Network Working Group
INTERNET-DRAFT
Updates: 2865, 2866, 2869, 3579
Category: Proposed Standard
<draft-ietf-radext-fixes-07.txt>
Expires: March 1, 2007
4 September 2007

David Nelson
Elbrys Networks, Inc
Alan DeKok
FreeRADIUS

Common Remote Authentication Dial In User Service (RADIUS) Implementation Issues and Suggested Fixes

Abstract

This document describes common issues seen in RADIUS implementations and suggests some fixes. Where applicable, ambiguities and errors in previous RADIUS specifications are clarified.

Table of Contents

1. Introduction

The last few years have seen an increase in the deployment of RADIUS clients and servers. This document describes common issues seen in RADIUS implementations and suggests some fixes. Where applicable, ambiguities and errors in previous RADIUS specifications are clarified.

1.1. Terminology

This document uses the following terms:

Network Access Server (NAS)
: The device providing access to the network. Also known as the Authenticator in IEEE 802.1X or Extensible Authentication Protocol (EAP) terminology, or RADIUS client.

service
: The NAS provides a service to the user, such as network access via 802.11 or Point to Point Protocol (PPP).

session
: Each service provided by the NAS to a peer constitutes a session, with the beginning of the session defined as the point where service is first provided and the end of the session defined as the point where service is ended. A peer may have multiple sessions in parallel or series if the NAS supports that, with each session generating a separate start and stop accounting record.

silently discard
: This means the implementation discards the packet without further processing. The implementation SHOULD provide the capability of logging the error, including the contents of the silently discarded packet, and SHOULD record the event in a statistics counter.

1.2. 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 [RFC2119].

2. Issues

2.1. Session Definition

2.1.1. State Attribute

Regarding the State attribute, [RFC2865] Section 5.24 states:

> This Attribute is available to be sent by the server to the client in an Access-Challenge and MUST be sent unmodified from the client to the server in the new Access-Request reply to that challenge, if any.
>
> This Attribute is available to be sent by the server to the client in an Access-Accept that also includes a Termination-Action Attribute with the value of RADIUS-Request. If the NAS performs the Termination-Action by sending a new Access-Request upon termination of the current session, it MUST include the State attribute unchanged in that Access-Request.

Some RADIUS client implementations do not properly use the State attribute in order to distinguish a restarted EAP authentication process from the continuation of an ongoing process (by the same user on the same NAS and port). Where an EAP-Message attribute is included in an Access-Challenge or Access-Accept attribute, RADIUS servers SHOULD also include a State attribute. See Section 2.1.2 on Request ID supplementation for additional benefits to using the State attribute in this fashion.

As defined in [RFC2865] Table 5.44, Access-Request packets may contain a State attribute. The table does not qualify this statement, while the text in Section 5.24 quoted above adds other requirements not specified in that table.

We extend the requirements of [RFC2865] to say that Access-Requests which are part of an ongoing Access-Request / Access-Challenge authentication process SHOULD contain a State attribute. It is the responsibility of the server to send a State attribute in an Access-Challenge packet, if that server needs a State attribute in a subsequent Access-Request to tie multiple Access-Requests togther into one authentication session. As defined in [RFC2865] Section 5.24, the State MUST be sent unmodified from the client to the server in the new Access-Request reply to that challenge, if any.

While most server implementations require the presence of a State attribute in an Access-Challenge packet, some challenge-response systems can distinguish the initial request from the response to the challenge without using a State attribute to track an authentication

session. The Access-Challenge and subsequent Access-Request packets for those systems do not need to contain a State attribute.

Other authentication mechanisms need to tie a sequence of Access-Request / Access-Challenge packets together into one ongoing authentication session. Servers implementing those authentication mechanisms SHOULD include a State attribute in Access-Challenge packets.

In general, if the authentication process involves one or more Access-Request / Access-Challenge sequences, the State attribute SHOULD be sent by the server in the Access-Challenge packets. Using the State attribute to create a multi-packet session is the simplest method available in RADIUS today. While other methods of creating multi-packet sessions are possible (e.g. [RFC3579] Section 2.6.1), those methods are NOT RECOMMENDED.

The only permissible values for a State attribute are values provided in an Access-Accept, Access-Challenge, CoA-Request or Disconnect-Request packet. A RADIUS client MUST use only those values for the State attribute that it has previously received from a server. An Access-Request sent as a result of a new or restarted authentication run MUST NOT include the State attribute, even if a State attribute has previously been received in an Access-Challenge for the same user and port.

In contrast, Access-Requests which are intended to perform authorization checks MUST contain a State attribute in order to tie the authorization check to a previous authentication session. This last requirement often means that an Access-Accept needs to contain a State attribute, which is then used in a later Access-Request that performs authorization checks.

Access-Request packets that contain a Service-Type attribute with value Authorize Only (17) MUST contain a State attribute. Access-Request packets that contain a Service-Type attribute with value Call Check (10) SHOULD NOT contain a State attribute. Any other Access-Request packet that performs authorization checks MUST contain a State attribute.

The standard use case for Call-Check is pre-screening authentication based solely on the end-point identifier information, such as phone number or MAC address in Calling-Station-ID and optionally Called-Station-ID. In that use case there is no source of the State attribute in the NAS. Other, non-standard, uses of Call-Check may require or permit the use of a State Attribute, but are beyond the scope of this document.

Implementing Call Check functionality via requests where the User-Name and User-Password contain the same information (e.g. MAC address) is NOT RECOMMENDED. This practice gives an attacker both the clear-text and cipher-text of the User-Password field, which permits many additional attacks on the security of the RADIUS protocol. For example, if the Request Authenticator does not satisfy the [RFC2865] requirements on global and temporal uniqueness, the practice described above may lead to the compromise of the User-Password attribute in other Access-Requests for unrelated users. Access to the cipher-text also greatly simplifies offline dictionary attacks, potentially exposing the shared secret, and compromising the entire RADIUS protocol.

Any Access-Request packet that performs authorization checks, including Call Check, MUST contain a Message-Authenticator attribute. Any response to an Access-Request performing an authorization check MUST NOT contain confidential information about any user (such as Tunnel-Password), unless that Access-Request contains a State attribute. The use of State here permits the authorization check to be tied to an earlier user authentication. In that case, the server MAY respond to the NAS with confidential information about that user. The server MUST NOT respond to that authorization check with confidential information about any other user.

2.1.2. Request-ID Supplementation

[RFC3579] Section 2.6.1 states:

> In EAP, each session has its own unique Identifier space. RADIUS server implementations MUST be able to distinguish between EAP packets with the same Identifier existing within distinct sessions, originating on the same NAS. For this purpose, sessions can be distinguished based on NAS and session identification attributes. NAS identification attributes include NAS-Identifier, NAS-IPv6-Address and NAS-IPv4-Address. Session identification attributes include User-Name, NAS-Port, NAS-Port-Type, NAS-Port-Id, Called-Station-Id, Calling-Station-Id and Originating-Line-Info.

There are issues with the suggested algorithm. Since proxies may modify Access-Request attributes such as NAS-IP-Address, depending on any attribute under control of the NAS to distinguish request identifiers can result in deployment problems.

The FreeRADIUS implementation does not track EAP identifiers by NAS-IP-Address or other non-EAP attributes sent by the NAS. Instead, it uses the EAP identifier, source Internet Protocol (IP) address, and the State attribute as a "key" to uniquely identify each EAP session.

Since the State attribute is under the control of the RADIUS server, this means that the uniqueness of each session is controlled by the server, not the NAS. The algorithm used in FreeRADIUS is as follows:

```
if (EAP start, or EAP identity) {
  allocate unique State Attribute
  insert session into "active session" table with
       key=(EAP identifier, State, source IP)
} else {
  look up active session in table, with above key
}
```

This algorithm appears to work well in variety of situations, including situations where home servers receive messages via intermediate RADIUS proxies.

Implementations that do not use this algorithm are often restricted to having an EAP Identifier space per NAS, or perhaps one that is global to the implementation. These restrictions are unnecessary when the above algorithm is used, which gives each session a unique EAP Identifier space. The above algorithm SHOULD be used to track EAP sessions in preference to any other method.

2.2. Overload Conditions

2.2.1. Retransmission Behavior

[RFC2865] Section 2.4 describes the retransmission requirements for RADIUS clients:

> At one extreme, RADIUS does not require a "responsive" detection of lost data. The user is willing to wait several seconds for the authentication to complete. The generally aggressive Transmission Control Protocol (TCP) retransmission (based on average round trip time) is not required, nor is the acknowledgment overhead of TCP.
>
> At the other extreme, the user is not willing to wait several minutes for authentication. Therefore the reliable delivery of TCP data two minutes later is not useful. The faster use of an alternate server allows the user to gain access before giving up.

Some existing RADIUS clients implement excessively aggressive retransmission behavior, utilizing default retransmission timeouts of one second or less without support for congestive backoff. When deployed at large scale, these implementations are susceptible to congestive collapse. For example, as the result of a power failure, a network with 3000 NAS devices with a fixed retransmission timer of one second will continuously generate 3000 RADIUS Access-Requests per

second. This is sufficient to overwhelm most RADIUS servers.

Suggested solutions include:

[a] Jitter. To avoid synchronization, a RADIUS client SHOULD incorporate induced jitter within its retransmission algorithm, as specified below.

[b] Congestive backoff. While it is not necessary for RADIUS client implementations to implement complex retransmission algorithms, implementations SHOULD support congestive backoff.

RADIUS retransmission timers are based on the model used in DHCPv6 [RFC3315]. Variables used here are also borrowed from this specification. RADIUS is a request/response-based protocol. The message exchange terminates when the requester successfully receives the answer or the message exchange is considered to have failed according to the RECOMMENDED retransmission mechanism described below. Other retransmission mechanisms are possible, so long as they satisfy the requirements on jitter and congestive backoff.

The following algorithms apply to any client that originates RADIUS packets, including but not limited to Access-Request, Accounting-Request, Disconnect-Request, and CoA-Request [RFC3576].

The retransmission behavior is controlled and described by the following variables:

RT	Retransmission timeout
IRT	Initial retransmission time (default 2 seconds)
MRC	Maximum retransmission count (default 10 attempts)
MRT	Maximum retransmission time (default 16 seconds)
MRD	Maximum retransmission duration (default 30 seconds)
RAND	Randomization factor

With each message transmission or retransmission, the sender sets RT according to the rules given below. If RT expires before the message exchange terminates, the sender recomputes RT and retransmits the message.

Each of the computations of a new RT include a randomization factor (RAND), which is a random number chosen with a uniform distribution between -0.1 and +0.1. The randomization factor is included to

minimize synchronization of messages.

The algorithm for choosing a random number does not need to be cryptographically sound. The algorithm SHOULD produce a different sequence of random numbers from each invocation.

RT for the first message transmission is based on IRT:

```
RT = IRT + RAND*IRT
```

RT for each subsequent message retransmission is based on the previous value of RT:

```
RT = 2*RTprev + RAND*RTprev
```

MRT specifies an upper bound on the value of RT (disregarding the randomization added by the use of RAND). If MRT has a value of 0, there is no upper limit on the value of RT. Otherwise:

```
if (RT > MRT)
   RT = MRT + RAND*MRT
```

MRD specifies an upper bound on the length of time a sender may retransmit a message. The message exchange fails once MRD seconds have elapsed since the client first transmitted the message. MRD MUST be set, and SHOULD have a value between 5 and 30 seconds. These values mirror the values for a servers duplicate detection cache, as described in the next section.

MRC specifies an upper bound on the number of times a sender may retransmit a message. if MRC is zero, the message exchange fails once MRD seconds have elapsed since the client first transmitted the message. If MRC is non-zero, the message exchange fails when the either the sender has transmitted the message MRC times, or when MRD seconds have elapsed since the client first transmitted the message.

For Accounting-Request packets, the default values for MRC, MRD, and MRT SHOULD be zero. These settings will enable a RADIUS client to continue sending accounting requests to a RADIUS server until the request is acknowledged. If any of MRC, MRD, or MRT are non-zero, then the accounting information could potentially be discarded without being recorded.

2.2.2. Duplicate detection and orderly delivery.

When packets are retransmitted by a client, the server may receive duplicate requests. The limitations of the transport protocol used by RADIUS, the User Datagram Protocol (UDP), means that the Access-

Request packets may be received, and potentially processed, in an order different from the order in which the packets were sent. However, the discussion of the Identifier field in Section 3 of [RFC2865] says:

> The RADIUS server can detect a duplicate request if it has the same client source IP address and source UDP port and Identifier within a short span of time.

Also, Section 7 of [RFC4669] defines a radiusAuthServDupAccessRequests object, as

> The number of duplicate Access-Request packets received.

This text has a number of implications. First, without duplicate detection, a RADIUS server may process an authentication request twice, leading to an erroneous conclusion that a user has logged in twice. That behavior is undesirable, so duplicate detection is desirable. Second, the server may track not only the duplicate request, but also the replies to those requests. This behavior permits the server to send duplicate replies in response to duplicate requests, increasing network stability.

Since Access-Request packets may also be sent by the client in response to an Access-Challenge from the server, those packets form a logically ordered stream, and therefore have additional ordering requirements over Access-Request packets for different sessions. Implementing duplicate detection results in new packets being processed only once, ensuring order.

RADIUS servers MUST therefore implement duplicate detection for Access-Request packets, as described in Section 3 of [RFC2865]. Implementations MUST also cache the Responses (Access-Accept, Access-Challenge, or Access-Reject) that they send in response to Access-Request packets. If a server receives a valid duplicate Access-Request for which is already has sent a Response, it MUST resend its original Response without reprocessing the request. The server MUST silently discard any duplicate Access-Requests for which a Response has not been sent yet.

Each cache entry SHOULD be purged after a period of time. This time SHOULD be no less than 5 seconds, and no more than 30 seconds. After about 30 seconds, most RADIUS clients and end users will have given up on the authentication request. Therefore, there is little value in having a larger cache timeout.

Cache entries MUST also be purged if the server receives a valid Access-Request packet that matches a cached Access-Request packet in

source address, source port, RADIUS Identifier, and receiving socket, but where the Request Authenticator field is different from the one in the cached packet. If the request contains a Message-Authenticator attribute, the request MUST be processed as described in [RFC3580] Section 3.2. Packets with invalid Message-Authenticators MUST NOT affect the cache in any way.

However, Access-Request packets not containing a Message-Authenticator attribute always affect the cache, even though they may be trivially forged. To avoid this issue, server implementations may be configured to require the presence of a Message-Authenticator attribute in Access-Request packets. Requests not containing a Message-Authenticator attribute MAY then be silently discarded.

Client implementations SHOULD include a Message-Authenticator attribute in every Access-Request, to further help mitigate this issue.

When sending requests, RADIUS clients MUST NOT re-use Identifiers for a source IP address and source UDP port until either a valid response has been received, or the request has timed out. Clients SHOULD allocate Identifiers via a least-recently-used (LRU) method for a particular source IP address and source UDP port

RADIUS clients do not have to perform duplicate detection. When a client sends a request, it processes the first response that has a valid Response Authenticator as defined in [RFC2865] Section 3. Any later responses MUST be silently discarded, as they do not match a pending request. That is, later responses are treated exactly the same as unsolicited responses, and are silently discarded.

2.2.3. Server Response to Overload

Some RADIUS server implementations are not robust in response to overload, dropping packets with even probability across multiple sessions. In an overload situation, this results in a high failure rate for multi-round authentication protocols such as EAP [RFC3579]. Typically, users will continually retry in an attempt to gain access, increasing the load even further.

A more sensible approach is for a RADIUS server to preferentially accept RADIUS Access-Request packets containing a valid State attribute, so that multi-round authentication conversations, once begun, will be more likely to succeed. Similarly, a server that is proxying requests should preferentially process Access-Accept, Access-Challenge, or Access-Reject packets from home servers, before processing new requests from a NAS.

These methods will allow some users to gain access to the network, reducing the load created by ongoing access attempts.

2.3. Accounting Issues

2.3.1. Attributes allowed in an Interim Update

[RFC2866] indicates that Acct-Input-Octets, Acct-Output-Octets, Acct-Session-Time, Acct-Input-Packets, Acct-Output-Packets and Acct-Terminate-Cause attributes "can only be present in Accounting-Request records where the Acct-Status-Type is set to Stop."

However [RFC2869] Section 2.1 states:

> It is envisioned that an Interim Accounting record (with Acct-Status-Type = Interim-Update (3)) would contain all of the attributes normally found in an Accounting Stop message with the exception of the Acct-Term-Cause attribute.

Although [RFC2869] does not indicate that it updates [RFC2866], this is an oversight, and the above attributes are allowable in an Interim Accounting record.

2.3.2. Acct-Session-Id and Acct-Multi-Session-Id

[RFC2866] Section 5.5 describes Acct-Session-Id as Text within the figure summarizing the attribute format, but then goes to state that "The String field SHOULD be a string of UTF-8 encoded 10646 characters."

[RFC2865] defines the "Text" type as "containing UTF-8 encoded 10646 characters", which is compatible with the description of Acct-Session-Id. Since other attributes are consistently described as "Text" within both the figure summarizing the attribute format, and the following attribute definition, it appears that this is a typographical error, and that Acct-Session-Id is of type Text, and not of type String.

The definition of the Acct-Multi-Session-Id attribute also has typographical errors. It says

> A summary of the Acct-Session-Id attribute format ...

This text should read

> A summary of the Acct-Multi-Session-Id attribute format ...

The Acct-Multi-Session-Id attribute is also defined as being of type

"String". However, the language in the text strongly recommends that implementors consider the attribute as being of type "Text". It is unclear why the type "String" was chosen for this attribute when the type "Text" would be sufficient. This attribute SHOULD be treated as "Text".

2.3.3. Request Authenticator

[RFC2866] Section 4.1 states:

> The Request Authenticator of an Accounting-Request contains a 16-octet MD5 hash value calculated according to the method described in "Request Authenticator" above.

However, the text does not indicate any action to take when an Accounting-Request packet contains an invalid Request Authenticator. The following text should be considered to be part of the above description:

> The Request Authenticator field MUST contain the correct data, as given by the above calculation. Invalid packets are silently discarded. Note that some early implementations always set the Request Authenticator to all zeros. New implementations of RADIUS clients MUST use the above algorithm to calculate the Request Authenticator field. New RADIUS server implementations MUST silently discard invalid packets.

2.3.4. Interim-Accounting-Interval

[RFC2869] Section 2.1 states:

> It is also possible to statically configure an interim value on the NAS itself. Note that a locally configured value on the NAS MUST override the value found in an Access-Accept.

This requirement may be phrased too strongly. It is conceivable that a NAS implementation has a setting for a "minimum" value of Interim-Accounting-Interval, based on resource constraints in the NAS, and network loading in the local environment of the NAS. In such cases, the value administratively provisioned in the NAS should not be over-ridden by a smaller value from an Access-Accept message. The NAS's value could be over-ridden by a larger one, however. The intent is that the NAS sends accounting information at fixed intervals, short enough such that the potential loss of billable revenue is limited, but also that the accounting updates are infrequent enough such that the NAS, network, and RADIUS server are not overloaded.

2.3.5. Counter values in the RADIUS Management Information Base (MIB)

The RADIUS Authentication and Authorization Client MIB module [RFC2618], [RFC4668] includes counters of packet statistics. In the descriptive text of the MIB module, formulas are provided for certain counter objects. Implementors have noted apparent inconsistencies in the formulas, which could result in negative values.

Since the original MIB module specified in [RFC2618] had been widely implemented, the RADEXT WG chose not to change the object definitions or to create new ones within the revised MIB module [RFC4668]. However, this section explains the issues and provides guidance for implementors regarding the interpretation of the textual description and comments for certain MIB objects.

The issues raised can be summarized as follows:

Issue (1):

```
-- TotalIncomingPackets = Accepts + Rejects + Challenges +
UnknownTypes
--
-- TotalIncomingPackets - MalformedResponses - BadAuthenticators -
-- UnknownTypes - PacketsDropped = Successfully received
--
-- AccessRequests + PendingRequests + ClientTimeouts =
-- Successfully Received
```

It appears that the value of "Successfully Received" could be negative, since various counters are subtracted from TotalIncomingPackets that are not included in the calculation of TotalIncomingPackets.

It also appears that "AccessRequests + PendingRequests + ClientTimeouts = Successfully Received" should read "AccessRequests + PendingRequests + ClientTimeouts = Successfully Transmitted".

"TotalIncomingPackets" and "Successfully Received" are temporary variables, i.e. not objects within the MIB module. The comment text in the MIB modules is intended, therefore, to aid in understanding. What's of consequence is the consistency of values of the objects in the MIB module, and that does not appear to be impacted by the inconsistencies noted above. It does appear, however, that the "Successfully Received" variable should be labeled "Successfully Transmitted".

In addition, the definition of Accept, Reject or Challenge counters

indicates that they MUST be incremented before the message is validated. If the message is invalid, one of MalformedResponses, BadAuthenticators or PacketsDropped counters will be additionally incremented. In that case the first two equations are consistent, i.e. "Successfully Received" could not be negative.

Issue (2):

It appears that the radiusAuthClientPendingRequests counter is decremented upon retransmission. That would mean a retransmitted packet is not considered as being as pending, although such retransmissions can still be considered as being pending requests.

The definition of this MIB object in [RFC2618] is as follows:

> The number of RADIUS Access-Request packets destined for this server that have not yet timed out or received a response. This variable is incremented when an Access-Request is sent and decremented due to receipt of an Access-Accept, Access-Reject or Access-Challenge, a timeout or retransmission.

This object purports to count the number of pending request packets. It is open to interpretation whether or not retransmissions of a request are to be counted as additional pending packets. In either event, it seems appropriate to treat retransmissions consistently with respect to incrementing and decrementing this counter.

2.4. Multiple Filter-ID Attributes

[RFC2865] Section 5.11 states:

> Zero or more Filter-Id attributes MAY be sent in an Access-Accept packet.

In practice the behavior of a RADIUS client receiving multiple Filter-ID attributes is implementation dependent. For example, some implementations treat multiple instances of the Filter-ID attribute as alternative filters; the first Filter-ID attribute having a name matching a locally defined filter is used, and the remaining ones are discarded. Other implementations may combine matching filters.

As a result, the interpretation of multiple Filter-ID attributes is undefined within RADIUS. The sending of multiple Filter-ID attributes within an Access-Accept SHOULD be avoided within heterogeneous deployments and roaming scenarios, where it is likely to produce unpredictable results.

2.5. Mandatory and Optional Attributes

RADIUS attributes do not explicitly state whether they are optional or mandatory. Nevertheless there are instances where RADIUS attributes need to be treated as mandatory.

[RFC2865] Section 1.1 states:

> A NAS that does not implement a given service MUST NOT implement the RADIUS attributes for that service. For example, a NAS that is unable to offer Apple Remote Access Protocol (ARAP) service MUST NOT implement the RADIUS attributes for ARAP. A NAS MUST treat a RADIUS access-accept authorizing an unavailable service as an access-reject instead.

With respect to the Service-Type attribute, [RFC2865] Section 5.6 says:

> This Attribute indicates the type of service the user has requested, or the type of service to be provided. It MAY be used in both Access-Request and Access-Accept packets. A NAS is not required to implement all of these service types, and MUST treat unknown or unsupported Service-Types as though an Access-Reject had been received instead.

[RFC2865] Section 5 states:

> A RADIUS server MAY ignore Attributes with an unknown Type.
>
> A RADIUS client MAY ignore Attributes with an unknown Type.

With respect to Vendor-Specific Attributes (VSAs), [RFC2865] Section 5.26 states:

> Servers not equipped to interpret the vendor-specific information sent by a client MUST ignore it (although it may be reported). Clients which do not receive desired vendor-specific information SHOULD make an attempt to operate without it, although they may do so (and report they are doing so) in a degraded mode.

It is possible for either a standard attribute or VSA to represent a request for an unavailable service. However, where the Type, Vendor-ID, or Vendor-Type is unknown, a RADIUS client will not know whether the attribute defines a service or not.

In general, it is best for a RADIUS clients to err on the side of caution. On receiving an Access-Accept including an attribute of known Type for an unimplemented service, a RADIUS client MUST treat

it as an Access-Reject, as directed in [RFC2865] Section 1.1. On receiving an Access-Accept including an attribute of unknown Type, a RADIUS client SHOULD assume that it is a potential service definition, and treat it as an Access-Reject. Unknown VSAs SHOULD be ignored by RADIUS clients.

In order to avoid introducing changes in default behavior, existing implementations that do not obey this recommendation should make the behavior configurable, with the legacy behavior being enabled by default. A configuration flag such as "treat unknown attributes as reject" can be exposed to the system administrator. If the flag is set to true, then Access-Accepts containing unknown attributes are treated as Access-Rejects. If the flag is set to false, then unknown attributes in Access-Accepts are be silently ignored.

On receiving a packet including an attribute of unknown type, RADIUS authentication server implementations SHOULD ignore such attributes. However, RADIUS accounting server implementations typically do not need to understand attributes in order to write them to stable storage or pass them to the billing engine. Therefore, accounting server implementations SHOULD be equipped to handle unknown attributes.

To avoid misinterpretation of service requests encoded within VSAs, RADIUS servers SHOULD NOT send VSAs containing service requests to RADIUS clients that are not known to understand them. For example, a RADIUS server should not send a VSA encoding a filter without knowledge that the RADIUS client supports the VSA.

2.6. Interpretation of Access-Reject

2.6.1. Improper Use of Access-Reject

The intent of an Access-Reject is to deny access to the requested service. [RFC2865] Section 2 states:

> If any condition is not met, the RADIUS server sends an "Access-Reject" response indicating that this user request is invalid. If desired, the server MAY include a text message in the Access-Reject which MAY be displayed by the client to the user. No other Attributes (except Proxy-State) are permitted in an Access-Reject.

This text makes it clear that RADIUS does not allow the provisioning of services within an Access-Reject. If the desire is to allow limited access, then an Access-Accept can be sent with attributes provisioning limited access. Attributes within an Access-Reject are restricted to those necessary to route the message (e.g. Proxy-State), attributes providing the user with an indication that access

has been denied (e.g. an EAP-Message attribute containing an EAP-Failure) or attributes conveying an error message (e.g. a Reply-Message or Error-Cause attribute).

Unfortunately, there are examples where this requirement has been misunderstood. [RFC2869] Section 2.2 states:

> If that authentication fails, the RADIUS server should return an Access-Reject packet to the NAS, with optional Password-Retry and Reply-Messages attributes. The presence of Password-Retry indicates the ARAP NAS MAY choose to initiate another challenge-response cycle,

This paragraph is problematic from two perspectives. Firstly, a Password-Retry attribute is being returned in an Access-Reject; this attribute does not fit into the categories established in [RFC2865]. Secondly, an Access-Reject packet is being sent in the context of a continuing authentication conversation; [RFC2865] requires use of an Access-Challenge for this. [RFC2869] uses the phrase "challenge-response" to describe this use of Access-Reject, indicating that the semantics of Access-Challenge are being used.

[RFC2865] Section 4.4, addresses the semantics of Access-Challenge being equivalent to Access-Reject in some cases:

> If the NAS does not support challenge/response, it MUST treat an Access-Challenge as though it had received an Access-Reject instead.

While it is difficult to correct existing deployments of [RFC2869], we make the following recommendations:

[1] New RADIUS specifications and implementations MUST NOT use Access-Reject where the semantics of Access-Challenge are intended.

[2] Access-Reject MUST mean denial of access to the requested service. In response to an Access-Reject, the NAS MUST NOT send any additional Access-Request packets for that user session.

[3] New deployments of ARAP [RFC2869] SHOULD use Access-Challenge instead of Access-Reject packets in the conversations described in [RFC2869] Section 2.2.

We also note that the table of attributes [RFC2869] Section 5.19 has an error for the Password-Retry attribute. It says:

Request	Accept	Reject	Challenge	#	Attribute
0	0	0-1	0	75	Password-Retry

However, the text in [RFC2869] Section 2.3.2 says that Password-Retry can be included within an Access-Challenge packet, for EAP authentication sessions. We recommend a correction to the table, which removes the "0-1" from the Reject column, moves it to the Challenge column. We also add a "Note 2" to follow the existing "Note 1" in the document, to clarify the use of this attribute.

Request	Accept	Reject	Challenge	#	Attribute
0	0	0	0-1	75	Password-Retry [Note 2]

[Note 2] As per RFC 3579, the use of the Password-Retry in EAP authentications is deprecated. The Password-Retry attribute can be used only for ARAP authentication.

2.6.2. Service Request Denial

RADIUS has been deployed for purposes outside network access authentication, authorization and accounting. For example, RADIUS has been deployed as a "back-end" for authenticating Voice Over IP (VOIP) connections, Hypertext Transfer Protocol (HTTP) sessions (e.g. Apache), File Transfer Protocol (FTP) sessions (e.g. proftpd), and machine logins for multiple operating systems (e.g. bsdi, pam, gina). In those contexts, an Access-Reject sent to the RADIUS client MUST be interpreted as a rejection of the request for service, and the RADIUS client MUST NOT offer that service to the user.

For example, when an authentication failure occurs in the context of an FTP session, the normal semantics for rejecting FTP services apply. The rejection does not necessarily cause the FTP server to terminate the underlying TCP connection, but the FTP server MUST NOT offer any services protected by user authentication.

Users may request multiple services from the NAS. Where those services are independent, the deployment MUST treat the RADIUS sessions as being independent.

For example, a NAS may offer multi-link services, where a user may have multiple simultaneous network connections. In that case, an Access-Reject for a later multi-link connection request does not necessarily mean that earlier multi-link connections are torn down. Similarly, if a NAS offers both dialup and VOIP services, the rejection of a VOIP attempt does not mean that the dialup session is torn down.

2.7. Addressing

2.7.1. Link-Local Addresses

Since Link-Local addresses are unique only on the local link, if the NAS and RADIUS server are not on the same link, then an IPv6 Link-Local address [RFC2462] or an IPv4 Link-Local Address [RFC3927] cannot be used to uniquely identify the NAS. A NAS SHOULD NOT utilize a link-scope address within a NAS-IPv6-Address or NAS-IP-Address attributes. A RADIUS server receiving a NAS-IPv6-Address or NAS-IP-Address attribute containing a Link-Local address SHOULD NOT count such an attribute toward satisfying the requirements of [RFC3162] Section 2.1:

> NAS-IPv6-Address and/or NAS-IP-Address MAY be present in an Access-Request packet; however, if neither attribute is present then NAS-Identifier MUST be present.

2.7.2. Multiple Addresses

There are situations in which a RADIUS client or server may have multiple addresses. For example, a dual stack host can have both IPv4 and IPv6 addresses; a host that is a member of multiple VLANs could have IPv4 and/or IPv6 addresses on each VLAN; a host can have multiple IPv4 or IPv6 addresses on a single interface. However, [RFC2865] Section 5.44 only permits zero or one NAS-IP-Address attribute within an Access-Request and [RFC3162] Section 3 only permits zero or one NAS-IPv6-Address attribute within an Access-Request. When a NAS has more than one global address and no ability to determine which is used for identification in a particular request, it is RECOMMENDED that the NAS include the NAS-Identifier attribute in an Access-Request in order to identify itself to the RADIUS server.

[RFC2865] Section 3 states:

> A RADIUS server MUST use the source IP address of the RADIUS UDP packet to decide which shared secret to use, so that RADIUS requests can be proxied.

Therefore if a RADIUS client sends packets from more than one source address, a shared secret will need to be configured on both the client and server for each source address.

2.8. Idle-Timeout

With respect to the Idle-Timeout attribute, [RFC2865] Section 5.28 states:

> This Attribute sets the maximum number of consecutive seconds of

> idle connection allowed to the user before termination of the session or prompt. This Attribute is available to be sent by the server to the client in an Access-Accept or Access-Challenge.

[RFC3580] Section 3.12 states:

> The Idle-Timeout attribute is described in [RFC2865]. For IEEE 802 media other than 802.11 the media are always on. As a result the Idle-Timeout attribute is typically only used with wireless media such as IEEE 802.11. It is possible for a wireless device to wander out of range of all Access Points. In this case, the Idle-Timeout attribute indicates the maximum time that a wireless device may remain idle.

In the above paragraphs "idle" may not necessarily mean "no traffic"; the NAS may support filters defining what traffic is included in the idle time determination. As a result, an "idle connection" is defined by local policy in the absence of other attributes.

2.9. Unknown Identity

[RFC3748] Section 5.1 states:

> If the Identity is unknown, the Identity Response field should be zero bytes in length.

However, [RFC2865] Section 5.1 describes the User-Name attribute as follows:

> The String field is one or more octets.

How should the RADIUS client behave if it receives an EAP-Response/Identity that is zero octets in length?

[RFC2865] Section 5.1 states:

> This Attribute indicates the name of the user to be authenticated. It MUST be sent in Access-Request packets if available.

This suggests that the User-Name attribute may be ommitted if it is unavailable.

However, [RFC3579] Section 2.1 states:

> In order to permit non-EAP aware RADIUS proxies to forward the Access-Request packet, if the NAS initially sends an EAP-Request/Identity message to the peer, the NAS MUST copy the contents of the Type-Data field of the EAP-Response/Identity

> received from the peer into the User-Name attribute and MUST include the Type-Data field of the EAP-Response/Identity in the User-Name attribute in every subsequent Access-Request.

This suggests that the User-Name attribute should contain the contents of the Type-Data field of the EAP-Response/Identity, even if it is zero octets in length.

Note that [RFC4282] does not permit a Network Access Identifier (NAI) of zero octets, so that an EAP-Response/Identity with a Type-Data field of zero octets MUST NOT be construed as a request for privacy (e.g. anonymous NAI).

When a NAS receives an EAP-Response/Identity with a Type-Data field that is zero octets in length, it is RECOMMENDED that it either omit the User-Name attribute in the Access-Request or include the Calling-Station-Id in the User-Name attribute, along with a Calling-Station-Id attribute.

2.10. Responses after retransmissions

Some implementations do not correctly handle the receipt of RADIUS responses after retransmissions. [RFC2865] Section 2.5 states

> If the NAS is retransmitting a RADIUS request to the same server as before, and the attributes haven't changed, you MUST use the same Request Authenticator, ID, and source port. If any attributes have changed, you MUST use a new Request Authenticator and ID.

Note that changing the Request ID for a retransmission may have undesirable side effects. Since RADIUS does not have a clear definition of a "session", it is perfectly valid for a RADIUS server to treat a retransmission as a new session request, and to reject it due to (say) multiple simultaneous login restrictions are enforced. In that situation, the NAS may receive a belated Access-Accept for the first request, and an Access-Reject for the retransmitted request, both of which apply to the same "session".

We suggest that the contents of Access-Request packets SHOULD NOT be changed during retransmissions. If they must be changed due to the inclusion of an Event-Timestampt attribute, for example, then responses to earlier transmissions MUST be silently discarded. Any response to the current request MUST be treated as the definitive response, even if as noted above, it disagrees with earlier responses.

This problem can be made worse by implementations that use a fixed

retransmission timeout (30 seconds is common). We reiterate the suggestions above in Section 2.1 about using congestive backoff. In that case, responses to earlier transmissions MAY be used as data points for congestive backoff, even if their contents are discarded.

2.11. Framed-IPv6-Prefix

[RFC3162] Section 2.3 says

> This Attribute indicates an IPv6 prefix (and corresponding route) to be configured for the user. It MAY be used in Access-Accept packets, and can appear multiple times. It MAY be used in an Access-Request packet as a hint by the NAS to the server that it would prefer these prefix(es), but the server is not required to honor the hint. Since it is assumed that the NAS will plumb a route corresponding to the prefix, it is not necessary for the server to also send a Framed-IPv6-Route attribute for the same prefix.

An Internet Service Provider (ISP) may desire to support Prefix Delegation [RFC4818] at the same time that it would like to assign a prefix for the link between the NAS and the user. The intent of the paragraph was to enable the NAS to advertise the prefix (such as via a Router Advertisement). If the Framed-Routing attribute is used, it is also possible that the prefix would be advertised in a routing protocol such as RIPNG. RFC 2865 Section 5.10 describes the purpose of Framed-Routing:

> This Attribute indicates the routing method for the user, when the user is a router to a network. It is only used in Access-Accept packets.

The description of the Prefix-Length field in RFC 3162 indicates excessively wide latitude:

> The length of the prefix, in bits. At least 0 and no larger than 128.

This length appears too broad, because it is not clear what a NAS should do with a prefix of greater granularity than /64. For example, the Framed-IPv6-Prefix may contain a /128. This does not imply that the NAS should assign an IPv6 address to the end user, because RFC 3162 already defines a Framed-IPv6-Identifier attribute to handle the Identifier portion.

It appears that the Framed-IPv6-Prefix is used for the link between the NAS and CPE only if a /64 prefix is assigned. When a /64 or larger prefix is sent, the intent is for the NAS to send a routing

advertisement containing the information present in the Framed-IPv6-Prefix attribute.

The CPE may also require a delegated prefix for its own use, if it is decrementing the Time To Live (TTL) field of IP headers. In that case, it should be delegated a prefix by the NAS via the Delegated-IPv6-Prefix attribute. [RFC4818]. If the CPE is not decrementing TTL, it does not require a delegated prefix.

3. IANA Considerations

This specification does not create any new registries, nor does it require assignment of any protocol parameters.

4. Security Considerations

The contents of the State attribute are available to both the RADIUS client and observers of the RADIUS protocol. RADIUS server implementations should ensure that the state attribute does not disclose sensitive information to a RADIUS client or third parties observing the RADIUS protocol.

The cache mechanism described in Section 2.2.2 is vulnerable to attacks when Access-Request packets do not contain a Message-Authenticator attribute. If the server accepts requests without a Message-Authenticator, then RADIUS packets can be trivially forged by an attacker. Cache entries can then be forcibly expired, negating the utility of the cache. This attack can be mitigated by following the suggestions in [RFC3579] Section 4, or by requiring the presence of Message-Authenticator, as described in Sections 2.1.1 and 2.2.2.

Since this document describes the use of RADIUS for purposes of authentication, authorization, and accounting in a wide variety of networks, applications using these specifications are vulnerable to all of the threats that are present in other RADIUS applications. For a discussion of these threats, see [RFC2865], [RFC2607], [RFC3162], [RFC3579], and [RFC3580].

5. References

5.1. Normative references

[RFC2865]
Rigney, C., Rubens, A., Simpson, W. and S. Willens, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC4818]
Salowey, J., Droms., R, "RADIUS Delegated-IPv6-Prefix Attribute",

RFC 4818, April 2007.

5.2. Informative references

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March, 1997.

[RFC2462] Thomson, S. and T. Narten, "IPv6 Stateless Address Autoconfiguration", RFC 2462, December 1998.

[RFC2607] Aboba, B. and J. Vollbrecht, "Proxy Chaining and Policy Implementation in Roaming", RFC 2607, June 1999.

[RFC2618] Aboba, B. and G. Zorn, "RADIUS Authentication Client MIB", RFC 2618, June 1999.

[RFC2866] Rigney, C., "RADIUS Accounting", RFC 2866, June 2000.

[RFC2869] Rigney, C., Willats, W. and P. Calhoun, "RADIUS Extensions", RFC 2869, June 2000.

[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.

[RFC3576] Chiba, M., Dommety, G., Eklund, M., Mitton, D. and B. Aboba, "Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS)", RFC 3576, July 2003.

[RFC3579] Aboba, B. and P. Calhoun, "RADIUS Support for Extensible Authentication Protocol (EAP)", RFC 3579, September 2003.

[RFC3580] Congdon, P., Aboba, B., Smith, A., Zorn, G. and J. Roese, "IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Usage Guidelines", RFC 3580, September 2003.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J. and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[RFC3927] Cheshire, S., Aboba, B. and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses", RFC 3927, May 2005.

[RFC4282] Aboba, B., Beadles, M., Arkko, J. and P. Eronen, "The Network Access Identifier", RFC 4282, December 2005.

[RFC4668] Nelson, D, "RADIUS Authentication Client MIB for IPv6", RFC 4668, August 2006.

[RFC4669] Nelson, D, "RADIUS Authentication Server MIB for IPv6", RFC 4669, August 2006.

Acknowledgments

The authors would like to acknowledge Glen Zorn and Bernard Aboba for contributions to this document.

The alternate algorithm to [RFC3579] Section 2.6.1 that is described in section 2.1.2 of this document was designed by Raghu Dendukuri.

The text discussing retransmissions in Section 2.2.1 is taken with minor edits from Section 9 of draft-ietf-pana-pana-17.txt

Alan DeKok wishes to acknowledge the support of Quiconnect Inc., where he was employed during much of the work on this document.

David Nelson wishes to acknowledge the support of Enterasys Networks, where he was employed during much of the work on this document.

Authors' Addresses

David B. Nelson
Elbrys Networks, Inc.
75 Rochester Ave., Unit 3
Portsmouth N.H. 03801 USA

Phone: +1.603.570.2636

Email: d.b.nelson@comcast.net

Alan DeKok
The FreeRADIUS Server Project
http://freeradius.org/

Email: aland@freeradius.org

Intellectual Property Statement

Disclaimer of Validity

Full Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.

Open issues

Open issues relating to this specification are tracked on the following web site:

http://www.drizzle.com/~aboba/RADEXT/