Internet Engineering Task Force (IETF)
Request for Comments: 7930
Updates: 6613
Category: Experimental
ISSN: 2070-1721

S. Hartman
Painless Security

August 2016

Larger Packets for RADIUS over TCP

Abstract

The RADIUS-over-TLS experiment described in RFC 6614 has opened RADIUS to new use cases where the 4096-octet maximum size limit of a RADIUS packet proves problematic. This specification extends the RADIUS-over-TCP experiment (RFC 6613) to permit larger RADIUS packets. This specification compliments other ongoing work to permit fragmentation of RADIUS authorization information. This document registers a new RADIUS code, an action that required IESG approval.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The experiment with Remote Authentication Dial-In User Service (RADIUS) over Transport Layer Security (TLS) [RFC6614] provides strong confidentiality and integrity for RADIUS [RFC2865]. This enhanced security has opened new opportunities for using RADIUS to convey additional authorization information. As an example, [RFC7833] describes a mechanism for using RADIUS to carry Security Assertion Markup Language (SAML) messages in RADIUS. Many attributes carried in these SAML messages will require confidentiality or integrity such as that provided by TLS.

These new use cases involve carrying additional information in RADIUS packets. The maximum packet length of 4096 octets is proving insufficient for some SAML messages and for other structures that may be carried in RADIUS.

One approach is to fragment a RADIUS message across multiple packets at the RADIUS layer. RADIUS fragmentation [RFC7499] provides a mechanism to split authorization information across multiple RADIUS messages. That mechanism is necessary in order to split authorization information across existing unmodified proxies.

However, there are some significant disadvantages to RADIUS fragmentation. First, RADIUS is a lock-step protocol, and only one fragment can be in transit at a time as part of a given request. Also, there is no current mechanism to discover the Path Maximum Transmission Unit (PMTU) across the entire path that the fragment will travel. As a result, fragmentation is likely both at the RADIUS layer and at the transport layer. When TCP is used, much better transport characteristics can be achieved by fragmentation only at the TCP layer. This specification provides a mechanism to achieve these better transport characteristics when TCP is used. As part of this specification, a new RADIUS code is registered.

This specification is published as an Experimental specification because the TCP extensions to RADIUS are currently experimental. The need for this specification arises from operational experience with the TCP extensions. However, this specification introduces no new experimental evaluation criteria beyond those in the base TCP specification; this specification can be evaluated along with that one for advancement on the Standards Track.

1.1. 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].

2. Changes to Packet Processing

The maximum length of a RADIUS message is increased from 4096 to 65535. A RADIUS Server implementing this specification MUST be able to receive a RADIUS packet of maximum length. Servers MAY have a maximum size over which they choose to return an error, as discussed in Section 5, rather than processing a received packet; this size MUST be at least 4096 octets.

Clients implementing this specification MUST be able to receive a RADIUS packet of maximum length; that is, clients MUST NOT close a TCP connection simply because a large packet is sent over it. Clients MAY include the Response-Length attribute defined in Section 6 to indicate the maximum size of a packet that they can successfully process. Clients MAY silently discard a packet greater than some configured size; this size MUST be at least 4096 octets. Clients MUST NOT retransmit an unmodified request whose response is larger than the client can process, as subsequent responses will likely continue to be too large.

Proxies MUST be able to receive a RADIUS packet of maximum length without closing the TCP connection. Proxies SHOULD be able to process and forward packets of maximum length. When a proxy receives a request over a transport with a 4096-octet maximum length and the proxy forwards that request over a transport with a larger maximum length, the proxy MUST include the Response-Length attribute with a value of 4096.

2.1. Status-Server Considerations

This section extends processing of Status-Server messages as described in Sections 4.1 and 4.2 of [RFC5997].

Clients implementing this specification SHOULD include the Response-Length attribute in Status-Server requests. Servers are already required to ignore unknown attributes received in this message. By including the attribute, the client indicates how large of a response it can process to its Status-Server request. It is very unlikely that a response to Status-Server is greater than 4096 octets. However, the client also indicates support for this specification, which triggers the server behavior below.

If a server implementing this specification receives a Response-Length attribute in a Status-Server request, it MUST include a Response-Length attribute indicating the maximum size request it can process in its response to the Status-Server request.

3. Forward and Backward Compatibility

An implementation of [RFC6613] will silently discard any RADIUS packet larger than 4096 octets and will close the TCP connection. This section provides guidelines for interoperability with these implementations. These guidelines are stated at the SHOULD level. In some environments, support for large packets will be important enough that roaming or other agreements will mandate their support. In these environments, all implementations might be required to support this specification, thus removing the need for interoperability with RFC 6613. It is likely that these guidelines will be relaxed to the MAY level and support for this specification made a requirement if RADIUS over TLS and TCP are moved to the Standards Track in the future.

Clients SHOULD provide configuration for the maximum size of a request sent to each server. Servers SHOULD provide configuration for the maximum size of a response sent to each client. If dynamic discovery mechanisms are supported, configuration SHOULD be provided for the default maximum size of RADIUS packets sent to clients and servers. If an implementation provides more granular configuration for some classes of dynamic resources, then the implementation SHOULD also provide configuration of default maximum packet sizes at the same granularity. As an example, an implementation that provided granular configuration for resources using a particular trust anchor or belonging to a particular roaming consortium SHOULD provide default packet size configuration at the same granularity.

If a client sends a request larger than 4096 octets and the TCP connection is closed without a response, the client SHOULD treat the request as if a "Request Too Big" error (Section 5) specifying a maximum size of 4096 is received. Clients or proxies sending multiple requests over a single TCP connection without waiting for responses SHOULD implement capability discovery as discussed in Section 3.2.

By default, a server SHOULD NOT generate a response larger than 4096 octets. The Response-Length attribute MAY be included in a request to indicate that larger responses are acceptable. Other attributes or configurations MAY be used as an indicator that large responses are likely to be acceptable.

A proxy that implements both this specification and RADIUS fragmentation [RFC7499] SHOULD use RADIUS fragmentation when the following conditions are met:

1. A RADIUS packet is being forwarded towards a next hop whose configuration does not support a packet that large.

2. RADIUS fragmentation can be used for the packet in question.

3.1. Rationale

The interoperability challenge appears at first significant. This specification proposes to introduce behavior where new implementations will fail to function with existing implementations.

However, these capabilities are introduced to support new use cases. If an implementation has 10000 octets of attributes to send, it cannot, in general, trim down the response to something that can be sent. Under this specification, a large packet would be generated that will be silently discarded by an existing implementation. Without this specification, no packet is generated because the required attributes cannot be sent.

The biggest risk to interoperability would be if requests and responses are expanded to include additional information that is not strictly necessary. So, avoiding creating situations where large packets are sent to existing implementations is mostly an operational matter. Interoperability is most impacted when the size of packets in existing use cases is significantly increased and least impacted when large packets are used for new use cases where the deployment is likely to require updated RADIUS implementations.

There is a special challenge for proxies or clients with a high request volume. When an implementation of RFC 6613 receives a packet that is too large, it closes the connection and does not respond to any requests in process. Such a client would lose requests and might find it difficult to distinguish "Request Too Big" situations from other failures. In these cases, the discovery mechanism described in Section 3.2 can be used.

Also, RFC 6613 is an experiment. Part of running that experiment is to evaluate whether additional changes are required to RADIUS. A lower bar for interoperability should apply to changes to Experimental protocols than Standard protocols.

This specification provides good facilities to enable implementations to understand packet size when proxying to/from Standards Track UDP RADIUS.

3.2. Discovery

As discussed in Section 2.1, a client MAY send a Status-Server message to discover whether an authentication or accounting server supports this specification. The client includes a Response-Length attribute; this signals the server to include a Response-Length attribute indicating the maximum packet size the server can process. In this one instance, Response-Length indicates the size of a request that can be processed rather than a response.

4. Protocol-Error Code

This document defines a new RADIUS code, 52, called Protocol-Error. This packet code may be used in response to any request packet, such as Access-Request, Accounting-Request, CoA-Request, or Disconnect-Request. It is a response packet sent by a server to a client. The packet indicates to the client that the server is unable to process the request for some reason.

A Protocol-Error packet MUST contain an Original-Packet-Code attribute, along with an Error-Cause attribute. Other attributes MAY be included if desired. The Original-Packet-Code contains the code from the request that generated the protocol error so that clients can disambiguate requests with different codes and the same ID. Regardless of the original packet code, the RADIUS Server calculates the Message-Authenticator attribute as if the original packet were an Access-Request packet. The identifier is copied from the original request.

Clients processing Protocol-Error MUST ignore unknown or unexpected attributes.

This RADIUS code is hop by hop. Proxies MUST NOT forward a Protocol-Error packet they receive.

5. Too Big Response

When a RADIUS Server receives a request that is larger than can be processed, it generates a Protocol-Error response as follows:

The code is Protocol-Error.

The Response-Length attribute MUST be included and its value is the maximum size of request that will be processed.

The Error-Cause attribute is included with a value of 601.

The Original-Packet-Code attribute is copied from the request.

Clients will not typically be able to adjust and resend requests when this error is received. In some cases, the client can fall back to RADIUS fragmentation. In other cases, this code will provide for better client error reporting and will avoid retransmitting requests guaranteed to fail.

6. IANA Considerations

A new RADIUS Packet Type Code is registered in the "RADIUS Packet Type Codes" registry discussed in Section 2.1 of RFC 3575 [RFC3575]. The name is "Protocol-Error" and the code is 52.

The following RADIUS attribute Type values [RFC3575] are assigned. The assignment rules in Section 10.3 of [RFC6929] are used.

Name	Attribute	Description
Response-Length	241.3	An attribute of type "integer" per Section 5 of RFC 2865 containing maximum response length.
Original-Packet-Code	241.4	An integer attribute containing the code from a packet resulting in a Protocol-Error response.

The Response-Length attribute MAY be included in any RADIUS request. In this context, it indicates the maximum length of a response the client is prepared to receive. Values are between 4096 and 65535. The attribute MAY also be included in a response to a Status-Server message. In this case, the attribute indicates the maximum size RADIUS request that is permitted.

A new Error-Cause value is registered in the "Values for RADIUS Attribute 101, Error-Cause Attribute" registry at <http://www.iana.org/assignments/radius-types> for "Response Too Big" with value 601. The range of valid values for the Error-Cause attribute in the "Values for RADIUS Attribute 101, Error-Cause Attribute" registry originally defined in RFC 5176 are extended. Two new ranges are defined:

6xx	fatal errors committed by a RADIUS server
7xx	fatal errors committed by a RADIUS client

7. Security Considerations

This specification updates [RFC6613] and will be used with [RFC6614]. When used over plain TCP, this specification creates new opportunities for an on-path attacker to impact availability. These attacks can be entirely mitigated by using TLS. If these attacks are acceptable, then this specification can be used over TCP without TLS.

8. References

8.1. Normative References

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

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

[RFC3575] Aboba, B., "IANA Considerations for RADIUS (Remote Authentication Dial In User Service)", RFC 3575, DOI 10.17487/RFC3575, July 2003, <http://www.rfc-editor.org/info/rfc3575>.

[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, <http://www.rfc-editor.org/info/rfc5997>.

[RFC6613] DeKok, A., "RADIUS over TCP", RFC 6613, DOI 10.17487/RFC6613, May 2012, <http://www.rfc-editor.org/info/rfc6613>.

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

[RFC6929] DeKok, A. and A. Lior, "Remote Authentication Dial In User Service (RADIUS) Protocol Extensions", RFC 6929, DOI 10.17487/RFC6929, April 2013, <http://www.rfc-editor.org/info/rfc6929>.

8.2. Informative References

[RFC7499] Perez-Mendez, A., Ed., Marin-Lopez, R., Pereniguez-Garcia, F., Lopez-Millan, G., Lopez, D., and A. DeKok, "Support of Fragmentation of RADIUS Packets", RFC 7499, DOI 10.17487/RFC7499, April 2015, <http://www.rfc-editor.org/info/rfc7499>.

[RFC7833] Howlett, J., Hartman, S., and A. Perez-Mendez, Ed., "A RADIUS Attribute, Binding, Profiles, Name Identifier Format, and Confirmation Methods for the Security Assertion Markup Language (SAML)", RFC 7833, DOI 10.17487/RFC7833, May 2016, <http://www.rfc-editor.org/info/rfc7833>.

Acknowledgements

Sam Hartman’s time on this document was funded by JANET as part of Project Moonshot.

Alan DeKok provided valuable review and text for the Protocol-Error packet code.

Alejandro Perez Mendez provided valuable review comments.

Author’s Address

Sam Hartman
Painless Security

Email: hartmans-ietf@mit.edu