Projekt nr. POIG.02 03. 00-00-028/08-01 www.platon.pionier.net.pl
PLATFORMA OBSŁUGI NAUKI PLATON

KOORDYNATOR: INSTYTUT CHEMII BIOORGANICZNEJ PAN
POZNAŃSKIE CENTRUM SUPERKOMPUTEROWO - SIECIOWE
ul. Noskowskiego 12/14, 61-704 Poznań, (+48 61) 858 20 00, fax: (+48 61) 852 59 54, e-mail: office@man.poznan.pl, www: http://www.man.poznan.pl

Konfiguracja regionalnych serwerów eduroam w Polsce

Maja Górecka-Wolniewicz, UCI UMK (mgw@umk.pl)
dokument przygotowany w ramach projektu B-R eduroam-PIONIER
zaktualizowany w ramach projketu PLATON
wersja 2.2 – lipiec 2012

Spis treści

1. Wstęp

W dokumencie *Koncepcja wdrożenia usługi eduroam w sieci PIONIER*[1] wskazano, że istotnym elementem struktury eduroam w sieci PIONIER są Regionalni Operatorzy eduroam. Serwery regionalne funkcjonują na poziomie trzecim hierarchii eduroam, tj. poniżej krajowych i, tak jak serwery krajowe, działają wyłącznie w roli elementu pośredniczącego w przekazywaniu zlecenia (proxy). Początkowo serwery te używały do komunikacji protokołu UDP. W opracowaniu [1] została omówiona potrzeba tworzenia rozwiązań opartych na protokole RadSec, gdyż właśnie takie podejście, jako znacząco bezpieczniejsze, będzie dominowało w przyszłości. Od 2009 roku sukcesywnie serwery regionalne przechodziły na użycie protokołu RadSec w ramach statycznego systemu zestawiania połączeń. Obecnie wszystkie serwery regionalne pracują w tym trybie i stopniowo jest realizowane na tych serwerach włączanie funkcji dynamicznego doboru partnera, co opisano w dokumencie *Realizacja wdrożenia usługi eduroam w sieci PIONIER* [2]. Po zakończeniu tej fazy polska infrastruktura stanie się elementem dynamicznej struktury *eduroam*, działającej zgodnie z zasadami opisanymi w dokumencie [7].

Protokół RadSec został zaprojektowany przez Open System Consultants Pty. Ltd., firmę, pod której szyldem jest tworzone oprogramowanie Radiator, w odpowiedzi na brak możliwości korzystania z protokołu RADIUS w oparciu o TLS. Dokument [3] jest pierwszą definicją protokołu i prezentacją sposobu jego implementacji. RadSec miał być z założenia rozwiązaniem pośrednim, nadającym się do szybkiego wdrożenia, zanim powstaną implementacje protokołu Diameter ([4]), uważanego

za następcę RADIUS-a. Okazało się, że tego typu podejście przyjęło się i właśnie w oparciu o nie jest obecnie definiowana wersja standardu uwzględniająca dynamiczne połączenia pomiędzy serwerami. W bieżącym roku zostały sfinalizowane dokumenty RFC poświęcony tej tematyce([5], [6]), a popularną już nazwę RadSec zastąpiono określeniem RADIUS over TLS.

Ogólnodostępne oprogramowanie FreeRADIUS na razie nie implementuje protokołu RadSec, ale zgodnie z zapowiedzią będzie wkrótce, gdy ukaże się wersja 3.0, dawało możliwość użycia połączeń TLS. RadSec jest dostępny od lat w komercyjnym oprogramowaniu Radiator. Na początku 2007 roku implementacją tego rozwiązania zainteresował się Stig Venaas (wówczas UNINETT, Norwegia). Wkrótce udostępnił on publicznie pierwszą wersję pakietu `radsecproxy`, realizującego funkcjonalność pośredniczącego serwera RADIUS, zarówno w warstwie transportowej UDP, jak i zgodnie z protokołem RadSec.

Obszerne informacje o pakiecie **`radsecproxy`** są dostępne na stronach http://software.uninett.no/radsecproxy.

Gdy projekt `radsecproxy` startował, podstawowym założeniem było zbudowanie oprogramowania pełniącego funkcję pośredniczącego serwera, które będzie oszczędne z punktu widzenia zasobów serwera, wydajne i proste w konfiguracji. Początkowo serwer pośredniczący miał wyłącznie udostępniać połączenia za pośrednictwem protokołu RadSec (RADIUS over TLS), gdyż właśnie tego typu rozwiązań brakowało, jednak szybko stwierdzono, że przydatna będzie również funkcjonalność typowego serwera pośredniczącego RADIUS. Dzięki temu oprogramowanie `radsecproxy` jest idealne w miejscach, gdzie jedynym zadaniem serwera jest przekazywanie pakietów do właściwego serwera. Taką rolę spełniają serwery krajowe i regionalne w polskiej usłudze eduroam.

Pakiet `radsecproxy` wspiera zarówno IPv4, jak i IPv6.

2. Pobieranie oprogramowania

Oprogramowanie można pobrać ze strony:

http://software.uninett.no/radsecproxy/index.php?page=download

Obecnie stabilną wersją jest 1.6.

Udostępniana jest wersja źródłowa. Ponieważ oprogramowanie ciągle rozwija się, zdarza się, że zalecane jest skorzystanie z wersji udostępnianej poprzez git. Aby pobrać najnowszą wersję rozwojową należy wykonać polecenie:

```
git clone git://git.nordu.net/radsecproxy.git
```

3. Kompilacja

Aby przygotować program do pracy, po rozpakowaniu pobranego pakietu należy przejść do katalogu źródłowego i zacząć od polecenia

```
./configure –prefix=/opt/radsecproxy --enable-fticks
```

W wierszu polecenia `configure` można: określić miejsce instalacji (`--prefix`), wskazać, że chcemy by serwer miał wbudowaną obsługę tworzenia danych statystycznych F-Ticks (`--enable-fticks`), albo wymusić korzystanie podczas kompilacji z określonych, niestandardowo umiejscowionych pakietów, np.: SSL (`--with-ssl=/opt/ssl`).

Następnie wykonujemy kompilację:

```
make
```

oraz

```
make install
```

Efektem kompilacji będzie program `radsecproxy`, zainstalowany w wybranym katalogu.

4. Konfiguracja

Domyślnie `radsecproxy` spodziewa się pliku konfiguracyjnego w drzewie instalacji oprogramowania (domyślnie `/usr/local/`), w podkatalogu `etc`, w pliku `radsecproxy.conf`. Przy starcie można wskazać programowi `radsecproxy` dowolną lokalizację konfiguracji. Podczas wczytywania tego pliku ignorowane są „puste" znaki, typu dodatkowe odstępy i znaki tabulacji, również w każdym wierszu ignorowane są znaki odstępu i tabulacji na początku i końcu wiersza. Jeśli pierwszym znakiem wiersza jest #, to wiersz jest traktowany jako komentarz. W konfiguracji na ogół nie ma znaczenia wysokość liter.

Dozwolone są dwa typy struktur konfiguracyjnych:

- pierwsza (prostsza) ma postać: `opcja wartość`, gdzie w polu opcja jest umieszczana dopuszczalna nazwa opcji, np. `LogLevel`, a w polu wartość, wartość nadana danej opcji, np. 4;
- druga (bardziej rozbudowana) to blok, czyli struktura mająca minimum dwa wiersze o postaci:

```
typ_bloku nazwa {
   opcja wartość
   opcja wartość
}
```

Jedna z opcji, ma specjalny charakter i pomaga zwiększyć czytelność konfiguracji:

```
Include <plik>
```

W ten sposób można w określonym miejscu wstawiać zawartość wskazanego pliku, który definiuje dany fragment konfiguracji.

Szczegółowy opis opcji oraz wszystkich dozwolonych bloków: `Client`, `Server`, `Realm`, `TLS`, `Rewrite` można znaleźć na stronie:

http://software.uninett.no/radsecproxy/index.php?page=documentation

4.1. Podstawowe opcje

Najistotniejsze podstawowe opcje to:

- ListenUDP / ListenTLS – służą do zadeklarowania interfejsu sieciowego (postaci IPv4 lub IPv6) oraz portu, na których nasłuchuje serwer, co jest szczególnie ważne, gdy nasz komputer ma kilka interfejsów sieciowych; klienci UDP i TCP domyślnie korzystają z portu 1812, klienci TLS/DTLS jako domyślny port przyjmują 2083;
- SourceUDP / SourceTLS – służą do ustalenia, jaki adres (postaci IPv4 lub IPv6) oraz jaki port pojawi się w pakietach wysyłanych przez serwer; należy pamiętać, że ustalenie, że serwer nasłuchuje na danym adresie, nie oznacza, że ten adres pojawi się w pakietach wyjściowych – jeśli nasz serwer działa na adresie 11.22.33.44 i taki adres ma pojawiać się w pakietach wyjściowych, to należy użyć w konfiguracji zarówno opcji ListenUDP / ListenTLS, jak i SourceUDP / SourceTLS;
- LoopPrevention – włączenie tej opcji (on) powoduje, że są pomijane pakiety, które przychodzą od klienta zdefiniowanego w bloku o nazwie AAA i na podstawie reguł obsługi domen, opisanych przez zestaw bloków Realm mają zostać przekierowane do serwera określonego w bloku o takiej samej nazwie AAA; w ten sposób w oprogramowaniu radsecproxy zaimplementowano zapobieganie pętlom.
- IPv4Only / IPv6Only – opcja umożliwia ustalenie, jakiego typu adresacji oczekuje serwer; włączenie IPv4Only oznacza, że wszystkie rozwikłania adresów domenowych będą zwracały adresy IPv4, włączenie IPv6Only oznacza, że chcemy wyłącznie otrzymywać adresy IPv6.

Uwaga: Jeśli serwer nie obsługuje IPv6, to zaleca się włączenie opcji IPv4Only w celu uniknięcia rozwikłania adresów domenowych (jeśli takie są stosowane w blokach Server/Client, patrz 4.2.1) na adresy IPv6.

4.2. Bloki

Dozwolone bloki to: Client, Server, Realm, TLS, Rewrite.

4.2.1. Bloki Client i Server

Bloki Client i Server ustalają dane dotyczące odpowiednio klienta i serwera.

Klientem nazywamy połączenie przysyłające do serwera pośredniczącego zlecenie Access-Request. Serwerem jest strona, do której serwer pośredniczący na podstawie analizy realmu przekazuje zapytanie, w celu jego dalszej realizacji.

Należy zwrócić uwagę na napis nazwa_bloku w strukturze

```
Client nazwa_bloku {...}
Server nazwa_bloku {...}
```

nazwa_bloku to nazwa unikatowa dla danego typu bloku. Jest to albo IP klienta/serwera, albo pełna nazwa domenowa, albo unikatowa nazwa identyfikująca konkretny blok. Jeśli w bloku nie definiujemy opcji host, to nazwa bloku musi być albo adresem IP, albo nazwą domenową klienta/serwera. Funkcjonowanie radsecproxy jako serwera pośredniczącego oznacza, że w większości przypadków klient jest jednocześnie serwerem. Zastosowanie tych samych nazw bloków Client i Server, przy włączonej opcji LoopPrevention pozwoli automatycznie eliminować ewentualne pętle.

Obecna wersja radsecproxy wymaga, by wszystkie nazwy serwerów oraz klientów podane w konfiguracji mogły zostać w chwili startu serwera przekształcone na adres IP. Jeśli nie jest to możliwe serwer nie startuje. Aby zapobiec problemom, zaleca się unikanie podawania nazwy domenowej partnera (serwera lub klienta), preferowane jest używane numerów IP.

Za pomocą bloków Server są również definiowane serwery, do których mają być przekazywane pakiety rozliczeniowe. Należy pamiętać, że, zgodnie z polską polityką eduroam, serwery pośredni-

czące nie mogą przekazywać pakietów rozliczeniowych (wyjątkiem jest specjalna umowa dotycząca przesyłania tego ruchu pomiędzy usługodawcą a operatorem).

Inną ważną opcją w bloku `Client` / `Server` jest `type` – opcja służąca do ustalenia, w jakim trybie ma pracować klient / serwer. Możliwe wartości – `udp / tcp / tls / dtls`, ustalają sposób komunikacji klienta / serwera:

- `udp` – za pomocą protokołu RADIUS UDP;
- `tcp` – za pomocą protokołu RADIUS TCP;
- `tls` – za pomocą protokołu RadSec (TLS+TCP);
- `dtls` – za pomocą protokołu RadSec (TLS+UDP);

W przypadku komunikacji RadSec istotna jest definicja zasad realizacji szyfrowanej komunikacji.

Każdy blok `Client` / `Server` mający wpis `type tls` musi mieć również ustawioną opcję `secret` z wartością `radsec`.

Domyślnie, jeśli klient/serwer używa TLS-a, to realizowane jest sprawdzenie, czy certyfikat klienta / serwera ma w polu `CN` lub `SubjectAltName` taką samą nazwę, jak wskazana dla klienta/serwera nazwa domenowa lub adres IP. Sprawdzenie to można wyłączyć umieszczając w bloku `Client` / `Server` opcję `certificateNameCheck off`. Wyłączenie sprawdzenia nazwy w certyfikacie jest niezbędne, jeśli definiujemy klienta lub serwer używając adresu IP, nie nazwy (co jest zalecane), a certyfikat został wystawiony dla nazwy serwera.

W przypadku realizacji połączeń na zasadach dynamicznego doboru partnerów serwer musi być przygotowany na odbiór zleceń od dowolnych klientów, kontaktujących się w oparciu o protokół TLS i posługujących certyfikatem europejskiej usługi eduroam. Służy do tego blok `Client` zawierający w opcjach: `host 0.0.0.0/0` oraz `type tls`.

Również blok `Server` może mieć specjalny charakter, jeśli ma wskazywać sposób dynamicznego doboru partnera. Wówczas blok taki nie zawiera nazwy, czy adresu IP serwera, występuje w nim natomiast dyrektywa `dynamicLookupCommand`, w której jest wskazany skrypt ustalający dynamicznie blok `Realm` dla realmu, którego dotyczy zapytanie. Pakiet `radsecproxy` zawiera w podkatalogu `tools` skrypty wspomagające dynamiczny dobór partnera w oparciu o wpisy DNS.

4.2.2. Blok `tls`

Reguły szyfrowanej komunikacji definiujemy w bloku `tls`. Możliwa jest definicja dowolnej liczby bloków `tls`. Domyślną nazwą bloku `tls` dla klienta (blok `Client`) jest `defaultClient`, a jeśli taki blok nie istnieje, używany jest blok `default`. Podobnie w przypadku serwera: w pierwszej kolejności szukany jest blok `defaultServer`, następnie blok `default`. Domyślny blok `tls` obowiązuje dla klienta / serwera, jeśli nie została w danym bloku `Client` / `Server` użyta opcja `tls nazwa`. Jeśli w konfiguracji `radsecproxy` definiujemy co najmniej jeden blok klienta / serwera, zawierający opcję `type tls`, to musi w niej zostać umieszczony co najmniej jeden blok `tls`.

Uwaga:

W domyślnych nazwach bloków `tls`: `defaultClient` i `defaultServer` istotna jest wielkość liter!

Blok `tls` wskazuje:

- plik certyfikatu serwera (`CertificateFile`),
- plik klucza prywatnego serwera (`CertificateKeyFile`) i hasło dla klucza prywatnego (`CertificateKeyPassword`), jeśli jest on zaszyfrowany ,
- plik certyfikatu urzędu certyfikacyjnego (`CACertificateFile`) lub ścieżkę do katalogu (`CACertificatePath`), w którym jest certyfikat i w którym utworzony jest sktót (*hash*) certyfikatów (polecenie `c_rehash`),
- dodatkowe wymagania, np. sprawdzanie listy odwołanych certyfikatów, obecność w certyfikacie partnera pola `policyOID` o określonej wartości,

- żądanie sprawdzania listy unieważnionych certyfikatów (CRLCheck On) – włączenie tej opcji wymaga, by była ustalona wartość CACertificatePath; we wskazanym tam katalogu musi znajdować się lista unieważnionych certyfikatów i należy w tym katalogu utworzyć skróty certyfikatów i list CRL-i.

radsecproxy nie wspiera obecnie protokołu OCSP.

4.2.3. Blok Realm

Kolejnym ważnym blokiem jest Realm, który realizuje powiązanie danej nazwy domenowej z serwerem. Jego nazwa to albo domena, albo wyrażenie regularne dopasowujące domenę, albo znak * oznaczający „wszystko". Jeśli domena pobrana z atrybutu User-Name zlecenia zostanie dopasowana do nazwy bloku Realm, to pakiet podlega obsłudze zgodnie z opcjami wskazanymi w danym bloku. W przypadku bloków Realm może być istotna ich kolejność, należy pamiętać, że poszukiwanie sposobu obsługi nazwy domenowej trwa do znalezienia dopasowania w nazwie bloku Realm. Dlatego na końcu umieszczamy blok obsługujący wszystkie pozostałe zapytania. Blok Realm typowo zawiera opcję Server nazwa, wskazującą serwer obsługujący tę domenę (można podać więcej niż jeden serwer, definiując w ten sposób serwery zastępcze). Można również w tym bloku określić serwer, który przyjmuje zlecenia accountingowe (accountingServer), ale, jak już wspomniano, w polskim projekcie nie jest to dozwolone. Serwer pośredniczący musi reagować na otrzymane pakiety Accounting-Request przez odesłanie Accounting-Response, ale nie przekazuje tego pakietu dalej. Z tego powodu istotną opcją w bloku Realm jest accountingResponse on – opcja powodująca, że w przypadku pakietu rozliczeniowego dotyczącego danego realmu, automatycznie wysyłane jest potwierdzenie przyjęcia pakietu, dzięki czemu unikamy retransmisji zbędnego pakietu. Takie ustawienie pozwala uchronić się przed zasypywaniem serwera pakietami rozliczeniowymi, w przypadku, gdy źle skonfigurowany partner wysyła takie pakiety. Jeśli w ramach bloku Realm nie podamy serwera (server), natomiast wskażemy replymessage, to radsecproxy wyśle odpowiedź Access-Reject.

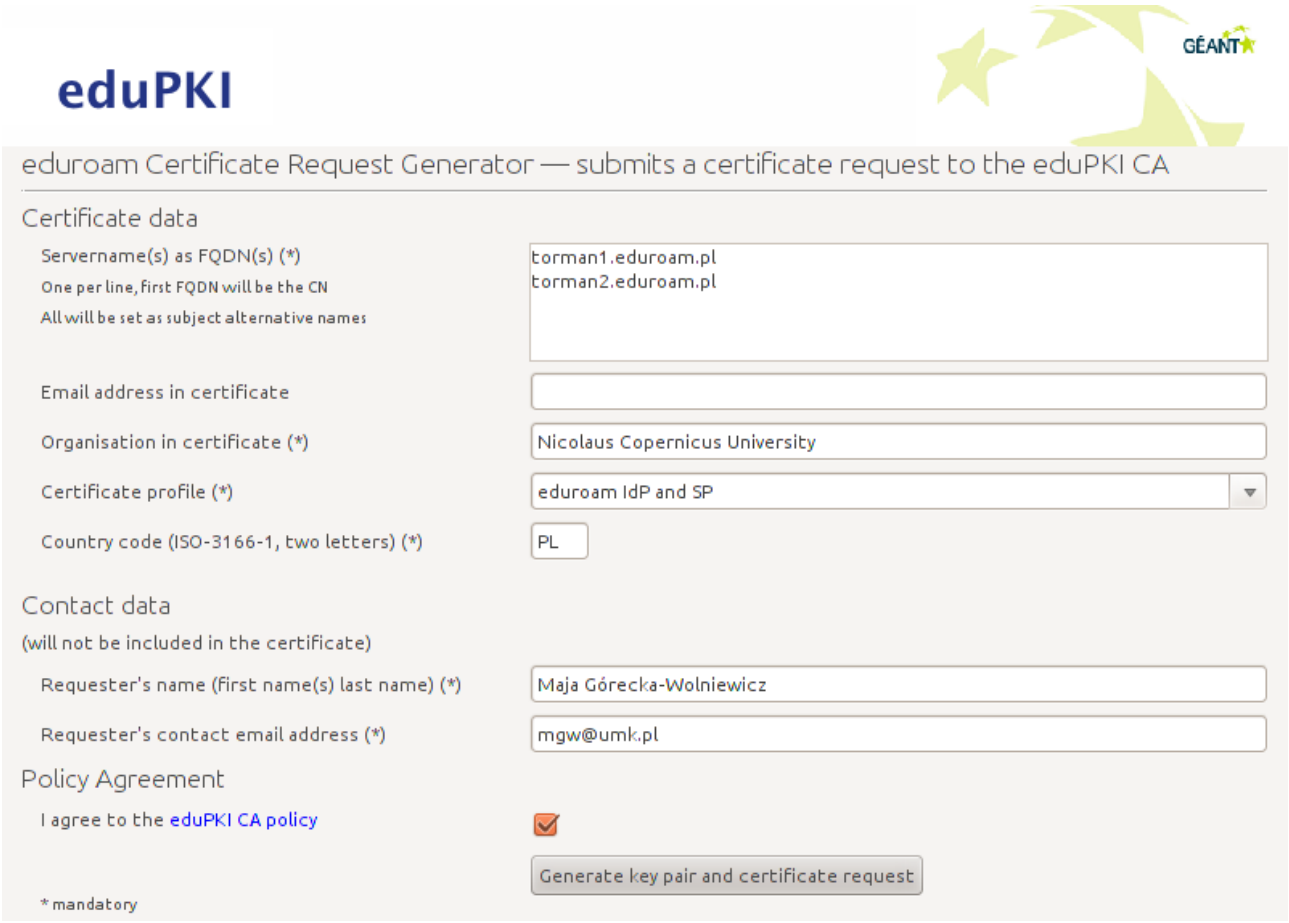
4.3. Pozyskiwanie certyfikatów do komunikacji TLS i ich konfiguracja

Aby była możliwa komunikacja według protokołu RADIUS over TLS, niezbędne jest dysponowanie odpowiednimi certyfikatami, które należy wskazać w konfiguracji. Zgodnie z opisami [1] i [2], wszystkie serwery regionalne, pracujące obecnie w trybie statycznej hierarchii realizowanej w oparciu o protokół RADIUS over TLS, będą docelowo wpięte w dynamiczną strukturę europejską, w której akceptowane są certyfikaty eduPKI (https://www.edupki.org), lub akredytowanego w usłudze eduroam urzędu. Proces przyłączania do tej struktury już się zaczął. Polskie serwery krajowe łączą się już z serwerami poziomu europejskiego za pomocą protokołu RADIUS over TLS, przy wykorzystaniu certyfikatów eduPKI. Docelowo wszystkie serwery regionalne będą korzystały z certyfikatów eduPKI. Obecnie część serwerów używa certyfikatów wystawionych przez dedykowany dla polskiego projektu eduroam urząd PIONIER CA, a kilka korzysta jeszcze z certyfikatów wystawionych w ramach projektu eduGAIN (Géant JRA5).

4.3.1. Certyfikaty eduPKI

Na stronie http://www.eduroam.org/index.php?p=europe&s=edupki opisano procedurę przyznawania certyfikatów. Administratorzy polskich serwerów regionalnych są zarejestrowani w europejskiej eduroam, dzięki czemu mogą samodzielnie ubiegać się o certyfikaty dla swoich serwerów. Zaleca się pobieranie jednego certyfikatu dla obu serwerów regionalnych, tj. podstawowego i zapasowego (patrz pole *Servername(s)* w zamieszczonym poniżej formularzu). Nazwa instytucji w polu *Organisation in certificate* musi być zgodna z danymi rejestracji instytucji. Jako profil certyfikatu (pole *Certificate profile*) wybieramy eduroam IdP and SP, ponieważ certyfikat będzie używany przez serwer pośredniczący, występujący zarówno w roli serwera jaki i klienta. Dane kontaktowe (*Contact data*) również muszą być spójne z informacją udostępniana w europejskiej bazie eduram. Zatwierdzamy formularz naciskając przycisk *Generate key pair and certificate request*. Utworzony w procesie generacji klucz prywatny umieszczamy w bezpiecznym miejscu. Po weryfikacji i za-

twierdzeniu wniosku, operator eduPKI podpisuje klucz publiczny i przesyła gotowy certyfikat w mailu przesłanym na wskazany adres.



Certyfikat razem z kluczem prywatnym oraz certyfikatem nadrzędnym `edupki-root-ca-cert.pem`, dostępnym pod adresem https://www.edupki.org/fileadmin/Documents/edupki-root-ca-cert.txt, należy umieścić w konfiguracji w bloku `tls`. Zaleca się dodanie w tym bloku dyrektywy sprawdzania listy unieważnionych certyfikatów (`CRLCheck On`). Lista odwołanych certyfikatów jest publikowana pod adresem: http://cdp.edupki.org/edupki-ca/pub/crl/cacrl.crl. Przed umieszczeniem tej listy w katalogu wskazanym parametrem `CACertificatePath` należy wykonać jej konwersję z postaci DER do PEM.

```
openssl crl -inform DER -outform PEM -in cacrl.crl -out cacrl.pem
```

Certyfikaty eduPKI zawierają obowiązujące w europejskiej usłudze eduroam identyfikatory polityki: `1.3.6.1.4.1.25178.3.1.1` (uprawnienia SP) oraz `1.3.6.1.4.1.25178.3.1.2` (uprawnienia IdP). Z tego powodu w konfiguracji należy oddzielnie specyfikować bloki `tls` dla klienta i dla serwera i w każdym z tych bloków umieścić opcję `policyOID` z wymaganym identyfikatorem polityki (patrz 4.2.2). Identyfikator polityki dla klienta to `1.3.6.1.4.1.25178.3.1.1`, dla serwera `1.3.6.1.4.1.25178.3.1.2`. Przykładowa konfiguracja bloków `tls` wygląda następująco:

```
# domyślny blok tls dla strony klienta
tls defaultClient {
CACertificatePath    /etc/radsecproxy.conf.d/certs/
CertificateFile      /etc/radsecproxy.conf.d/certs/radius.pem
CertificateKeyFile   /etc/radsecproxy.conf.d/certs/radius.key
policyOID 1.3.6.1.4.1.25178.3.1.1
CRLCheck On
}
# domyślny blok tls dla serwera
tls defaultServer {
CACertificatePath    /etc/radsecproxy.conf.d/certs
CertificateFile      /etc/radsecproxy.conf.d/certs/radius.pem
CertificateKeyFile   /etc/radsecproxy.conf.d/certs/radius.key
policyOID 1.3.6.1.4.1.25178.3.1.2
CRLCheck On
}
```

4.4. Konfiguracja raportowania F-Ticks

Serwer `radsecproxy` umożliwia tworzenie logów zgodnie z wymaganiami systemu F-Ticks (*Federated Ticker System*), który jest stosowany w europejskiej usłudze eduroam (http://monitor.eduroam.org/f-ticks). Szczegółowy opis przyjętego rozwiązania przy tworzeniu statystyk przedstawiono w [8].

Polski projekt eduroam korzysta z logowania F-Ticks do przekazywania statystyk dotyczących ruchu. Serwery regionalne powinny zostać tak skonfigurowane, by raporty F-Ticks były przekazywane do ustalonej lokalizacji.

Aby korzystać z logowania F-Ticks niezbędna jest specjalne przygotowanie programu `radsecproxy` do kompilacji poprzez użycie opcji `–enable-fticks` w poleceniu `configure`.

W polskiej usłudze eduroam zalecamy następujące ustawienia w pliku konfiguracyjnym `radesecproxy.conf`:

- `FticksReporting Full` – służy do włączenia raportowania F-Ticks i ustaleniu stylu raportu,
- `FticksMAC Original` – służy do ustalenia czy należy logować adres MAC, jeśli tak to w jakiej postaci; wartość `Original` wskazuje, że będzie logowana rzeczywista postać adresu MAC,
- `FticksSyslogFacility LOG_LOCALn` – służy do wskazania dedykowanego celu syslog, przeznaczonego dla komunikatów F-Ticks.

Dodatkowo w blokach `Client` dodajemy dyrektywę `fticksVISCOUNTRY PL`, gdyż tylko klienci z ustawioną wartością w `fticksVISCOUNTRY` podlegają logowaniu F-Ticks. Opcja `fticksVISINST` w bloku `Client` pozwala ustalić, jaka nazwa klienta ma być pokazywana w logu (domyślnie jest to nazwa bloku `Client`, dzięki tej opcji możliwa jest spójna prezentacja nazw klientów).

Ustalony w dyrektywie `FticksSyslogFacility` sposób identyfikowania komunikatów musi zostać odpowiednio obsłużony w używanym lokalnie systemie `syslog`. Jeśli np. w konfiguracji używamy:

```
FTicksSyslogFacility LOG_LOCAL3
```

to system `syslog` musi kierować komunikaty ze źródła local3 do centralnego serwera o IP `158.75.1.99` (zaleca się zachowanie lokalnej kopii logów F-Ticks).

Przykładowo dla systemu `rsyslog` wpis ma postać:

```
$RepeatedMsgReduction off
local3.* /var/log/fticks-local.log
:msg, contains, "F-TICKS/eduroam/1.0#" @158.75.1.99:514
:msg, contains, "F-TICKS/eduroam/1.0#" ~
```

a dla `syslog-ng`:

```
Destination df_local3 { file("/var/log/fticks-local.log");
udp("158.75.1.99"); };
```

4.5. Konfiguracja połączeń dynamicznych

Docelowy model usługi eduroam będzie opierał się na dynamicznym doborze partnera. Polska usługa przewiduje wykorzystanie serwerów regionalnych do realizacji tej funkcji.

Wdrożenie mechanizmów dynamicznego wyszukiwania partnerów wymaga dostosowania konfiguracji po stronie instytucji realizujących uwierzytelnianie użytkowników (dostawca tożsamości, czyli instytucja macierzysta użytkownika eduroam) oraz po stronie serwera regionalnego.

Rolą serwera regionalnego w tym procesie jest umiejętność przekazywania zleceń najkrótszą drogą, z pominięciem działającej obecnie hierarchii eduroam. Przed podjęciem decyzji o przesłaniu zlecenia, które nie dotyczy użytkowników danego regionu do serwera krajowego, serwer regionalny powinien sprawdzić, czy istnieje możliwość bezpośredniego kontaktu z serwerem obsługującym realm występujący w zleceniu. Służy do tego opisany poniżej mechanizm zaimplementowany w oprogramowaniu `radsecproxy`.

Z kolei instytucje, w których jest realizowane uwierzytelnienie, aby wykorzystać mechanizm dynamicznego doboru partnera, muszą wykonać specjalną konfigurację usługi eduroam w systemie DNS.

4.5.1. Przygotowanie ustawień DNS

Jeśli na serwerze regionalnym jest realizowana obsługa nazwy domenowej związanej z określoną instytucją w danym regionie, to, aby istniała możliwość dynamicznego dostępu z serwerów zewnętrznych, administratorzy instytucji muszą wykonać poniższe czynności.

Załóżmy, że zajmujemy się domeną UMK: umk.pl.

Administrator domeny umk.pl umieszcza w DNS rekordy NAPTR wskazujące sposób wyszukiwania serwera odpowiadającego za obsługę zleceń dotyczących użytkowników danej domeny.

```
umk.pl.  IN NAPTR 10 10 "s" "x-eduroam:radius.tls"
      "" _radsec._tcp.torman.eduroam.pl
```

Powyższy wpis definiuje rekord NAPTR dla domeny umk.pl z wartościami porządkowymi równymi 10 (*order* i *preference*). Flaga "s" wskazuje, że kolejne sprawdzenie, będzie dotyczyć rekordu SRV. Pole "x-eduroam:radius.tls" to nazwa usługi, której dotyczy rekord. Następne pola to pusty łańcuch wyrażenia regularnego oraz wartość _radsec._tcp.torman.eduroam.pl w polu określającym nazwę – ta nazwa zostanie użyta w celu wyszukania rekordu SRV.

Rekord NAPTR wskazuje usługę _radsec._tcp.torman.eduroam.pl, konieczne jest więc dokonanie również wpisu DNS dotyczącego rekordu SRV w domenie eduroam.pl.

```
_radsec._tcp.torman.eduroam.pl. IN SRV    5 0 2083 torman1.eduroam.pl
_radsec._tcp.torman.eduroam.pl. IN SRV 10 0 2083 torman2.eduroam.pl
```

W ten sposób ustalamy, że usługa _radsec._tcp.torman.eduroam.pl jest realizowana przez serwer torman1.eduroam.pl, działający na porcie 2083, a jeśli ten serwer nie jest dostępny, to przez serwer torman2.eduroam.pl:2083.

4.5.2. Przygotowanie serwera radsecproxy

System dynamicznego wyszukiwania partnerów zrealizowany w oprogramowaniu `radsecproxy` korzysta z zewnętrznych skryptów do wykonania zapytań DNS. Jeśli przychodzi zapytanie dotyczące określonego realmu i w konfiguracji nie jest określony sposób obsługi takiego zlecenia (nie ma bloku Realm, który dopasowuje się do przekazanej nazwy), to serwer powinien sprawdzić, czy dany realm jest obsługiwany dynamicznie. W tym celu wywołuje odpowiedni skrypt, który wykonuje zapytania NAPTR i SRV. Jeśli zapytania te dadzą konkretną odpowiedź, to jest tworzony blok `Server` dla realmu, którego dotyczy zlecenie.

Przygotowanie serwera regionalnego do realizacji dynamicznych wyszukiwań partnerów wymaga wykonania trzech ustawień:

1. dodajemy blok `Server`, przykładowo o nazwie `dynamic`, zawierający opcję `dynamicLookupCommand`, wskazującą skrypt odpowiedzialny za dynamiczne tworzenie bloków `Server` dla realmów;
2. na końcu sekwencji bloków klient dodajemy blok Client zawierający opcje `host 0.0.0.0/0`: i `type tls`;
3. w bloku `Realm`, umieszczonym na końcu konfiguracji realmów dodajemy wskazanie `Server dynamic`.

```
Server dynamic {
type tls
secret radsec
dynamicLookupCommand /opt/radsecproxy/scripts/naptr-eduroam.sh
}
...
Client incoming-tls {
type tls
secret radsec
host 0.0.0.0/0
certificateNameCheck off
}
...
Realm /.*@.+\..+/ {
server dynamic
server PL1TLS
server PL2TLS
}
```

5. Start serwera

Jeśli proces `radsecproxy` jest startowany bez dodatkowych opcji, to jest czytana domyślna konfiguracja z pliku `radsecproxy.conf`, znajdującego się w podkatalogu `etc/` drzewa instalacji oprogramowania. Można utrzymywać konfigurację w innym miejscu, np. w katalogu `/opt/radsecproxy.conf.d/` w pliku `main.conf`, ale wówczas startując serwer musimy dodać opcję `-c <plik>`:

```
radsecproxy -c /opt/radsecproxy.conf.d/main.cf
```

Serwer domyślnie startuje w tle. Jeśli zależy nam na tym, by pracował w pierwszym planie, to należy dodać opcję `-f` *(run in foreground)*. Gdy chcemy wyłącznie przetestować konfigurację, to pozwoli na to opcja `-p` *(pretend)*. W fazie uruchamiania oprogramowania warto ustawić najwyższy poziom debugowania, co robimy albo poprzez wpis

```
LogLevel 5
```

w pliku konfiguracyjnym, albo dodając opcję `-d 5` w wywołaniu.

Domyślny poziom logowania to 2 (błędy, ostrzeżenia).

6. Przykład konfiguracji

Poniższa konfiguracja pokazuje jak powinien zostać skonfigurowany serwer regionalny, by był przygotowany do włączenia do dynamicznej infrastruktury europejskiej usługi eduroam.

6.1. Plik `/etc/radsecproxy.conf`

Podstawową konfigurację przygotowujemy według poniższego wzorca:

```
ListenUDP       11.22.33.44:1812
ListenUDP       11.22.33.44:1813
listenTLS       11.22.33.44:2083
SourceUDP       11.22.33.44
SourceTLS       11.22.44.44
# logowanie: poziom 3, do pliku
LogLevel        3
LogDestination  file:///opt/radsecproxy/rp.log
# zapobiegamy pętlom
LoopPrevention on
# konfiguracja F-Ticks - przekazujemy do systemu syslog, local2
FTicksSyslogFacility LOG_LOCAL2
FTicksReporting Full
FTicksMAC Original
# definiujemy bloki tls dla strony serwera i klienta
tls defaultClient {
CACertificatePath    /etc/radsecproxy.conf.d/certs/
CertificateFile      /etc/radsecproxy.conf.d/certs/radius.pem
CertificateKeyFile   /etc/radsecproxy.conf.d/certs/radius.key
policyOID 1.3.6.1.4.1.25178.3.1.1
CRLCheck On
# jeśli klucz prywatny nie jest zaszyfrowany, zakomentowujemy
CertificateKeyPassword "aaaa"
}
tls defaultServer {
CACertificateFile    /etc/radsecproxy.conf.d/certs/
CertificateFile      /etc/radsecproxy.conf.d/certs/radius.pem
CertificateKeyFile   /etc/radsecproxy.conf.d/certs/radius.key
policyOID 1.3.6.1.4.1.25178.3.1.2
CRLCheck On
# jeśli klucz prywatny nie jest zaszyfrowany, zakomentowujemy
CertificateKeyPassword "aaaa"
}
# klienci w pliku clients.conf
include /etc/radsecproxy.conf.d/clients.conf
# serwery w pliku servers.conf
include /etc/radsecproxy.conf.d/servers.conf
# domeny w pliku realms.conf
include /etc/radsecproxy.conf.d/realms.conf
```

6.2. Plik `/etc/radsecproxy.conf.d/clients.conf`

Zawiera definicje bloków `Client`, np.:

```
# przykładowy klient korzystający z UDP
Client UNI {
type udp
host 22.33.44.55
secret klucz_tajny_uni
}
# pozostali klienci korzystają z RadSeca
Client UNITLS {
type tls
secret radsec
host ip_serwera
certificateNameCheck off
fticksVISCOUNTRY PL
fticksVISINST UMKTORUN
}
# pierwszy serwer krajowy radius1.eduroam.pl
Client radius1.eduroam.pl {
host 158.75.1.25
type tls
secret radsec
certificateNameCheck off
fticksVISCOUNTRY PL
}
# drugi serwer krajowy radius2.eduroam.pl
Client radius2.eduroam.pl {
host 150.254.173.60
type tls
secret radsec
certificateNameCheck off
fticksVISCOUNTRY PL
}
# klient domyślny, wymagamy by posługiwał się certyfikatem
# eduPKI (zgodnie z blokiem defaultClient)
Client incoming-tls {
type tls
secret radsec
host 0.0.0.0/0
certificateNameCheck off
}
```

6.3. Plik `/etc/radsecproxy.conf.d/servers.conf`

Zawiera definicje bloków `Server`, np.:

```
# przykładowy serwer korzystający z połączeń UDP
Server UNI {
type udp
host 22.33.44.55
secret klucz_tajny_uni1
}
# pozostali klienci korzystają z RADIUS over TLS
Server UNI1TLS {
host IP_serwera1
type tls
secret radsec
certificateNameCheck off
}
Server UNI2TLS {
host IP_serwera1
type tls
secret radsec
certificateNameCheck off
}
# pierwszy serwer krajowy radius1.eduroam.pl
Server radius1.eduroam.pl {
host 158.75.1.25
type tls
secret radsec
certificateNameCheck off
}
# drugi serwer krajowy radius2.eduroam.pl
Server radius2.eduroam.pl {
host 150.254.173.60
type tls
secret radsec
certificateNameCheck off
}
# połączenia dynamiczne
Server dynamic {
type tls
secret radsec
dynamicLookupCommand /opt/radsecproxy/scripts/naptr-eduroam.sh
}
```

6.4. Plik /etc/radsecproxy.conf.d/realms.conf

Zawiera definicję obsługiwanych domen.

```
# domena uni.lodz.pl i jej poddomeny – obsługuje ją serwer UNI1, RADIUS
Realm /uni\.lodz\.pl$/ {
server UNI
}
# domena umed.lodz.pl i jej poddomeny – obsługuje serwer UNI2, RADIUS
Realm /umed\.lodz\.pl$/ {
server UNI1TLS
}
# domena p.lodz.pl i jej poddomeny – obsługuje serwer UNI2, RadSec
Realm /p\.lodz\.pl$/ {
server UNI2TLS
accountingResponse on
}
# odrzucamy pakiety z pustą poddomeną
realm /^.*@$/ {
 replymessage "Misconfigured client: empty realm!"
}
# odrzucamy pakiety z nieobsługiwanych poddomen lodz.pl
realm /lodz\.pl/ {
replymessage "Nie obslugujemy tej poddomeny"
}
# reszta dynamicznie lub do serwerów krajowych, jeśli jest poprawny realm
Realm /.*@.+\..+/ {
server dynamic
server  radius1.eduroam.pl
server  radius2.eduroam.pl
}
# pozostałe – odrzucamy
Realm * {
replymessage "Misconfigured client: bad realm!"
}
```

Materiały towarzyszące

[1] *Koncepcja wdrożenia usługi eduroam w sieci Pionier*, dokument przygotowany w ramach projektu B-R eduroam-PIONIER
[2] *Realizacja wdrożenia usługi eduroam w sieci PIONIER*, T. Wolniewicz, dokument przygotowany w ramach projektu PLATON
[3] *RadSec, a secure, reliable RADIUS Protocol*, Open System Consultants Pty. Ltd., http://www.open.com.au/radiator/radsec-whitepaper.pdf
[4] *Diameter Base Protocol,* RFC 3588
[5] RADIUS over TCP, RFC 6613, A. DeKok, http://www.rfc-editor.org/rfc/rfc6613.txt
[6] *Transport Layer Security (TLS) Encryption for RADIU, RFC 6614, S. Winter, M. McCauley, S. Venaas, K. Wierenga, http://www.rfc-editor.org/rfc/rfc6614.txt*
[7] NAI-based Dynamic Peer Discovery for RADIUS/TLS and RADIUS/DTLS, S. Winter, M. McCauley, http://tools.ietf.org/html/draft-ietf-radext-dynamic-discovery-04

[8] Deliverable DJ3.1.2,1: Roaming Developments, rozdz. 3.1, http://www.geant.net/Media_Centre/Media_Library/Media%20Library/GN3-11-354_DJ3-1.2-2_Roaming_Developments_v1.0.pdf