Internet Engineering Task Force (IETF)
Request for Comments: 7585
Category: Experimental
ISSN: 2070-1721

S. Winter
RESTENA
M. McCauley
AirSpayce

October 2015

Dynamic Peer Discovery for RADIUS/TLS and RADIUS/DTLS Based on the Network Access Identifier (NAI)

Abstract

This document specifies a means to find authoritative RADIUS servers for a given realm. It is used in conjunction with either RADIUS over Transport Layer Security (RADIUS/TLS) or RADIUS over Datagram Transport Layer Security (RADIUS/DTLS).

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

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

Where more than one administrative entity collaborates for RADIUS authentication of their respective customers (a "roaming consortium"), the Network Access Identifier (NAI) [RFC7542] 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 and 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 synchronizing the changed data across all branches.

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

```
                                    +-------+
                                    |       |
                                    |   .   |
                                    |       |
                                    +---+---+
                                      / | \
                    +----------------/  |  \---------------------+
                    |                   |                        |
                    |                   |                        |
                    |                   |                        |
                 +--+---+            +--+--+                +----+---+
                 |      |            |     |                |        |
                 | .edu |    . . .   | .nl |      . . .     | .ac.uk |
                 |      |            |     |                |        |
                 +--+---+            +--+--+                +----+---+
                  / | \                 | \                      |
                 /  |  \                |  \                     |
                /   |   \               |   \                    |
         +-----+    |    +-----+        |    +------+            |
         |          |          |        |           |            |
         |          |          |        |           |            |
     +---+---+ +----+---+ +----+---+ +--+---+ +-----+----+ +-----+-----+
     |       | |        | |        | |      | |          | |           |
     |utk.edu| |utah.edu| |case.edu| |hva.nl| |surfnet.nl| |soton.ac.uk|
     |       | |        | |        | |      | |          | |           |
     +----+--+ +--------+ +--------+ +------+ +----+-----+ +-----------+
          |                                        |
          |                                        |
       +--+--+                                  +--+--+
       |     |                                  |     |
     +-+-----+-+                                |     |
     |         |                                +-----+
     +---------+
    user: paul@surfnet.nl             surfnet.nl Authentication server
```

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 organized along country borders, but generic top-level domains such as .edu do not map to this choice of boundaries (see [RFC7593] 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 realized without a static hierarchy such as in the following figure:

```
                                    ---------
                                   /         \
                          ---------           ------------
                         /                                \
                         |    DNS                          -
               ----------|                                  \
              /          \          surfnet.nl NAPTR?       |
        (1)  /            ----       -> radius.surfnet.nl   /
            /                 \                            /
           /                   --------           ---------
          /                            \---------/
         |
         |   ---------------------------------------
         |  /              (2) RADIUS               \
         |  |                                       |
     +---+---+ +----+---+ +----+---+ +--+---+ +-----+----+ +-----+-----+
     |       | |        | |        | |      | |          | |           |
     |utk.edu| |utah.edu| |case.edu| |hva.nl| |surfnet.nl| |soton.ac.uk|
     |       | |        | |        | |      | |          | |           |
     +----+--+ +--------+ +--------+ +------+ +----+-----+ +-----------+
          |                                        |
          |                                        |
       +--+--+                                  +--+--+
       |     |                                  |     |
     +-+-----+-+                                |     |
     |         |                                +-----+
     +---------+
     user: paul@surfnet.nl             surfnet.nl Authentication server
```

Figure 2: RADIUS Hierarchy Based on Top-Level Domain Partitioning

This document also specifies various approaches for verifying that server information that was retrieved from DNS was from an authorized party; for example, an organization that 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 that initiates a new connection.

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

RADIUS/TLS Node: a RADIUS/TLS client or server.

[RFC7542] defines the terms NAI, realm, and 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 that match NAI realms) is not of a very experimental nature.

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

- provisions for determining the authority of a server to act for users of a realm (declared out of scope for Diameter)

- much more in-depth guidance on DNS regarding timeouts, failure conditions, and alteration of Time-To-Live (TTL) information than the Diameter counterpart

- 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 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 Authorization, 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 Port 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 that 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 that 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. 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 the 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 that 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 discovery algorithm does not have provisions to establish confidential keying material between the RADIUS/TLS client (i.e., the server that executes the discovery algorithm) and the RADIUS/TLS server that was discovered, Pre-Shared Key (PSK) ciphersuites for TLS 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 that allows verification of whether the contacted host is authoritative for an NAI realm or not. It also gives one example of another mechanism that is currently in widespread deployment and one possible approach based on DNSSEC, which is yet unimplemented.

For the approaches that 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 that is independent from default "browser" trust stores. Often, this will be one or a few Certification Authorities (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 that operate outside the control of the roaming consortium, since their issuance of certificates with the properties important for authorization (such as NAIRealm and policyOID below) is difficult to verify. Therefore, clients SHOULD NOT be preconfigured 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 Authentication, Authorization, and Accounting (AAA) services over RADIUS/TLS is a two-step process.

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

Step 2 is to compare the value of the 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 well-formedness and validity as per [RFC5280] and whether it was issued from a root certificate that 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 that 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 that provide mutual authentication and credential protection (i.e., Extensible Authentication Protocol (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; that is, the entity that 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 recognized 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 and DNS-Based Authentication of Named Entities (DANE) as follows: DANE/TLSA records of all authorized servers are put into a DNSSEC zone that contains all known and authorized 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 RRs) 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 Authorization

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 prioritized list. [RFC2782] does not specify rules regarding "Definition of Conditions for Retry/Failure" nor "Server Identification and Handshake". This specification states that the rules for these two topics as defined in Sections 2.1.1.2 and 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).

2.1.3. 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 that 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 [RFC7593]) 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 that 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 that 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 that was contacted is authorized to service requests for the user that 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 that 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 an NAI realm to be expressed; 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 a 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 used to look up the SRV record 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 does 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 8 }

   ub-naiRealm-length INTEGER ::= 255

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

The NAIRealm, if present, MUST contain an NAI realm as defined in [RFC7542]. 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 an "*" 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 match the NAI realm, the server is considered authorized; if none match, 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 that 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 sure 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 that contains the RR that was queried for.

Negative DNS response: A response that 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 it is 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 the "Usage Rules" section, 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 is 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 that fail the conversion rules as per [RFC5891].

UTF-8 realms that 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", Section 3.4.5, 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 tags "aaa+auth", "aaa+acct", and "aaa+dynauth" as appropriate. Keep note of the protocol tag and remaining TTL of each of the discovered NAPTR records.

8. If no records are 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 look up 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’ = (set of {hostname; port; protocol; order/preference; Effective TTL (all DNS TTLs that led to this hostname) } for all terminal lookup results).

12. Proceed with step 18.

13. Generate R’ = (prefix R with "_radiustls._tcp." and/or "_radiustls._udp.").

14. Using the host’s name resolution library, perform SRV lookup with R’ as label (see "Delay Considerations", Section 3.4.5, below).

15. If name resolution returns with error, O-1 = { empty set }, O-2 = BACKOFF_TIME, and terminate.

16. If the result is a negative DNS response, O-1 = { empty set }, O-2 = min { O-2, Effective TTL (TTL value of the SOA record) }, and terminate.

17. O’ = (set of {hostname; port; protocol; order/preference; Effective TTL (all DNS TTLs that led to this result) } for all hostnames).

18. Generate O-1 by resolving hostnames in O’ into corresponding A and/or AAAA addresses: O-1 = (set of {IP address; port; protocol; order/preference; Effective TTL (all DNS TTLs that led to this result) } for all hostnames), O-2 = 0.

19. For each element in O-1, test if the original request that triggered dynamic discovery was received on {IP address; port}. If yes, O-1 = { empty set }, O-2 = BACKOFF_TIME, log error, and terminate (see next section for a rationale). If no, O is the result of dynamic discovery; terminate.

20. O-1 = { empty set }, O-2 = BACKOFF_TIME, log error, and terminate.

3.4.4. Validity of Results

The discovery algorithm is used by servers that 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 recognize 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 that might introduce a loop includes the possibility that a subsequent hop has a statically configured next hop that 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 server’s own address was in a lower-priority branch of the result set that 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 that was learned during discovering the server has not expired, subsequent new user sessions for the realm that corresponds to that open connection SHOULD reuse the existing connection and SHOULD NOT re-execute the discovery algorithm nor open a new connection. To allow for a change of configuration, a RADIUS server SHOULD re-execute the discovery algorithm after the Effective TTL that is associated with this connection has expired. The server SHOULD keep the session open during this reassessment to avoid closure and immediate reopening 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, upped 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 are use of NAPTR vs. SRV; how to determine the authorization status of a contacted server for a given realm; and which trust anchors to consider trustworthy for the RADIUS conversation setup.

Random parties that 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, back-office compensations, procedures, etc.).

A roaming consortium’s roaming agreement must include a profile of which choice points in this document to use. So as 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 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 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 that 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 the amount of new executions of the 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 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. When the port for a TLS-based service is open, the RADIUS server shares all the typical attack vectors for services based on TLS (such as HTTPS and SMTPS). Deployments of RADIUS/TLS with dynamic discovery should consider these attack vectors and take appropriate countermeasures (e.g., blacklisting known bad IPs on a firewall, rate limiting new connection attempts, etc.).

6. Privacy Considerations

The classic RADIUS operational model (known, preconfigured peers, shared secret security, and 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,

- 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.

- 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.

- 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 that are cryptographically secured and do not use the attribute User-Password -- such as certain EAP types.

- 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,

- 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 are 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 sanitize RADIUS datagrams (filter attributes, tag requests with new attributes, etc.).

- 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 that are only deployed for pure request routing reasons. This reduces the number of entities that can inspect the RADIUS traffic.

- RADIUS clients that 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.

- 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 that do not make use of this algorithm, and they possibly use 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 authentication, while with dynamic discovery, many entities learn only very little about recently authenticated realms.

7. IANA Considerations

Per this document, IANA has added the following entries in existing registries:

- S-NAPTR Application Service Tags registry

    * aaa+auth

    * aaa+acct

    * aaa+dynauth

- 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 Section 8.1.1 of [RFC6335] 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 7585

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.

Per this document, a number of Object Identifiers have been assigned. They are now under the control of IANA following [RFC7299].

IANA has assigned the following identifiers:

85 has been assigned from the "SMI Security for PKIX Module Identifier" registry. The description is id-mod-nai-realm-08.

8 has been assigned from the "SMI Security for PKIX Other Name Forms" registry. The description is id-on-naiRealm.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <http://www.rfc-editor.org/info/rfc2782>.

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, DOI 10.17487/RFC2865, June 2000, <http://www.rfc-editor.org/info/rfc2865>.

[RFC2866] Rigney, C., "RADIUS Accounting", RFC 2866, DOI 10.17487/RFC2866, June 2000, <http://www.rfc-editor.org/info/rfc2866>.

[RFC3958] Daigle, L. and A. Newton, "Domain-Based Application Service Location Using SRV RRs and the Dynamic Delegation Discovery Service (DDDS)", RFC 3958, DOI 10.17487/RFC3958, January 2005, <http://www.rfc-editor.org/info/rfc3958>.

[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, DOI 10.17487/RFC5176, January 2008, <http://www.rfc-editor.org/info/rfc5176>.

[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, DOI 10.17487/RFC5280, May 2008, <http://www.rfc-editor.org/info/rfc5280>.

[RFC5580] Tschofenig, H., Ed., Adrangi, F., Jones, M., Lior, A., and B. Aboba, "Carrying Location Objects in RADIUS and Diameter", RFC 5580, DOI 10.17487/RFC5580, August 2009, <http://www.rfc-editor.org/info/rfc5580>.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, DOI 10.17487/RFC5891, August 2010, <http://www.rfc-editor.org/info/rfc5891>.

[RFC6614] Winter, S., McCauley, M., Venaas, S., and K. Wierenga, "Transport Layer Security (TLS) Encryption for RADIUS", RFC 6614, DOI 10.17487/RFC6614, May 2012, <http://www.rfc-editor.org/info/rfc6614>.

[RFC7360] DeKok, A., "Datagram Transport Layer Security (DTLS) as a Transport Layer for RADIUS", RFC 7360, DOI 10.17487/RFC7360, September 2014, <http://www.rfc-editor.org/info/rfc7360>.

[RFC7542] DeKok, A., "The Network Access Identifier", RFC 7542, DOI 10.17487/RFC7542, May 2015, <http://www.rfc-editor.org/info/rfc7542>.

8.2. Informative References

[RFC4017] Stanley, D., Walker, J., and B. Aboba, "Extensible Authentication Protocol (EAP) Method Requirements for Wireless LANs", RFC 4017, DOI 10.17487/RFC4017, March 2005, <http://www.rfc-editor.org/info/rfc4017>.

[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, DOI 10.17487/RFC6335, August 2011, <http://www.rfc-editor.org/info/rfc6335>.

[RFC6733] Fajardo, V., Ed., Arkko, J., Loughney, J., and G. Zorn, Ed., "Diameter Base Protocol", RFC 6733, DOI 10.17487/RFC6733, October 2012, <http://www.rfc-editor.org/info/rfc6733>.

[RFC7299] Housley, R., "Object Identifier Registry for the PKIX Working Group", RFC 7299, DOI 10.17487/RFC7299, July 2014, <http://www.rfc-editor.org/info/rfc7299>.

[RFC7593] Wierenga, K., Winter, S., and T. Wolniewicz, "The eduroam Architecture for Network Roaming", RFC 7593, DOI 10.17487/RFC7593, September 2015, <http://www.rfc-editor.org/info/rfc7593>.

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(85) }

 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 RFCs 5280 and 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 RFCs 5280 and 5912
 ;


 -- Service Name Object Identifier

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

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

 -- 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