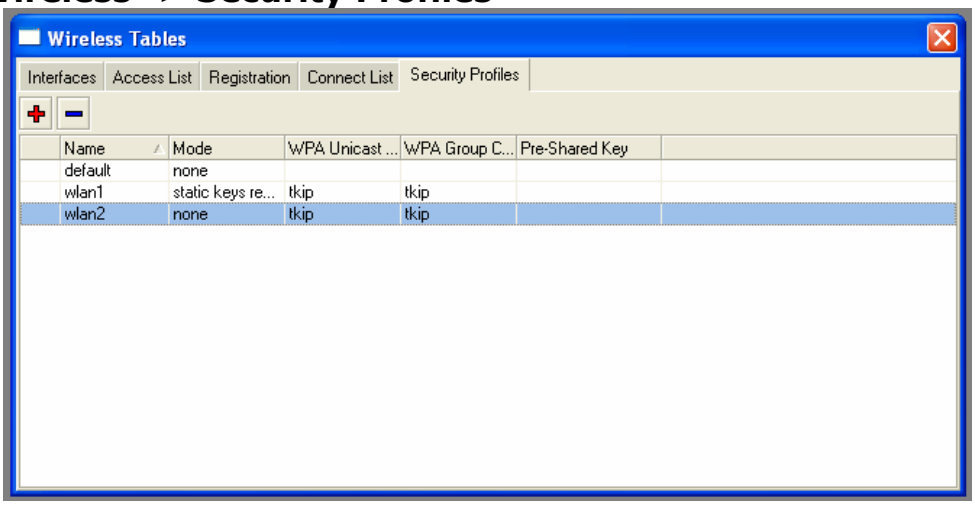
W dalszej części wypełniamy pole *Address*. Jest to adres IP naszego serwera RADIUS. Pole *Secret* zawiera hasło, jakie służyć będzie zarówno do autoryzacji klienta RADIUSA (pamiętajmy, że jest to nasz Mikrotik a nie klient końcowy naszej bazy) jak i podpisywania (uwierzytelnienia) przesyłanych wiadomości pomiędzy NASem a RADIUSem. To samo hasło pojawi się w dalszej części dokumentu przy okazji konfiguracji samego serwera RADIUS.



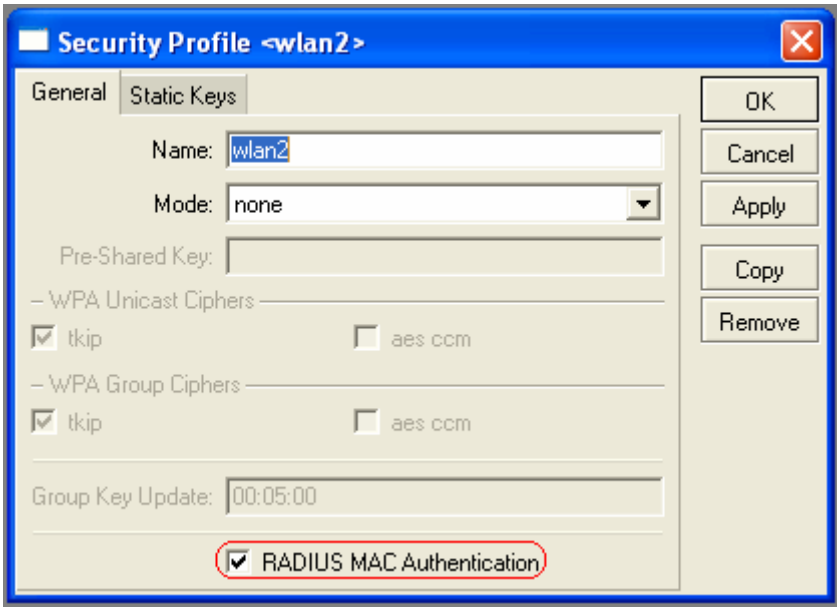
Resztę pól pozostawiamy na tym etapie bez zmian z wartościami domyślnymi.

b. Stworzyć odpowiedni *security profile* i przypisać go do odpowiedniego interfejsu radiowego.

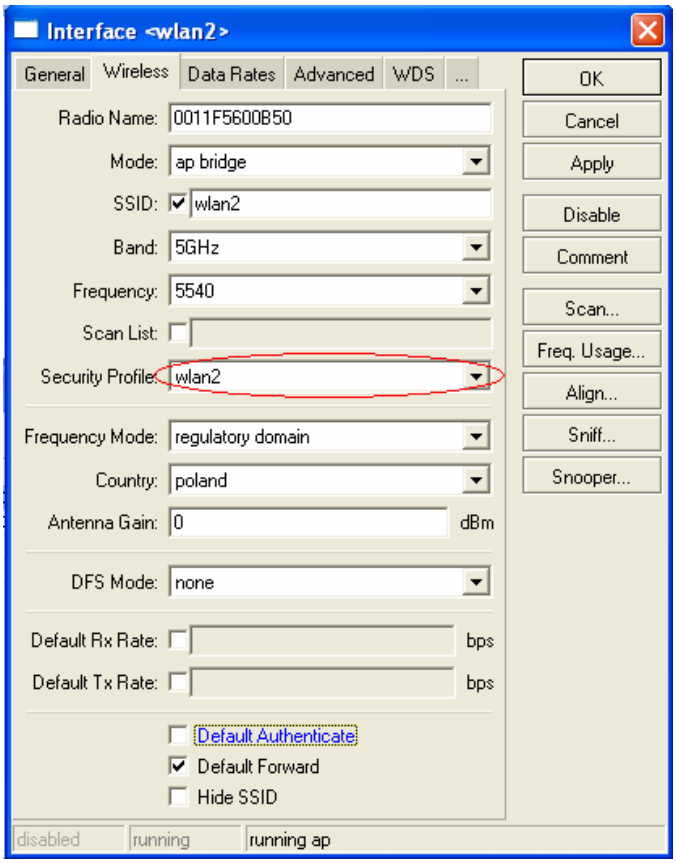
Wireless -> Security Profiles



W zakładce *General* należy zaznaczyć odpowiednią opcję zlecającą weryfikację adresów MAC w serwerze RADIUS.



Tak utworzony profil przypisuje do wybranego interfejsu



c. UWAGA!!! Adresy MAC, które mają podlegać autentykacji w serwerze RADIUS nie mogą znajdować się w lokalnym ACL. Obecność danego MAC adresu w lokalnym ACL powoduje, iż jest on „wpuszczany do systemu" bez względu na to czy jest on zdefiniowany w RADIUSie czy nie.

W celu zwiększenia czytelności, ułatwienia znalezienia ewentualnych błędów, zwiększymy sobie w Mikrotiku poziom logowania informacji związanych z RADIUSem do poziomu DEBUG korzystając z tego polecenia:

```
/system logging add topics=debug,packet,radius
```

W tej chwili każda próba podłączenia się klienta radiowego powinna wygenerować tego typu komunikat:

```
15:30:49 radius,debug,packet sending Access-Request with id 163 to 192.168.1.128:1812
15:30:49 radius,debug,packet     Signature = 0x1ed410426dccf44f4aaf4e404f87164a
15:30:49 radius,debug,packet     Service-Type = 2
15:30:49 radius,debug,packet     NAS-Port-Id = "wlan1"
15:30:49 radius,debug,packet     User-Name = "00:0B:6B:20:3A:2E"
15:30:49 radius,debug,packet     User-Password = 0x
15:30:49 radius,debug,packet     NAS-Identifier = "TechnoLogic"
15:30:49 radius,debug,packet     NAS-IP-Address = 192.168.1.5
15:30:49 wireless,debug RADIUS got failure for 00:0B:6B:20:3A:2E on wlan2
```

lub

```
15:49:19 radius,debug,packet sending Access-Request with id 196 to 192.168.1.128:1812
15:49:19 radius,debug,packet     Signature = 0xff9eb538ca9579304d02e252b544e30c
15:49:19 radius,debug,packet     Service-Type = 2
15:49:19 radius,debug,packet     NAS-Port-Id = "wlan1"
15:49:19 radius,debug,packet     User-Name = "00:0B:6B:20:3A:2E"
15:49:19 radius,debug,packet     User-Password = 0x
15:49:19 radius,debug,packet     NAS-Identifier = "TechnoLogic"
15:49:19 radius,debug,packet     NAS-IP-Address = 192.168.1.5
15:49:19 radius,debug,packet received Access-Reject with id 196 from 192.168.1.128:1812
15:49:19 radius,debug,packet     Signature = 0xd472ba8dc1fdbbcb3a3969f95312786c
15:49:19 wireless,debug RADIUS got failure for 00:0B:6B:20:3A:2E on wlan2
```

Różnica pomiędzy tymi dwoma komunikatami polega na tym, iż pierwszy ma miejsce w przypadku, gdy RADIUS na serwerze 192.168.1.128 nie jest uruchomiony, a drugi komunikat pojawia się gdy usługa RADIUS działa jednak klient od adresie MAC `00:0B:6B:20:3A:2E` został odrzucony.

Konfiguracja RADIUSA

Przejdziemy teraz do konfiguracji samej usługi RADIUS.

UWAGA!!!
Każdorazowo, po zmianie plików konfiguracyjnych należy wykonać restart serwera ***radiusd***, wysyłając sygnał HUP do odpowiedniego *daemona*, np.:

```
kill -HUP `cat /usr/local/var/run/radiusd/radiusd.pid `
```

lub

```
bash-2.05b# ps auxc |grep radiusd
root  50132  0.0  1.5  4688  3324  ??  Ss    9:11PM   0:00.27 radiusd
bash-2.05b# kill -HUP 50132
```

w efekcie powinniśmy zobaczyć w naszym logu podobny opis:

```
Sun Oct 30 21:37:54 2005 : Info: Reloading configuration files.
Sun Oct 30 21:37:54 2005 : Info: Ready to process requests.
```

PORADA
W celu śledzenia działania daemona radiusd w trybie online, uruchamiamy do w trybie "single server" poleceniem

radiusd –X

Taki tryb działania pozwala nam śledzić zachowanie daemona oraz przychodzące do niego pakiety radiusowe z naszego Mikrotika.

Plik konfiguracyjny, jaki w pierwszym kroku musimy w odpowiedni sposób zmodyfikować to:

`/usr/local/etc/raddb/clients.conf`

Może on, w zależności od systemu operacyjnego, znajdować się w innej lokalizacji, jak np. `/etc/raddb`. Plik ten zawiera definicję klientów naszego serwera RADIUS.

UWAGA!!!
Klienci definiowani w tym pliku to nie końcowe urządzenia, jakie mamy zamiar autoryzować (np. klienci radiowi naszej sieci WIFI). Urządzenia te to Network Access Server'y (NAS) – urządzenia wykorzystywane w sieci

do kontrolowania dostępu, np. stacje bazowe pracujące pod kontrolą Mikrotik'a.

```
client 192.168.1.5/32 {
       secret          = haslo01
       shortname       = TechnoLogicWIFI
}
```

Dodajemy tym wpisem NAS o adresie IP 192.168.1.5. Oczywiście możliwe jest dodanie całej sieci NAS np. 192.168.1.0/28. Hasło podane w polu *secret* to właśnie hasło, jakie podaliśmy w Mikrotiku definiując używany serwer Radiusa.

W następnym kroku popracujemy nad plikiem *users* znajdującym się w tym samym katalogu co plik *clients.conf*. Ten plik służy właśnie do definiowania naszych końcowych użytkowników. To tutaj dodaje się adresy MAC kart radiowych naszych klientów.

Zanim jednak rozpoczniemy, należy zmodyfikować już istniejący zapis w tym pliku.

Znajdujemy pierwszy wpis DEFAULT opatrzony poniższym komentarzem.

```
#
# First setup all accounts to be checked against the UNIX /etc/passwd.
# (Unless a password was already given earlier in this file).
#
DEFAULT     Auth-Type = System
            Fall-Through = yes
```

Znaczenie tego wpisu jest takie, iż serwer RADIUS ma uwierzytelniać użytkowników bazując na systemowym pliku haseł. W naszym przypadku, działanie to nie jest tym czego oczekujemy. Dlatego wyłączamy takie zachowanie poprzez wykomentowanie fragmentu tekstu zaznaczonego pogrubioną czcionką:

```
#
# First setup all accounts to be checked against the UNIX /etc/passwd.
# (Unless a password was already given earlier in this file).
#
#DEFAULT    Auth-Type = System
#           Fall-Through = yes
```

Po tym zabiegu przystępujemy do dodawania naszych użytkowników.

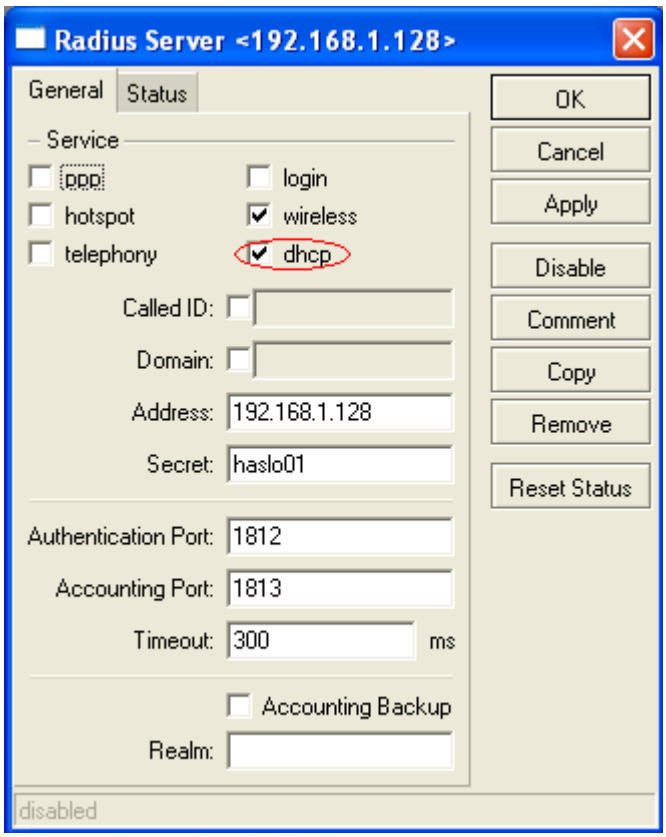
Ta sekcja składa się z zestawu swego rodzaju rekordów. Pierwsze pole to „nazwa użytkownika". Oczywiście w naszym przypadku jest to właśnie

MAC adres karty sieciowej. Kolejne pola (co ważne – znajdujące się w tej samej linii) oddzielone przecinkami zawierają listę atrybutów i ich wartości wymaganych do autentykacji danego użytkownika. Mogą to być takie pola i wartości jak np. hasło, nazwa karty radiowej Mikrotika, nazwa samego Access Pointa itp.
W kolejnych liniach, rozpoczynających się od znaku tabulatora, znajdują się atrybuty i ich wartości, jakie po pozytywnym zweryfikowaniu użytkownika przez serwer RADIUS zostaną odesłane do NAS'a. Mogą to być np. informacje niezbędne dla serwera PPP. Lub adres IP i maska jaką przyzna serwer DHCP w Mikrtotiku, np.:

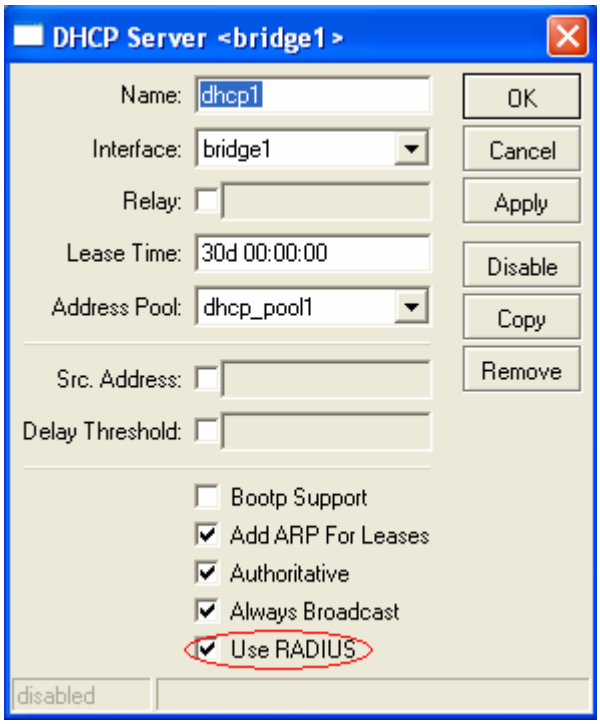
```
"00:0C:76:71:04:D9" User-Password == ""
        Framed-IP-Address = 192.168.1.113,
        Framed-IP-Netmask = 255.255.255.0
```

Ten wpis spowoduje oczywiście pozytywne uwierzytelnienie karty o adresie MAC „:04:D9" i wysłanie do serwera DHCP pracującego na naszej bazie informacji o tym, iż temu klientowi należy przyznać adres IP 192.168.1.113/24. Aby ta funkcjonalność zadziałała należy uaktywnić obsługę DHCP<->RADIUS w Mikrotiku w dwóch miejscach:

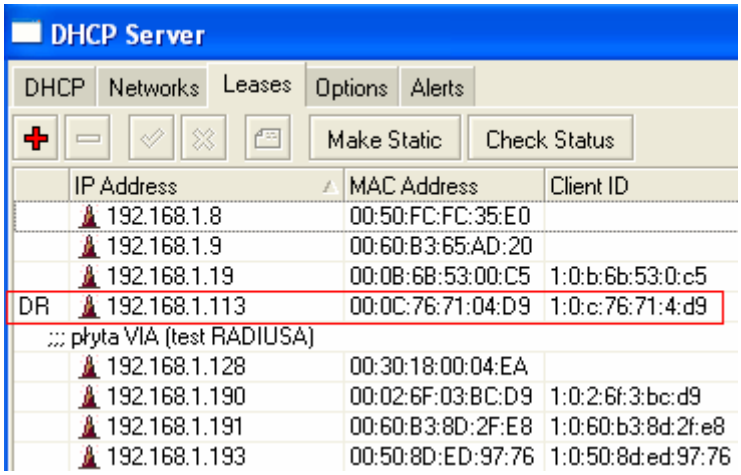
1. Serwer RADIUS



2. Serwer DHCP



Po tym zabiegu widzimy, że serwer DHCP wykorzystał informację zwrotną z RADIUSa i przypisał skonfigurowany przez administratora adres IP.



Świadczy o tym oznaczenie „DR” przy naszym przykładzie (dynamic, radius).

Lista innych atrybutów jakie mogą być obsługiwane przez serwer RADIUS i system Mikrotik znajduje się w **Załączniku 1**.

Minimalna forma wpisu jaka jest niezbędna do kontynuowania naszego tematu to:

```
"00:0B:6B:20:3A:2E" User-Password == ""
```

weryfikuje ona jedynie, że klient o podanym adresie MAC jest nam znany i nie stawiamy żadnych innych wymagań względem tego użytkownika.

Bardziej rozbudowana forma to np:

```
"00:0B:6B:20:3A:2E" User-Password == "", NAS-Port-Id == "wlan2"
```

zezwalająca zalogowanie się użytkownika pod warunkiem że loguje się do karty o nazwie wlan2 w naszym Mikrotiku.

Zobaczmy, co dzieje się w logach naszego Mikrotika po dodaniu tej linii i uruchomieniu demona ponownie:

```
19:17:05 wireless,debug send RADIUS request for 00:0B:6B:20:3A:2E on wlan2
19:17:05 radius,debug,packet sending Access-Request with id 255 to 192.168.1.128:1812
19:17:05 radius,debug,packet     Signature = 0x2bc31313bc7d92504284324894506f5a
19:17:05 radius,debug,packet     Service-Type = 2
19:17:05 radius,debug,packet     NAS-Port-Id = "wlan2"
19:17:05 radius,debug,packet     User-Name = "00:0B:6B:20:3A:2E"
19:17:05 radius,debug,packet     User-Password = 0x
19:17:05 radius,debug,packet     NAS-Identifier = "TechnoLogic"
19:17:05 radius,debug,packet     NAS-IP-Address = 192.168.1.5
19:17:05 radius,debug,packet received Access-Accept with id 255 from 192.168.1.128:1812
19:17:05 radius,debug,packet     Signature = 0xbd8e5a183762674f8d26b28425894f31
19:17:05 radius,debug,packet     Framed-IP-Address = 255.255.255.254
19:17:05 radius,debug,packet     Framed-MTU = 576
19:17:05 radius,debug,packet     Service-Type = 2
19:17:05 wireless,debug RADIUS got accept for 00:0B:6B:20:3A:2E on wlan2
19:17:05 wireless,info wlan2: 00:0B:6B:20:3A:2E connected
```

a oto co dzieje się po stronie naszego daemona radiusd:

```
rad_recv: Access-Request packet from host 192.168.1.5:1411, id=255, length=89
        Service-Type = Framed-User
        NAS-Port-Id = "wlan1"
        User-Name = "00:0B:6B:20:3A:2E"
        User-Password = ""
        NAS-Identifier = "TechnoLogic"
        NAS-IP-Address = 192.168.1.5
  Processing the authorize section of radiusd.conf
modcall: entering group authorize for request 0
  modcall[authorize]: module "preprocess" returns ok for request 0
  modcall[authorize]: module "chap" returns noop for request 0
  modcall[authorize]: module "mschap" returns noop for request 0
    rlm_realm: No '@' in User-Name = "00:0B:6B:20:3A:2E", looking up realm NULL
    rlm_realm: No such realm "NULL"
  modcall[authorize]: module "suffix" returns noop for request 0
  rlm_eap: No EAP-Message, not doing EAP
  modcall[authorize]: module "eap" returns noop for request 0
    users: Matched entry DEFAULT at line 171
    users: Matched entry 00:0B:6B:20:3A:2E at line 215
  modcall[authorize]: module "files" returns ok for request 0
modcall: group authorize returns ok for request 0
auth: type Local
auth: user supplied User-Password matches local User-Password
```

```
Login OK: [00:0B:6B:20:3A:2E] (from client TechnoLogicWIFI port 0)
Sending Access-Accept of id 255 to 192.168.1.5:1411
        Framed-IP-Address = 255.255.255.254
        Framed-MTU = 576
        Service-Type = Framed-User
Finished request 0
```

Jak widzimy nasza praca dobiegła końca z wynikiem pozytywnym.

Logowanie aktywności usługi radiusd

TODO

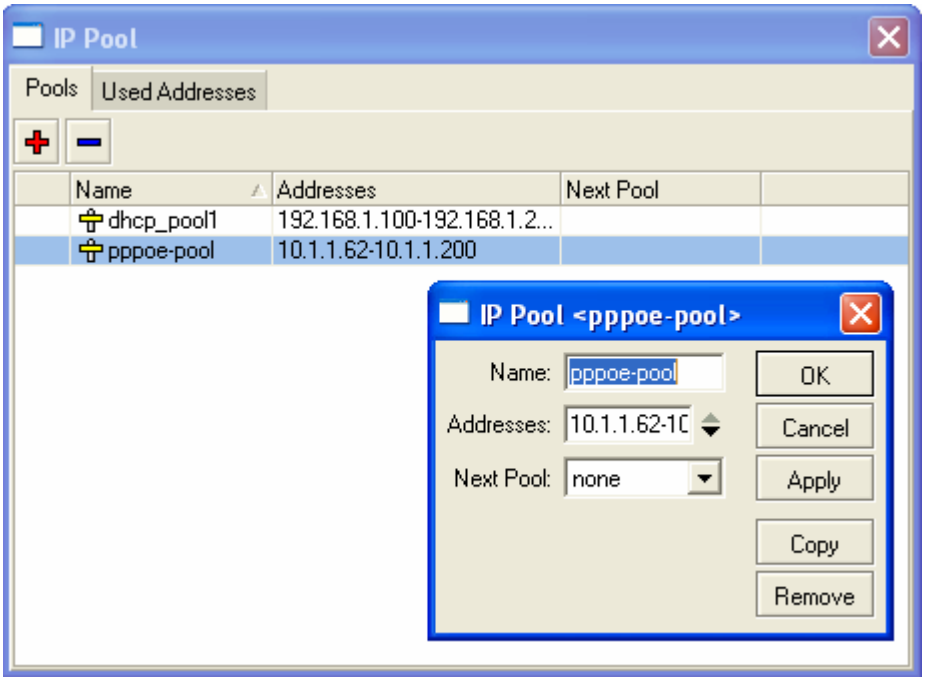
Konfiguracja połączenia PPPoE

Protokół PPPoE (Point to Point Protocol over Ethernet) dostarcza rozbudowanych mechanizmów zarządzania użytkownikami, zasobami sieciowymi a także dostarcza informacji niezbędnych do billingu. Protokół ten stanowi rozwinięcie protokołu PPP z tą różnicą, iż zastosowany w nim transport bazuje na warstwie Ethernet podczas gdy protokół PPP operuje bezpośrednio na połączeniu modemowym.

Działająca sesja PPPoE składa się ze stacji klienckiej np.: komputer pracujący pod kontrolą systemu Windows oraz serwera występującego w roli koncentratora PPPoE. Komputery osobiste działające pod kontrolą Windows XP posiadają wbudowanego klienta PPPoE w system. Starsze wersje tego systemu należy „dozbroić" w klienta np. popularne oprogramowanie RASPPPoE (*uwaga: prosimy zapoznać się z warunkami licencji wybranego oprogramowania klienckiego)*.

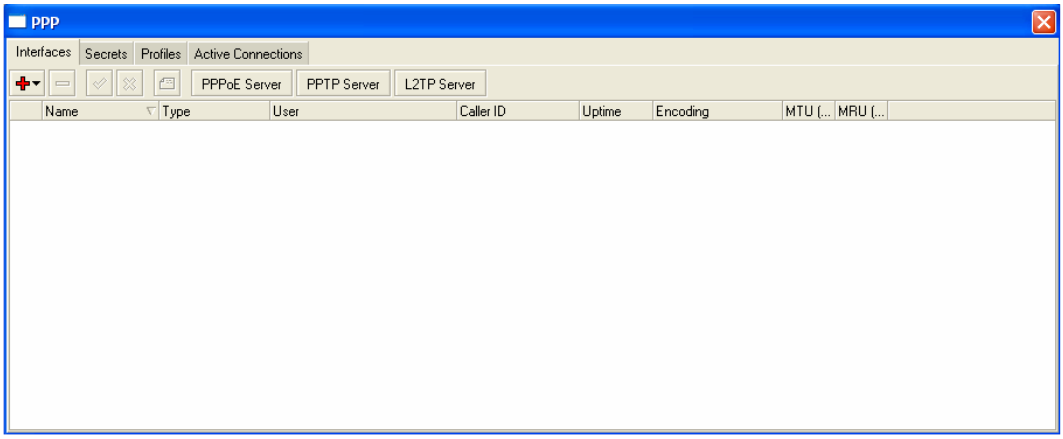
Przygotowanie Access Pointa - Mikrotik

Rozpoczniemy od dodania dedykowanej puli adresowej jaka wykorzystywana będzie do przypisywania adresów IP komputerom łączącym się poprzez PPPoE. Robimy to poprzez polecenia IP -> Pool.

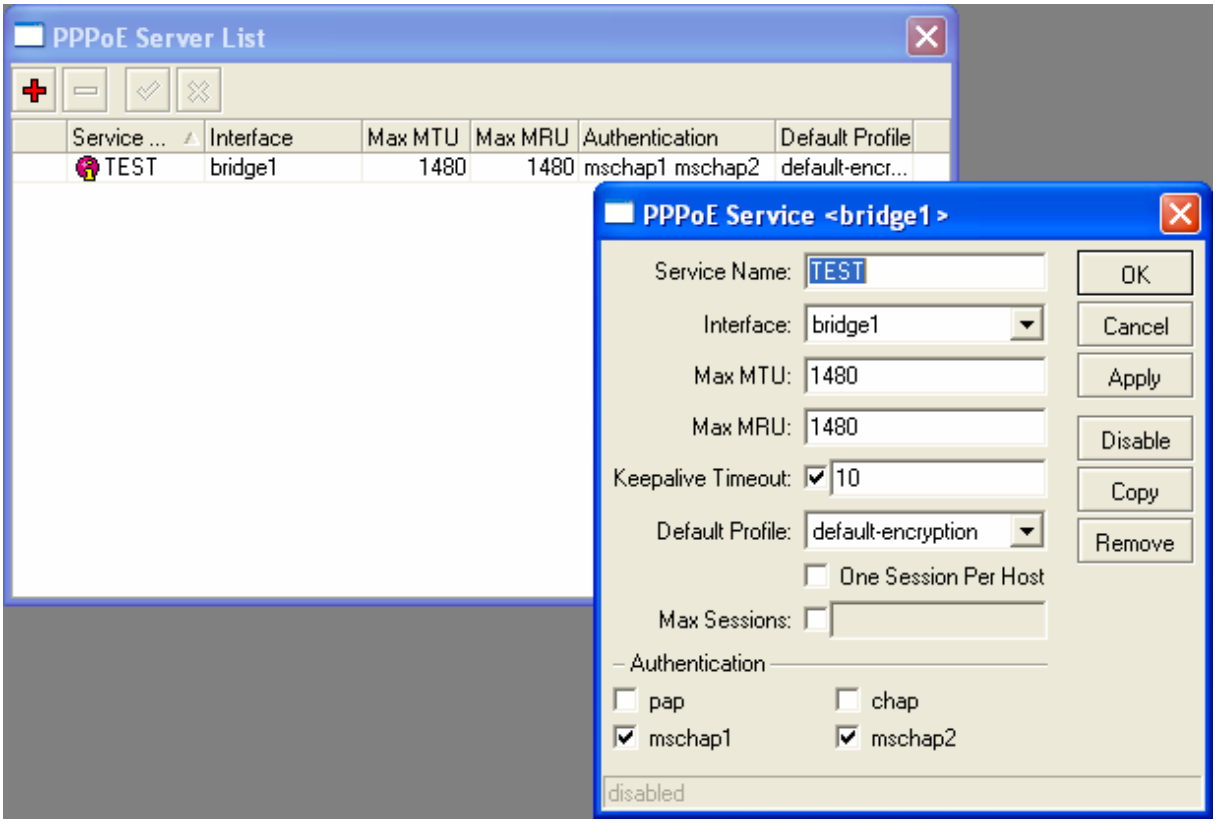


Zgodnie z przykładem dodajemy poolę adresową 10.1.1.62-10.1.1.200.

Następnie otwieramy okienko PPP.



Korzystając z klawisza „**PPPoE Server**” otwieramy okienko PPPoE Server List. Dodajemy nasz nowy serwer PPPoE i dokonujemy jego konfiguracji, jak na przykładzie.



Kolejne wartości oznaczają:

Service Name – nazwa
Interface – nazwa interfejsu, na którym ma być aktywny nasz serwer PPPoE. W naszym przypadku jest to interfejs bridge1 co gwarantuje nam iż PPPoE będzie widziane na wszystkich interfejsach dodanych do naszego bridge'a
Default Profile – nazwa profilu PPPoE jaki dany serwer będzie wykorzystywał. Do konfiguracji profili PPPoE przejdziemy w następnych krokach
Authentication – lista aktywnych metod uwierzytelniania jakie nasz serwer będzie wspierał. Ich opis i sposób wykorzystania w połączeniu z serwerem RADIUS znajdzie się w następnych częściach dokumentu. Zwróćmy na razie uwagę na fakt „odznaczenia" metody PAP, jako że nie gwarantuje ona wystarczającego poziomu bezpieczeństwa w sieci.

Przygotowanie serwera RADIUS

Plik RADIUSD.CONF

Rozpoczniemy od konfiguracji odpowiednich opcji w pliku radiusd.conf.

Znajdźmy w tym pliku sekcję rozpoczynającą się od

```
# MODULE CONFIGURATION
```

Służy ona do konfiguracji modułów jakie są wykorzystywane przez FreeRADIUS'a.

Tutaj podobnie jak w trakcie konfiguracji AP na Mikrotiku wyłączamy protokół PAP poprzez wstawienie znaku „#" w całej sekcji konfiguracyjnej tego protokołu.

```
#pap {
#       encryption_scheme = crypt
#}
```

Wyłączamy także protokół CHAP z uwagi na to iż jest on potencjalnie prosty do złamania przez atakującego naszą sieć

```
# CHAP module
#
#  To authenticate requests containing a CHAP-Password attribute.
#
#chap {
#       authtype = CHAP
```

```
#}
```

Postępujemy tak ze wszystkimi modułami jakie chcemy wyłączyć. Zostawiamy moduł **mschap** gdyż na nim skupimy naszą uwagę w dalszej części dokumentu.

```
# Microsoft CHAP authentication
        #
        #  This module supports MS-CHAP and MS-CHAPv2 authentication.
        #  It also enforces the SMB-Account-Ctrl attribute.
        #
        mschap {
                authtype = MS-CHAP
                use_mppe = yes
                require_encryption = yes
                require_strong = yes
                with_ntdomain_hack = no
}
```

…co ważne; to jedynie przykładowa konfiguracja, użytkownik może sam zadecydować jaki algorytm i jakie parametry są odpowiednie w jego środowisku. Wyłączenie modułów w tej sekcji powoduje, że serwer nie ładuje ich przy starcie. Musimy jeszcze wskazać w odpowiednich sekcjach pliku konfiguracyjnego, że nie chcemy wykorzystywać tych algorytmów.

Robimy to w sekcji *authenticate* oraz *authorize*

```
authenticate {
        #Auth-Type PAP {
        #       pap
        #}

        #Auth-Type CHAP {
        #       chap
        #}

#  MSCHAP authentication.
        Auth-Type MS-CHAP {
                mschap
        }


#       digest
#       pam
#       unix
#       Auth-Type LDAP {
#               ldap
#       }
#        eap
}
```

Plik USERS

Dokonamy teraz konfiguracji pliku *users* na potrzebę obsługi protokołu PPPoE.

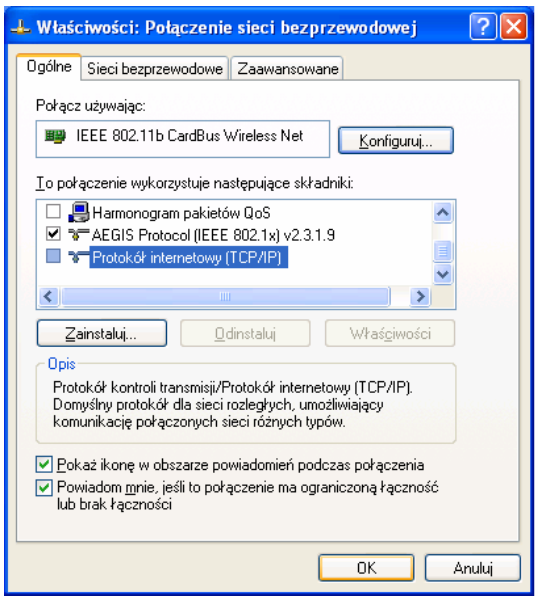
Nasz przykładowy wpis w tym pliku odpowiadający pojedynczemu użytkownikowi będzie składał się z dwóch linii, np.:

```
"00:0C:76:71:04:D9" User-Password == ""
"tomek" User-Password == "ttt", Calling-Station-Id == "00:0C:76:71:04:D9"
```

Do poprawnego działania protokołu PPPoE wystarcza jedynie linia druga. Jednak zdecydowaliśmy się na pozostawienie pierwszej linii (analogicznej do etapu konfiguracji *autoryzacji adresów MAC*) dlatego, iż wprowadza ona dodatkowy krok w zabezpieczeniu naszej sieci. Polega on na tym, dodatkowa autoryzacja adresów MAC chroni nas już w warstwie „fizycznej" przed dostępem osób niepowołanych do naszego AP oraz RADIUS'a. Dopiero w momencie pozytywnej autoryzacji MAC przechodzimy do nawiązania sesji PPPoE.

Konfiguracja stacji klienckiej – WindowsXP

Na potrzeby niniejszego dokumentu skupimy naszą uwagę jedynie na wbudowanym w system Windows XP kliencie PPPoE. W kolejnych krokach zobaczymy jak należy skonfigurować nasze połączenie.

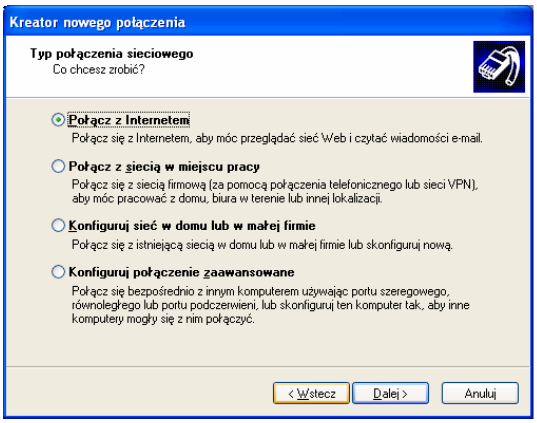


Rozpoczniemy tę czynność od wyłączenia protokołu TCP/IP na naszej karcie radiowej którą jesteśmy podłączeni do naszego AccesPointa.

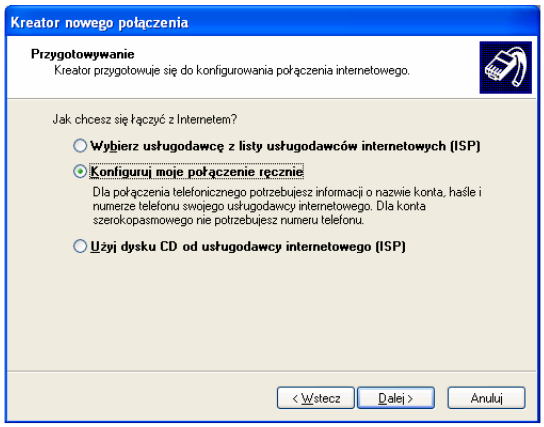
Protokół ten jest zbędny do poprawnej pracy PPPoE. Wykonujemy to poprzez „odznaczenie" odpowiedniej opcji we właściwościach naszej karty radiowej.

Następnie tworzymy właściwe połączenie PPPoE wykorzystując wbudowanego w system „kreatora połączeń"

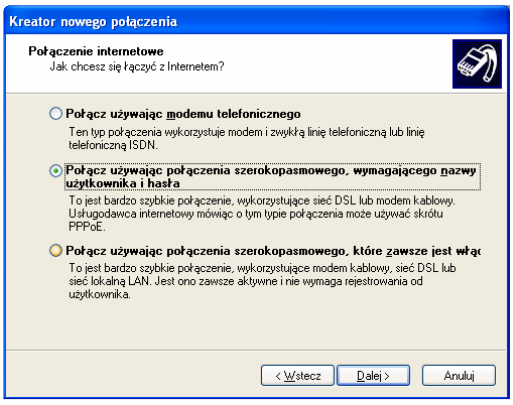
Krok pierwszy – *połącz z Internetem*



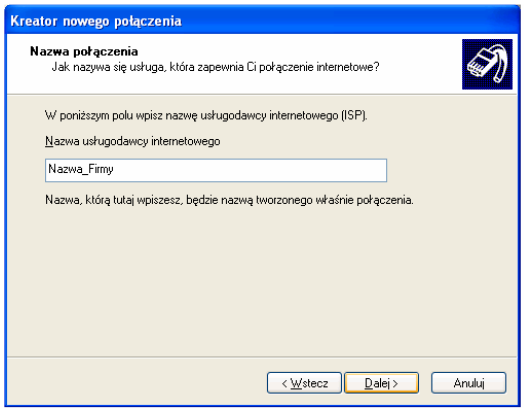
Krok drugi – *Konfiguruj moje połączenie ręcznie*



Krok trzeci – *Połącz używając połączenia szerokopasmowego….*



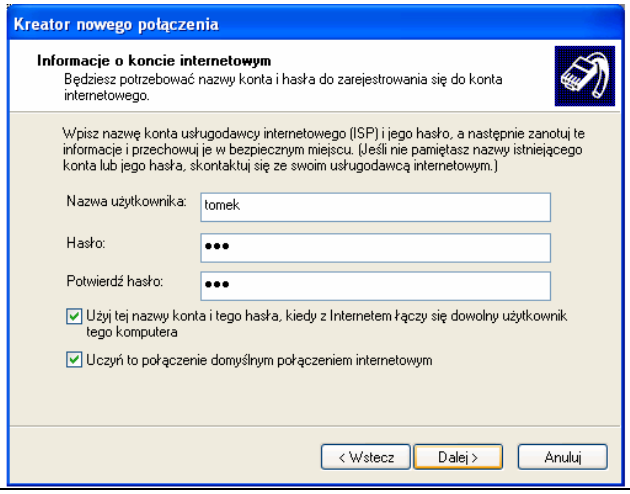
Krok czwarty – *opisowa nazwa naszego połączenia*



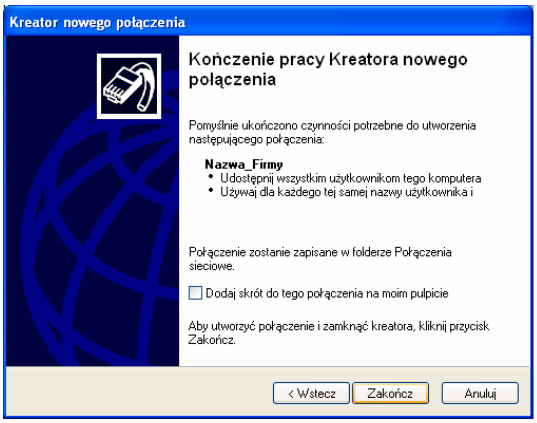
Krok piąty – *Nazwa użytkownika i hasło*. Te wartości wprowadziliśmy w pliku *sers* serwera RADIUS. W naszym przypadku to

```
"tomek" User-Password == "ttt"
```

Czyli Nazwa to „tomek” a hasło to „ttt”.

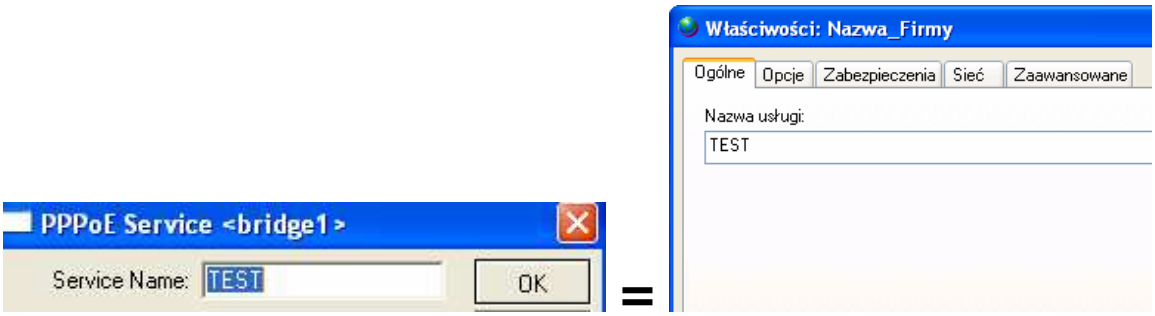


Krok szósty – *zakończenie pracy kreatora*



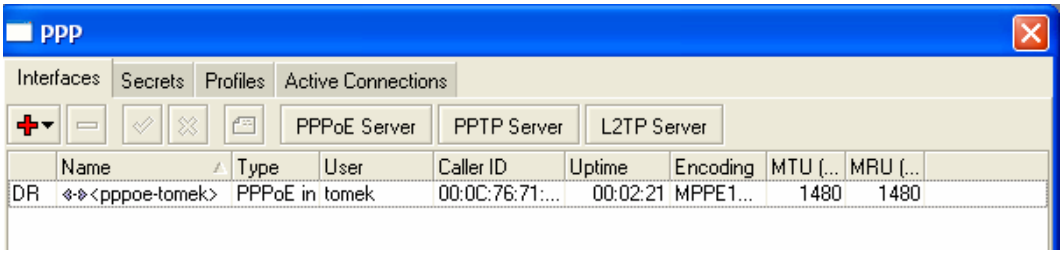
Krok siódmy – *zmodyfikowanie nazwy serwisu*.

W celu umożliwienia pracy komputerowi z naszym AccessPointem musimy jeszcze wprowadzić ***nazwę usługi*** (niestety w trakcie działania kreatora połączeń pole to zostaje pominięte). Wartość tego pola musi być zgodna z polem *ServiceName* jaką podaliśmy na etapie konfiguracji koncentratora PPPoE w Mikrotiku

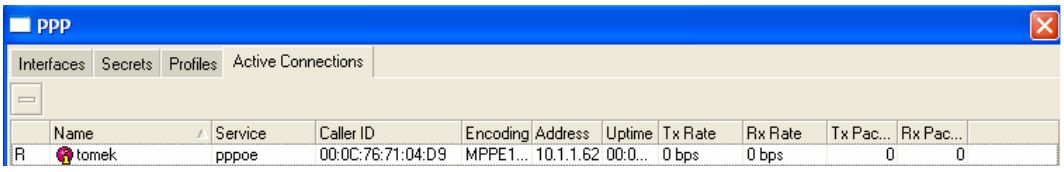


Na tym kończymy konfigurację naszego nowego połączenia. Uruchamiamy je. W systemie Windows nawiązanie tego połączenia powinno odbyć się błyskawicznie. Teraz sprawdzamy w systemie Mikrotik oraz w serwerze RADIUS co się wydarzyło.

W Mikrotiku widzimy, iż utworzony został na potrzeby naszego połaczenia nowy interfejs



W połączeniach PPPoE widzimy także aktywną sesję.



W logach naszego serwera widzimy także dobre informacje:

```
rad_recv: Access-Request packet from host 192.168.1.5:1118, id=110, length=186
        Service-Type = Framed-User
        Framed-Protocol = PPP
        NAS-Port = 46
        NAS-Port-Type = Ethernet
        User-Name = "tomek"
        Calling-Station-Id = "00:0C:76:71:04:D9"
        Called-Station-Id = "TEST"
        NAS-Port-Id = "bridge1"
        MS-CHAP-Challenge = 0xc87049d986b943c7773272b9f6c076dd
        MS-CHAP2-Response =
0x0100c82d8b02dcace4a58754ab00a8c6bec90000000000000000045bf65ad4c13809dc429bb2e301af756907697a
8f7f62bf
        NAS-Identifier = "TechnoLogic"
        NAS-IP-Address = 192.168.1.5
    Processing the authorize section of radiusd.conf
    modcall: entering group authorize for request 4
    modcall[authorize]: module "preprocess" returns ok for request 4
    radius_xlat:  '/usr/local/var/log/radius/radacct/192.168.1.5/auth-detail-20051108'
    rlm_detail: /usr/local/var/log/radius/radacct/%{Client-IP-Address}/auth-detail-%Y%m%d
expands to /usr/local/var/log/radius/radacct/192.168.1.5/auth-detail-20051108
    modcall[authorize]: module "auth_log" returns ok for request 4
    rlm_mschap: Found MS-CHAP attributes.  Setting 'Auth-Type  = MS-CHAP'
    modcall[authorize]: module "mschap" returns ok for request 4
    rlm_realm: No '@' in User-Name = "tomek", looking up realm NULL
    rlm_realm: No such realm "NULL"
    modcall[authorize]: module "suffix" returns noop for request 4
    rlm_eap: No EAP-Message, not doing EAP
    modcall[authorize]: module "eap" returns noop for request 4
    users: Matched entry tomek at line 2
    modcall[authorize]: module "files" returns ok for request 4
    modcall: group authorize returns ok for request 4
    rad_check_password:  Found Auth-Type MS-CHAP
    auth: type "MS-CHAP"
    Processing the authenticate section of radiusd.conf
    modcall: entering group Auth-Type for request 4
    rlm_mschap: Told to do MS-CHAPv2 for tomek with NT-Password
    rlm_mschap: adding MS-CHAPv2 MPPE keys
    modcall[authenticate]: module "mschap" returns ok for request 4
    modcall: group Auth-Type returns ok for request 4
    Login OK: [tomek] (from client TechnoLogicWIFI port 46 cli 00:0C:76:71:04:D9)
```

```
        Sending Access-Accept of id 110 to 192.168.1.5:1118
                MS-CHAP2-Success =
0x01533d34393744303039464435313241314543443031414242313638353142453946313435463939323832
                MS-MPPE-Recv-Key = 0xb81de3a8918490c4339fee3828f1d06f
                MS-MPPE-Send-Key = 0xd091f3dba5df959b7a763ab5ac08766b
                MS-MPPE-Encryption-Policy = 0x00000002
                MS-MPPE-Encryption-Types = 0x00000004
        Finished request 4
```

Załącznik 1

Lista atrybutów obsługiwanych przez RADIUS'a i system Mikrotik.

Attribute Numeric Values

Name	VendorID	Value	RFC where it is defined
Acct-Authentic		45	RFC2866
Acct-Delay-Time		41	RFC2866
Acct-Input-Gigawords		52	RFC2869
Acct-Input-Octets		42	RFC2866
Acct-Input-Packets		47	RFC2866
Acct-Interim-Interval		85	RFC2869
Acct-Output-Gigawords		53	RFC2869
Acct-Output-Octets		43	RFC2866
Acct-Output-Packets		48	RFC2866
Acct-Session-Id		44	RFC2866
Acct-Session-Time		46	RFC2866
Acct-Status-Type		40	RFC2866
Acct-Terminate-Cause		49	RFC2866
Ascend-Client-Gateway	529	132	
Ascend-Data-Rate	529	197	
Ascend-Xmit-Rate	529	255	
Called-Station-Id		30	RFC2865
Calling-Station-Id		31	RFC2865
CHAP-Challenge		60	RFC2866
CHAP-Password		3	RFC2865
Class		25	RFC2865
Filter-Id		11	RFC2865
Framed-IP-Address		8	RFC2865

Framed-IP-Netmask		9	RFC2865
Framed-Pool		88	RFC2869
Framed-Protocol		7	RFC2865
Framed-Route		22	RFC2865
Group	14988	3	
Idle-Timeout		28	RFC2865
MS-CHAP-Challenge	311	11	RFC2548
MS-CHAP-Domain	311	10	RFC2548
MS-CHAP-Response	311	1	RFC2548
MS-CHAP2-Response	311	25	RFC2548
MS-CHAP2-Success	311	26	RFC2548
MS-MPPE-Encryption-Policy	311	7	RFC2548
MS-MPPE-Encryption-Types	311	8	RFC2548
MS-MPPE-Recv-Key	311	17	RFC2548
MS-MPPE-Send-Key	311	16	RFC2548
NAS-Identifier		32	RFC2865
NAS-Port		5	RFC2865
NAS-Port-Id		87	RFC2869
NAS-Port-Type		61	RFC2865
Rate-Limit	14988	8	
Realm	14988	9	
Recv-Limit	14988	1	
Service-Type		6	RFC2865
Session-Timeout		27	RFC2865
User-Name		1	RFC2865
User-Password		2	RFC2865
Wireless-Enc-Algo	14988	6	
Wireless-Enc-Key	14988	7	

Wireless-Forward	14988	4	
Wireless-Skip-Dot1x	14988	5	
Xmit-Limit	14988	2	