                 Use of Status-Server Packets in the
        Remote Authentication Dial In User Service (RADIUS) Protocol
                    draft-ietf-radext-status-server-06

Status of this Memo

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

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF), its areas, and its working groups. Note that
   other groups may also distribute working documents as Internet-
   Drafts.

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

   The list of current Internet-Drafts can be accessed at
   http://www.ietf.org/ietf/1id-abstracts.txt.

   The list of Internet-Draft Shadow Directories can be accessed at
   http://www.ietf.org/shadow.html.

   This Internet-Draft will expire on August 16, 2010.

Copyright Notice

Abstract

This document describes a deployed extension to the Remote
Authentication Dial In User Service (RADIUS) protocol, enabling
clients to query the status of a RADIUS server. This extension
utilizes the Status-Server (12) Code, which was reserved for
experimental use in RFC 2865.

Table of Contents

1. Introduction

   This document specifies a deployed extension to the Remote
   Authentication Dial In User Service (RADIUS) protocol, enabling
   clients to query the status of a RADIUS server. While the Status-
   Server Code (12) was defined as experimental in [RFC2865] Section 3,
   details of the operation and potential uses of the Code were not
   provided.

   As with the core RADIUS protocol, the Status-Server extension is
   stateless, and queries do not otherwise affect the normal operation
   of a server, nor do they result in any side effects, other than
   perhaps incrementing of an internal packet counter. Most of the
   implementations of this extension have utilized it alongside
   implementations of RADIUS as defined in [RFC2865], so that this
   document focuses solely on the use of this extension with UDP
   transport.

   The rest of this document is laid out as follows. Section 2 contains
   the problem statement, and explanations as to why some possible
   solutions can have unwanted side effects. Section 3 defines the
   Status-Server packet format. Section 4 contains client and server
   requirements, along with some implementation notes. Section 5 lists
   additional considerations not covered in the other sections. The
   remaining text contains a RADIUS table of attributes, and discusses
   security considerations not covered elsewhere in the document.

1.1. Applicability

   This protocol is being recommended for publication as an
   Informational RFC rather than as a standards-track RFC because of
   problems with deployed implementations. This includes security
   vulnerabilities. The fixes recommended here are compatible with
   existing servers that receive Status-Server packets, but impose new
   security requirements on clients that send Status-Server packets.

   Some existing implementations of this protocol do not support the
   Message-Authenticator attribute. This enables an unauthorized client
   to spoofing Status-Server packets, potentially leading to incorrect
   Access-Accepts. In order to remedy this problem, this specification
   requires the use of the Message-Authenticator attribute to provide
   per-packet authentication and integrity protection.

   With existing implementations of this protocol, the potential exists
   for Status-Server requests to be in conflict with Access-Request or
   Accounting-Requests packets using the same Identifier. This
   specification recommends techniques to avoid this problem.

These limitations are discussed in more detail below.

1.2. 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 terminology) or RADIUS client.

RADIUS Proxy
     In order to provide for the routing of RADIUS authentication and
     accounting requests, a RADIUS proxy can be employed. To the NAS,
     the RADIUS proxy appears to act as a RADIUS server, and to the
     RADIUS server, the proxy appears to act as a RADIUS client.

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

1.3. 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. Overview

   Status-Server packets are sent by a RADIUS client to a RADIUS server
   in order to test the status of that server. A Message-Authenticator
   attribute MUST be included so as to provide per-packet authentication
   and integrity protection. A single Status-Server packet MUST be
   included within a UDP datagram. RADIUS proxies MUST NOT forward
   Status-Server packets.

   Since a Status-Server packet MUST NOT be forwarded by a RADIUS proxy
   or server, the destination of a Status-Server packet is set to the IP
   address of the server which is being tested. As a result, the client
   is provided with an indication of the status of that server only,
   since no RADIUS proxies are on the path between the RADIUS client and
   server. Since servers respond to a Status-Server packet without
   examining the User-Name attribute, the response to a Status-Server
   packet cannot be used to infer any information about the reachability
   of specific realms.

   A RADIUS server or proxy implementing this specification SHOULD
   respond to a Status-Server packet with an Access-Accept
   (authentication port) or Accounting-Message (accounting port). An
   Access-Challenge response is NOT RECOMMENDED. An Access-Reject
   response MAY be used. The list of attributes that are permitted in
   Status-Server and Access-Accept packets responding to Status-Server
   packets are provided in the Section 6.

   The "hop by hop" functionality of Status-Server packets is useful to
   RADIUS clients attempting to determine the status of elements on the
   path between the client and a server. Since the Status-Server packet
   is non-forwardable, lack of a response may only be due to packet loss
   or the failure of the server in the destination IP address, not due
   to faults in downstream links, proxies or servers. It therefore
   provides an unambiguous indication of the status of a server.

   This information may be useful in situations where the RADIUS client
   does not receive a response to an Access-Request. A client may have
   multiple proxies configured, with one proxy marked as primary and
   another marked as secondary. If the client does not receive a
   response to a request sent to the primary proxy, it can "fail over"
   to the secondary, and send requests to the secondary instead of to
   the primary proxy.

   However, it is possible that the lack of a response to requests sent
   to the primary proxy was due not to a failure within the the primary,
   but to alternative causes such as a failed link along the path to the
   destination server, or the failure of the destination server.

In such a situation, it may be useful for the client to be able to
distinguish between failure causes so that it does not trigger
failover inappropriately. For example, if the primary proxy is down,
then a quick failover to the secondary proxy would be prudent.
Whereas if a downstream failure is the cause, then the value of
failover to a secondary proxy will depend on whether packets
forwarded by the secondary will utilize independent links,
intermediaries or destination servers.

The Status-Server packet is not a "Keep-Alive" as discussed in
[RFC2865] Section 2.6. "Keep-Alives" are Access-Request packets sent
to determine whether a downstream server is responsive. These
packets are typically sent only when a server is suspected to be
down, and are no longer sent as soon as the server is available
again.

2.1. Why Access-Request is inappropriate

One possible solution to the problem of querying server status is for
a NAS to send specially formed Access-Request packets to a RADIUS
server's authentication port. The NAS can then look for a response,
and use this information to determine if the server is active or
unresponsive.

However, the server may see the request as a normal login request for
a user, and conclude that a real user has logged onto that NAS. The
server may then perform actions that are undesirable for a simple
status query. The server may alternatively respond with an Access-
Challenge, indicating that it believes an extended authentication
conversation is necessary.

Another possibility is that the server responds with an Access-
Reject, indicating that the user is not authorized to gain access to
the network. As above, the server may also perform local site
actions, such as warning an administrator of failed login attempts.
The server may also delay the Access-Reject response, in the
traditional manner of rate-limiting failed authentication attempts.
This delay in response means that the querying administrator is
unsure as to whether or not the server is down, is slow to respond,
or is intentionally delaying its response to the query.

In addition, using Access-Request queries may mean that the server
may have local users configured whose sole reason for existence is to
enable these query requests. Unless the server's policy is designed
carefully, it may be possible for an attacker to use those
credentials to gain unauthorized network access.

We note that some NAS implementations currently use Access-Request

   packets as described above, with a fixed (and non configurable) user
   name and password. Implementation issues with that equipment means
   that if a RADIUS server does not respond to those queries, it may be
   marked as unresponsive by the NAS. This marking may happen even if
   the server is actively responding to other Access-Requests from that
   same NAS. This behavior is confusing to administrators who then need
   to determine why an active server has been marked as "unresponsive".

2.1.1. Recommendation against Access-Request

   For the reasons outlined above, NAS implementors SHOULD NOT generate
   Access-Request packets solely to see if a server is alive.
   Similarly, site administrators SHOULD NOT configure test users whose
   sole reason for existence is to enable such queries via Access-
   Request packets.

   Note that it still may be useful to configure test users for the
   purpose of performing end-to-end or in-depth testing of a servers
   policy. While this practice is widespread, we caution administrators
   to use it with care.

2.2. Why Accounting-Request is inappropriate

   A similar solution for the problem of querying server status may be
   for a NAS to send specially formed Accounting-Request packets to a
   RADIUS servers accounting port. The NAS can then look for a
   response, and use this information to determine if the server is
   active or unresponsive.

   As seen above with Access-Request, the server may then conclude that
   a real user has logged onto a NAS, and perform local site actions
   that are undesirable for a simple status query.

   Another consideration is that some attributes are mandatory to
   include in an Accounting-Request. This requirement forces the
   administrator to query an accounting server with fake values for
   those attributes in a test packet. These fake values increase the
   work required to perform a simple query, and may pollute the server's
   accounting database with incorrect data.

2.2.1. Recommendation against Accounting-Request

   For the reasons outlined above, NAS implementors SHOULD NOT generate
   Accounting-Request packets solely to see if a server is alive.
   Similarly, site administrators SHOULD NOT configure accounting
   policies whose sole reason for existence is to enable such queries
   via Accounting-Request packets.

Note that it still may be useful to configure test users for the
purpose of performing end-to-end or in-depth testing of a servers
policy. While this practice is widespread, we caution administrators
to use it with care.

3. Packet Format

Status-Server packets reuse the RADIUS packet format, with the fields
and values for those fields as defined [RFC2865] Section 3. We do
not include all of the text or diagrams of that section here, but
instead explain the differences required to implement Status-Server.

The Authenticator field of Status-Server packets MUST be generated
using the same method as that used for the Request Authenticator
field of Access-Request packets, as given below.

The role of the Identifier field is the same for Status-Server as for
other packets. However, as Status-Server is taking the role of
Access-Request or Accounting-Request packets, there is the potential
for Status-Server requests to be in conflict with Access-Request or
Accounting-Request packets with the same Identifier. In Section 4.2,
below, we describe a method for avoiding these problems. This method
MUST be used to avoid conflicts between Status-Server and other
packet types.

   Request Authenticator

      In Status-Server Packets, the Authenticator value is a 16 octet
      random number, called the Request Authenticator. The value
      SHOULD be unpredictable and unique over the lifetime of a
      secret (the password shared between the client and the RADIUS
      server), since repetition of a request value in conjunction
      with the same secret would permit an attacker to reply with a
      previously intercepted response. Since it is expected that the
      same secret MAY be used to authenticate with servers in
      disparate geographic regions, the Request Authenticator field
      SHOULD exhibit global and temporal uniqueness.

      The Request Authenticator value in a Status-Server packet
      SHOULD also be unpredictable, lest an attacker trick a server
      into responding to a predicted future request, and then use the
      response to masquerade as that server to a future Status-Server
      request from a client.

Similarly, the Response Authenticator field of an Access-Accept
packet sent in response to Status-Server queries MUST be generated
using the same method as used for calculating the Response
Authenticator of the Access-Accept sent in response to an Access-

Request, with the Status-Server Request Authenticator taking the
place of the Access-Request Request Authenticator.

The Response Authenticator field of an Accounting-Response packet
sent in response to Status-Server queries MUST be generated using the
same method as used for for calculating the Response Authenticator of
the Accounting-Response sent in response to an Accounting-Request,
with the Status-Server Request Authenticator taking the place of the
Accounting-Request Request Authenticator.

Note that when a server responds to a Status-Server request, it MUST
NOT send more than one response packet.

   Response Authenticator

      The value of the Authenticator field in Access-Accept, or
      Accounting-Response packets is called the Response
      Authenticator, and contains a one-way MD5 hash calculated over
      a stream of octets consisting of: the RADIUS packet, beginning
      with the Code field, including the Identifier, the Length, the
      Request Authenticator field from the Status-Server packet, and
      the response Attributes (if any), followed by the shared
      secret. That is, ResponseAuth =
      MD5(Code+ID+Length+RequestAuth+Attributes+Secret) where +
      denotes concatenation.

In addition to the above requirements, all Status-Server packets MUST
include a Message-Authenticator attribute. Failure to do so would
mean that the packets could be trivially spoofed.

Status-Server packets MAY include NAS-Identifier, and one of NAS-IP-
Address or NAS-IPv6-Address. These attributes are not necessary for
the operation of Status-Server, but may be useful information to a
server that receives those packets.

Other attributes SHOULD NOT be included in a Status-Server packet,
and MUST be ignored if they are included. User authentication
credentials such as User-Name, User-Password, CHAP-Password, EAP-
Message, etc. MUST NOT appear in a Status-Server packet sent to a
RADIUS authentication port. User or NAS accounting attributes such
as Acct-Session-Id, Acct-Status-Type, Acct-Input-Octets, etc. MUST
NOT appear in a Status-Server packet sent to a RADIUS accounting
port.

The Access-Accept MAY contain a Reply-Message or Message-
Authenticator attribute. It SHOULD NOT contain other attributes.
The Accounting-Response packets sent in response to a Status-Server
query SHOULD NOT contain any attributes. As the intent is to

   implement a simple query instead of user authentication or
   accounting, there is little reason to include other attributes in
   either the query or the corresponding response.

   Examples of Status-Server packet flows are given below in Section 7.

3.1. Single definition for Status-Server

   When sent to a RADIUS accounting port, contents of the Status-Server
   packets are calculated as described above. That is, even though the
   packets are being sent to an accounting port, they are not created
   using the same method as for Accounting-Requests. This difference
   has a number of benefits.

   Having a single definition for Status-Server packets is simpler than
   having different definitions for different destination ports. In
   addition, if we were to define Status-Server as being similar to
   Accounting-Request but containing no attributes, then those packets
   could be trivially forged.

   We therefore define Status-Server consistently, and vary the response
   packets depending on the port to which the request is sent. When
   sent to an authentication port, the response to a Status-Server query
   is an Access-Accept packet. When sent to an accounting port, the
   response to a Status-Server query is an Accounting-Response packet.

4. Implementation notes

   There are a number of considerations to take into account when
   implementing support for Status-Server. This section describes
   implementation details and requirements for RADIUS clients and
   servers that support Status-Server.

   The following text applies to the authentication and accounting
   ports. We use the generic terms below to simplify the discussion:

      * Request packet

        An Access-Request packet sent to an authentication port, or
        an Accounting-Request packet sent to an accounting port.

      * Response packet

        An Access-Accept, Access-Challenge, or Access-Reject packet sent
        from an authentication port, or an Accounting-Response packet
        sent from an accounting port.

   We also refer to "client" as the originator of the Status-Server

packet, and "server" as the receiver of that packet, and the
originator of the Response packet.

Using generic terms to describe the Status-Server conversations is
simpler than duplicating the text for authentication and accounting
packets.

4.1. Client Requirements

Clients SHOULD permit administrators to globally enable or disable
the generation of Status-Server packets. The default SHOULD be that
it is disabled. As it is undesirable to send queries to servers that
do not support Status-Server, clients SHOULD also have a per-server
configuration indicating whether or not to enable Status-Server for a
particular destination. The default SHOULD be that it is disabled.

The client SHOULD also have a configurable global timer (Tw) that is
used when sending periodic Status-Server queries during server fail-
over. The default value SHOULD be 30 seconds, and the value MUST NOT
be permitted to be set below 6 seconds. If a response has not been
received within the timeout period, the Status-Server packet is
deemed to have received no corresponding Response packet, and MUST be
discarded.

Clients SHOULD use a jitter of +/- 2 seconds when sending periodic
Status-Server packets, in order to avoid synchronization.

When Status-Server packets are sent from a client, they MUST NOT be
retransmitted. Instead, the Identity field MUST be changed every
time a packet is transmitted. The old packet should be discarded,
and a new Status-Server packet should be generated and sent, with new
Identity and Authenticator fields.

Clients MUST include the Message-Authenticator attribute in all
Status-Server packets. Failure to do so would mean that the packets
could be trivially spoofed, leading to potential denial of service
(DoS) attacks. Other attributes SHOULD NOT appear in a Status-Server
packet, except as outlined below in Section 6. As the intent of the
packet is a simple status query, there is little reason for any
additional attributes to appear in Status-Server packets.

The client MAY increment packet counters as a result of sending a
Status-Server request, or receiving a Response packet. The client
MUST NOT perform any other action that is normally performed when it
receives a Response packet, such as permitting a user to have login
access to a port.

Clients MAY send Status-Server requests to the RADIUS destination

ports from the same source port used to send normal Request packets.
Other clients MAY choose to send Status-Server requests from a unique
source port, that is not used to send Request packets.

The above suggestion for a unique source port for Status-Server
packets aids in matching responses to requests. Since the response
to a Status-Server packet is an Access-Accept or Accounting-Response
packet, those responses are indistinguishable from other packets sent
in response to a Request packet. Therefore, the best way to
distinguish them from other traffic is to have a unique port.

A client MAY send a Status-Server packet from a source port also used
to send Request packets. In that case, the Identifer field MUST be
unique across all outstanding Request packets for that source port,
independent of the value of the RADIUS Code field for those
outstanding requests. Once the client has either received a response
to the Status-Server packet, or has determined that the Status-Server
packet has timed out, it may reuse that Identifier in another packet.

Robust implementations SHOULD accept any Response packet as a valid
response to a Status-Server packet, subject to the validation
requirements defined above for the Response Authenticator. The code
field of the packet matters less than the fact that a valid, signed,
response has been received.

That is, prior to accepting the response as valid, the client should
check that the Response packet Code field is either Access-Accept (2)
or Accounting-Response (5). If the code does not match any of these
values, the packet MUST be silently discarded. The client MUST then
validate the Response Authenticator via the algorithm given above in
Section 3. If the Response Authenticator is not valid, the packet
MUST be silently discarded. If the Response Authenticator is valid,
then the packet MUST be deemed to be a valid response from the
server.

If the client instead discarded the response because the packet code
did not match what it expected, then it could erroneously discard
valid responses from a server, and mark that server as unresponsive.
This behavior would affect the stability of a RADIUS network, as
responsive servers would erroneously be marked as unresponsive. We
therefore recommend that clients should be liberal in what they
accept as responses to Status-Server queries.

4.2. Server Requirements

Servers SHOULD permit administrators to globally enable or disable
the acceptance of Status-Server packets. The default SHOULD be that
it is enabled. Servers SHOULD also permit adminstrators to enable or

disable acceptance of Status-Server packets on a per-client basis.
The default SHOULD be that it is enabled.

Status-Server packets originating from clients that are not permitted
to send the server Request packets MUST be silently discarded. If a
server does not support Status-Server packets, or is configured to
not respond to them, then it MUST silently discard the packet.

We note that [RFC2865] Section 3 defines a number of RADIUS Codes,
but does not make statements about which Codes are valid for port
1812. In contrast, [RFC2866] Section 3 specifies that only RADIUS
Accounting packets are to be sent to port 1813. This specification
is compatible with [RFC2865], as it uses a known Code for packets to
port 1812. This specification is not compatible with [RFC2866], as
it adds a new code (Status-Server) that is valid for port 1812.
However, as the category of [RFC2866] is Informational, this conflict
is acceptable.

Servers SHOULD silently discard Status-Server packets if they
determine that a client is sending too many Status-Server requests in
a particular time period. The method used by a server to make this
determination is implementation-specific, and out of scope for this
specification.

If a server supports Status-Server packets, and is configured to
respond to them, and receives a packet from a known client, it MUST
validate the Message-Authenticator attribute as defined in [RFC3579]
Section 3.2. Packets failing that validation MUST be silently
discarded.

Servers SHOULD NOT otherwise discard Status-Server packets if they
have recently sent the client a Response packet. The query may have
originated from an administrator who does not have access to the
Response packet stream, or who is interested in obtaining additional
information about the server.

The server MAY prioritize the handling of Status-Server packets over
the handling of other requests, subject to the rate limiting
described above.

The server MAY decide to not respond to a Status-Server, depending on
local site policy. For example, a server that is running but is
unable to perform its normal activities MAY silently discard Status-
Server packets. This situation can happen, for example, when a
server requires access to a database for normal operation, but the
connection to that database is down. Or, it may happen when the
accepted load on the server is lower than the offered load.

   Some server implementations require that Access-Request packets are
   accepted only on "authentication" ports, (e.g. 1812/udp), and that
   Accounting-Request packets are accepted only on "accounting" ports
   (e.g. 1813/udp). Those implementations SHOULD reply to Status-Server
   packets sent to an "authentication" port with an Access-Accept
   packet. Those implementations SHOULD reply to Status-Server packets
   sent to an "accounting" port with an Accounting-Response packet.

   Some server implementations accept both Access-Request and
   Accounting-Request packets on the same port, and do not distinguish
   between "authentication only" ports, and "accounting only" ports.
   Those implementations SHOULD reply to Status-Server packets with an
   Access-Accept packet.

   The server MAY increment packet counters as a result of receiving a
   Status-Server, or sending a Response packet. The server SHOULD NOT
   perform any other action that is normally performed when it receives
   a Request packet, other than sending a Response packet.

4.3. Fail-over with Status-Server

   A client may wish to failover from one proxy to another in the event
   that it does not receive a response to an Access-Request or
   Accounting-Request. In order to determine whether the lack of
   response is due to a problem with the proxy or a downstream server,
   the client can send periodic Status-Server packets to a proxy after
   lack of a response. The inter-packet period is Tw, as described in
   Section 4.1.

   These packets will help the client determine if the failure was due
   to an issue on the path between the client and proxy or the proxy
   itself, or whether the issue is occurring downstream.

   If no response is received to Status-Server packets, the RADIUS
   client can initiate failover to another proxy. By continuing to send
   Status-Server packets to the original proxy at an interval of Tw, the
   RADIUS client can determine when the original proxy becomes
   responsive again.

   Once three time periods have passed where Status-Server packets have
   been sent and responded to, the server should be deemed responsive
   and RADIUS requests may sent to it again. This determination should
   be made separately for each server that the client has a relationship
   with. The same algorithm should be used for both authentication and
   accounting ports. The client MUST treat each destination (ip, port)
   combination as a unique server for the purposes of this
   determination.

The above behavior is modelled after [RFC3539] Section 3.4.1. We
note that if a reliable transport is used for RADIUS, then the
algorithms specified in [RFC3539] MUST be used in preference to the
ones given here.

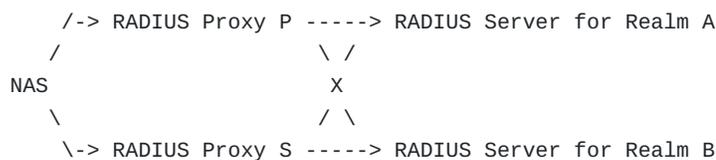4.4. Proxy Server handling of Status-Server

Many RADIUS servers can act as proxy servers, and can forward
requests to another RADIUS server. Such servers MUST NOT proxy
Status-Server packets. The purpose of Status-Server as specified
here is to permit the client to query the responsiveness of a server
that it has a direct relationship with. Proxying Status-Server
queries would negate any usefulness that may be gained by
implementing support for them.

Proxy servers MAY be configured to respond to Status-Server queries
from clients, and MAY act as clients sending Status-Server queries to
other servers. However, those activities MUST be independent of one
another.

4.5. Limitations of Status-Server

RADIUS servers are commonly used in an environment where Network
Access Identifiers (NAIs) are used as routing identifiers [RFC4282].
In this practice, the User-Name attribute is decorated with realm
routing information, commonly in the format of "user@realm". Since a
particular RADIUS server may act as a proxy for more than one realm,
we need to explain how the behavior defined above in Section 4.3,
above, affects realm routing.

The schematic below demonstrates this scenario.

```
            /-> RADIUS Proxy P -----> RADIUS Server for Realm A
           /                  \ /
        NAS                    X
           \                  / \
            \-> RADIUS Proxy S -----> RADIUS Server for Realm B
```

That is, the NAS has relationships with two RADIUS Proxies, P and S.
Each RADIUS Proxyhas relationships with RADIUS Servers for both Realm
A and Realm B.

In this scenario, the RADIUS Proxies can determine if one or both of
the RADIUS Servers are dead or unreachable. The NAS can determine if
one or both of the RADIUS Proxies are dead or unreachable. There is
an additional case to consider, however.

If RADIUS Proxy P cannot reach the RADIUS Server for Realm A, but the

RADIUS Proxy S can reach that RADIUS Server, then the NAS cannot
discover this information using the Status-Server queries as outlined
above. It would therefore be useful for the NAS to know that Realm A
is reachable from RADIUS Proxy S, as it can then route all requests
for Realm A to that RADIUS Proxy. Without this knowledge, the client
may route requests to RADIUS Proxy P, where they may be discarded or
rejected.

To complicate matters, the behavior of RADIUS Proxies P and S in this
situation is not well defined. Some implementations simply fail to
respond to the request, and other implementations respond with an
Access-Reject. If the implementation fails to respond, then the NAS
cannot distinguish between the RADIUS Proxy being down, or the next
server along the proxy chain being unreachable.

In the worst case, failures in routing for Realm A may affect users
of Realm B. For example, if RADIUS Proxy P can reach Realm B but not
Realm A, and RADIUS Proxy S can reach Realm A but not Realm B, then
active paths exist to handle all RADIUS requests. However, depending
on the NAS and RADIUS Proxy implementation choices, the NAS may not
be able to determine which server requests may be sent to in order to
maintain network stability.

This problem cannot, unfortunately be solved by using Status-Server
requests. A robust solution would involve either a RADIUS routing
table for the NAI realms, or a RADIUS "destination unreachable"
response to authentication requests. Either solution would not fit
into the traditional RADIUS model, and both are therefore outside of
the scope of this specification.

The problem is discussed here in order to define how best to use
Status-Server in this situation, rather than to define a new
solution.

When a server has responded recently to a request from a client, that
client MUST mark the server as "responsive". In the above case, a
RADIUS Proxy may be responding to requests destined for Realm A, but
not responding to requests destined for Realm B. The client
therefore considers the server to be responsive, as it is receiving
responses from the server.

The client will then continue to send requests to the RADIUS Proxy
for destination Realm B, even though the RADIUS Proxy cannot route
the requests to that destination. This failure is a known limitation
of RADIUS, and can be partially addressed through the use of failover
in the RADIUS Proxies.

A more realistic situation than the one outlined above is where each

   RADIUS Proxy also has multiple choices of RADIUS Servers for a realm,
   as outlined below.

```
             /-> RADIUS Proxy P -----> RADIUS Server P
            /                 \ /
        NAS                    X
            \                 / \
             \-> RADIUS Proxy S -----> RADIUS Server S
```
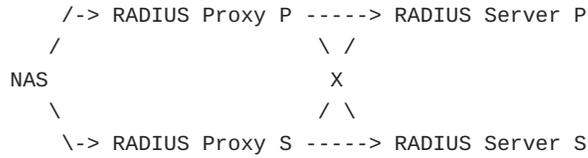
   In this situation, if all participants implement Status-Server as
   defined herein, any one link may be broken, and all requests from the
   NAS will still reach a RADIUS Server. If two links are broken at
   different places, (i.e. not both links from the NAS), then all
   requests from the NAS will still reach a RADIUS Server. In many
   situations where three or more links are broken, then requests from
   the NAS may still reach a RADIUS Server.

   It is RECOMMENDED, therefore, that implementations desiring the most
   benefit from Status-Server also implement server failover. The
   combination of these two practices will maximize network reliability
   and stability.

4.6. Management Information Base (MIB) Considerations

4.6.1. Interaction with RADIUS Server MIB modules

   Since Status-Server packets are sent to the defined RADIUS ports,
   they can affect the [RFC4669] and [RFC4671] RADIUS server MIB
   modules. [RFC4669] defines a counter named
   radiusAuthServTotalUnknownTypes that counts "The number of RADIUS
   packets of unknown type that were received". [RFC4671] defines a
   similar counter named radiusAcctServTotalUnknownTypes.
   Implementations not supporting Status-Server, or implementations that
   are configured to not respond to Status-Server packets MUST use these
   counters to track received Status-Server packets.

   If, however, Status-Server is supported and the server is configured
   to respond as described above, then the counters defined in [RFC4669]
   and [RFC4671] MUST NOT be used to track Status-Server requests or
   responses to those requests. That is, when a server fully implements
   Status-Server, the counters defined in [RFC4669] and [RFC4671] MUST
   be unaffected by the transmission or reception of packets relating to
   Status-Server.

   If a server supports Status-Server and the [RFC4669] or [RFC4671] MIB
   Modules, then it SHOULD also support vendor-specific MIB extensions
   dedicated solely to tracking Status-Server requests and responses.

   Any definition of the server MIB modules for Status-Server is outside
   of the scope of this document.

4.6.2. Interaction with RADIUS Client MIB modules

   Clients implementing Status-Server MUST NOT increment [RFC4668] or
   [RFC4670] counters upon reception of Response packets to Status-
   Server queries. That is, when a server fully implements Status-
   Server, the counters defined in [RFC4668] and [RFC4670] MUST be
   unaffected by the transmission or reception of packets relating to
   Status-Server.

   If an implementation supports Status-Server and the [RFC4668] or
   [RFC4670] MIB modules, then it SHOULD also support vendor-specific
   MIB extensions dedicated solely to tracking Status-Server requests
   and responses. Any definition of the client MIB module extensions
   for Status-Server is outside of the scope of this document.

5. Table of Attributes

   The following table provides a guide to which attributes may be found
   in Status-Server packets, and in what quantity. Attributes other
   than the ones listed below SHOULD NOT be found in a Status-Server
   packet.

Status- Server	Access- Accept	Accounting- Response	#	Attribute
0	0	0	1	User-Name,
0	0	0	2	User-Password
0	0	0	3	CHAP-Password
0-1	0	0	4	NAS-IP-Address (Note 1)
0	0+	0	18	Reply-Message
0+	0+	0+	26	Vendor-Specific
0-1	0	0	32	NAS-Identifier (Note 1)
0	0	0	70	EAP-MEssage
1	0-1	0-1	80	Message-Authenticator
0-1	0	0	95	NAS-IPv6-Address (Note 1)
0	0	0	103-121	Digest-*

   Note 1: A Status-Server SHOULD contain one of (NAS-IP-Address or NAS-
   IPv6-Address), or NAS-Identifier, or both NAS-Identifier and one of
   (NAS-IP-Address or NAS-IPv6-Address).

   The following table defines the meaning of the above table entries.

0   This attribute MUST NOT be present in packet.
0+   Zero or more instances of this attribute MAY be present in packet.

0-1 Zero or one instance of this attribute MAY be present in packet.
1   Exactly one instance of this attribute MUST be present in packet.

6. Examples

   A few examples are presented to illustrate the flow of packets to
   both the authentication and accounting ports. These examples are not
   intended to be exhaustive, many others are possible. Hexadecimal
   dumps of the example packets are given in network byte order, using
   the shared secret "xyzzy5461".

6.1. Minimal Query to Authentication Port

   The NAS sends a Status-Server UDP packet with minimal content to a
   RADIUS server on port 1812.

   The Request Authenticator is a 16 octet random number generated by
   the NAS. Message-Authenticator is included in order to authenticate
   that the request came from a known client.

      0c da 00 26 8a 54 f4 68 6f b3 94 c5 28 66 e3 02
      18 5d 06 23 50 12 5a 66 5e 2e 1e 84 11 f3 e2 43
      82 20 97 c8 4f a3

       1 Code = Status-Server (12)
       1 ID = 218
       2 Length = 38
      16 Request Authenticator

      Attributes:
      18 Message-Authenticator (80) = 5a665e2e1e8411f3e243822097c84fa3

   The Response Authenticator is a 16 octet MD5 checksum of the code
   (2), id (218), Length (20), the Request Authenticator from above, and
   the shared secret.

      02 da 00 14 ef 0d 55 2a 4b f2 d6 93 ec 2b 6f e8
      b5 41 1d 66

       1 Code = Access-Accept (2)
       1 ID = 218
       2 Length = 20
      16 Request Authenticator

      Attributes:
         None.

6.2. Minimal Query to Accounting Port

   The NAS sends a Status-Server UDP packet with minimal content to a
   RADIUS server on port 1813.

   The Request Authenticator is a 16 octet random number generated by
   the NAS. Message-Authenticator is included in order to authenticate
   that the request came from a known client.

```
      0c b3 00 26 92 5f 6b 66 dd 5f ed 57 1f cb 1d b7
      ad 38 82 60 50 12 e8 d6 ea bd a9 10 87 5c d9 1f
      da de 26 36 78 58

       1 Code = Status-Server (12)
       1 ID = 179
       2 Length = 38
      16 Request Authenticator

      Attributes:
      18 Message-Authenticator (80) = e8d6eabda910875cd91fdade26367858
```

   The Response Authenticator is a 16 octet MD5 checksum of the code
   (5), id (179), Length (20), the Request Authenticator from above, and
   the shared secret.

```
      02 b3 00 14 0f 6f 92 14 5f 10 7e 2f 50 4e 86 0a
      48 60 66 9c

       1 Code = Accounting-Response (5)
       1 ID = 179
       2 Length = 20
      16 Request Authenticator

      Attributes:
         None.
```

6.3. Verbose Query and Response

   The NAS at 192.0.2.16 sends a Status-Server UDP packet to the RADIUS
   server on port 1812.

   The Request Authenticator is a 16 octet random number generated by
   the NAS.

```
      0c 47 00 2c bf 58 de 56 ae 40 8a d3 b7 0c 85 13
      f9 b0 3f be 04 06 c0 00 02 10 50 12 85 2d 6f ec
      61 e7 ed 74 b8 e3 2d ac 2f 2a 5f b2
```

```
   1 Code = Status-Server (12)
   1 ID = 71
   2 Length = 44
  16 Request Authenticator

  Attributes:
   6  NAS-IP-Address (4) = 192.0.2.16
  18 Message-Authenticator (80) = 852d6fec61e7ed74b8e32dac2f2a5fb2
```

The Response Authenticator is a 16-octet MD5 checksum of the code
(2), id (71), Length (52), the Request Authenticator from above, the
attributes in this reply, and the shared secret.

The Reply-Message is "RADIUS Server up 2 days, 18:40"

```
  02 47 00 34 46 f4 3e 62 fd 03 54 42 4c bb eb fd
  6d 21 4e 06 12 20 52 41 44 49 55 53 20 53 65 72
  76 65 72 20 75 70 20 32 20 64 61 79 73 2c 20 31
  38 3a 34 30

   1 Code = Access-Accept (2)
   1 ID = 71
   2 Length = 52
  16 Request Authenticator

  Attributes:
  32 Reply-Message (18)
```

7. IANA Considerations

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

8. Security Considerations

   This document defines the Status-Server packet as being similar in
   treatment to the Access-Request packet, and is therefore subject to
   the same security considerations as described in [RFC2865], Section
   8. Status-Server packets also use the Message-Authenticator
   attribute, and are therefore subject to the same security
   considerations as [RFC3579], Section 4.

   We reiterate that Status-Server packets MUST contain a Message-
   Authenticator attribute. Early implementations supporting Status-
   Server did not enforce this requirement, and were vulnerable to the
   following attacks:

      * Servers not checking Message-Authenticator could respond to
        Status-Server packets from an attacker, potentially enabling
        a reflected DoS attack onto a real client.

      * Servers not checking Message-Authenticator could be subject to a
        race condition, where an attacker could see an Access-Request
        packet from a valid client, and synthesise a Status-Server packet
        containing the same Request Authenticator. If the attacker won the
        race against the valid client, the server could respond with an
        Access-Accept, and potentially authorize unwanted service.

   The last attack is similar to a related attack when Access-Request
   packets contain a CHAP-Password but no Message-Authenticator. We re-
   iterate the suggestion of [RFC5080] Section 2.2.2, which suggests
   that all clients send a Message-Authenticator in every Access-Request
   packet, and that all servers configurably require that a Message-
   Authenticator attribute be used in every Access-Request packet.

   Failure to include a Message-Authenticator attribute in a Status-
   Server packet means that any RADIUS client or server may be
   vulnerable to the attacks outlined above. For this reason,
   implementations of this specification which fail to require use of
   the Message-Authenticator attribute are NOT RECOMMENDED.

   Where this document differs from [RFC2865] is that it defines a new
   request/response method in RADIUS; the Status-Server request. As
   this use is based on previously described and implemented standards,
   we know of no additional security considerations that arise from the
   use of Status-Server as defined herein.

9. References

9.1. Normative references

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

[RFC4282]
     Aboba, B., and Beadles, M. at al, "The Network Access Identifier",
     RFC 4282, December 2005.

9.2. Informative references

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

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

[RFC3539] Aboba, B., Wood, J., "Authentication, Authorization, and
          Accounting (AAA) Transport Profile", RFC 3539, June 2003.

[RFC3579] Aboba, B., Calhoun, P., "RADIUS (Remote Authentication Dial In
          User Service) Support For Extensible Authentication Protocol
          (EAP)", RFC 3579, September 2003.

[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.

[RFC4670] Nelson, D., "RADIUS Accounting Client MIB for IPv6", RFC 4670,
          August 2006.

[RFC4671] Nelson, D., "RADIUS Accounting Server MIB for IPv6", RFC 4671,
          August 2006.

[RFC5080] Nelson, D., DeKok, A, "Common Remote Authentication Dial In
          User Service (RADIUS) Implementation Issues and Suggested
          Fixes", RFC 5080, December 2007.

Acknowledgments

   Parts of the text in Section 3 defining the Request and Response
   Authenticators were taken with minor edits from [RFC2865] Section 3.

   The author would like to thank Mike McCauley of Open Systems
   Consultants for making a Radiator server available for
   interoperability testing.

   Ignacio Goyret provided valuable feedback on the history and security
   of the Status-Server packet.

Authors' Addresses

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

   Email: aland@freeradius.org