

RADIUS Extensions Working Group
Internet-Draft
Intended status: Experimental
Expires: November 1, 2015

S. Winter
RESTENA
M. McCauley
AirSpayce
April 30, 2015

NAI-based Dynamic Peer Discovery for RADIUS/TLS and RADIUS/DTLS
draft-ietf-radext-dynamic-discovery-15

Abstract

This document specifies a means to find authoritative RADIUS servers for a given realm. It is used in conjunction with either RADIUS/TLS and RADIUS/DTLS.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 1, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Requirements Language	5
1.2. Terminology	6
1.3. Document Status	6
2. Definitions	7
2.1. DNS Resource Record (RR) definition	7
2.1.1. S-NAPTR	7
2.1.2. SRV	11
2.1.3. Optional name mangling	12
2.2. Definition of the X.509 certificate property SubjectAltName:otherName:NAIRealm	13
3. DNS-based NAPTR/SRV Peer Discovery	15
3.1. Applicability	15
3.2. Configuration Variables	16
3.3. Terms	16
3.4. Realm to RADIUS server resolution algorithm	17
3.4.1. Input	17
3.4.2. Output	18
3.4.3. Algorithm	18
3.4.4. Validity of results	19
3.4.5. Delay considerations	20
3.4.6. Example	21
4. Operations and Manageability Considerations	23
5. Security Considerations	24
6. Privacy Considerations	25
7. IANA Considerations	26
8. References	28
8.1. Normative References	28
8.2. Informative References	29
Appendix A. Appendix A: ASN.1 Syntax of NAIRealm	30

1. Introduction

RADIUS in all its current transport variants (RADIUS/UDP, RADIUS/TCP, RADIUS/TLS, RADIUS/DTLS) requires manual configuration of all peers (clients, servers).

Where more than one administrative entity collaborates for RADIUS authentication of their respective customers (a "roaming consortium"), the Network Access Identifier (NAI) [I-D.ietf-radext-nai] is the suggested way of differentiating users between those entities; the part of a username to the right of the @ delimiter in an NAI is called the user's "realm". Where many realms and RADIUS forwarding servers are in use, the number of realms to be forwarded and the corresponding number of servers to configure may be significant. Where new realms with new servers are added or details

of existing servers change on a regular basis, maintaining a single monolithic configuration file for all these details may prove too cumbersome to be useful.

Furthermore, in cases where a roaming consortium consists of independently working branches (e.g. departments, national subsidiaries), each with their own forwarding servers, and who add or change their realm lists at their own discretion, there is additional complexity in synchronising the changed data across all branches.

Where realms can be partitioned (e.g. according to their top-level domain ending), forwarding of requests can be realised with a hierarchy of RADIUS servers, all serving their partition of the realm space. Figure 1 show an example of this hierarchical routing.

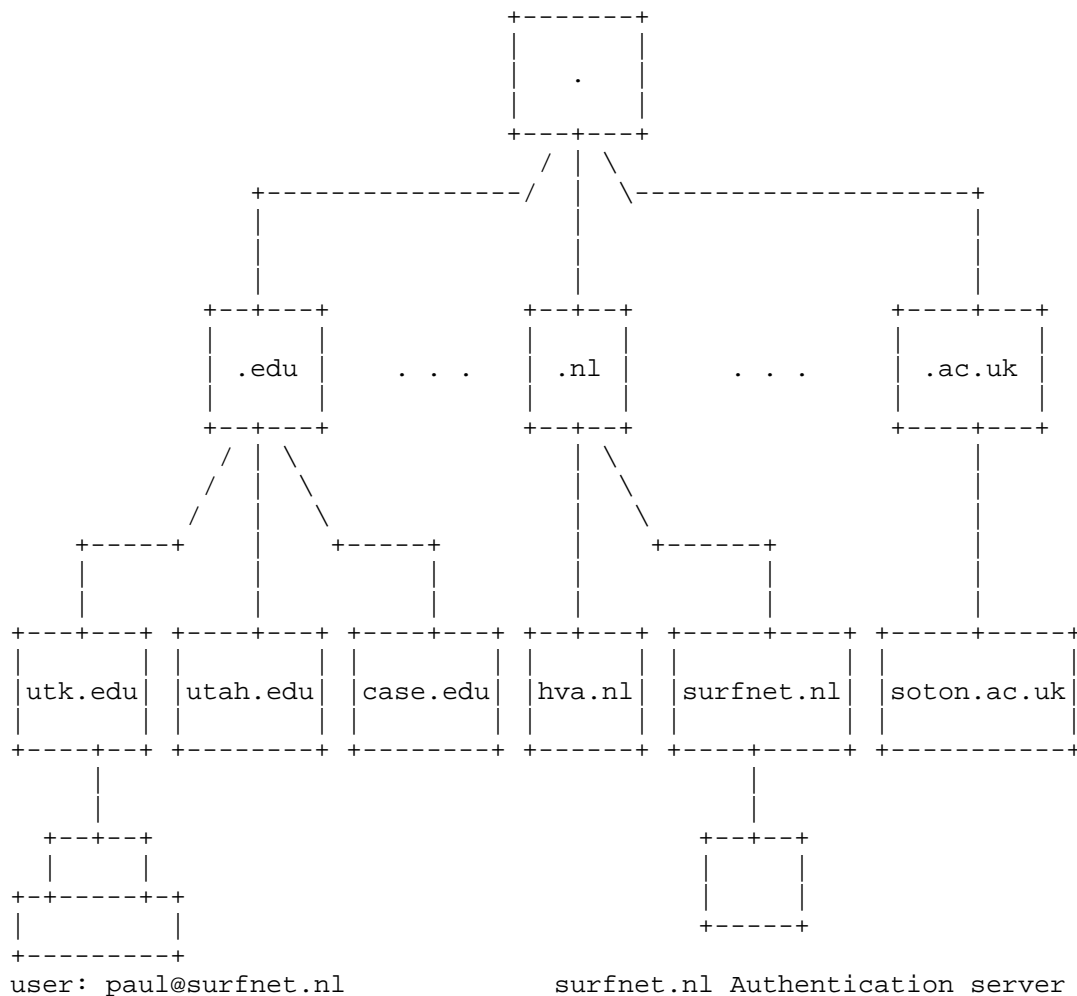


Figure 1: RADIUS hierarchy based on Top-Level Domain partitioning

However, such partitioning is not always possible. As an example, in one real-life deployment, the administrative boundaries and RADIUS forwarding servers are organised along country borders, but generic top-level domains such as .edu do not map to this choice of boundaries (see [I-D.wierenga-ietf-eduroam] for details). These situations can benefit significantly from a distributed mechanism for storing realm and server reachability information. This document describes one such mechanism: storage of realm-to-server mappings in DNS; realm-based request forwarding can then be realised without a static hierarchy such as in the following figure:

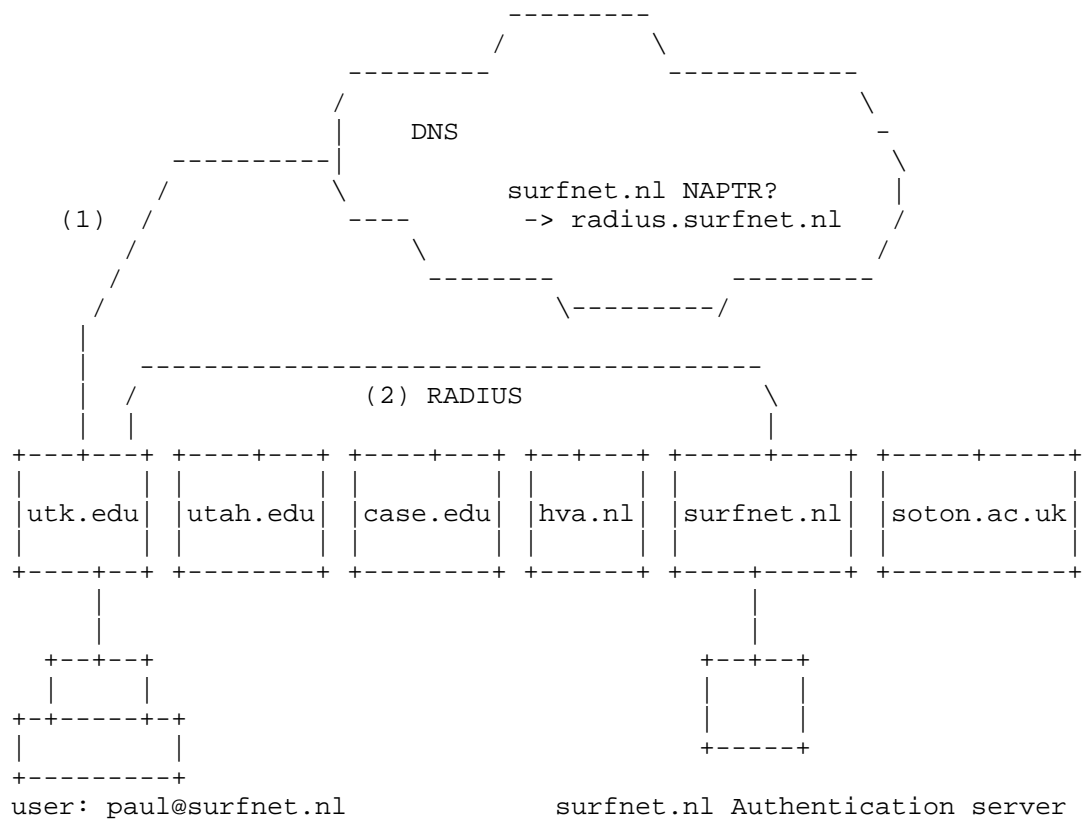


Figure 2: RADIUS hierarchy based on Top-Level Domain partitioning

This document also specifies various approaches for verifying that server information which was retrieved from DNS was from an authorised party; e.g. an organisation which is not at all part of a given roaming consortium may alter its own DNS records to yield a result for its own realm.

1.1. Requirements Language

In this document, several words are used to signify the requirements of the specification. The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119. [RFC2119]

1.2. Terminology

RADIUS/TLS Client: a RADIUS/TLS [RFC6614] instance which initiates a new connection.

RADIUS/TLS Server: a RADIUS/TLS [RFC6614] instance which listens on a RADIUS/TLS port and accepts new connections

RADIUS/TLS node: a RADIUS/TLS client or server

[I-D.ietf-radext-nai] defines the terms NAI, realm, consortium.

1.3. Document Status

This document is an Experimental RFC.

The communities expected to use this document are roaming consortia whose authentication services are based on the RADIUS protocol.

The duration of the experiment is undetermined; as soon as enough experience is collected on the choice points mentioned below, it is expected to be obsoleted by a standards-track version of the protocol which trims down the choice points.

If that removal of choice points obsoletes tags or service names as defined in this document and allocated by IANA, these items will be returned to IANA as per the provisions in [RFC6335].

The document provides a discovery mechanism for RADIUS which is very similar to the approach that is taken with the Diameter protocol [RFC6733]. As such, the basic approach (using Naming Authority Pointer (NAPTR) records in DNS domains which match NAI realms) is not of very experimental nature.

However, the document offers a few choice points and extensions which go beyond the provisions for Diameter. The list of major additions/deviations is

- o provisions for determining the authority of a server to act for users of a realm (declared out of scope for Diameter)
- o much more in-depth guidance on DNS regarding timeouts, failure conditions, alteration of Time-To-Live (TTL) information than the Diameter counterpart
- o a partially correct routing error detection during DNS lookups

2. Definitions

2.1. DNS Resource Record (RR) definition

DNS definitions of RADIUS/TLS servers can be either S-NAPTR records (see [RFC3958]) or Service Record (SRV) records. When both are defined, the resolution algorithm prefers S-NAPTR results (see Section 3.4 below).

2.1.1. S-NAPTR

2.1.1.1. Registration of Application Service and Protocol Tags

This specification defines three S-NAPTR service tags:

Service Tag	Use
aaa+auth	RADIUS Authentication, i.e. traffic as defined in [RFC2865]
aaa+acct	RADIUS Accounting, i.e. traffic as defined in [RFC2866]
aaa+dynauth	RADIUS Dynamic Authorisation, i.e. traffic as defined in [RFC5176]

Figure 3: List of Service Tags

This specification defines two S-NAPTR protocol tags:

Protocol Tag	Use
radius.tls.tcp	RADIUS transported over TLS as defined in [RFC6614]
radius.dtls.udp	RADIUS transported over DTLS as defined in [RFC7360]

Figure 4: List of Protocol Tags

Note well:

The S-NAPTR service and protocols are unrelated to the IANA Service Name and Transport Protocol Number registry.

The delimiter '.' in the protocol tags is only a separator for human reading convenience - not for structure or namespacing; it MUST NOT be parsed in any way by the querying application or resolver.

The use of the separator '.' is common also in other protocols' protocol tags. This is coincidence and does not imply a shared semantics with such protocols.

2.1.1.2. Definition of Conditions for Retry/Failure

RADIUS is a time-critical protocol; RADIUS clients which do not receive an answer after a configurable, but short, amount of time, will consider the request failed. Due to this, there is little leeway for extensive retries.

As a general rule, only error conditions which generate an immediate response from the other end are eligible for a retry of a discovered target. Any error condition involving timeouts, or the absence of a reply for more than one second during the connection setup phase is to be considered a failure; the next target in the set of discovered NAPTR targets is to be tried.

Note that [RFC3958] already defines that a failure to identify the server as being authoritative for the realm is always considered a failure; so even if a discovered target returns a wrong credential instantly, it is not eligible for retry.

Furthermore, the contacted RADIUS/TLS server verifies during connection setup whether or not it finds the connecting RADIUS/TLS client authorized or not. If the connecting RADIUS/TLS client is not found acceptable, the server will close the TLS connection immediately with an appropriate alert. Such TLS handshake failures are permanently fatal and not eligible for retry, unless the connecting client has more X.509 certificates to try; in this case, a retry with the remainder of its set of certificates SHOULD be attempted. Not trying all available client certificates potentially creates a DoS for the end-user whose authentication attempt triggered the discovery; one of the neglected certificates might have led to a successful RADIUS connection and subsequent end-user authentication.

If the TLS session setup to a discovered target does not succeed, that target (as identified by IP address and port number) SHOULD be ignored from the result set of any subsequent executions of the discovery algorithm at least until the target's Effective TTL (see

Section 3.3) has expired or until the entity which executes the algorithm changes its TLS context to either send a new client certificate or expect a different server certificate.

2.1.1.3. Server Identification and Handshake

After the algorithm in this document has been executed, a RADIUS/TLS session as per [RFC6614] is established. Since the dynamic discovery algorithm does not have provisions to establish confidential keying material between the RADIUS/TLS client (i.e. the server which executes the discovery algorithm) and the RADIUS/TLS server which was discovered, TLS-PSK ciphersuites cannot be used in the subsequent TLS handshake. Only TLS ciphersuites using X.509 certificates can be used with this algorithm.

There are numerous ways to define which certificates are acceptable for use in this context. This document defines one mandatory-to-implement mechanism which allows to verify whether the contacted host is authoritative for an NAI realm or not. It also gives one example of another mechanism which is currently in wide-spread deployment, and one possible approach based on DNSSEC which is yet unimplemented.

For the approaches which use trust roots (see the following two sections), a typical deployment will use a dedicated trust store for RADIUS/TLS certificate authorities, particularly a trust store which is independent from default "browser" trust stores. Often, this will be one or few CAs, and they only issue certificates for the specific purpose of establishing RADIUS server-to-server trust. It is important not to trust a large set of CAs which operate outside the control of the roaming consortium, for their issuance of certificates with the properties important for authorisation (such as NAIRealm and policyOID below) is difficult to verify. Therefore, clients SHOULD NOT be pre-configured with a list of known public CAs by the vendor or manufacturer. Instead, the clients SHOULD start off with an empty CA list. The addition of a CA SHOULD be done only when manually configured by an administrator.

2.1.1.3.1. Mandatory-to-implement mechanism: Trust Roots + NAIRealm

Verification of authority to provide AAA services over RADIUS/TLS is a two-step process.

Step 1 is the verification of certificate wellformedness and validity as per [RFC5280] and whether it was issued from a root certificate which is deemed trustworthy by the RADIUS/TLS client.

Step 2 is to compare the value of algorithm's variable "R" after the execution of step 3 of the discovery algorithm in Section 3.4.3 below

(i.e. after a consortium name mangling, but before conversion to a form usable by the name resolution library) to all values of the contacted RADIUS/TLS server's X.509 certificate property "subjectAlternativeName:otherName:NAIRealm" as defined in Section 2.2.

2.1.1.3.2. Other mechanism: Trust Roots + policyOID

Verification of authority to provide AAA services over RADIUS/TLS is a two-step process.

Step 1 is the verification of certificate wellformedness and validity as per [RFC5280] and whether it was issued from a root certificate which is deemed trustworthy by the RADIUS/TLS client.

Step 2 is to compare the values of the contacted RADIUS/TLS server's X.509 certificate's extensions of type "Policy OID" to a list of configured acceptable Policy OIDs for the roaming consortium. If one of the configured OIDs is found in the certificate's Policy OID extensions, then the server is considered authorized; if there is no match, the server is considered unauthorized.

This mechanism is inferior to the mandatory-to-implement mechanism in the previous section because all authorized servers are validated by the same OID value; the mechanism is not fine-grained enough to express authority for one specific realm inside the consortium. If the consortium contains members which are hostile against other members, this weakness can be exploited by one RADIUS/TLS server impersonating another if DNS responses can be spoofed by the hostile member.

The shortcomings in server identification can be partially mitigated by using the RADIUS infrastructure only with authentication payloads which provide mutual authentication and credential protection (i.e. EAP types passing the criteria of [RFC4017]): using mutual authentication prevents the hostile server from mimicking the real EAP server (it can't terminate the EAP authentication unnoticed because it does not have the server certificate from the real EAP server); protection of credentials prevents the impersonating server from learning usernames and passwords of the ongoing EAP conversation (other RADIUS attributes pertaining to the authentication, such as the EAP peer's Calling-Station-ID, can still be learned though).

2.1.1.3.3. Other mechanism: DNSSEC / DANE

Where DNSSEC is used, the results of the algorithm can be trusted; i.e. the entity which executes the algorithm can be certain that the realm that triggered the discovery is actually served by the server

that was discovered via DNS. However, this does not guarantee that the server is also authorized (i.e. a recognised member of the roaming consortium). The server still needs to present an X.509 certificate proving its authority to serve a particular realm.

The authorization can be sketched using DNSSEC+DANE as follows: DANE/TLSA records of all authorized servers are put into a DNSSEC zone which contains all known and authorised realms; the zone is rooted in a common, consortium-agreed branch of the DNS tree. The entity executing the algorithm uses the realm information from the authentication attempt, and then attempts to retrieve TLSA Resource Records (TLSA RR) for the DNS label "realm.commonroot". It then verifies that the presented server certificate during the RADIUS/TLS handshake matches the information in the TLSA record.

Example:

```
Realm = "example.com"

Common Branch = "idp.roaming-consortium.example.

label for TLSA query = "example.com.idp.roaming-
consortium.example.

result of discovery algorithm for realm "example.com" =
192.0.2.1:2083

( TLS certificate of 192.0.2.1:2083 matches TLSA RR ? "PASS" :
"FAIL" )
```

2.1.1.3.4. Client Authentication and Authorisation

Note that RADIUS/TLS connections always mutually authenticate the RADIUS server and the RADIUS client. This specification provides an algorithm for a RADIUS client to contact and verify authorization of a RADIUS server only. During connection setup, the RADIUS server also needs to verify whether it considers the connecting RADIUS client authorized; this is outside the scope of this specification.

2.1.2. SRV

This specification defines two SRV prefixes (i.e. two values for the "_service._proto" part of an SRV RR as per [RFC2782]):

SRV Label	Use
_radiustls._tcp	RADIUS transported over TLS as defined in [RFC6614]
_radiusdtls._udp	RADIUS transported over DTLS as defined in [RFC7360]

Figure 5: List of SRV Labels

Just like NAPTR records, the lookup and subsequent follow-up of SRV records may yield more than one server to contact in a prioritised list. [RFC2782] does not specify rules regarding "Definition of Conditions for Retry/Failure", nor "Server Identification and Handshake". This specification defines that the rules for these two topics as defined in Section 2.1.1.2 and Section 2.1.1.3 SHALL be used both for targets retrieved via an initial NAPTR RR as well as for targets retrieved via an initial SRV RR (i.e. in the absence of NAPTR RRs).

2.1.1.3. Optional name mangling

It is expected that in most cases, the SRV and/or NAPTR label used for the records is the DNS A-label representation of the literal realm name for which the server is the authoritative RADIUS server (i.e. the realm name after conversion according to section 5 of [RFC5891]).

However, arbitrary other labels or service tags may be used if, for example, a roaming consortium uses realm names which are not associated to DNS names or special-purpose consortia where a globally valid discovery is not a use case. Such other labels require a consortium-wide agreement about the transformation from realm name to lookup label, and/or which service tag to use.

Examples:

- a. A general-purpose RADIUS server for realm example.com might have DNS entries as follows:

```
example.com. IN NAPTR 50 50 "s" "aaa+auth:radius.tls.tcp" ""
_radiustls._tcp.foobar.example.com.

_radiustls._tcp.foobar.example.com. IN SRV 0 10 2083
radsec.example.com.
```

- b. The consortium "foo" provides roaming services for its members only. The realms used are of the form enterprise-name.example. The consortium operates a special purpose DNS server for the (private) TLD "example" which all RADIUS servers use to resolve realm names. "Company, Inc." is part of the consortium. On the consortium's DNS server, realm company.example might have the following DNS entries:

```
company.example. IN NAPTR 50 50 "a"  
"aaa+auth:radius.dtls.udp" "" roamserv.company.example.
```

- c. The eduroam consortium (see [I-D.wierenga-ietf-eduroam]) uses realms based on DNS, but provides its services to a closed community only. However, a AAA domain participating in eduroam may also want to expose AAA services to other, general-purpose, applications (on the same or other RADIUS servers). Due to that, the eduroam consortium uses the service tag "x-eduroam" for authentication purposes and eduroam RADIUS servers use this tag to look up other eduroam servers. An eduroam participant example.org which also provides general-purpose AAA on a different server uses the general "aaa+auth" tag:

```
example.org. IN NAPTR 50 50 "s" "x-eduroam:radius.tls.tcp" ""  
_radiustls._tcp.eduroam.example.org.
```

```
example.org. IN NAPTR 50 50 "s" "aaa+auth:radius.tls.tcp" ""  
_radiustls._tcp.aaa.example.org.
```

```
_radiustls._tcp.eduroam.example.org. IN SRV 0 10 2083 aaa-  
eduroam.example.org.
```

```
_radiustls._tcp.aaa.example.org. IN SRV 0 10 2083 aaa-  
default.example.org.
```

2.2. Definition of the X.509 certificate property SubjectAltName:otherName:NAIRealm

This specification retrieves IP addresses and port numbers from the Domain Name System which are subsequently used to authenticate users via the RADIUS/TLS protocol. Regardless whether the results from DNS discovery are trustworthy or not (e.g. DNSSEC in use), it is always important to verify that the server which was contacted is authorized to service requests for the user which triggered the discovery process.

The input to the algorithm is an NAI realm as specified in Section 3.4.1. As a consequence, the X.509 certificate of the server which is ultimately contacted for user authentication needs to be

able to express that it is authorized to handle requests for that realm.

Current subjectAltName fields do not semantically allow to express an NAI realm; the field subjectAltName:dNSName is syntactically a good match but would inappropriately conflate DNS names and NAI realm names. Thus, this specification defines a new subjectAltName field to hold either a single NAI realm name or a wildcard name matching a set of NAI realms.

The subjectAltName:otherName:SRVName field certifies that a certificate holder is authorized to provide a service; this can be compared to the target of DNS label's SRV resource record. If the Domain Name System is insecure, it is required that the label of the SRV record itself is known-correct. In this specification, that label is not known-correct; it is potentially derived from a (potentially untrusted) NAPTR resource record of another label. If DNS is not secured with DNSSEC, the NAPTR resource record may have been altered by an attacker with access to the Domain Name System resolution, and thus the label to lookup the SRV record for may already be tainted. This makes subjectAltName:otherName:SRVName not a trusted comparison item.

Further to this, this specification's NAPTR entries may be of type "A" which do not involve resolution of any SRV records, which again makes subjectAltName:otherName:SRVName unsuited for this purpose.

This section defines the NAIRealm name as a form of otherName from the GeneralName structure in SubjectAltName defined in [RFC5280].

```
id-on-naiRealm OBJECT IDENTIFIER ::= { id-on XXX }
```

```
ub-naiRealm-length INTEGER ::= 255
```

```
NAIRealm ::= UTF8String (SIZE (1..ub-naiRealm-length))
```

The NAIRealm, if present, MUST contain an NAI realm as defined in [I-D.ietf-radext-nai]. It MAY substitute the leftmost dot-separated label of the NAI with the single character "*" to indicate a wildcard match for "all labels in this part". Further features of regular expressions, such as a number of characters followed by a * to indicate a common prefix inside the part, are not permitted.

The comparison of an NAIRealm to the NAI realm as derived from user input with this algorithm is a byte-by-byte comparison, except for the optional leftmost dot-separated part of the value whose content is a single "*" character; such labels match all strings in the same dot-separated part of the NAI realm. If at least one of the

sAN:otherName:NAIRealm values matches the NAI realm, the server is considered authorized; if none matches, the server is considered unauthorized.

Since multiple names and multiple name forms may occur in the subjectAltName extension, an arbitrary number of NAIRealms can be specified in a certificate.

Examples:

NAI realm (RADIUS)	NAIRealm (cert)	MATCH?
foo.example	foo.example	YES
foo.example	*.example	YES
bar.foo.example	*.example	NO
bar.foo.example	*ar.foo.example	NO (NAIRealm invalid)
bar.foo.example	bar.*.example	NO (NAIRealm invalid)
bar.foo.example	*.*.example	NO (NAIRealm invalid)
sub.bar.foo.example	*.*.example	NO (NAIRealm invalid)
sub.bar.foo.example	*.bar.foo.example	YES

Figure 6: Examples for NAI realm vs. certificate matching

Appendix A contains the ASN.1 definition of the above objects.

3. DNS-based NAPTR/SRV Peer Discovery

3.1. Applicability

Dynamic server discovery as defined in this document is only applicable for new AAA transactions and per service (i.e. distinct discovery is needed for Authentication, Accounting, and Dynamic Authorization) where a RADIUS entity which acts as a forwarding server for one or more realms receives a request with a realm for which it is not authoritative, and which no explicit next hop is configured. It is only applicable for

- a. new user sessions, i.e. for the initial Access-Request. Subsequent messages concerning this session, for example Access-Challenges and Access-Accepts use the previously-established communication channel between client and server.
- b. the first accounting ticket for a user session.
- c. the first RADIUS DynAuth packet for a user session.

3.2. Configuration Variables

The algorithm contains various variables for timeouts. These variables are named here and reasonable default values are provided. Implementations wishing to deviate from these defaults should make they understand the implications of changes.

DNS_TIMEOUT: maximum amount of time to wait for the complete set of all DNS queries to complete: Default = 3 seconds

MIN_EFF_TTL: minimum DNS TTL of discovered targets: Default = 60 seconds

BACKOFF_TIME: if no conclusive DNS response was retrieved after DNS_TIMEOUT, do not attempt dynamic discovery before BACKOFF_TIME has elapsed. Default = 600 seconds

3.3. Terms

Positive DNS response: a response which contains the RR that was queried for.

Negative DNS response: a response which does not contain the RR that was queried for, but contains an SOA record along with a TTL indicating cache duration for this negative result.

DNS Error: Where the algorithm states "name resolution returns with an error", this shall mean that either the DNS request timed out, or a DNS response which is neither a positive nor a negative response (e.g. SERVFAIL).

Effective TTL: The validity period for discovered RADIUS/TLS target hosts. Calculated as: Effective TTL (set of DNS TTL values) = max { MIN_EFF_TTL, min { DNS TTL values } }

SRV lookup: for the purpose of this specification, SRV lookup procedures are defined as per [RFC2782], but excluding that RFCs "A" fallback as defined in its section "Usage Rules", final "else" clause.

Greedy result evaluation: The NAPTR to SRV/A/AAAA resolution may lead to a tree of results, whose leafs are the IP addresses to contact. The branches of the tree are ordered according to their order/preference DNS properties. An implementation is executing greedy result evaluation if it uses a depth-first search in the tree along the highest order results, attempts to connect to the corresponding resulting IP addresses, and only backtracks to other branches if the higher ordered results did not end in successful connection attempts.

3.4. Realm to RADIUS server resolution algorithm

3.4.1. Input

For RADIUS Authentication and RADIUS Accounting server discovery, input I to the algorithm is the RADIUS User-Name attribute with content of the form "user@realm"; the literal @ sign being the separator between a local user identifier within a realm and its realm. The use of multiple literal @ signs in a User-Name is strongly discouraged; but if present, the last @ sign is to be considered the separator. All previous instances of the @ sign are to be considered part of the local user identifier.

For RADIUS DynAuth Server discovery, input I to the algorithm is the domain name of the operator of a RADIUS realm as was communicated during user authentication using the Operator-Name attribute ([RFC5580], section 4.1). Only Operator-Name values with the namespace "1" are supported by this algorithm - the input to the algorithm is the actual domain name, preceded with an "@" (but without the "1" namespace identifier byte of that attribute).

Note well: The attribute User-Name is defined to contain UTF-8 text. In practice, the content may or may not be UTF-8. Even if UTF-8, it may or may not map to a domain name in the realm part. Implementors MUST take possible conversion error paths into consideration when parsing incoming User-Name attributes. This document describes server discovery only for well-formed realms mapping to DNS domain names in UTF-8 encoding. The result of all other possible contents of User-Name is unspecified; this includes, but is not limited to:

- Usage of separators other than @.

- Encoding of User-Name in local encodings.

- UTF-8 realms which fail the conversion rules as per [RFC5891].

- UTF-8 realms which end with a . ("dot") character.

For the last bullet point, "trailing dot", special precautions should be taken to avoid problems when resolving servers with the algorithm below: they may resolve to a RADIUS server even if the peer RADIUS server only is configured to handle the realm without the trailing dot. If that RADIUS server again uses NAI discovery to determine the authoritative server, the server will forward the request to localhost, resulting in a tight endless loop.

3.4.2. Output

Output O of the algorithm is a two-tuple consisting of: O-1) a set of tuples {hostname; port; protocol; order/preference; Effective TTL} - the set can be empty; and O-2) an integer: if the set in the first part of the tuple is empty, the integer contains the Effective TTL for backoff timeout, if the set is not empty, the integer is set to 0 (and not used).

3.4.3. Algorithm

The algorithm to determine the RADIUS server to contact is as follows:

1. Determine P = (position of last "@" character) in I.
2. generate R = (substring from P+1 to end of I)
3. modify R according to agreed consortium procedures if applicable
4. convert R to a representation usable by the name resolution library if needed
5. Initialize TIMER = 0; start TIMER. If TIMER reaches DNS_TIMEOUT, continue at step 20.
6. Using the host's name resolution library, perform a NAPTR query for R (see "Delay considerations" below). If the result is a negative DNS response, O-2 = Effective TTL (TTL value of the SOA record) and continue at step 13. If name resolution returns with error, O-1 = { empty set }, O-2 = BACKOFF_TIME and terminate.
7. Extract NAPTR records with service tag "aaa+auth", "aaa+acct", "aaa+dynauth" as appropriate. Keep note of the protocol tag and remaining TTL of each of the discovered NAPTR records.
8. If no records found, continue at step 13.
9. For the extracted NAPTRs, perform successive resolution as defined in [RFC3958], section 2.2. An implementation MAY use greedy result evaluation according to the NAPTR order/preference fields (i.e. can execute the subsequent steps of this algorithm for the highest-order entry in the set of results, and only lookup the remainder of the set if necessary).
10. If the set of hostnames is empty, O-1 = { empty set }, O-2 = BACKOFF_TIME and terminate.

11. $O' = (\text{set of } \{\text{hostname; port; protocol; order/preference; Effective TTL (all DNS TTLs that led to this hostname) } \} \text{ for all terminal lookup results}).$
12. Proceed with step 18.
13. Generate $R' = (\text{prefix R with } _radiustls._tcp. \text{ and/or } _radiustls._udp.)$
14. Using the host's name resolution library, perform SRV lookup with R' as label (see "Delay considerations" below).
15. If name resolution returns with error, $O-1 = \{ \text{empty set} \}$, $O-2 = \text{BACKOFF_TIME}$ and terminate.
16. If the result is a negative DNS response, $O-1 = \{ \text{empty set} \}$, $O-2 = \min \{ O-2, \text{Effective TTL (TTL value of the SOA record) } \}$ and terminate.
17. $O' = (\text{set of } \{\text{hostname; port; protocol; order/preference; Effective TTL (all DNS TTLs that led to this result) } \} \text{ for all hostnames}).$
18. Generate $O-1$ by resolving hostnames in O' into corresponding A and/or AAAA addresses: $O-1 = (\text{set of } \{\text{IP address; port; protocol; order/preference; Effective TTL (all DNS TTLs that led to this result) } \} \text{ for all hostnames }), O-2 = 0.$
19. For each element in $O-1$, test if the original request which triggered dynamic discovery was received on $\{\text{IP address; port}\}$. If yes, $O-1 = \{ \text{empty set} \}$, $O-2 = \text{BACKOFF_TIME}$, log error, Terminate (see next section for a rationale). If no, O is the result of dynamic discovery. Terminate.
20. $O-1 = \{ \text{empty set} \}$, $O-2 = \text{BACKOFF_TIME}$, log error, Terminate.

3.4.4. Validity of results

The dynamic discovery algorithm is used by servers which do not have sufficient configuration information to process an incoming request on their own. If the discovery algorithm result contains the server's own listening address (IP address and port), then there is a potential for an endless forwarding loop. If the listening address is the DNS result with the highest priority, the server will enter a tight loop (the server would forward the request to itself, triggering dynamic discovery again in a perpetual loop). If the address has a lower priority in the set of results, there is a potential loop with intermediate hops in between (the server could

forward to another host with a higher priority, which might use DNS itself and forward the packet back to the first server). The underlying reason that enables these loops is that the server executing the discovery algorithm is seriously misconfigured in that it does not recognise the request as one that is to be processed by itself. RADIUS has no built-in loop detection, so any such loops would remain undetected. So, if step 18 of the algorithm discovers such a possible-loop situation, the algorithm should be aborted and an error logged. Note that this safeguard does not provide perfect protection against routing loops. One reason which might introduce a loop include the possibility that a subsequent hop has a statically configured next-hop which leads to an earlier host in the loop. Another reason for occurring loops is if the algorithm was executed with greedy result evaluation, and the own address was in a lower-priority branch of the result set which was not retrieved from DNS at all, and thus can't be detected.

After executing the above algorithm, the RADIUS server establishes a connection to a home server from the result set. This connection can potentially remain open for an indefinite amount of time. This conflicts with the possibility of changing device and network configurations on the receiving end. Typically, TTL values for records in the name resolution system are used to indicate how long it is safe to rely on the results of the name resolution. If these TTLs are very low, thrashing of connections becomes possible; the Effective TTL mitigates that risk. When a connection is open and the smallest of the Effective TTL value which was learned during discovering the server has not expired, subsequent new user sessions for the realm which corresponds to that open connection SHOULD re-use the existing connection and SHOULD NOT re-execute the dynamic discovery algorithm nor open a new connection. To allow for a change of configuration, a RADIUS server SHOULD re-execute the dynamic discovery algorithm after the Effective TTL that is associated with this connection has expired. The server SHOULD keep the session open during this re-assessment to avoid closure and immediate re-opening of the connection should the result not have changed.

Should the algorithm above terminate with O-1 = empty set, the RADIUS server SHOULD NOT attempt another execution of this algorithm for the same target realm before the timeout O-2 has passed.

3.4.5. Delay considerations

The host's name resolution library may need to contact outside entities to perform the name resolution (e.g. authoritative name servers for a domain), and since the NAI discovery algorithm is based on uncontrollable user input, the destination of the lookups is out of control of the server that performs NAI discovery. If such

outside entities are misconfigured or unreachable, the algorithm above may need an unacceptably long time to terminate. Many RADIUS implementations time out after five seconds of delay between Request and Response. It is not useful to wait until the host name resolution library signals a timeout of its name resolution algorithms. The algorithm therefore controls execution time with TIMER. Execution of the NAI discovery algorithm SHOULD be non-blocking (i.e. allow other requests to be processed in parallel to the execution of the algorithm).

3.4.6. Example

Assume

a user from the Technical University of Munich, Germany, has a RADIUS User-Name of "foobar@tu-m[U+00FC]nchen.example".

The name resolution library on the RADIUS forwarding server does not have the realm tu-m[U+00FC]nchen.example in its forwarding configuration, but uses DNS for name resolution and has configured the use of Dynamic Discovery to discover RADIUS servers.

It is IPv6-enabled and prefers AAAA records over A records.

It is listening for incoming RADIUS/TLS requests on 192.0.2.1, TCP /2083.

May the configuration variables be

DNS_TIMEOUT = 3 seconds

MIN_EFF_TTL = 60 seconds

BACKOFF_TIME = 3600 seconds

If DNS contains the following records:

xn--tu-mnchen-t9a.example. IN NAPTR 50 50 "s"
"aaa+auth:radius.tls.tcp" "" _myradius._tcp.xn--tu-mnchen-t9a.example.

xn--tu-mnchen-t9a.example. IN NAPTR 50 50 "s"
"fooservice:bar.dccp" "" _abc123._def.xn--tu-mnchen-t9a.example.

_myradius._tcp.xn--tu-mnchen-t9a.example. IN SRV 0 10 2083
radsecserver.xn--tu-mnchen-t9a.example.

```
_myradius._tcp.xn--tu-mnchen-t9a.example.  IN SRV 0 20 2083  
backupserver.xn--tu-mnchen-t9a.example.
```

```
radsecserver.xn--tu-mnchen-t9a.example.  IN AAAA  
2001:0DB8::202:44ff:fe0a:f704
```

```
radsecserver.xn--tu-mnchen-t9a.example.  IN A 192.0.2.3
```

```
backupserver.xn--tu-mnchen-t9a.example.  IN A 192.0.2.7
```

Then the algorithm executes as follows, with I =
"foobar@tu-m[U+00FC]nchen.example", and no consortium name mangling
in use:

1. P = 7
2. R = "tu-m[U+00FC]nchen.example"
3. NOOP
4. name resolution library converts R to xn--tu-mnchen-t9a.example
5. TIMER starts.
6. Result:

```
(TTL = 47) 50 50 "s" "aaa+auth:radius.tls.tcp" ""  
_myradius._tcp.xn--tu-mnchen-t9a.example.  
  
(TTL = 522) 50 50 "s" "fooservice:bar.dccp" ""  
_abc123._def.xn--tu-mnchen-t9a.example.
```
7. Result:

```
(TTL = 47) 50 50 "s" "aaa+auth:radius.tls.tcp" ""  
_myradius._tcp.xn--tu-mnchen-t9a.example.
```
8. NOOP
9. Successive resolution performs SRV query for label
_myradius._tcp.xn--tu-mnchen-t9a.example, which results in

```
(TTL 499) 0 10 2083 radsec.xn--tu-mnchen-t9a.example.  
  
(TTL 2200) 0 20 2083 backup.xn--tu-mnchen-t9a.example.
```
10. NOOP

11. $O' = \{$
 (radsec.xn--tu-mnchen-t9a.example.; 2083; RADIUS/TLS; 10;
 60),
 (backup.xn--tu-mnchen-t9a.example.; 2083; RADIUS/TLS; 20; 60)
 } // minimum TTL is 47, up'ed to MIN_EFF_TTL
12. Continuing at 18.
13. (not executed)
14. (not executed)
15. (not executed)
16. (not executed)
17. (not executed)
18. $O-1 = \{$
 (2001:0DB8::202:44ff:fe0a:f704; 2083; RADIUS/TLS; 10; 60),
 (192.0.2.7; 2083; RADIUS/TLS; 20; 60)
 }; $O-2 = 0$
19. No match with own listening address; terminate with tuple (O-1,
 O-2) from previous step.

The implementation will then attempt to connect to two servers, with preference to [2001:0DB8::202:44ff:fe0a:f704]:2083 using the RADIUS/TLS protocol.

4. Operations and Manageability Considerations

The discovery algorithm as defined in this document contains several options; the major ones being use of NAPTR vs. SRV; how to determine the authorization status of a contacted server for a given realm; which trust anchors to consider trustworthy for the RADIUS conversation setup.

Random parties which do not agree on the same set of options may not be able to interoperate. However, such a global interoperability is not intended by this document.

Discovery as per this document becomes important inside a roaming consortium, which has set up roaming agreements with the other partners. Such roaming agreements require much more than a technical means of server discovery; there are administrative and contractual considerations at play (service contracts, backoffice compensations, procedures, ...).

A roaming consortium's roaming agreement must include a profile of which choice points of this document to use. So long as the roaming consortium can settle on one deployment profile, they will be able to interoperate based on that choice; this per-consortium interoperability is the intended scope of this document.

5. Security Considerations

When using DNS without DNSSEC security extensions and validation for all of the replies to NAPTR, SRV and A/AAAA requests as described in section Section 3, the result of the discovery process can not be trusted. Even if it can be trusted (i.e. DNSSEC is in use), actual authorization of the discovered server to provide service for the given realm needs to be verified. A mechanism from section Section 2.1.1.3 or equivalent MUST be used to verify authorization.

The algorithm has a configurable completion timeout `DNS_TIMEOUT` defaulting to three seconds for RADIUS' operational reasons. The lookup of DNS resource records based on unverified user input is an attack vector for DoS attacks: an attacker might intentionally craft bogus DNS zones which take a very long time to reply (e.g. due to a particularly byzantine tree structure, or artificial delays in responses).

To mitigate this DoS vector, implementations SHOULD consider rate-limiting either their amount of new executions of the dynamic discovery algorithm as a whole, or the amount of intermediate responses to track, or at least the number of pending DNS queries. Implementations MAY choose lower values than the default for `DNS_TIMEOUT` to limit the impact of DoS attacks via that vector. They MAY also continue their attempt to resolve DNS records even after `DNS_TIMEOUT` has passed; a subsequent request for the same realm might benefit from retrieving the results anyway. The amount of time to spent waiting for a result will influence the impact of a possible DoS attack; the waiting time value is implementation dependent and outside the scope of this specification.

With Dynamic Discovery being enabled for a RADIUS Server, and depending on the deployment scenario, the server may need to open up its target IP address and port for the entire internet, because arbitrary clients may discover it as a target for their

authentication requests. If such clients are not part of the roaming consortium, the RADIUS/TLS connection setup phase will fail (which is intended) but the computational cost for the connection attempt is significant. With the port for a TLS-based service open, the RADIUS server shares all the typical attack vectors for services based on TLS (such as HTTPS, SMTPS, ...). Deployments of RADIUS/TLS with Dynamic Discovery should consider these attack vectors and take appropriate counter-measures (e.g. blacklisting known-bad IPs on a firewall, rate-limiting new connection attempts, etc.).

6. Privacy Considerations

The classic RADIUS operational model (known, pre-configured peers, shared secret security, mostly plaintext communication) and this new RADIUS dynamic discovery model (peer discovery with DNS, PKI security and packet confidentiality) differ significantly in their impact on the privacy of end users trying to authenticate to a RADIUS server.

With classic RADIUS, traffic in large environments gets aggregated by statically configured clearinghouses. The packets sent to those clearinghouses and their responses are mostly unprotected. As a consequence,

- o All intermediate IP hops can inspect most of the packet payload in clear text, including the User-Name and Calling-Station-Id attributes, and can observe which client sent the packet to which clearinghouse. This allows the creation of mobility profiles for any passive observer on the IP path.
- o The existence of a central clearinghouse creates an opportunity for the clearinghouse to trivially create the same mobility profiles. The clearinghouse may or may not be trusted not to do this, e.g. by sufficiently threatening contractual obligations.
- o In addition to that, with the clearinghouse being a RADIUS intermediate in possession of a valid shared secret, the clearinghouse can observe and record even the security-critical RADIUS attributes such as User-Password. This risk may be mitigated by choosing authentication payloads which are cryptographically secured and do not use the attribute User-Password - such as certain EAP types.
- o There is no additional information disclosure to parties outside the IP path between the RADIUS client and server (in particular, no DNS servers learn about realms of current ongoing authentications).

With RADIUS and dynamic discovery,

- o This protocol allows for RADIUS clients to identify and directly connect to the RADIUS home server. This can eliminate the use of clearinghouses to do forwarding of requests, and it also eliminates the ability of the clearinghouse to then aggregate the user information that flows through it. However, there exist reasons why clearinghouses might still be used. One reason to keep a clearinghouse is to act as a gateway for multiple backends in a company; another reason may be a requirement to sanitise RADIUS datagrams (filter attributes, tag requests with new attributes, ...).
- o Even where intermediate proxies continue to be used for reasons unrelated to dynamic discovery, the number of such intermediates may be reduced by removing those proxies which are only deployed for pure request routing reasons. This reduces the number of entities which can inspect the RADIUS traffic.
- o RADIUS clients which make use of dynamic discovery will need to query the Domain Name System, and use a user's realm name as the query label. A passive observer on the IP path between the RADIUS client and the DNS server(s) being queried can learn that a user of that specific realm was trying to authenticate at that RADIUS client at a certain point in time. This may or may not be sufficient for the passive observer to create a mobility profile. During the recursive DNS resolution, a fair number of DNS servers and the IP hops in between those get to learn that information. Not every single authentication triggers DNS lookups, so there is no one-to-one relation of leaked realm information and the number of authentications for that realm.
- o Since dynamic discovery operates on a RADIUS hop-by-hop basis, there is no guarantee that the RADIUS payload is not transmitted between RADIUS systems which do not make use of this algorithm, and possibly using other transports such as RADIUS/UDP. On such hops, the enhanced privacy is jeopardized.

In summary, with classic RADIUS, few intermediate entities learn very detailed data about every ongoing authentications, while with dynamic discovery, many entities learn only very little about recently authenticated realms.

7. IANA Considerations

This document requests IANA registration of the following entries in existing registries:

- o S-NAPTR Application Service Tags registry

- * aaa+auth
- * aaa+acct
- * aaa+dynauth
- o S-NAPTR Application Protocol Tags registry
 - * radius.tls.tcp
 - * radius.dtls.udp

This document reserves the use of the "radiustls" and "radiusdtls" service names. Registration information as per [RFC6335] section 8.1.1 is as follows:

Service Name: radiustls; radiusdtls

Transport Protocols: TCP (for radiustls), UDP (for radiusdtls)

Assignee: IESG <iesg@ietf.org>

Contact: IETF Chair <chair@ietf.org>

Description: Authentication, Accounting and Dynamic authorization via the RADIUS protocol. These service names are used to construct the SRV service labels "_radiustls" and "_radiusdtls" for discovery of RADIUS/TLS and RADIUS/DTLS servers, respectively.

Reference: RFC Editor Note: please insert the RFC number of this document. The protocol does not use broadcast, multicast or anycast communication.

This specification makes use of the SRV Protocol identifiers "_tcp" and "_udp" which are mentioned as early as [RFC2782] but do not appear to be assigned in an actual registry. Since they are in widespread use in other protocols, this specification refrains from requesting a new registry "RADIUS/TLS SRV Protocol Registry" and continues to make use of these tags implicitly.

This document requires that a number of Object Identifiers be assigned. They are now under the control of IANA following [RFC7299]

IANA is requested to assign the following identifiers:

TBD99 is to be assigned from the "SMI Security for PKIX Module Identifier Registry". The suggested description is id-mod-nai-realm-08.

TBD98 is to be assigned from the "SMI Security for PKIX Other Name Forms Registry." The suggested description is id-on-naiRealm.

RFC Editor Note: please replace the occurrences of TBD98 and TBD99 in Appendix A of the document with the actually assigned numbers.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.
- [RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.
- [RFC2866] Rigney, C., "RADIUS Accounting", RFC 2866, June 2000.
- [RFC3958] Daigle, L. and A. Newton, "Domain-Based Application Service Location Using SRV RRs and the Dynamic Delegation Discovery Service (DDDS)", RFC 3958, January 2005.
- [RFC5176] Chiba, M., Dommety, G., Eklund, M., Mitton, D., and B. Aboba, "Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS)", RFC 5176, January 2008.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.
- [RFC5580] Tschofenig, H., Adrangi, F., Jones, M., Lior, A., and B. Aboba, "Carrying Location Objects in RADIUS and Diameter", RFC 5580, August 2009.
- [RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, August 2010.
- [RFC6614] Winter, S., McCauley, M., Venaas, S., and K. Wierenga, "Transport Layer Security (TLS) Encryption for RADIUS", RFC 6614, May 2012.

[RFC7360] DeKok, A., "Datagram Transport Layer Security (DTLS) as a Transport Layer for RADIUS", RFC 7360, September 2014.

[I-D.ietf-radext-nai]

DeKok, A., "The Network Access Identifier", draft-ietf-radext-nai-15 (work in progress), December 2014.

8.2. Informative References

[RFC4017] Stanley, D., Walker, J., and B. Aboba, "Extensible Authentication Protocol (EAP) Method Requirements for Wireless LANs", RFC 4017, March 2005.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, August 2011.

[RFC6733] Fajardo, V., Arkko, J., Loughney, J., and G. Zorn, "Diameter Base Protocol", RFC 6733, October 2012.

[RFC7299] Housley, R., "Object Identifier Registry for the PKIX Working Group", RFC 7299, July 2014.

[I-D.wierenga-ietf-eduroam]

Wierenga, K., Winter, S., and T. Wolniewicz, "The eduroam architecture for network roaming", draft-wierenga-ietf-eduroam-05 (work in progress), March 2015.

Appendix A. Appendix A: ASN.1 Syntax of NAIRealm

```
PKIXNaiRealm08 {iso(1) identified-organization(3) dod(6)
  internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
  id-mod-nai-realm-08 (TBD99) }

DEFINITIONS EXPLICIT TAGS ::=

BEGIN

-- EXPORTS ALL --

IMPORTS

    id-pkix
    FROM PKIX1Explicit-2009
        {iso(1) identified-organization(3) dod(6) internet(1)
        security(5) mechanisms(5) pkix(7) id-mod(0)
        id-mod-pkix1-explicit-02(51)}
        -- from RFC 5280, RFC 5912

    OTHER-NAME
    FROM PKIX1Implicit-2009
        {iso(1) identified-organization(3) dod(6) internet(1) security(5)
        mechanisms(5) pkix(7) id-mod(0) id-mod-pkix1-implicit-02(59)}
        -- from RFC 5280, RFC 5912
;

-- Service Name Object Identifier

id-on    OBJECT IDENTIFIER ::= { id-pkix 8 }

id-on-naiRealm OBJECT IDENTIFIER ::= { id-on TBD98 }

-- Service Name

naiRealm OTHER-NAME ::= { NAIRealm IDENTIFIED BY { id-on-naiRealm }}

ub-naiRealm-length INTEGER ::= 255

NAIRealm ::= UTF8String (SIZE (1..ub-naiRealm-length))

END
```

Authors' Addresses

Stefan Winter
Fondation RESTENA
6, rue Richard Coudenhove-Kalergi
Luxembourg 1359
LUXEMBOURG

Phone: +352 424409 1
Fax: +352 422473
EMail: stefan.winter@restena.lu
URI: <http://www.restena.lu>.

Mike McCauley
AirSpayce Pty Ltd
9 Bulbul Place
Currumbin Waters QLD 4223
AUSTRALIA

Phone: +61 7 5598 7474
EMail: mikem@airspayce.com
URI: <http://www.airspayce.com>

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 16, 2013

W. Dec, Ed.
Cisco Systems, Inc.
B. Sarikaya
Huawei USA
G. Zorn
Network Zen
D. Miles
Google
B. Lourdelet
Juniper Networks
February 12, 2013

RADIUS attributes for IPv6 Access Networks
draft-ietf-radext-ipv6-access-16

Abstract

This document specifies additional IPv6 RADIUS attributes useful in residential broadband network deployments. The attributes, which are used for authorization and accounting, enable assignment of a host IPv6 address and IPv6 DNS server address via DHCPv6; assignment of an IPv6 route announced via router advertisement; assignment of a named IPv6 delegated prefix pool; and assignment of a named IPv6 pool for host DHCPv6 addressing.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 16, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Deployment Scenarios	3
2.1. IPv6 Address Assignment	4
2.2. DNS Servers	4
2.3. IPv6 Route Information	5
2.4. Delegated IPv6 Prefix Pool	5
2.5. Stateful IPv6 address pool	5
3. Attributes	6
3.1. Framed-IPv6-Address	6
3.2. DNS-Server-IPv6-Address	7
3.3. Route-IPv6-Information	8
3.4. Delegated-IPv6-Prefix-Pool	9
3.5. Stateful-IPv6-Address-Pool	10
3.6. Table of attributes	10
4. Diameter Considerations	11
5. Security Considerations	11
6. IANA Considerations	11
7. Acknowledgements	12
8. References	12
8.1. Normative References	12
8.2. Informative References	12
Authors' Addresses	13

1. Introduction

This document specifies additional RADIUS attributes used to support configuration of DHCPv6 and/or ICMPv6 Router Advertisement (RA) parameters on a per-user basis. The attributes, which complement those defined in [RFC3162] and [RFC4818], support the following:

- o Assignment of specific IPv6 addresses to hosts via DHCPv6.
- o Assignment of an IPv6 DNS server address, via DHCPv6 or Router Advertisement [RFC6106].
- o Configuration of more specific routes to be announced to the user via the Route Information Option defined in [RFC4191] Section 2.3.
- o The assignment of a named delegated prefix pool for use with "IPv6 Prefix Options for DHCPv6" [RFC3633].
- o The assignment of a named stateful address pool for use with DHCPv6 stateful address assignment [RFC3315].

2. Deployment Scenarios

The extensions in this draft are intended to be applicable across a wide variety of network access scenarios where RADIUS is involved. One such typical network scenario is illustrated in Figure 1. It is composed of a IP Routing Residential Gateway (RG) or host, a Layer 2 Access-Node (AN) e.g. a Digital Subscriber Line Access Multiplexer - DSLAM, an IP Network Access Servers (NASes), and an Authentication Authorization & Accounting (AAA) server.

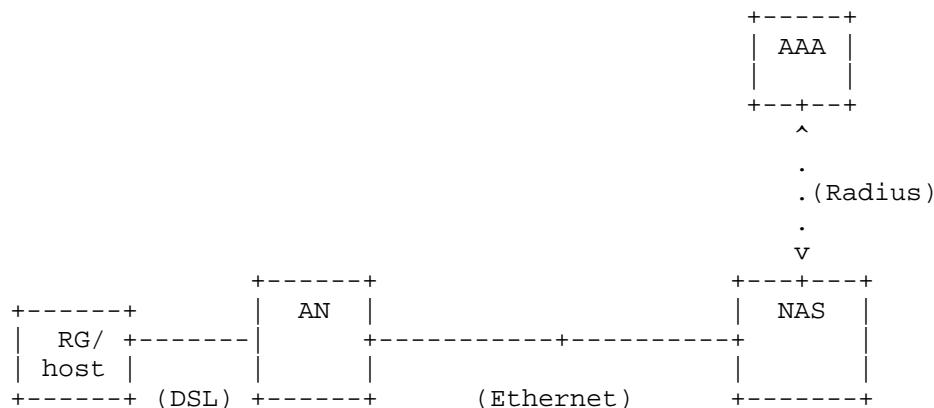


Figure 1

In the depicted scenario the NAS may utilize an IP address configuration protocol (e.g. a DHCPv6 server) to handle address assignment to RGs/hosts. The RADIUS server authenticates each RG/host and returns to the attributes used for authorization and accounting. These attributes can include a host's IPv6 address, a DNS server address and a set of IPv6 routes to be advertised via any suitable protocol, eg ICMPv6 (Neighbour Discovery). The name of a prefix pool to be used for DHCPv6 Prefix Delegation, or the name of an address pool to be used for DHCPv6 address assignment can also be attributes provided to the NAS by the RADIUS AAA server.

The following sub-sections discuss how these attributes are used in more detail.

2.1. IPv6 Address Assignment

DHCPv6 [RFC3315] provides a mechanism to assign one or more non-temporary IPv6 addresses to hosts. To provide a DHCPv6 server residing on a NAS with one or more IPv6 addresses to be assigned, this document specifies the Framed-IPv6-Address Attribute.

While [RFC3162] permits an IPv6 address to be specified via the combination of the Framed-Interface-Id and Framed-IPv6-Prefix attributes, this separation is more natural for use with PPP's IPv6 Control Protocol than it is for use with DHCPv6, and the use of a single IPv6 address attribute makes for easier processing of accounting records.

Since DHCPv6 can be deployed on the same network as ICMPv6 stateless (SLAAC) [RFC4862], it is possible that the NAS will require both stateful and stateless configuration information. Therefore it is possible for the Framed-IPv6-Address, Framed-IPv6-Prefix and Framed-Interface-Id attributes [RFC3162] to be included within the same packet. To avoid ambiguity in this case, the Framed-IPv6-Address attribute is intended for authorization and accounting of DHCPv6-assigned addresses and the Framed-IPv6-Prefix and Framed-Interface-Id attributes used for authorization and accounting of addresses assigned via SLAAC.

2.2. DNS Servers

DHCPv6 provides an option for configuring a host with the IPv6 address of a DNS server. The IPv6 address of a DNS server can also be conveyed to the host using ICMPv6 with Router Advertisements, via the [RFC6106] option. To provide the NAS with the IPv6 address of a DNS server, this document specifies the DNS-Server-IPv6-Address Attribute.

2.3. IPv6 Route Information

An IPv6 Route Information option, defined in [RFC4191] is intended to be used to inform a host connected to the NAS that a specific route is reachable via any given NAS.

This document specifies the RADIUS attribute that allows the AAA server to provision the announcement by the NAS of a specific Route Information Option to an accessing host. The NAS may advertise this route using the method defined in [RFC4191], or using other equivalent methods. Any other information, such as preference or life-time values, that is to be present in the actual announcement using a given method is assumed to be determined by the NAS using means not scoped by this document (e.g. Local configuration on the NAS).

While the Framed-IPv6-Prefix attribute defined in [RFC3162] Section 2.3 causes the route to be advertised in an RA, it cannot be used to configure more specific routes. While the Framed-IPv6-Route attribute defined in [RFC3162] Section 2.5 causes the route to be configured on the NAS, and potentially announced via an IP routing protocol, depending on the value of Framed-Routing, it does not result in the route being announced in an RA.

2.4. Delegated IPv6 Prefix Pool

DHCPv6 Prefix Delegation (DHCPv6-PD) [RFC3633] involves a delegating router selecting a prefix and delegating it on a temporary basis to a requesting router. The delegating router may implement a number of strategies as to how it chooses what prefix is to be delegated to a requesting router, one of them being the use of a local named prefix pool. The Delegated-IPv6-Prefix-Pool attribute allows the RADIUS server to convey a prefix pool name to a NAS hosting a DHCPv6-PD server and acting as a delegating router.

Since DHCPv6 Prefix Delegation can be used with SLAAC on the same network, it is possible for the Delegated-IPv6-Prefix-Pool and Framed-IPv6-Pool attributes to be included within the same packet. To avoid ambiguity in this scenario, use of the Delegated-IPv6-Prefix-Pool attribute should be restricted to authorization and accounting of prefix pools used in DHCPv6 Prefix Delegation and the Framed-IPv6-Pool attribute should be used for authorization and accounting of prefix pools used in SLAAC.

2.5. Stateful IPv6 address pool

DHCPv6 [RFC3315] provides a mechanism to assign one or more non-temporary IPv6 addresses to hosts. Section 3.1 introduces the

Framed-IPv6-Address attribute to be used for providing a DHCPv6 server residing on a NAS with one or more IPv6 addresses to be assigned to the clients. An alternative way to achieve a similar result is for the NAS to select the IPv6 address to be assigned from an address pool configured for this purpose on the NAS. This document specifies the Stateful-IPv6-Address-Pool attribute to allow the RADIUS server to convey a pool name to be used for such stateful DHCPv6 based addressing, and any subsequent accounting.

3. Attributes

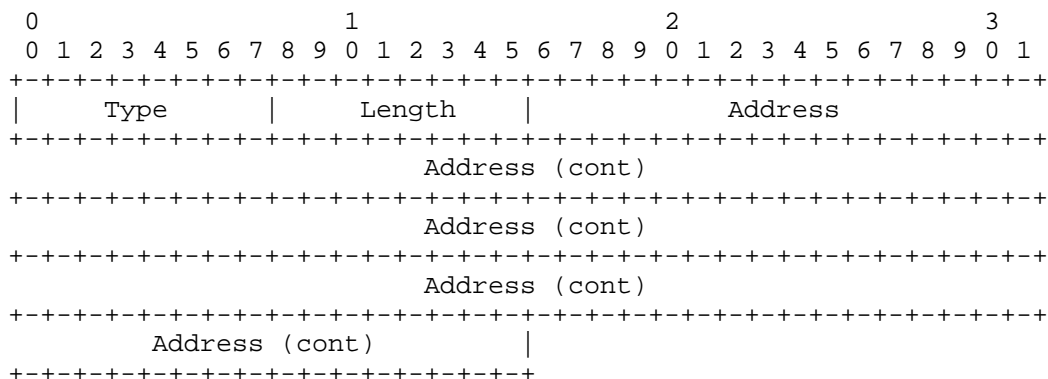
The fields shown in the diagrams below are transmitted from left to right.

3.1. Framed-IPv6-Address

This attribute indicates an IPv6 address that is assigned to the NAS-facing interface of the RG/host. It MAY be used in Access-Accept packets, and MAY appear multiple times. It MAY be used in an Access-Request packet as a hint by the NAS to the RADIUS server that it would prefer these IPv6 address(es), but the RADIUS server is not required to honor the hint. Since it is assumed that the NAS will add a route corresponding to the address, it is not necessary for the RADIUS server to also send a host Framed-IPv6-Route attribute for the same address.

This attribute can be used by a DHCPv6 process on the NAS to assign a unique IPv6 address to the RG/host.

A summary of the Framed-IPv6-Address attribute format is shown below. The format of the address is as per [RFC3162].



Type

TBA2 for DNS-Server-IPv6-Address

Length

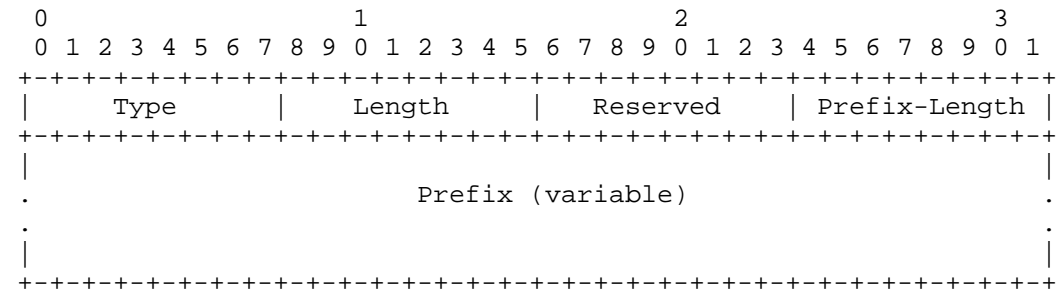
18

Address

The 128-bit IPv6 address of a DNS server.

3.3. Route-IPv6-Information

This attribute specifies a prefix (and corresponding route) for the user on the NAS, which is to be announced using the Route Information Option defined in "Default Router Preferences and More Specific Routes" [RFC4191] Section 2.3. It is used in the Access-Accept packet and can appear multiple times. It MAY be used in an Access-Request packet as a hint by the NAS to the RADIUS server, but the RADIUS server is not required to honor the hint. The Route-IPv6-Information attribute format is depicted below. The format of the prefix is as per [RFC3162].



Type

TBA3 for Route-IPv6-Information

Length

Length in bytes. At least 4 and no larger than 20; typically 12 or less.

Prefix Length

8-bit unsigned integer. The number of leading bits in the prefix that are valid. The value ranges from 0 to 128. The prefix field is 0, 8 or 16 octets depending on Length.

Prefix

Variable-length field containing an IP prefix. The prefix length field contains the number of valid leading bits in the prefix. The bits in the prefix after the prefix length (if any) are reserved and MUST be initialized to zero.

3.4. Delegated-IPv6-Prefix-Pool

This attribute contains the name of an assigned pool that SHOULD be used to select an IPv6 delegated prefix for the user on the NAS. If a NAS does not support prefix pools, the NAS MUST ignore this attribute. It MAY be used in an Access-Request packet as a hint by the NAS to the RADIUS server regarding the pool, but the RADIUS server is not required to honor the hint.

A summary of the Delegated-IPv6-Prefix-Pool attribute format is shown below.

0										1										2																													
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3																										
Type																														Length										String...									

Type

TBA4 for Delegated-IPv6-Prefix-Pool

Length

Length in bytes. At least 3.

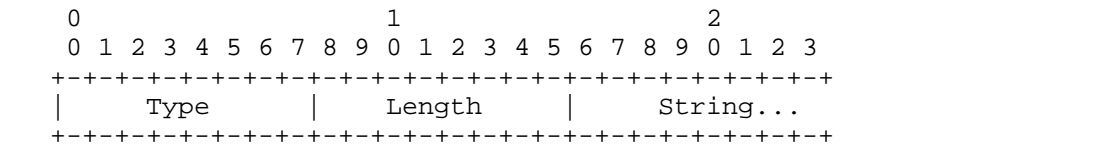
String

The string field contains the name of an assigned IPv6 prefix pool configured on the NAS. The field is not NULL (hexadecimal 00) terminated.

Note: The string data type is as documented in [RFC6158], and carries binary data that is external to the Radius protocol, eg the name of a pool of prefixes configured on the NAS.

3.5. Stateful-IPv6-Address-Pool

This attribute contains the name of an assigned pool that SHOULD be used to select an IPv6 address for the user on the NAS. If a NAS does not support address pools, the NAS MUST ignore this attribute. A summary of the Stateful-IPv6-Address-Pool attribute format is shown below. It MAY be used in an Access-Request packet as a hint by the NAS to the RADIUS server regarding the pool, but the RADIUS server is not required to honor the hint.



Type

TBA5 for Stateful-IPv6-Address-Pool

Length

Length in bytes. At least 3.

String

The string field contains the name of an assigned IPv6 stateful address pool configured on the NAS. The field is not NULL (hexadecimal 00) terminated.

Note: The string data type is as documented in [RFC6158], and carries binary data that is external to the Radius protocol, eg the name of a pool of addresses configured on the NAS.

3.6. Table of attributes

The following table provides a guide to which attributes may be found in which kinds of packets, and in what quantity. The optional inclusion of the options in Access Request messages is intended to allow for a network access server (NAS) to provide the RADIUS server with a hint of the attributes in advance of user authentication, which may be useful in cases where a user re-connects or has a static address. The server is under no obligation to honor such hints.

Request	Accept	Reject	Challenge	Accounting	#	Attribute
				Request		
0+	0+	0	0	0+	TBA1	Framed-IPv6-Address
0+	0+	0	0	0+	TBA2	DNS-Server-IPv6-Address
0+	0+	0	0	0+	TBA3	Route-IPv6-Information
0+	0+	0	0	0+	TBA4	Delegated-IPv6-Prefix-Pool
0+	0+	0	0	0+	TBA5	Stateful-IPv6-Address-Pool

4. Diameter Considerations

Given that the attributes defined in this document are allocated from the standard RADIUS type space (see Section 6), no special handling is required by Diameter entities.

5. Security Considerations

This document specifies additional IPv6 RADIUS attributes useful in residential broadband network deployments. In such networks, the RADIUS protocol may run either over IPv4 or over IPv6 and known security vulnerabilities of the RADIUS protocol, e.g. [SECI], apply to the attributes defined in this document. A trust relationship between a NAS and RADIUS server is expected to be in place, with communication optionally secured by IPsec or TLS [RFC6614] .

6. IANA Considerations

This document requires the assignment of five new RADIUS attribute types in the "Radius Types" registry (currently located at <http://www.iana.org/assignments/radius-types> for the following attributes:

- o Framed-IPv6-Address
- o DNS-Server-IPv6-Address
- o Route-IPv6-Information
- o Delegated-IPv6-Prefix-Pool
- o Stateful-IPv6-Address-Pool

7. Acknowledgements

The authors would like to thank Bernard Aboba, Benoit Claise, Peter Deacon, Alan DeKok, Alfred Hines, Jouni Korhonen, Roberta Maglione, Leaf Yeh, Mark Smith, Pete Resnik, Ralph Droms, Stephen Farrell, Brian Haberman, for their help and comments in reviewing this document.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

8.2. Informative References

- [RFC3162] Aboba, B., Zorn, G., and D. Mitton, "RADIUS and IPv6", RFC 3162, August 2001.
- [RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.
- [RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, December 2003.
- [RFC3646] Droms, R., "DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3646, December 2003.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.
- [RFC4818] Salowey, J. and R. Droms, "RADIUS Delegated-IPv6-Prefix Attribute", RFC 4818, April 2007.
- [RFC6106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 6106, November 2010.
- [RFC6158] DeKok, A. and G. Weber, "RADIUS Design Guidelines", BCP 158, RFC 6158, March 2011.

- [RFC6614] Winter, S., McCauley, M., Venaas, S., and K. Wierenga,
"Transport Layer Security (TLS) Encryption for RADIUS",
RFC 6614, May 2012.
- [SECI] -,
"http://regul.uni-mb.si/~meolic/ptk-seminarske/
radius.pdf", November 2001.

Authors' Addresses

Wojciech Dec (editor)
Cisco Systems, Inc.
Haarlerbergweg 13-19
Amsterdam , NOORD-HOLLAND 1101 CH
Netherlands

Email: wdec@cisco.com

Behcet Sarikaya
Huawei USA
1700 Alma Dr. Suite 500
Plano, TX
US

Phone: +1 972-509-5599
Email: sarikaya@ieee.org

Glen Zorn
Network Zen
1310 East Thomas Street
Seattle, WA
US

Email: gwz@net-zen.net

David Miles
Google

Phone:
Fax:
Email: david.miles@google.com
URI:

Benoit Lourdelet
Juniper Networks
France

Email: blourdel@juniper.net

RADIUS Extensions Working Group
Internet-Draft
Intended status: Experimental
Expires: August 17, 2012

S. Winter
RESTENA
M. McCauley
OSC
S. Venaas
K. Wierenga
Cisco
February 14, 2012

Transport Layer Security (TLS) encryption for RADIUS
draft-ietf-radext-radsec-12

Abstract

This document specifies a transport profile for RADIUS using Transport Layer Security (TLS) over TCP as the transport protocol. This enables dynamic trust relationships between RADIUS servers.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 17, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	3
1.1. Requirements Language	3
1.2. Terminology	4
1.3. Document Status	4
2. Normative: Transport Layer Security for RADIUS/TCP	5
2.1. TCP port and packet types	5
2.2. TLS negotiation	5
2.3. Connection Setup	5
2.4. Connecting Client Identity	7
2.5. RADIUS Datagrams	8
3. Informative: Design Decisions	10
3.1. Implications of Dynamic Peer Discovery	10
3.2. X.509 Certificate Considerations	10
3.3. Ciphersuites and Compression Negotiation Considerations	11
3.4. RADIUS Datagram Considerations	11
4. Compatibility with other RADIUS transports	12
5. Diameter Compatibility	13
6. Security Considerations	13
7. IANA Considerations	14
8. Notes to the RFC Editor	15
9. Acknowledgements	15
10. References	15
10.1. Normative References	15
10.2. Informative References	16
Appendix A. Implementation Overview: Radiator	18
Appendix B. Implementation Overview: radsecproxy	19
Appendix C. Assessment of Crypto-Agility Requirements	20

1. Introduction

The RADIUS protocol [RFC2865] is a widely deployed authentication and authorisation protocol. The supplementary RADIUS Accounting specification [RFC2866] also provides accounting mechanisms, thus delivering a full Authentication, Authorization, and Accounting (AAA) solution. However, RADIUS is experiencing several shortcomings, such as its dependency on the unreliable transport protocol UDP and the lack of security for large parts of its packet payload. RADIUS security is based on the MD5 algorithm, which has been proven to be insecure.

The main focus of RADIUS over TLS is to provide a means to secure the communication between RADIUS/TCP peers using TLS. The most important use of this specification lies in roaming environments where RADIUS packets need to be transferred through different administrative domains and untrusted, potentially hostile networks. An example for a world-wide roaming environment that uses RADIUS over TLS to secure communication is "eduroam", see [eduroam].

There are multiple known attacks on the MD5 algorithm which is used in RADIUS to provide integrity protection and a limited confidentiality protection (see [MD5-attacks]). RADIUS over TLS wraps the entire RADIUS packet payload into a TLS stream and thus mitigates the risk of attacks on MD5.

Because of the static trust establishment between RADIUS peers (IP address and shared secret) the only scalable way of creating a massive deployment of RADIUS-servers under control by different administrative entities is to introduce some form of a proxy chain to route the access requests to their home server. This creates a lot of overhead in terms of possible points of failure, longer transmission times as well as middleboxes through which authentication traffic flows. These middleboxes may learn privacy-relevant data while forwarding requests. The new features in RADIUS over TLS obsolete the use of IP addresses and shared MD5 secrets to identify other peers and thus allow the use of more contemporary trust models, e.g. checking a certificate by inspecting the issuer and other certificate properties.

1.1. Requirements Language

In this document, several words are used to signify the requirements of the specification. The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119. [RFC2119]

1.2. Terminology

RADIUS/TLS node: a RADIUS over TLS client or server

RADIUS/TLS Client: a RADIUS over TLS instance which initiates a new connection.

RADIUS/TLS Server: a RADIUS over TLS instance which listens on a RADIUS over TLS port and accepts new connections

RADIUS/UDP: classic RADIUS transport over UDP as defined in [RFC2865]

1.3. Document Status

This document is an Experimental RFC.

It is one out of several approaches to address known cryptographic weaknesses of the RADIUS protocol (see also Section 4). The specification does not fulfill all recommendations on a AAA transport profile as per [RFC3539]; in particular, by being based on TCP as a transport layer, it does not prevent head-of-line blocking issues.

If this specification is indeed selected for advancement to standards track, certificate verification options (section 2.3.2) need to be refined.

Another experimental characteristic of this specification is the question of key management between RADIUS/TLS peers. RADIUS/UDP only allowed for manual key management, i.e. distribution of a shared secret between a client and a server. RADIUS/TLS allows manual distribution of long-term proofs of peer identity as well (by using TLS-PSK cipher suites, or identifying clients by a certificate fingerprint), but as a new feature enables use of X.509 certificates in a PKIX infrastructure. It remains to be seen if one of these methods prevail, or if both will find their place in real-life deployments. The authors can imagine pre-shared keys to be popular in small-scale deployments (SOHO or isolated enterprise deployments) where scalability is not an issue and the deployment of a CA is considered too much a hassle; but can also imagine large roaming consortia to make use of PKIX. Readers of this specification are encouraged to read the discussion of key management issues within [RFC6421] as well as [RFC4107].

It has yet to be decided whether this approach is to be chosen for standards track. One key aspect to judge whether the approach is usable at large scale is by observing the uptake, usability and operational behaviour of the protocol in large-scale, real-life deployments.

An example for a world-wide roaming environment that uses RADIUS over TLS to secure communication is "eduroam", see [eduroam].

2. Normative: Transport Layer Security for RADIUS/TCP

2.1. TCP port and packet types

The default destination port number for RADIUS over TLS is TCP/2083. There are no separate ports for authentication, accounting and dynamic authorisation changes. The source port is arbitrary. See section Section 3.4 for considerations regarding separation of authentication, accounting and dynamic authorization traffic.

2.2. TLS negotiation

RADIUS/TLS has no notion of negotiating TLS in an established connection. Servers and clients need to be preconfigured to use RADIUS/TLS for a given endpoint.

2.3. Connection Setup

RADIUS/TLS nodes

1. establish TCP connections as per [I-D.ietf-radext-tcp-transport]. Failure to connect leads to continuous retries, with exponentially growing intervals between every try. If multiple servers are defined, the node MAY attempt to establish a connection to these other servers in parallel, in order to implement quick failover.
2. after completing the TCP handshake, immediately negotiate TLS sessions according to [RFC5246] or its predecessor TLS 1.1. The following restrictions apply:
 - * Support for TLS v1.1 [RFC4346] or later (e.g. TLS 1.2 [RFC5246]) is REQUIRED. To prevent known attacks on TLS versions prior to 1.1, implementations MUST NOT negotiate TLS versions prior to 1.1.
 - * Support for certificate-based mutual authentication is REQUIRED.
 - * Negotiation of mutual authentication is REQUIRED.
 - * Negotiation of a ciphersuite providing for confidentiality as well as integrity protection is REQUIRED. Failure to comply with this requirement can lead to severe security problems, like user passwords being recoverable by third parties. See

Section 6 for details.

- * Support for and negotiation of compression is OPTIONAL.
 - * Support for TLS-PSK mutual authentication [RFC4279] is OPTIONAL.
 - * RADIUS/TLS implementations MUST at a minimum support negotiation of the TLS_RSA_WITH_3DES_EDE_CBC_SHA), and SHOULD support TLS_RSA_WITH_RC4_128_SHA and TLS_RSA_WITH_AES_128_CBC_SHA as well (see Section 3.3).
 - * In addition, RADIUS/TLS implementations MUST support negotiation of the mandatory-to-implement ciphersuites required by the versions of TLS that they support.
3. Peer authentication can be performed in any of the following three operation models:
- * TLS with X.509 certificates using PKIX trust models (this model is mandatory to implement):
 - + Implementations MUST allow to configure a list of trusted Certification Authorities for incoming connections.
 - + Certificate validation MUST include the verification rules as per [RFC5280].
 - + Implementations SHOULD indicate their trusted Certification Authorities (CAs). For TLS 1.2, this is done using [RFC5246] section 7.4.4 "certificate authorities" (server side) and [RFC6066] Section 6 "Trusted CA Indication" (client side). See also Section 3.2.
 - + Peer validation always includes a check on whether the locally configured expected DNS name or IP address of the server that is contacted matches its presented certificate. DNS names and IP addresses can be contained in the Common Name (CN) or subjectAltName entries. For verification, only one of these entries is to be considered. The following precedence applies: for DNS name validation, subjectAltName:DNS has precedence over CN; for IP address validation, subjectAltName:iPAddr has precedence over CN. Implementors of this specification are advised to read [RFC6125] Section 6 for more details on DNS name validation.

- + Implementations MAY allow to configure a set of additional properties of the certificate to check for a peer's authorisation to communicate (e.g. a set of allowed values in subjectAltName:URI or a set of allowed X509v3 Certificate Policies)
 - + When the configured trust base changes (e.g. removal of a CA from the list of trusted CAs; issuance of a new CRL for a given CA) implementations MAY re-negotiate the TLS session to re-assess the connecting peer's continued authorisation.
 - * TLS with X.509 certificates using certificate fingerprints (this model is optional to implement): Implementations SHOULD allow to configure a list of trusted certificates, identified via fingerprint of the DER encoded certificate octets. Implementations MUST support SHA-1 as the hash algorithm for the fingerprint. To prevent attacks based on hash collisions, support for a more contemporary hash function such as SHA-256 is RECOMMENDED.
 - * TLS using TLS-PSK (this model is optional to implement)
4. start exchanging RADIUS datagrams (note Section 3.4 (1)). The shared secret to compute the (obsolete) MD5 integrity checks and attribute encryption MUST be "radsec" (see Section 3.4 (2)).

2.4. Connecting Client Identity

In RADIUS/UDP, clients are uniquely identified by their IP address. Since the shared secret is associated with the origin IP address, if more than one RADIUS client is associated with the same IP address, then those clients also must utilize the same shared secret, a practice which is inherently insecure as noted in [RFC5247].

RADIUS/TLS supports multiple operation modes.

In TLS-PSK operation, a client is uniquely identified by its TLS identifier.

In TLS-X.509 mode using fingerprints, a client is uniquely identified by the fingerprint of the presented client certificate.

In TLS-X.509 mode using PKIX trust models, a client is uniquely identified by the tuple (serial number of presented client certificate;Issuer).

Note well: having identified a connecting entity does not mean the

server necessarily wants to communicate with that client. E.g. if the Issuer is not in a trusted set of Issuers, the server may decline to perform RADIUS transactions with this client.

There are numerous trust models in PKIX environments, and it is beyond the scope of this document to define how a particular deployment determines whether a client is trustworthy. Implementations which want to support a wide variety of trust models should expose as many details of the presented certificate to the administrator as possible so that the trust model can be implemented by the administrator. As a suggestion, at least the following parameters of the X.509 client certificate should be exposed:

- o Originating IP address
- o Certificate Fingerprint
- o Issuer
- o Subject
- o all X509v3 Extended Key Usage
- o all X509v3 Subject Alternative Name
- o all X509v3 Certificate Policies

In TLS-PSK operation, at least the following parameters of the TLS connection should be exposed:

- o Originating IP address
- o TLS Identifier

2.5. RADIUS Datagrams

Authentication, Accounting and Authorization packets are sent according to the following rules:

RADIUS/TLS clients transmit the same packet types on the connection they initiated as a RADIUS/UDP client would (see Section 3.4 (3) and (4)). E.g. they send

- o Access-Request
- o Accounting-Request

- o Status-Server
- o Disconnect-ACK
- o Disconnect-NAK
- o ...

and they receive

- o Access-Accept
- o Accounting-Response
- o Disconnect-Request
- o ...

RADIUS/TLS servers transmit the same packet types on connections they have accepted as a RADIUS/UDP server would. E.g. they send

- o Access-Challenge
- o Access-Accept
- o Access-Reject
- o Accounting-Response
- o Disconnect-Request
- o ...

and they receive

- o Access-Request
- o Accounting-Request
- o Status-Server
- o Disconnect-ACK
- o ...

Due to the use of one single TCP port for all packet types, it is required for a RADIUS/TLS server to signal to a connecting peer which types of packets are supported on a server. See also section

Section 3.4 for a discussion of signaling.

- o When receiving an unwanted packet of type 'CoA-Request' or 'Disconnect-Request', it needs to be replied to with a 'CoA-NAK' or 'Disconnect-NAK' respectively. The NAK SHOULD contain an attribute Error-Cause with the value 406 ("Unsupported Extension"); see [RFC5176] for details.
- o When receiving an unwanted packet of type 'Accounting-Request', the RADIUS/TLS server SHOULD reply with an Accounting-Response containing an Error-Cause attribute with value 406 "Unsupported Extension" as defined in [RFC5176]. A RADIUS/TLS accounting client receiving such an Accounting-Response SHOULD log the error and stop sending Accounting-Request packets.

3. Informative: Design Decisions

This section explains the design decisions that led to the rules defined in the previous section.

3.1. Implications of Dynamic Peer Discovery

One mechanism to discover RADIUS over TLS peers dynamically via DNS is specified in [I-D.ietf-radext-dynamic-discovery]. While this mechanism is still under development and therefore is not a normative dependency of RADIUS/TLS, the use of dynamic discovery has potential future implications that are important to understand.

Readers of this document who are considering the deployment of DNS-based dynamic discovery are thus encouraged to read [I-D.ietf-radext-dynamic-discovery] and follow its future development.

3.2. X.509 Certificate Considerations

(1) If a RADIUS/TLS client is in possession of multiple certificates from different CAs (i.e. is part of multiple roaming consortia) and dynamic discovery is used, the discovery mechanism possibly does not yield sufficient information to identify the consortium uniquely (e.g. DNS discovery). Subsequently, the client may not know by itself which client certificate to use for the TLS handshake. Then it is necessary for the server to signal which consortium it belongs to, and which certificates it expects. If there is no risk of confusing multiple roaming consortia, providing this information in the handshake is not crucial.

(2) If a RADIUS/TLS server is in possession of multiple certificates from different CAs (i.e. is part of multiple roaming consortia), it

will need to select one of its certificates to present to the RADIUS/TLS client. If the client sends the Trusted CA Indication, this hint can make the server select the appropriate certificate and prevent a handshake failure. Omitting this indication makes it impossible to deterministically select the right certificate in this case. If there is no risk of confusing multiple roaming consortia, providing this indication in the handshake is not crucial.

3.3. Ciphersuites and Compression Negotiation Considerations

Not all TLS ciphersuites in [RFC5246] are supported by available TLS tool kits, and licenses may be required in some cases. The existing implementations of RADIUS/TLS use OpenSSL as cryptographic backend, which supports all of the ciphersuites listed in the rules in the normative section.

The TLS ciphersuite TLS_RSA_WITH_3DES_EDE_CBC_SHA is mandatory-to-implement according to [RFC4346] and thus has to be supported by RADIUS/TLS nodes.

The two other ciphersuites in the normative section are widely implemented in TLS toolkits and are considered good practice to implement.

3.4. RADIUS Datagram Considerations

(1) After the TLS session is established, RADIUS packet payloads are exchanged over the encrypted TLS tunnel. In RADIUS/UDP, the packet size can be determined by evaluating the size of the datagram that arrived. Due to the stream nature of TCP and TLS, this does not hold true for RADIUS/TLS packet exchange. Instead, packet boundaries of RADIUS packets that arrive in the stream are calculated by evaluating the packet's Length field. Special care needs to be taken on the packet sender side that the value of the Length field is indeed correct before sending it over the TLS tunnel, because incorrect packet lengths can no longer be detected by a differing datagram boundary. See section 2.6.4 of [I-D.ietf-radext-tcp-transport] for more details.

(2) Within RADIUS/UDP [RFC2865], a shared secret is used for hiding of attributes such as User-Password, as well as in computation of the Response Authenticator. In RADIUS accounting [RFC2866], the shared secret is used in computation of both the Request Authenticator and the Response Authenticator. Since TLS provides integrity protection and encryption sufficient to substitute for RADIUS application-layer security, it is not necessary to configure a RADIUS shared secret. The use of a fixed string for the obsolete shared secret eliminates possible node misconfigurations.

(3) RADIUS/UDP [RFC2865] uses different UDP ports for authentication, accounting and dynamic authorisation changes. RADIUS/TLS allocates a single port for all RADIUS packet types. Nevertheless, in RADIUS/TLS the notion of a client which sends authentication requests and processes replies associated with it's users' sessions and the notion of a server which receives requests, processes them and sends the appropriate replies is to be preserved. The normative rules about acceptable packet types for clients and servers mirror the packet flow behaviour from RADIUS/UDP.

(4) RADIUS/UDP [RFC2865] uses negative ICMP responses to a newly allocated UDP port to signal that a peer RADIUS server does not support reception and processing of the packet types in [RFC5176]. These packet types are listed as to be received in RADIUS/TLS implementations. Note well: it is not required for an implementation to actually process these packet types; it is only required to send the NAK as defined above.

(5) RADIUS/UDP [RFC2865] uses negative ICMP responses to a newly allocated UDP port to signal that a peer RADIUS server does not support reception and processing of RADIUS Accounting packets. There is no RADIUS datagram to signal an Accounting NAK. Clients may be misconfigured to send Accounting packets to a RADIUS/TLS server which does not wish to process their Accounting packet. To prevent a regression of detectability of this situation, the Accounting-Response + Error-Cause signaling was introduced.

4. Compatibility with other RADIUS transports

Ongoing work in the IETF defines multiple alternative transports to the classic UDP transport model as defined in [RFC2865], namely RADIUS over TCP [I-D.ietf-radext-tcp-transport], RADIUS over Datagram Transport Layer Security (DTLS) [I-D.ietf-radext-dtls] and this present document on RADIUS over TLS.

RADIUS/TLS does not specify any inherent backwards compatibility to RADIUS/UDP or cross compatibility to the other transports, i.e. an implementation which implements RADIUS/TLS only will not be able to receive or send RADIUS packet payloads over other transports. An implementation wishing to be backward or cross compatible (i.e. wishes to serve clients using other transports than RADIUS/TLS) will need to implement these other transports along with the RADIUS/TLS transport and be prepared to send and receive on all implemented transports, which is called a multi-stack implementation.

If a given IP device is able to receive RADIUS payloads on multiple transports, this may or may not be the same instance of software, and it may or may not serve the same purposes. It is not safe to assume

that both ports are interchangeable. In particular, it can not be assumed that state is maintained for the packet payloads between the transports. Two such instances MUST be considered separate RADIUS server entities.

5. Diameter Compatibility

Since RADIUS/TLS is only a new transport profile for RADIUS, compatibility of RADIUS/TLS - Diameter [RFC3588] vs. RADIUS/UDP [RFC2865] - Diameter [RFC3588] is identical. The considerations regarding payload size in [I-D.ietf-radext-tcp-transport] apply.

6. Security Considerations

The computational resources to establish a TLS tunnel are significantly higher than simply sending mostly unencrypted UDP datagrams. Therefore, clients connecting to a RADIUS/TLS node will more easily create high load conditions and a malicious client might create a Denial-of-Service attack more easily.

Some TLS ciphersuites only provide integrity validation of their payload, and provide no encryption. This specification forbids the use of such ciphersuites. Since the RADIUS payload's shared secret is fixed to the well-known term "radsec" (see Section 2.3 (4)), failure to comply with this requirement will expose the entire datagram payload in plain text, including User-Password, to intermediate IP nodes.

By virtue of being based on TCP, there are several generic attack vectors to slow down or prevent the TCP connection from being established; see [RFC4953] for details. If a TCP connection is not up when a packet is to be processed, it gets re-established, so such attacks in general lead only to a minor performance degradation (the time it takes to re-establish the connection). There is one notable exception where an attacker might create a bidding-down attack though: If peer communication between two devices is configured for both RADIUS/TLS (i.e. TLS security over TCP as a transport, shared secret fixed to "radsec") and RADIUS/UDP (i.e. shared secret security with a secret manually configured by the administrator), and where the RADIUS/UDP transport is the failover option if the TLS session cannot be established, a bidding-down attack can occur if an adversary can maliciously close the TCP connection, or prevent it from being established. Situations where clients are configured in such a way are likely to occur during a migration phase from RADIUS/UDP to RADIUS/TLS. By preventing the TLS session setup, the attacker can reduce the security of the packet payload from the selected TLS cipher suite packet encryption to the classic MD5 per-attribute encryption. The situation should be avoided by disabling the weaker

RADIUS/UDP transport as soon as the new RADIUS/TLS connection is established and tested. Disabling can happen at either the RADIUS client or server side:

- o Client side: de-configure the failover setup, leaving RADIUS/TLS as the only communication option
- o Server side: de-configure the RADIUS/UDP client from the list of valid RADIUS clients

RADIUS/TLS provides authentication and encryption between RADIUS peers. In the presence of proxies, the intermediate proxies can still inspect the individual RADIUS packets, i.e. "end-to-end" encryption is not provided. Where intermediate proxies are untrusted, it is desirable to use other RADIUS mechanisms to prevent RADIUS packet payload from inspection by such proxies. One common method to protect passwords is the use of the Extensible Authentication Protocol (EAP) and EAP methods which utilize TLS.

When using certificate fingerprints to identify RADIUS/TLS peers, any two certificates which produce the same hash value (i.e. which have a hash collision) will be considered the same client. It is therefore important to make sure that the hash function used is cryptographically uncompromised so that an attacker is very unlikely to be able to produce a hash collision with a certificate of his choice. While this specification mandates support for SHA-1, a later revision will likely demand support for more contemporary hash functions because as of issuance of this document there are already attacks on SHA-1.

7. IANA Considerations

No new RADIUS attributes or packet codes are defined. IANA is requested to update the already-assigned TCP port number 2083 in the following ways:

- o Reference: list the RFC number of this document as the reference
- o Assignment Notes: add the text "The TCP port 2083 was already previously assigned by IANA for "RadSec", an early implementation of RADIUS/TLS, prior to issuance of this RFC. This early implementation can be configured to be compatible to RADIUS/TLS as specified by the IETF. See RFC (RFC number of this document), Appendix A for details."

8. Notes to the RFC Editor

[I-D.ietf-radext-tcp-transport] is currently in the publication queue because it has a normative reference on this draft; it has no other blocking dependencies. The two drafts should be published as an RFC simultaneously, ideally with consecutive numbers. The references in this draft to [I-D.ietf-radext-tcp-transport] should be changed to references to the corresponding RFC prior to publication.

This section, "Notes to the RFC Editor" should be deleted from the draft prior to publication.

9. Acknowledgements

RADIUS/TLS was first implemented as "RADSec" by Open Systems Consultants, Currumbin Waters, Australia, for their "Radiator" RADIUS server product (see [radsec-whitepaper]).

Funding and input for the development of this Internet Draft was provided by the European Commission co-funded project "GEANT2" [geant2] and further feedback was provided by the TERENA Task Force Mobility [terena].

10. References

10.1. Normative References

- | | |
|-----------|--|
| [RFC2119] | Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997. |
| [RFC2865] | Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000. |
| [RFC2866] | Rigney, C., "RADIUS Accounting", RFC 2866, June 2000. |
| [RFC4279] | Eronen, P. and H. Tschofenig, "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, December 2005. |
| [RFC5280] | Cooper, D., Santesson, S., Farrell, S., Boeyen, S., |

Housley, R., and W. Polk,
"Internet X.509 Public Key
Infrastructure Certificate and
Certificate Revocation List
(CRL) Profile", RFC 5280,
May 2008.

[RFC5176]

Chiba, M., Dommety, G., Eklund,
M., Mitton, D., and B. Aboba,
"Dynamic Authorization
Extensions to Remote
Authentication Dial In User
Service (RADIUS)", RFC 5176,
January 2008.

[RFC5246]

Dierks, T. and E. Rescorla, "The
Transport Layer Security (TLS)
Protocol Version 1.2", RFC 5246,
August 2008.

[RFC5247]

Aboba, B., Simon, D., and P.
Eronen, "Extensible
Authentication Protocol (EAP)
Key Management Framework",
RFC 5247, August 2008.

[RFC6066]

Eastlake, D., "Transport Layer
Security (TLS) Extensions:
Extension Definitions",
RFC 6066, January 2011.

[I-D.ietf-radext-tcp-transport]

DeKok, A., "RADIUS Over TCP", dr
aft-ietf-radext-tcp-transport-09
(work in progress),
October 2010.

10.2. Informative References

[I-D.ietf-radext-dtls]

DeKok, A., "DTLS as a Transport
Layer for RADIUS",
draft-ietf-radext-dtls-01 (work
in progress), October 2010.

[I-D.ietf-radext-dynamic-discovery]

Winter, S. and M. McCauley,
"NAI-based Dynamic Peer
Discovery for RADIUS/TLS and
RADIUS/DTLS", draft-ietf-radext-
dynamic-discovery-03 (work in

progress), July 2011.

- [RFC3539] Aboba, B. and J. Wood,
"Authentication, Authorization
and Accounting (AAA) Transport
Profile", RFC 3539, June 2003.
- [RFC3588] Calhoun, P., Loughney, J.,
Guttman, E., Zorn, G., and J.
Arkko, "Diameter Base Protocol",
RFC 3588, September 2003.
- [RFC4107] Bellovin, S. and R. Housley,
"Guidelines for Cryptographic
Key Management", BCP 107,
RFC 4107, June 2005.
- [RFC4346] Dierks, T. and E. Rescorla, "The
Transport Layer Security (TLS)
Protocol Version 1.1", RFC 4346,
April 2006.
- [RFC4953] Touch, J., "Defending TCP
Against Spoofing Attacks",
RFC 4953, July 2007.
- [RFC6125] Saint-Andre, P. and J. Hodges,
"Representation and Verification
of Domain-Based Application
Service Identity within Internet
Public Key Infrastructure Using
X.509 (PKIX) Certificates in the
Context of Transport Layer
Security (TLS)", RFC 6125,
March 2011.
- [RFC6421] Nelson, D., "Crypto-Agility
Requirements for Remote
Authentication Dial-In User
Service (RADIUS)", RFC 6421,
November 2011.
- [radsec-whitepaper] Open System Consultants, "RadSec
- a secure, reliable RADIUS
Protocol", May 2005, <[http://
www.open.com.au/radiator/
radsec-whitepaper.pdf](http://www.open.com.au/radiator/radsec-whitepaper.pdf)>.

- [MD5-attacks] Black, J., Cochran, M., and T. Highland, "A Study of the MD5 Attacks: Insights and Improvements", October 2006, <<http://www.springerlink.com/content/40867185727r7084/>>.
- [radsecproxy-impl] Venaas, S., "radsecproxy Project Homepage", 2007, <<http://software.uninett.no/radsecproxy/>>.
- [eduroam] Trans-European Research and Education Networking Association, "eduroam Homepage", 2007, <<http://www.eduroam.org/>>.
- [geant2] Delivery of Advanced Network Technology to Europe, "European Commission Information Society and Media: GEANT2", 2008, <<http://www.geant2.net/>>.
- [terena] TERENA, "Trans-European Research and Education Networking Association", 2008, <<http://www.terena.org/>>.

Appendix A. Implementation Overview: Radiator

Radiator implements the RadSec protocol for proxying requests with the <Authby RADSEC> and <ServerRADSEC> clauses in the Radiator configuration file.

The <AuthBy RADSEC> clause defines a RadSec client, and causes Radiator to send RADIUS requests to the configured RadSec server using the RadSec protocol.

The <ServerRADSEC> clause defines a RadSec server, and causes Radiator to listen on the configured port and address(es) for connections from <Authby RADSEC> clients. When an <Authby RADSEC> client connects to a <ServerRADSEC> server, the client sends RADIUS requests through the stream to the server. The server then handles the request in the same way as if the request had been received from a conventional UDP RADIUS client.

Radiator is compliant to RADIUS/TLS if the following options are used:

<AuthBy RADSEC>

- * Protocol tcp
- * UseTLS
- * TLS_CertificateFile
- * Secret radsec

<ServerRADSEC>

- * Protocol tcp
- * UseTLS
- * TLS_RequireClientCert
- * Secret radsec

As of Radiator 3.15, the default shared secret for RadSec connections is configurable and defaults to "mysecret" (without quotes). For compliance with this document, this setting needs to be configured for the shared secret "radsec". The implementation uses TCP keepalive socket options, but does not send Status-Server packets. Once established, TLS connections are kept open throughout the server instance lifetime.

Appendix B. Implementation Overview: radsecproxy

The RADIUS proxy named radsecproxy was written in order to allow use of RadSec in current RADIUS deployments. This is a generic proxy that supports any number and combination of clients and servers, supporting RADIUS over UDP and RadSec. The main idea is that it can be used on the same host as a non-RadSec client or server to ensure RadSec is used on the wire, however as a generic proxy it can be used in other circumstances as well.

The configuration file consists of client and server clauses, where there is one such clause for each client or server. In such a clause one specifies either "type tls" or "type udp" for RadSec or UDP transport. For RadSec the default shared secret "mysecret" (without quotes), the same as Radiator, is used. For compliance with this document, this setting needs to be configured for the shared secret "radsec". A secret may be specified by putting say "secret somesharedsecret" inside a client or server clause.

In order to use TLS for clients and/or servers, one must also specify

where to locate CA certificates, as well as certificate and key for the client or server. This is done in a TLS clause. There may be one or several TLS clauses. A client or server clause may reference a particular TLS clause, or just use a default one. One use for multiple TLS clauses may be to present one certificate to clients and another to servers.

If any RadSec (TLS) clients are configured, the proxy will at startup listen on port 2083, as assigned by IANA for the OSC RadSec implementation. An alternative port may be specified. When a client connects, the client certificate will be verified, including checking that the configured FQDN or IP address matches what is in the certificate. Requests coming from a RadSec client are treated exactly like requests from UDP clients.

The proxy will at startup try to establish a TLS connection to each (if any) of the configured RadSec (TLS) servers. If it fails to connect to a server, it will retry regularly. There is some back-off where it will retry quickly at first, and with longer intervals later. If a connection to a server goes down it will also start retrying regularly. When setting up the TLS connection, the server certificate will be verified, including checking that the configured FQDN or IP address matches what is in the certificate. Requests are sent to a RadSec server just like they would to a UDP server.

The proxy supports Status-Server messages. They are only sent to a server if enabled for that particular server. Status-Server requests are always responded to.

This RadSec implementation has been successfully tested together with Radiator. It is a freely available open-source implementation. For source code and documentation, see [radsecproxy-impl].

Appendix C. Assessment of Crypto-Agility Requirements

The RADIUS Crypto-Agility Requirements [RFC6421] defines numerous classification criteria for protocols that strive to enhance the security of RADIUS. It contains mandatory (M) and recommended (R) criteria which crypto-agile protocols have to fulfill. The authors believe that the following assessment about the crypto-agility properties of RADIUS/TLS are true.

By virtue of being a transport profile using TLS over TCP as a transport protocol, the cryptographically agile properties of TLS are inherited, and RADIUS/TLS subsequently meets the following points:

(M) negotiation of cryptographic algorithms for integrity and auth

- (M) negotiation of cryptographic algorithms for encryption
- (M) replay protection
- (M) define mandatory-to-implement cryptographic algorithms
- (M) generate fresh session keys for use between client and server
- (R) support for Perfect Forward Secrecy in session keys
- (R) support X.509 certificate based operation
- (R) support Pre-Shared keys
- (R) support for confidentiality of the entire packet
- (M/R) support Automated Key Management

The remainder of the requirements is discussed individually below in more detail:

- (M) "avoid security compromise, even in situations where the existing cryptographic algorithms used by RADIUS implementations are shown to be weak enough to provide little or no security" - The existing algorithm, based on MD5, is not of any significance in RADIUS/TLS; its compromise does not compromise the outer transport security.
- (R) mandatory-to-implement algorithms are to be NIST-Acceptable with no deprecation date - The mandatory-to-implement algorithm is TLS_RSA_WITH_3DES_EDE_CBC_SHA. This ciphersuite supports three-key 3DES operation, which is classified as Acceptable with no known deprecation date by NIST.
- (M) demonstrate backward compatibility with RADIUS - There are multiple implementations supporting both RADIUS and RADIUS/TLS, and the translation between them.
- (M) After legacy mechanisms have been compromised, secure algorithms MUST be used, so that backward compatibility is no longer possible - In RADIUS, communication between client and server is always a manual configuration; after a compromise, the legacy client in question can be de-configured by the same manual configuration.
- (M) indicate a willingness to cede change control to the IETF - Change control of this protocol is with the IETF.

(M) be interoperable between implementations based purely on the information in the specification - At least one implementation was created exclusively based on this specification and is interoperable with other RADIUS/TLS implementations.

(M) apply to all packet types - RADIUS/TLS operates on the transport layer, and can carry all packet types.

(R) message data exchanged with Diameter SHOULD NOT be affected - The solution is Diameter-agnostic.

(M) discuss any inherent assumptions - The authors are not aware of any implicit assumptions which would be yet-unarticulated in the draft

(R) provide recommendations for transition - The Security Considerations section contains a transition path.

(R) discuss legacy interoperability and potential for bidding-down attacks - The Security Considerations section contains an corresponding discussion.

Summarizing, it is believed that this specification fulfills all the mandatory and all the recommended requirements for a crypto-agile solution and should thus be considered UNCONDITIONALLY COMPLIANT.

Authors' Addresses

Stefan Winter
Fondation RESTENA
6, rue Richard Coudenhove-Kalergi
Luxembourg 1359
LUXEMBOURG

Phone: +352 424409 1
Fax: +352 422473
EMail: stefan.winter@restena.lu
URI: <http://www.restena.lu>.

Mike McCauley
Open Systems Consultants
9 Bulbul Place
Currumbin Waters QLD 4223
AUSTRALIA

Phone: +61 7 5598 7474
Fax: +61 7 5598 7070
EMail: mikem@open.com.au
URI: <http://www.open.com.au>.

Stig Venaas
cisco Systems
Tasman Drive
San Jose, CA 95134
USA

EMail: stig@cisco.com

Klaas Wierenga
Cisco Systems International BV
Haarlerbergweg 13-19
Amsterdam 1101 CH
The Netherlands

Phone: +31 (0)20 3571752
Fax:
EMail: kwiereng@cisco.com
URI: <http://www.cisco.com>.

