RADIUS Extensions Working Group S. Winter
Internet-Draft RESTENA
Intended status: Experimental M. McCauley
Expires: August 29, 2013 OSC
February 25, 2013

NAI-based Dynamic Peer Discovery for RADIUS/TLS and RADIUS/DTLS
draft-ietf-radext-dynamic-discovery-06

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

Copyright Notice

Table of Contents

1. Introduction

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

Where 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, 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.

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.

This document does not specify how to verify 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.

RADIUS/TLS and RADIUS/DTLS have their own ways how to verify that a contacted peer is authorised (e.g. by presenting PKIX certificates from a agreed-upon CA).

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

2. DNS-based NAPTR/SRV Peer Discovery

2.1. Applicability

Dynamic server discovery as defined in this document is only applicable for AAA transactions 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. Furthermore, it is only applicable for 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.

2.2. DNS 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 2.3 below).

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 1: List of Service Tags

This specification defines two S-NAPTR protocol tags:

Protocol Tag	Use
radius.tls	RADIUS transported over TLS as defined in [RFC6614]
radius.dtls	RADIUS transported over DTLS as defined in [I-D.ietf-radext-dtls]

Figure 2: 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.

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

SRV Label	Use

```
| _radiustls._tcp | RADIUS transported over TLS as defined  |
|                 | in [RFC6614]                             |
| - - - - - - - - | - - - - - - - - - - - - - - - - - - - - |
| _radiustls._udp | RADIUS transported over DTLS as defined |
|                 | in [I-D.ietf-radext-dtls]                |
+-----------------+-----------------------------------------+
```

Figure 3: List of SRV Labels

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 SRV and/or NAPTR labels 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.

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" ""
_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. "Bad, Inc." is part of the consortium. On the consortium's DNS server, realm bad.example might have the following DNS entries:

```
bad.example IN NAPTR 50 50 "a" "aaa+auth:radius.dtls" ""
very.bad.example
```

c. The eduroam consortium 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" ""
_radiustls._tcp.eduroam.example.org.

example.org.  IN NAPTR 50 50 "s" "aaa+auth:radius.tls" ""
_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.
```

2.3. Realm to RADIUS server resolution algorithm

This algorithm can be used to discover RADIUS servers (for RADIUS Authentication and RADIUS Accounting) or to discover RADIUS DynAuth servers.

2.3.1. 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, preceeded 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.

2.3.2. Output

Output O of the algorithm is a set of tuples {hostname; port; order/ preference; TTL} - the set can be empty.

2.3.3. 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. Optional: modify R according to agreed consortium procedures

4. Using the host's name resolution library, perform a NAPTR query for R (see "Delay considerations" below). The name resolution library may need to convert R to a different respresentation, depending on the resolution backend used. If no result, continue at step 9. If name resolution returns with error, O = { empty set } and terminate.

5. Extract NAPTR records with service tag "aaa+auth", "aaa+acct", "aaa+dynauth" as appropriate. Keep note of the remaining TTL of each of the discovered NAPTR records.

6. If no records found, continue at step 9.

7. Evaluate NAPTR result(s) for desired protocol tag, perform subsequent lookup steps until lookup yields one or more hostnames. O' = (set of {hostname; port; order/preference; min{all TTLs that led to this result} } for all lookup results). Keep note of the remaining TTL of each of the discovered records (e.g. SRV and AAAA).

8. Proceed with step 15.

9. Generate R' = (prefix R with "_radiustls._tcp." or "_radiustls._udp")

10. Using the host's name resolution library, perform SRV lookup with R' as label (see "Delay considerations" below). Keep note of the TTL of each of the discovered SRV records.

11. If name resolution returns with error, O = { empty set } and terminate.

12. If no result, O = { empty set } and terminate.

13. O' = (set of {hostname; port; order/preference; min{all TTLs that led to this result} } for all hostnames).

14. Generate O by resoving hostnames to corresponding A and/or AAAA addresses: O = (set of {IP address; port; order/preference; min{all TTLs that led to this result}} for all hostnames).

15. For each element in O, test if the original request which triggered dynamic discovery was received on {IP address; port}. If yes, O = { empty set }, log error, Terminate. If no, O is the result of dynamic discovery. Terminate.

2.3.4. Validity of results

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. When a connection is open and the smallest of the TTL values which were used for 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 lowest of the TTL values that are associated with this connection have expired. The server MAY 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 an empty set (but no error), the RADIUS server SHOULD NOT attempt another execution of this algorithm for the same target realm before the negative TTL has expired.

Should the algorithm above terminate due to an error with no TTL value known (e.g. DNS SERVFAIL), the RADIUS server SHOULD NOT attempt another execution of this algorithm for the same target realm before a configurable timeout interval has passed.

2.3.5. 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 time-out of its name resolution algorithms; instead, implementations of NAI discovery SHOULD terminate the algorithm after the fixed upper bound of time of three seconds. If no final output of the algorithm is available after this timeout, the RADIUS server MUST assume the empty set as a result and treat the pending request according to its static configuration (e.g., fallback to a default route to a home server). 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).

2.3.6. Example

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 192.37.5.1, TCP/2083.

If DNS contains the following records:

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

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

_radiustls._tcp.xn--tu-mnchen-t9a.example. IN SRV 0 10 2083 radsec.xn--tu-mnchen-t9a.example.

_radiustls._tcp.xn--tu-mnchen-t9a.example. IN SRV 0 20 2083 backup.xn--tu-mnchen-t9a.example.

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

radsec.xn--tu-mnchen-t9a.example. IN A 192.0.2.3

backup.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] Query result: (50 50 "s" "aaa+auth:radius.tls" "" _radiustls._tcp.xn--tu-mnchen-t9a.example. ; 50 50 "s" "fooservice:bar.dccp" "" _abc._def.xn--tu-mnchen-t9a.example.)

5. Result: 50 50 "s" "aaa+auth:radius.tls" "" _radiustls._tcp.xn --tu-mnchen-t9a.example.

6. NOOP

7. O' = {(radsec.xn--tu-mnchen-t9a.example.; 2083; 10; TTL A),(backup.xn--tu-mnchen-t9a. example.;2083; 20; TTL B)}

8. Go to step 15.

9. (not executed)

10. (not executed)

11. (not executed)

12. (not executed)

13. (not executed)

14. O = {(2001:0DB8::202:44ff:fe0a:f704; 2083; 10; TTL A),(192.0.2.7; 2083; 20; TTL B)}

15. O = {(2001:0DB8::202:44ff:fe0a:f704; 2083; 10; TTL A),(192.0.2.7; 2083; 20; TTL B)}. Terminate.

The implementation will then attempt to connect to two servers, with preference to radsec.xn--tu-mnchen-t9a.example.:2083, using either the AAAA or A addresses depending on the host configuration and its IP stack's capabilities.

3. Security Considerations

When using DNS without DNSSEC security extensions, the replies to NAPTR, SRV and A/AAAA requests as described in section Section 2 can not be trusted. RADIUS transports have an out-of-DNS-band means to verify that the discovery attempt led to the intended target: certificate verification or TLS-PSK keys.

4. IANA Considerations

This document requests IANA registration of 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

   * radius.dtls

This document reserves the use of the "_radiustls" and "_radiusdtls" Service labels.

This document requests the creation of a new IANA registry named "RADIUS/TLS SRV Protocol Registry" with the following initial entries:

- _tcp

- _udp

5. 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.

[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.

[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.

[I-D.ietf-radext-dtls]
DeKok, A., "DTLS as a Transport Layer for RADIUS", draft-ietf-radext-dtls-02 (work in progress), July 2012.

[RFC6614] Winter, S., McCauley, M., Venaas, S., and K. Wierenga, "Transport Layer Security (TLS) Encryption for RADIUS", RFC 6614, May 2012.

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.