Workgroup: radextra BOF
Internet-Draft:
draft-cullen-radextra-status-realm-00
Published: 24 October 2022
Intended Status: Standards Track
Expires: 27 April 2023
Authors: M. Cullen
Painless Security
A. DeKok
FreeRADIUS
M. Donnelly
Painless Security
J. Howlett
Federated Solutions

Status-Realm and Loop Prevention for the Remote Dial-In User Service (RADIUS)

Abstract

This document describes extension to the Remote Authentication Dial-In User Service (RADIUS) protocol to allow participants in a multi-hop RADIUS proxy fabric to check the status of a remote RADIUS authentication realm, gain visibility into the path that a RADIUS request will take across the RADIUS proxy fabric, and mitigate or prevent RADIUS proxy loops.

Status of This Memo

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

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

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

This Internet-Draft will expire on 27 April 2023.

Table of Contents

1. Introduction

This document describes an extension to the Remote Authentication Dial-In User Service (RADIUS) protocol [RFC2865], to allow participants in a multi-hop RADIUS proxy fabric to check the status of a remote RADIUS authentication realm, gain visibility into the path that a RADIUS request will take across the RADIUS proxy fabric, and mitigate or prevent RADIUS proxy forwarding loops.

This document defines two new RADIUS Packet Type Codes:

*Status-Realm-Request (TBD)

*Status-Realm-Response (TBD)

This document also defines the following RADIUS Attributes:

*Status-Realm-Response-Code (TBD)

*Max-Hop-Count (TBD)

*Server-Identifier (TBD)

2. Requirements Notation

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].

3. Terminology

The following terms are used throughout this document. Their definitions are included here for consistency and clarity.

RADIUS Request A RADIUS Request is the first message in a RADIUS message exchange. RADIUS request message types include: Access-Request, Accounting-Request, and Status-Server. This document defines a new RADIUS Request message type: Status-Realm-Request.

RADIUS Response A RADIUS Response is any RADIUS message sent in reply to a RADIUS Request. RADIUS reponse message types include: Access-Accept, Access-Challenge, Access-Reject, Accounting-Response. This document defines a new RADIUS Response message type: Status-Realm-Response.

RADIUS Instance A single device or software module that implements the RADIUS protocol.

RADIUS Client A RADIUS Client is a RADIUS Instance that sends RADIUS Request messages and recevies RADIUS Reponse messages in reply.

RADIUS Server A RADIUS Server is a RADIUS Instance that receives RADIUS Requests and sends RADIUS Response messages in reply.

Authentication Request An Authentication Request is sent to authenticate a particular user within a particular realm. The user and realm information are typically included in a User-Name Attribute [RFC2865] within the Authentication Request.

Authentication Server

An Authentication Server is a RADIUS Server that receives Access-Requests for a given RADIUS Realm, and sends Access-Access, Access-Challenge or Access-Reject messages in response. A single Authentication Server may serve more than one Authentication Realm.

Authentication Realm An Authentication Realm consists of a group of users within a single organization that can be authenticated using RADIUS. A single Authentication Realm MAY be served by more than one Authentication Server.

Target Realm The Target Realm of a RADIUS Request is the RADIUS Realm toward which the Request is directed. The Target Realm is typically contained within the "User-Name" attribute of a Request.

RADIUS Proxy A RADIUS Proxy receives RADIUS Requests and forwards then towards the Target Realm included in the RADIUS Request message. It also receives the corresponding RADIUS Respone message and fowards them back towards the RADIUS Client that originated the request. In this context forwarding a RADIUS Requst consists of generating a new RADIUS Request containing information from the original Request, and sending it to the configured next-hop RADIUS server for the Target Realm. Forwarding a RADIUS Response consists of sending it to the RADIUS Server from which the corresponding Request was received.

RADIUS Proxy Fabric A multi-hop group of inter-connected RADIUS Servers that Proxy requests among themselves towards a set of Target Realms.

RADIUS Proxy Path The RADIUS Server Path is a the set of RADIUS Servers that a RADIUS Request traverses from the first RADIUS Server that is contacted by the RADIUS Client to the final RADIUS Server that responds to the Request.

Proxy Loop A Proxy Loop may occur when two or more RADIUS Proxies are configured such that a RADIUS Request follow a circular path through the Proxy Fabric, never reaching the Target Realm. This is a pathological and potentially damaging misconfiguration.

First-Hop Server The First-Hop Server is the first RADIUS Server within a Proxy Fabric to recieve a RADIUS Request. In some cases, the First-Hop RADIUS Server may receive the request from a separate RADIUS Client. In other case, the First-Hop RADIUS

Server and the RADIUS Client may be running in a single RADIUS Instance.

Last-Hop Proxy The Last-Hop Proxy is the last RADIUS Proxy to forward a RADIUS Request before it reaches the Authentication Server. Depending on its configuraiton, the Last-Hop Proxy may or may not know that is the Last-Hop Proxy for a given RADIUS Request.

Note: It is possible for a single RADIUS instance to server in multiple roles. For example, it is common for a RADIUS Server to act as an Authentication Server for some Realms, while acting as a Proxy for other Realms. A RADIUS Proxy will, by its nature, act as a RADIUS Server for some RADIUS messages while acting as a RADIUS Client for others. The requirements in this document apply to all RADIUS instances whenever they are acting in the role to which the requirement applies.

4. Overview

This document defines two functional extensions to RADIUS: Querying the status of a remote RADIUS Realm (Status-Realm), and mitigating, detecting and preventing loops in a RADIUS Proxy forwarding loops (Proxy Loop Prevention). This section contains a short overview of each function. Detailed definitions and requirements are covered in later sections of this document.

4.1. Status-Realm Overview

Status-Realm-Request messages are sent by RADIUS Clients to to query the reachability and status of a particular Target Realm. In some cases, the Status-Realm RADIUS Client may be able to reach an Authentication Server for the Target Realm directly. In other cases, the RADIUS Client will send the initial Status-Realm request to a RADIUS Proxy, which will forward the Status-Realm-Request toward the indicated realm.

Status-Realm-Requests may be sent to the RADIUS authentication port or the RADIUS accounting port of the first-hop RADIUS server. RADIUS proxies should forward Status-Realm-Requests received on the authentication port to the authentication port of the next-hop RADIUS server. Status-Realm-Requests received on the accounting port should, similarly, be forwarded to the accounting port of the next-hop server.

When a Status-Realm-Request packet is received by an Authentication Server for the Target Realm, the Authentication Server MUST respond with a Status-Realm-Response packet.

If an intermediate RADIUS Proxy is unable to forward a Status-Realm-Request packet towards the Target Realm, either because it has no information about how to reach the Target Realm, or because there are no reachable Authentication Servers for the Target Realm, the RADIUS Proxy MUST return a Status-Realm-Response packet containing a Status-Realm-Response-Code attribute.

Status-Realm packets allow the sender to determine the reachability and status of a Authentication Realm, without requiring a direct RADIUS connection to a RADIUS Server for the Target realm, and without requiring credentials for an authorized user within that realm. This can be useful for debugging RADIUS authentication issues, identifying routing issues within a RADIUS proxy fabric, or monitoring realm availability.

Using the Max-Hop-Count attribute defined in this document, RADIUS Clients can also implement "traceroute-like" functionality, discovering a series of proxies on route to a target realm.

4.2. RADIUS Loop Prevention Overview

RADIUS Proxies are configured to know which next-hop RADIUS Server to use for a given Target Realm. There is no dynamic routing protocol or tree-spanning protocol in use, so Proxy Loops are a common occurence due to misconfiguration. These loops can be controlled or prevented using implementation-specific or operator-specific mechanisms, but it would be useful to have well-defined, common mechanism.

The Max-Hop-Count attribute described in this document can be used to mitigate the damage caused by Proxy Loops. The Max-Hop-Count attribute is set to a small integer by the RADIUS Client or First-Hop RADIUS Server. The value is deprecated each time a RADIUS message is proxied. When the Max-Hop-Count reaches zero, the request is discarded, ending the loop.

This document also defines a more effective method of detecting and preventing Proxy Forwarding Loops: RADIUS Loop Prevention. This document defines a RADIUS Server-Identifier attribute that is used to uniquely identify a RADIUS Server. When a RADIUS Proxy receives a RADIUS Request packet, it checks to see if the Request contains a Server-Identifier attribute indicating that it has already processed this packet. If so, it discards the packet. If not, it adds its own Server Identifier to the packet before forwarding it.

5. Packet Formats

This section describes the RADIUS packet formats for Status-Realm-Request and Status-Realm-Response packets. Status-Realm-Requests are

sent in the same format, whether they are sent to the authentication port or the accounting port.

5.1. Status-Realm-Request Packet

Status-Realm-Request packets reuse the RADIUS packet format, with the fields and values for those fields as defined in [RFC2865], Section 3.

A Status-Realm-Request packet MUST include a Message-Authenticator attribute, as defined in [RFC2869], section 5.14. The Message-Authenticator provides per-packet authentication and integrity protection. The Authenticator field of a Status-Realm-Request packet MUST be generated using the same method as that used for the Request Authenticator field of Access-Request packets.

A Status-Realm-Request packets MUST include a User-Name Attribute, containing the Target Realm for the Request. The 'user' portion of the User-Name SHOULD be ignored, if present.

A Status-Realm-Request message MUST also include a Max-Hop-Count attribute, as defined above.

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

Status-Realm-Request packets MUST NOT contain authentication credentials (such as User-Password, CHAP-Password, EAP-Message) or User or NAS accounting attributes (such as Acct-Session-Id, Acct-Status-Type, Acct-Input-Octets).

5.2. Status-Realm-Response Packet

Status-Realm-Response packets reuse the RADIUS packet format, with the fields and values for those fields as defined in [RFC2865], Section 3.

The Response Authenticator field of a Status-Realm-Response packet MUST be generated using the same method used for calculating the Response Authenticator of an Access-Accept or an Access-Reject sent in response to an Access-Request, with the Status-Realm-Request Request Authenticator taking the place of the Access-Request Request Authenticator.

The Status-Realm-Response packet MUST contain a Status-Realm-Response-Code attribute, as defined below, indicating the results of the Status-Realm request.

The Status-Realm-Response packet MAY contain the following attributes: Reply-Message, Message-Authenticator, Server-Information.

Note that when a server responds to a Status-Realm-Request packet, it MUST NOT send more than one Status-Realm-Response packet.

6. Max-Hop-Count Attribute

This section defines a new RADIUS attribute, Max-Hop-Count (TBD). The value of the Max-Hop-Count attribute is an integer, as defined in [RFC8044], Section 3.1. Valid values are small positive integers, 0 to 255.

This attribute is used to limit the number of RADIUS servers that will proxy a packet before it reaches its final destination. When a RADIUS server that implements the Max-Hop-Count Attribute determines that it wants to proxy a RADIUS Request to another RADIUS Server, it will check the Max-Hop-Count attribute. If the Max-Hop-Count attribute is present and the value is zero, the Request MUST NOT be forwarded and an error response SHOULD be returned, as appropriate to the request type. If the Max-Hop-Count is greater than zero, the proxy server MUST decrement the hop count by 1 before forwarding the request.

In the context of Status-Realm-Requests, this attribute can be used to implement "traceroute-like" functionality. By sending a series of Status-Realm-Requests with incremented values of Max-Hop-Count, starting with a Max-Hop-Count value of O, the RADIUS Client will receive a series of Status-Realm-Responses from the RADIUS Proxies on the Proxy Path to a given Target Realm.

When used on other types of RADIUS Request messages, this option can mitigate the damage caused by RADIUS proxy loops. It is therefore possible that a RADIUS Client or a RADIUS proxy server will support the Max-Hop-Count attribute, even if they do not support Status-Realm. When used to limit RADIUS proxy loops, it is RECOMENDED that the value of the Max-Hop-Count attribute be set to 32, by default.

If this attribute is not present on a RADIUS Request received from a RADIUS Client, the First-Hop RADIUS Server MAY add this option, setting it to the default value of 32, or to any valid, configured value.

7. Status-Realm-Response-Code Attribute

This section defines a new RADIUS attribute, Status-Realm-Response-Code (TBD). This is of type tlv, as defined in [RFC8044], section 3.13. It contains 3 sub-attributes:

*Response-Code (Type = 1)

*Hop-Count (Type = 2)

*Responding-Server (Type = 3)

Response-Code is of type 'integer', as defined in [RFC8044], Section 3.1. Exactly one Response-Code sub-attribute MUST be included in in every Status-Realm-Response-Code attribute. It will contain one of the following values:

0	The target realm is available
1	No proxy route to the target realm
2	No available servers for the target realm
3	The target realm is missing or invalid
4	Max-Hop-Count exceeded
5-255	Unspecified error, the is unreachable
257	Administratively prohibited, target realm status unknown
258	Internal error, target realm status unknown
259	Bad Status-Realm-Request, missing or invalid Target Realm
260	Bad Status-Realm-Request, missing or invalid Max-Hop-Count, Target Realm status unknown
260-511	Unspecified error, Target Realm status unknown
512+	Reserved

Hop-Count is of type 'integer'. Valid values are 0=255. The value of this sub-attribute MUST be set to the value of the Max-Hop-Count attribute in the received Status-Realm-Request. If no Max-Hop-Count is included in the Status-Realm-Request message, this sub-attribute MUST be omitted.

Responding-Server is of type 'tlv', as defined in [RFC8044], Section 3.13. This sub-attribute MUST be returned in every Status-Realm-Response attribute. The value field of this sub-attribute contains a

Server-Information Attribute for the responding server, as described below.

8. Server-Information Attribute

The Server-Information attribute is used to identify a specific RADIUS Server. It MAY be added to any RADIUS Request message to indicate that a particular RADIUS Server has processed the Request. If present in a RADIUS Request, it should be copied into the corresponding RADIUS Response. RADIUS Servers SHOULD NOT add Server-Information attributes to Response messages when processing Responses.

This attribute is also included as a sub-attribute within the Status-Realm-Response-Code attribute, defined above, to indicate which RADIUS Server has sent the Status-Realm-Response message.

This attribute is of type 'tlv', as defined in [RFC8044], Section 3.13. The value of this attribute consists of a set of sub-attributes, all of type 'tlv'. Each sub-attribute contains an identifier for a RADIUS proxy. The Proxy-Identifier MUST have at least one sub-attribute and MAY have more than one sub-attribute. If multiple sub-attributes are present, a RADIUS proxy MUST match all of the sub-attributes in order to match the identifier.

The following sub-attributes may be included in the value field of a Proxy-Information Attribute. The Type code for each sub-attribute is included in parenthesis.

*Server-Name (Type = 1)

*Server-Identifier (Type = 2)

*Hop-Count (Type = 3)

The Server-Operator is of type 'string'. It is the analogue of the Operator-Name, as defined in [RFC5580].

The Server-Identifier in an analogue of the NAS-Identifier defined in [RFC2865]. It indicates the name of this particular proxy server. This field is used to identify which server processed the Request, among those operated by the organization indicated in the Server-Operator sub-attribute.

9. Status-Realm Implementation Requirements

This section describes implementation details and requirements for RADIUS Clients and servers that support Status-Realm.

9.1. RADIUS Client Requirements

When Status-Realm-Request packets are sent from a RADIUS 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-Realm-Request packet should be generated and sent, with new Identity and Authenticator fields.

RADIUS Clients MUST include the Message-Authenticator attribute in all Status-Realm-Request packets. Failure to do so would mean that the packets could be trivially spoofed, leading to potential denial-of-service (DoS) attacks.

The RADIUS Client MUST include a User-Name attribute in the request. The "user" portion of the username SHOULD be omitted. The "realm" portion of the username is the target realm for the Status-Realm request.

RADIUS Clients that support Status-Realm-Requests SHOULD allow a user or administrator to set or configure the Count value of the Max-Hop-Count Attribute described above. If a different value is not indicated, the RADIUS Client SHOULD include a Max-Hop-Count attribute with a Count value of 32 in the Status-Realm-Request packet to prevent the possibility that Status-Realm-Requests will loop indefinitely.

The RADIUS Client MAY increment packet counters as a result of sending a Status-Realm-Resquest or receiving a Status-Realm-Response. The RADIUS 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.

RADIUS Clients MAY send Status-Realm-Request packets to the RADIUS destination ports from the same source port(s) used to send other Request packets. Other RADIUS Clients MAY choose to send Status-Realm-Request packets from a unique source port that is not used to send other Request packets.

In the case where a RADIUS Client sends a Status-Realm-Request packets from a source port also used to send other Request packets, the Identifier 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 RADIUS Client has either received a corresponding Status-Realm-Response packet or determined that the Status-Realm-Request has timed out, it may reuse the Identifier in another Request packet.

The RADIUS Client MUST validate the Response Authenticator in the Status-Realm-Response. 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.

9.2. Server Requirements

Servers SHOULD permit administrators to globally enable or disable the acceptance of Status-Realm-Request packets. The default SHOULD be that acceptance is enabled. Servers SHOULD also permit administrators to enable or disable acceptance of Status-Realm-Request packets on a per-RADIUS Client basis. The default SHOULD be that acceptance is enabled.

If a server does not support Status-Realm, or if it is configured not to respond to Status-Realm-Requests, then it MUST silently discard any Status-Realm-Requests messages that it receives. If a server receives a Status-Realm-Request packet from a RADIUS Client from which it is configured not to accept Status-Realm-Requests, then it MUST silently discard the message.

If a server supports Status-Realm, is configured to respond to Status-Realm-Requets, and receives a Status-Realm-Request packet from a permitted RADIUS Client, it MUST first validate the Message-Authenticator attribute as defined in [RFC3579], Section 3.2. Packets failing this validation MUST be silently discarded.

If the Status-Realm-Request passes Message-Authenticator validation, then the server should check if the Target Realm matches a local realm served by this Server. If it does match, the server should send a Status-Realm-Response packet indicating that status of the Target Realm, reachable or unreachable (Status-Server-Response-Code = 0 or 2).

If the Target Realm does not match a local realm, then the server should determine whether it is configured to proxy packets towards the Target Realm. If so, the server should implement the Proxy Server Requirements, below. Servers SHOULD ignore the value of the "user" portion of the User-Name attribute, if any.

Servers SHOULD NOT discard Status-Realm packets merely because they have recently sent the RADIUS Client a response packet. The query may have originated from an administrator who does not have access to the response packet stream or one who is interested in obtaining additional information about the server.

The server MAY decide to send an error response to a Status-Realm-Request packet based on local-site policy. For example, a server that is running but is unable to perform its normal duties SHOULD send a Status-Realm-Response packet indicating an internal error (Status-Server-Response-Code = 257). 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 current load.

The server MAY increment packet counters or create log entries as a result of receiving a Status-Realm-Request packet or sending a Status-Realm-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.

If the Status-Realm-Request packet includes a Max-Hop-Count attribute, that attribute (with its current value) MUST be returned in any corresponding Status-Realm-Response packet.

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 the standards-track specification [RFC2865], as it defines a new Message Type Code for packets to port 1812. This specification is not compatible with the informational document [RFC2866], as it adds a new Code (Status-Realm-Request) that is valid for port 1813.

9.3. Proxy Server Requirements

Many RADIUS servers act as RADIUS proxies, forwarding requests to other RADIUS servers. Such servers SHOULD proxy Status-Realm-Request packets to enable RADIUS Clients to determine the status of Authentication Realms that are not directly connected to the RADIUS Client.

RADIUS proxies that support Status-Realm-Requests MUST support the Max-Hop-Count attribute defined above. Before forwarding a Status-Realm-Request packet, a proxy MUST check the Max-Hop-Count Attribute. If the Max-Hop-Count attribute is present and the Count is zero (0), the proxy MUST send a Status-Realm-Response indicating that the hop count has been exceeded (Status-Server-Response-Code = 4), and MUST NOT forward the packet. If the Max-Hop-Count attribute is present, and the Count value is not zero, the proxy MUST decrement the Max-Hop-Count value before forwarding the packet.

The RADIUS proxy MUST check the "realm" portion of the User-Name attribute in the Status-Realm-Request to determine the Target Realm for the request. If the target realm is missing or malformed, the RADIUS proxy MUST send a Status-Realm-Response indicating an invalid realm (Status-Server-Response-Code = 3). If the realm is properly formed, the Status-Realm-Request packet should be proxied toward the Target Realm, using the same next-hop RADIUS server that the proxy

server would use for other request packets received on the same port.

In some cases, a RADIUS proxy may not have an available next-hop RADIUS server for the Target Realm. In that case, the RADIUS proxy server MUST send a Status-Realm-Response packet indicating that there is no proxy route to the Target Realm (Status-Server-Response-Code = 1).

In cases where a RADIUS proxy is configured to have a direct connection to the RADIUS server(s) of the Target Realm, but is configured not to forward Status-Realm-Request packets to the target server(s), the proxy MAY use other methods to determine the status of the Target Realm (such as Status-Server packets or recent Access-Request state information), and send a Status-Realm-Response indicating the determined state of the Target Realm (Status-Server-Response-Code = 0 or 2). If the proxy is configured not to forward Status-Realm-Request packet to the Target Realm and does not have other methods to detect the status of the Target Realm, it SHOULD return a Status-Realm-Response packet indicating that the request is administrative prohibited (Status-Server-Response-Code = 257).

If the Status-Realm-Request packet includes a Max-Hop-Count attribute, that attribute (with its current value) MUST be returned in any corresponding Status-Realm-Response packet.

10. Status-Realm Implementation Status

There is an initial implementation of Status-Realm available here:

https://github.com/alandekok/freeradius-server/tree/Status-Realm

10.1. Status-Realm Message Exchange Examples

Message exchange examples are TBD.

11. Proxy Loop Detection Implementation Requirements

This section describes implementation details and requirements for RADIUS Clients, Servers and Proxies that support Proxy Loop Detection.

11.1. Server Requirements

A RADIUS Server that implements Proxy Loop Prevention add its own Server-Information Attribute to any RADIUS message that it generates, including RADIUS Response messages. It MUST also copy all Server-Information atributes from a received RADIUS Request into any RADIUS Response that it generates in reply to that Request.

11.2. Proxy Requirements

A RADIUS Proxy that implements the Loop Prevention mechanism defined in this document MUST be configured with information to populate a Server-Information attribute, and matching criteria to determine if a Server-Information attribute in an incoming request indicates the existence of a Proxy Loop.

Before forwarding a RADIUS Request towards the Target Realm, a RADIUS Proxy that implements Proxy Loop Prevention MUST examine each of the Server-Information attributes included in the Request message to determine whether the message is caught in a Proxy Loop. If so, the Proxy should discard the message. If a Proxy Loop is not detected, the RADIUS Proxy MUST add its own Server-Information attribute to any RADIUS Request that they forward toward the Target Realm.

12. Proxy Loop Detection Implementation Status

The Proxy Loop Detection mechanism is similar to RADIUS Vendor-Specific attribute used today to detect RADIUS Proxy Loops. Unlike the Vendor-Specific attributes in use today, this mechanism includes server information within a single, globally-defrined attribute, rather than requiring that a unique vendor identifiers be allocated for each RADIUS Server operator.

12.1. Loop Detection Message Exchange Examples

Message exchange examples are TBD.

13. Management Information Base (MIB) Considerations

Status-Realm-Request packets are sent to the defined RADIUS ports, so 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 radiusAccServTotalUnknownTypes. Implementations not supporting Status-Realm-Requests or implementations that are configured not to respond to Status-Realm-Request packets MUST use these counters to track received Status-Realm packets.

If, however, Status-Realm-Requests are 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-Realm-Request or Status-Realm-Response packets. That is, when a server fully implements Status-Realm, the counters defined in [RFC4669] and [RFC4671] MUST be unaffected by the transmission or reception of packets relating to Status-Realm-Requests.

If a server supports Status-Realm-Request and the [RFC4669] or [RFC4671] MIB modules, then it SHOULD also support vendor-specific MIB extensions dedicated solely to tracking Status-Realm-Request and Status-Realm-Response packets. Any definition of the server MIB modules for Status-Realm-Requests is outside of the scope of this document.

14. Interaction with RADIUS Client MIB Modules

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

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

15. Table of Attributes

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

Status-Realm-Request	Status-Realm-Response		
1	1	1	User-Name
0	0	2	User-Password
0	0	3	CHAP-Password
0-1	0	4	NAS-IP-Address (Note 1)
0	0+	18	Reply-Message
0+	0+	26	Vendor-Specific
0-1	0	32	NAS-Identifier (Note 1)
0	0	79	EAP-Message
1	0-1	80	Message-Authenticator
0-1	0	95	NAS-IPv6-Address (Note 1)
0	1	(TBD)	Status-Realm-Response-Code
1	0	(TBD)	Max-Hop-Count
0+	0+	(TBD)	Server-Information
0	0	103-121	Digest-*

Note 1: Status-Realm-Request packet 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 table entries included above:

0	This attribute MUST NOT be present in packet.
0+	Zero or more instances of this attribute MAY be present in the packet.
0-1	Zero or one instance of this attribute MAY be present in the packet.
1	Exactly one instance of this attribute MUST be present in the packet.

16. IANA Considerations

This document defines the Status-Realm-Request (TBD) and the Status-Realm-Response (TBD) RADIUS Packet Type Codes, both of which should be assigned by IANA from the Unassigned block of RADIUS Packet Type Codes.

This document defines three new RADIUS attributes, Max-Hop-Count (TBD) and Status-Realm-Response-Code (TBD) and Server-Identifier (TBD), which should be assigned by IANA from an Unassigned block of RADIUS Attribute Types, such as the Unassigned block for Extended-Attribute-1.

This document also defines two new Protocol Registries that need to be created: "Values for RADIUS Attribute (TBD), Status-Realm-Response-Code" and "Valies for RADIUS Attribute (TBD), Server-Identifier". Initial values for these registries are defined above.

17. Security Considerations

Status-Realm-Request packets are similar to Access-Request packets, and are therefore subject to the same security considerations as described in [RFC2865], Section 8. Status-Realm packets also use the Message-Authenticator attribute, and are therefore subject to the same security considerations as [RFC3579], Section 4.

We reiterate that all Status-Realm-Request packets MUST contain a Message-Authenticator. Servers not checking the Message-Authenticator attribute could respond to Status-Realm packets from an attacker, potentially enabling a reflected DoS attack onto a real RADIUS Client.

Where this document differs from [RFC2865] is that it defines a new request/response method in RADIUS: the Status-Realm-Request and Status-Realm-Response. The Status-Realm-Request is similar to the

previously described and widely implemented Status-Server message [RFC5997], and no additional security considerations are known to relate to the implementation or use of Status-Server. This option differs from Status-Server because it is forwarded through proxies, so it can be sent to a RADIUS Server that does not have a direct connection to the Status-Realm RADIUS Client. However, Access-Request packets are also forwarded, and there should be no additional attacks other than those incurred by forwarding Status-Realm-Request packets.

Attacks on cryptographic hashes are well known [RFC4270] and getting better with time. RADIUS uses the MD5 hash [RFC1321] for packet authentication and attribute obfuscation. There are ongoing efforts in the IETF to analyze and address these issues for the RADIUS protocol.

Security Considerations for Loop Prevention are TBD.

18. Acknowledgements

This document was written using xml2rfc, as described in [RFC7991]

Some of the sections in this document were adapted from the description of the Status-Server RADIUS Packet Type Code in [RFC5997].

19. References

19.1. Normative References

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

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

[RFC8044] DeKok, A., "Data Types in RADIUS", RFC 8044, DOI 10.17487/RFC8044, January 2017, <https://www.rfc-editor.org/info/rfc8044>.

19.2. Informative References

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <https://www.rfc-editor.org/info/rfc1321>.

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

[RFC2869] Rigney, C., Willats, W., and P. Calhoun, "RADIUS Extensions", RFC 2869, DOI 10.17487/RFC2869, June 2000, <https://www.rfc-editor.org/info/rfc2869>.

[RFC3579] Aboba, B. and P. Calhoun, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", RFC 3579, DOI 10.17487/RFC3579, September 2003, <https://www.rfc-editor.org/info/rfc3579>.

[RFC4270] Hoffman, P. and B. Schneier, "Attacks on Cryptographic Hashes in Internet Protocols", RFC 4270, DOI 10.17487/RFC4270, November 2005, <https://www.rfc-editor.org/info/rfc4270>.

[RFC4668] Nelson, D., "RADIUS Authentication Client MIB for IPv6", RFC 4668, DOI 10.17487/RFC4668, August 2006, <https://www.rfc-editor.org/info/rfc4668>.

[RFC4669] Nelson, D., "RADIUS Authentication Server MIB for IPv6", RFC 4669, DOI 10.17487/RFC4669, August 2006, <https://www.rfc-editor.org/info/rfc4669>.

[RFC4670] Nelson, D., "RADIUS Accounting Client MIB for IPv6", RFC 4670, DOI 10.17487/RFC4670, August 2006, <https://www.rfc-editor.org/info/rfc4670>.

[RFC4671] Nelson, D., "RADIUS Accounting Server MIB for IPv6", RFC 4671, DOI 10.17487/RFC4671, August 2006, <https://www.rfc-editor.org/info/rfc4671>.

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

[RFC5997] DeKok, A., "Use of Status-Server Packets in the Remote Authentication Dial In User Service (RADIUS) Protocol", RFC 5997, DOI 10.17487/RFC5997, August 2010, <https://www.rfc-editor.org/info/rfc5997>.

[RFC7991] Hoffman, P., "The "xml2rfc" Version 3 Vocabulary", RFC 7991, DOI 10.17487/RFC7991, December 2016, <https://www.rfc-editor.org/info/rfc7991>.

Authors' Addresses

Margaret Cullen
Painless Security

Phone: +1 (781)405-7464
Email: margaret@painless-security.com

Alan DeKok
FreeRADIUS

Email: aland@freeradius.org

Mark Donnelly
Painless Security

Phone: +1 (857)928-5967
Email: mark@painless-security.com

Josh Howlett
Federated Solutions

Phone: +44 (0)7510 666 950
Email: josh@federated-solutions.com