RADEXT Working Group
Internet-Draft
Updates: 2865, 2866, 5176, 7585 (if approved)
Intended status: Standards Track
Expires: 14 September 2026

A. DeKok
InkBridge Networks

13 March 2026

Deprecating Insecure Practices in RADIUS
draft-ietf-radext-deprecating-radius-09

Abstract

RADIUS crypto-agility was first mandated as future work by RFC 6421. The outcome of that work was the publication of RADIUS over TLS (RFC 6614) and RADIUS over DTLS (RFC 7360) as experimental documents. Those transport protocols have been in wide-spread use for many years in a wide range of networks, and have recently been standardized in [I-D.ietf-radext-radiusdtls-bis]. TLS has proven to be a useful replacment for UDP (RFC 2865) and TCP (RFC 6613) transports. With that knowledge, the continued use of insecure transports for RADIUS has serious and negative implications for privacy and security.

The publication of the "BlastRADIUS" exploit has also shown that RADIUS security needs to be updated. It is no longer acceptable for RADIUS to rely on MD5 for security. It is no longer acceptable to send device or location information in clear text across the wider Internet. This document therefore deprecates many insecure practices in RADIUS, and mandates support for secure TLS-based transport layers. Related security issues with RADIUS are discussed, and recommendations are made for practices which increase both security and privacy.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-radext-deprecating-radius/.

Discussion of this document takes place on the RADEXT Working Group mailing list (mailto:radext@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/radext/. Subscribe at https://www.ietf.org/mailman/listinfo/radext/.

Source for this draft and an issue tracker can be found at https://github.com/freeradius/deprecating-radius.git.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

With the publication of [I-D.ietf-radext-radiusdtls-bis], the [RFC6421] work on crypto-agility is nearing completion. The RADIUS protocol now has a secure transport which is standards-track. This specification therefore completes the work of [RFC6421] by deprecating insecure uses of RADIUS, including RADIUS/UDP and RADIUS/TCP.

This specification mandates new behavior for RADIUS to address those issues, most notably the BlastRADIUS vulnerability [BLAST]. In the interest of keeping this document simple, these mandates are given with minimal explanation.

1.1. Overview of RADIUS Security and Privacy

The reader is directed to [I-D.dekok-radext-review-radius] for a detailed review of of the security and privacy issues in RADIUS. That document explains the background behind the mandates in this document, which provides design motivation that is missing from this specification. In the interest of providing some justification in this document, we provide a brief overview here.

The RADIUS protocol [RFC2865] was first standardized in 1997, though its roots go back much earlier to 1993. The protocol uses MD5 [RFC1321] to authenticate some packets types, and to obfuscate certain attributes such as User-Password. As originally designed, Access-Request packets were entirely unauthenticated, and could be trivially spoofed ([RFC2869], Section 7.1 and [RFC3579], Section 4.3.2).

The insecurity of MD5 was first noted in relation to RADIUS in 1996 on the IETF RADIUS working group mailing list [MD5-1996], which also discussed using an HMAC construct to increase security. While it was common knowledge at the time, the earliest documented concern being raised about Access-Request packets spoofing was on the RADIUS working group mailing list in 1998 [DATTACK]. There was substantial further discussions about the lack of integrity checks on that list over the next few years. The outcome of that discussion was the definition of Message-Authenticator as an optional HMAC-based attribute in [RFC2869], Section 5.14.

The packet forgery issue was further discussed in 2004 in [RFC3579], Section 4, and again in 2007 in [RFC5080], Section 2.2.2. The state of MD5 security was again discussed in [RFC6151], which states in Section 2:

> MD5 is no longer acceptable where collision resistance is required such as digital signatures.

That statement led to RADIUS security being reviewed in [RFC6421], Section 3, but no protocol changes were made at that time. The outcome of that review was the text in the remainder of [RFC6421], which created crypto-agility requirements for RADIUS. The work of [RFC6421] was completed in [I-D.ietf-radext-radiusdtls-bis].

Another issue with RADIUS is that most information (but not passwords) is sent "in the clear". This practice has obvious privacy implications. The data which is publicly available in RADIUS includes information such as names, MAC addresses, locations, etc., which allows individuals to be tracked with minimal effort. The reader is refered to [RFC6973], and specifically to [RFC6973], Section 5 for detailed discussion, and to [RFC6973], Section 6 for recommendations on threat mitigations.

It is no longer acceptable for RADIUS to rely on MD5 for security. It is no longer acceptable to send device or location information in clear text across the wider Internet. This document therefore deprecates all insecure uses of RADIUS, and mandates the use of secure TLS-based transport layers.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

* RADIUS

  The Remote Authentication Dial-In User Service protocol, as defined in [RFC2865], [RFC2866], and [RFC5176] among others.

* RADIUS/UDP

  RADIUS over the User Datagram Protocol as define above.

* RADIUS/TCP

  RADIUS over the Transport Control Protocol [RFC6613]

* RADIUS/TLS

  RADIUS over the Transport Layer Security protocol [RFC6614]

* RADIUS/DTLS

  RADIUS over the Datagram Transport Layer Security protocol [RFC7360]

* RadSec

  Either RADIUS/TLS or RADIUS/DTLS.

* TLS

  the Transport Layer Security protocol.

* NAS

  Network Access Server, which is a RADIUS client.

* MS-CHAP

  Microsoft Challenge-Handshake authentication, as defined for MS-CHAPv1 in [RFC2433], MS-CHAPv2 in [RFC2759], and EAP-MSCHAPv2 [KAMATH]

In order to be consistent with the terminology of [RFC2865], this document describes the Request Authenticator, Response Authenticator, and Message-Authenticator as "signing" the packets. This terminology is not consistent with modern cryptographic terms, but using other terminology could be misleading to long-term RADIUS implementers. The reader is assured that no modern cryptographic methods are used with RADIUS/UDP.

3. Deprecating Insecure Practices

The solution to an insecure protocol which uses thirty year-old cryptography is to deprecate the use insecure cryptography, and to mandate modern cryptographic transport. This section deprecates insecure transports, mandates the use of secure transports, officially deprecates MS-CHAP nearly two decades after it was broken, and finally closes out the [RFC6421] crypto-agility requirements for RADIUS.

3.1. RADIUS/UDP and RADIUS/TCP are Deprecated

RADIUS/UDP and RADIUS/TCP MUST NOT be used outside of secure networks. A secure network is one which is believed to be safe from eavesdroppers, attackers, etc. For example, if IPsec is used between two systems, then those systems may use RADIUS/UDP or RADIUS/TCP over the IPsec connection.

However, administrators should not assume that such uses are always secure. An attacker who breaks into a critical system could use that access to view RADIUS traffic, and thus be able to attack it. Similarly, a network misconfiguration could result in the RADIUS traffic being sent over an insecure network.

Neither the RADIUS client nor the RADIUS server would be aware of any network misconfiguration (e.g. such as could happen with IPsec). Neither the RADIUS client nor the RADIUS server would be aware of any attacker snooping on RADIUS/UDP or RADIUS/TCP traffic.

In contrast, when RadSec is used, the RADIUS endpoints are aware of all security issues, and can enforce any necessary security policies.

Any use of RADIUS/UDP and RADIUS/TCP is therefore NOT RECOMMENDED, even when the underlying network is believed to be secure.

3.2. Secure Transports are Mandated

All systems which send RADIUS packets outside of secure networks MUST use either RadSec, or transport-layer security such as IPSec. For operational and security reasons, it is RECOMMENDED to use RadSec instead of IPsec.

Unlike RadSec, use of IPsec means that the RADIUS server is unaware of transport-layer security. Any problem with IPsec such as configuration issues, negotiation or re-keying problems are typically presented to the RADIUS servers as 100% packet loss. These issues may occur at any time, independent of any changes to a RADIUS application using that transport. Further, network misconfigurations which remove all security are completely transparent to the RADIUS application. A transport layer misconfiguration can cause packets can be sent over an insecure link, and the RADIUS server will be unaware of the failure of security at the transport layer.

In contrast, RadSec gives the RADIUS application completely knowledge and control over transport-layer security. The failure cases around RadSec are therefore often clearer, easier to diagnose and faster to resolve than failures in IPsec. For example, a failed TLS connection may return a "connection refused" error to the application, or any one of many TLS errors indicating which exact part of the TLS conversion failed during negotiation.

3.3. MS-CHAP is Deprecated

MS-CHAP (as defined for v1 in [RFC2433], v2 in [RFC2759], and EAP-MSCHAPv2 [KAMATH]) have major design flaws, as discussed in [I-D.dekok-radext-review-radius], Part TBD. MS-CHAP MUST NOT be used in any situation where it is not protected by RadSec. MS-CHAP MUST NOT be sent over RADIUS/UDP or RADIUS/TCP, unless that data is protected by a a secure transport layer such as IPSec.

As packets can be proxied outside of a secure transport, MS-CHAP MUST NOT be sent over RADIUS/UDP or RADIUS/TCP. For authentication protocols such as EAP, MS-CHAP methods MUST NOT be used outside of a secure tunnel such as PEAP or TTLS. This recommendation includes EAP-MSCHAPv2 [KAMATH].

Due to the [ASLEAP] attack, implementers and administrators MUST treat MS-CHAP as being equivalent to sending passwords in the clear, without any encryption or obfuscation. That is, the User-Password attribute with the [RFC2865], Section 5.2 obfuscation is substantially more secure than MS-CHAP.

Existing RADIUS client implementations which originate Access-Request packets SHOULD therefore deprecate all uses of MS-CHAP. Clients SHOULD forbid new configurations from enabling MS-CHAP authentication. New RADIUS clients MUST NOT implement MS-CHAPv1, MS-CHAPv2, or EAP-MSCHAPv2.

These prohibitions do not apply to EAP methods which transport MS-CHAP inside of a TLS tunnel.

3.4. Crypto-Agility

The crypto-agility requirements of [RFC6421] are defined in [RFC6614] Appendix C, and in Section 10.1 of [RFC7360]. For clarity, we repeat the text of [RFC7360] here, with some minor modifications to update references, without changing the content.

Section 4.2 of [RFC6421] makes a number of recommendations about security properties of new RADIUS proposals. All of those recommendations are satisfied by using RadSec. as the transport layer.

Section 4.3 of [RFC6421] makes a number of recommendations about backwards compatibility with RADIUS. [RFC7360] Section 3 addresses these concerns in detail.

Section 4.4 of [RFC6421] recommends that change control be ceded to the IETF, and that interoperability is possible. Both requirements are satisfied.

Section 4.5 of [RFC6421] requires that the new security methods apply to all packet types. This requirement is satisfied by allowing RadSec to be used for all RADIUS traffic. In addition, [RFC7360] Section 3, addresses concerns about documenting the transition from legacy RADIUS to crypto-agile RADIUS.

Section 4.6 of [RFC6421] requires automated key management. This requirement is satisfied by using TLS or DTLS key management.

This specification finalizes the work began in [RFC6421].

3.4.1. All new Cryptographic work in RADIUS is forbidden

This document mandates that new RADIUS specifications MUST NOT introduce new cryptographic primitives to authenticate packets (e.g. [RFC6218]). Specifications MUST NOT introduct new cryptographic primitives to obfuscate attributes (e.g. User-Password in [RFC2865], Section 5.2, and Tunnel-Password in [RFC2868], Section 3.5).

RADIUS-specific cryptographic methods which exist at the time of the publication of this document MAY continue to be used for historical compatibility. However, all new cryptographic work which is specific to the RADIUS protocol is forbidden.

As the BlastRADIUS attack shows ([BLAST] and [I-D.dekok-radext-review-radius], Part TBD), RADIUS/UDP security is inadequate for modern networks. The solution is not to fix RADIUS/UDP. The solution is to deprecate it entirely, and to instead use modern cryptographic methods which provide security and privacy.

All new security and privacy requirements in RADIUS therefore MUST be provided by a secure transport layer such as TLS or IPsec. As noted above, simply using IPsec is not always enough, as the use (or not) of IPsec is unknown to the RADIUS application.

The restriction which forbids new cryptographic work in RADIUS does not apply to the data being transported in RADIUS attributes. For example, a new authentication method could use new cryptographic methods, and would be permitted to be transported in RADIUS. This authentication method could be a new EAP method, or any other data which is opaque to the RADIUS transport. In those cases, RADIUS serves as a transport layer for the authentication method. The authentication data is treated as opaque data for the purposes of Access-Request, Access-Challenge, etc. packets.

For those situations, there is no need for the RADIUS protocol to define any new cryptographic methods in order to transport the data. As a result, those new cryptographic methods do not require chantes to the RADIUS protocol, and are not affected by this prohibition.

4. Securing Access-Request Packets

Despite the above mandates to use secure transports for RADIUS, the reality is that RADIUS/UDP is likely to remain in wide-spread use for many years. It is therefore important to update RADIUS/UDP and RADIUS/TCP in order to secure them from the BlastRADIUS attack ([BLAST]).

These updates require a number of changes to both clients and servers in order for all possible attack vectors to be closed. Implementing only some of these mitigations means that an attacker could bypass those partial mitigations, and still perform the attack.

This section outlines the mitigations which protect RADIUS/UDP and RADIUS/TCP systems from the BlastRADIUS attack. These mitigations MUST be applied to RADIUS/UDP and RADIUS/TCP, and MUST NOT be applied to RADIUS/TLS or RADIUS/DTLS.

Unless otherwise noted, the mitigations here apply only to Access-Request packets, and to responses to Access-Request (i.e. Access-Accept, Access-Reject, Access-Challenge, and Protocol-Error packets). All behavior involving other types of request and response packets MUST remain unchanged from legacy RADIUS.

The mitigation methods outlined here allow updated systems to protect themselves from the attack, while ensuring that they are interoperable with legacy systems. That is, there is no global “flag day” required for these changes to take effect. Systems which implement these recommendations are fully compatible with legacy RADIUS implementations, and can help to protect those legacy implementations. However, when these mitigations are not fully implemented, systems may still be vulnerable to the attack.

Note that when the RADIUS system does not do proxying, the attack can be mitigated simply by upgrading the RADIUS server, so that it sends Message-Authenticator as the first attribute in all responses to Access-Request packets. However, the goal of this specification is to fix all architectures supported by RADIUS systems, rather than only a limited subset. We therefore mandate new behavior for all RADIUS clients and servers, while acknowledging that some organizations may choose to not configure all of the new functionality.

For overall network security and good practice, is RECOMMENDED that all RADIUS clients and servers be upgraded to use the new software which contains the mitigations, and also be configured with the highest level of security. Doing so will ensure that configuration mistakes on one system will not reintroduce the vulnerability.

4.1. New Configuration Flags

The goal of these flags is to secure the RADIUS protocol without preventing communication between clients and servers, even when only one party has been upgraded. These flags are designed to allow a gradual migration from both parties using legacy RADIUS, to fully upgraded and secured systems with all of the mitigations in place.

Clients and servers MUST implement the new configuration flags defined below when RADIUS/UDP or RADIUS/TCP is used. These flags MUST NOT be exposed in any administrative interface or be examined by code when RADIUS/DTLS or RADIUS/TLS is used.

The behavior and meaning of these flags will be discussed in the following sections. Introducing these flags before discussing their meaning makes the subsequent discussion simpler and easier to understand.

Clients MUST have a per-server boolean configuration flag, which we call “require Message-Authenticator”.

Servers MUST have a per-client boolean configuration flag, which we call “require Message-Authenticator”.

Servers MUST have a per-client boolean configuration flag, which we call “limit Proxy-State”.

The default value of all three configuration flags SHOULD be "false", in order to maintain compatibility with legacy clients. Implementers and Vendors MAY choose to set the default value for these values to "true" instead, when they have determined that the added security benefit outweighs the configuration effort to disable the security measures for legacy clients.

It is RECOMMENDED that implementations support both a global setting, and per-client or per-server setting for the above flags. For example, an implementation could support a global setting which is over-ridden by a more specific per-client or per-server setting. The global setting could also be used if there was no client-specific or server-specific setting defined.

The combination of global and any more narrow configuration flags allows administrators to upgrade systems gradually, without requiring a "flag day" when all systems are required to change at the same time.

The following sections explain how these flags are used, by following the flow of an Access-Request packet being sent from the client, to being received by the server, to the server sending a response, and finally to that response being received by the client. To be clear: the requirements in this section apply only to RADIUS/UDP and RADIUS/TCP, and do not apply to RadSec.

4.2. Clients and Access-Request

The following new behavior is mandated for RADIUS/UDP and RADIUS/TCP clients:

> Clients MUST add Message-Authenticator to all Access-Request packets.

This behavior MUST NOT be configurable. Disabling it would open the system up to attack, and would prevent the other mitigation methods from working. The root cause of the attack is that Access-Request packets lack integrity checks. Therefore, the most important fix is to add integrity checks to those packets.

The Message-Authenticator SHOULD be the first attribute in all Access-Request packets. That is, it should be placed immediately after the packet header. Implementations MAY place the Message-Authenticator elsewhere in an Access-Request packet.

From a cryptographic point of view, the location of Message-Authenticator does not matter for Access-Request packets, it just needs to exist somewhere in the packet. However, the location of Message-Authenticator does matter for responses to Access-Request (Access-Accept, etc.). It is better to have consistent and clear messaging for addressing this attack, instead of having different recommendations for different kinds of packets.

However, many existing RADIUS clients do not currently send Message-Authenticator. It also may be difficult to upgrade some client equipment, as the relevant vendor may have gone out of business, or may have marked equipment as “end of life” and thus unsupported. It is therefore necessary for servers to work with such systems so as to not break existing RADIUS deployments, while at the same time protecting them as much as practically possible.

4.3. Servers and Access-Request

The following new behavior is mandated for for RADIUS/UDP and RADIUS/TCP servers:

> When receiving an Access-Request packet, servers MUST consult the value of the "require Message-Authenticator" flag prior to accepting the packet for processing. This flag MUST NOT be consulted for other types of request packets.
>
> If the "require Message-Authenticator" flag is set to “false”, servers MUST follow legacy behavior for validating and enforcing the existence of Message-Authenticator in Access-Request packets. For example, enforcing the requirement that all packets containing EAP-Message also contain a Message-Authenticator attributes, but otherwise accepting and validating the Message-Authenticator attribute if it is present, while taking no action if the attribute is missing.
>
> If the "require Message-Authenticator" flag is set to "false", servers MUST also check the value of the "limit Proxy-State" flag and either accept or discard the packet, based on the checks discussed in Section 4.4, below.
>
> If the "require Message-Authenticator" flag is set to “true”, the server MUST examine all Access-Request packets for the existence of the Message-Authenticator attribute. Access-Request packets which do not contain Message-Authenticator MUST be silently discarded. This discard process MUST occur before the Message-Authenticator or Request Authenticator have been validated.
>
> For packets which are not discarded by the preceding check, the server MUST then validate the contents of any Message-Authenticator and then discard packets which fail this validation as per [RFC2869], Section 5.14.
>
> Servers MUST NOT discard a packet based on the location of the Message-Authenticator attribute. We extend [RFC2865], Section 5 to state that RADIUS clients and servers MUST NOT discard packets based on the order or location of any attribute. If Message-Authenticator passes validation, then the packet is authentic and it has not been modified. The location of Message-Authenticator within the packet does not matter if the packet can be authenticated.

The default value for the "require Message-Authenticator" flag is “false” because many clients do not send the Message-Authenticator attribute in all Access-Request packets. Defaulting to a value of "true" would mean that the server would be unable to accept packets from many legacy clients, and existing networks could break.

We note that if this flag is “false”, the server can be vulnerable to the attack, even if the client has been updated to always send Message-Authenticator in all Access-Requests. An attacker could simply strip the Message-Authenticator from the Access-Request, and proceed with the attack as if client had not been updated. The server then does not see any Message-Authenticator in the Access-Request, and would accept the modified packet for processing.

When the "require Message-Authenticator" flag is set to "true", the server is protected from the BlastRADIUS attack on this client to server link. Any packet which has been modified by the attacker to remove Message-Authenticator will be discarded by the server. Any packet containing Message-Authenticator will be validated using the HMAC-MD5 construct, which is not vulnerable to this attack.

The server may still, however, be vulnerable to the attack if it proxies packets to another server. That is, the RADIUS infrastructure as a whole is secure only when all possible client to server links are secured.

Unfortunately, there is no way in RADIUS for clients and servers to negotiate protocol-layer features. A server cannot know if invalid packets are being discarded due to an ongoing attack, or if they are being discarded due to a mismatched configuration between client and server. Servers SHOULD therefore log the fact that an Access-Request packet was discarded (with rate limits) in order to inform the administrator that either an attack is underway, or that there is a configuration mismatch between client and server.

4.3.1. Detecting Configuration Mismatches

As a special case for debugging purposes, instead of discarding the packet, servers MAY instead send a Protocol-Error ([RFC7930], Section 4) or Access-Reject response packet. This packet MUST contain a Message-Authenticator attribute as the first attribute in the packet, otherwise an attacker could rewrite this response into an Access-Accept. The response MUST also contain an Error-Cause attribute with value 510 (Missing Message-Authenticator). The server MUST not send this response by default, as this behavior could cause the server to respond to forged Access-Request packets.

The purpose of this Protocol-Error response is to allow administrators to signal misconfigurations between client and server. It is intended to only be used temporarily when new client to server connections are being configured, and MUST be disabled permanently once a client-server connection is verified to work.

This behavior SHOULD only be enabled when specifically configured by an administrator. It MUST also be rate-limited, as there is no need to signal this error on every packet received by the server. It SHOULD be automatically disabled when the server receives an Access-Request from a client which contains Message-Authenticator. Implementations MAY instead automate this process, by sending a few such responses when packets from a client are first seen, and then not sending responses thereafter.

As RADIUS clients are upgraded over time, RADIUS server implementations SHOULD enable the “require Message-Authenticator” flag by default.

The next step is to protect servers when legacy clients do not send Message-Authenticator.

4.4. Updated Servers and Legacy Clients

The following new behavior is mandated for RADIUS/UDP and RADIUS/TCP servers:

> When receiving an Access-Request and where the "require Message-Authenticator" flag is set to "false", servers MUST then consult the value of the "limit Proxy-State" flag for the client.
>
> If the "limit Proxy-State" flag is set to "false", servers MUST follow legacy behavior for validating and enforcing the existence of Message-Authenticator in Access-Request packets. For example, enforcing the requirement that all packets containing EAP-Message also contain a Message-Authenticator attributes, but otherwise accepting and validating the Message-Authenticator attribute if it is present, while taking no action if the attribute is missing. This behavior is the same as mandated by the previous section.
>
> If the "limit Proxy-State" flag is set to "true", servers MUST require that all Access-Request packets which contain a Proxy-State attribute also contain a Message-Authenticator attribute. Access-Request packets which contain Proxy-State but no Message-Authenticator MUST be silently discarded.

> If the packet does contain a Message-Authenticator. servers MUST validate its contents, and discard packets which fail this validation ([RFC2869], Section 5.14).

This flag is motivated by the realization that NASes (i.e. not proxies) will never send Proxy-State in an Access-Request packet. If a server sees Proxy-State in a packet from a NAS, it is a strong signal that an attacker is attempting the BlastRADIUS attack. The BlastRADIUS attack depends on the construction and behavior of Proxy-State, so the attack is difficult or impossible when there is no Proxy-State in an Access-Request.

It is therefore useful to add a configuration flag which checks for Proxy-State, because well-behaving NASes will never send it. The only time the server will see a Proxy-State from a NAS is when the attack is taking place.

The behavior of this flag is not to simply discard Access-Request packets which contain an "unexpected" Proxy-State. Instead, the behavior is to require such packets to be authenticated. If a packet is authenticated via the existence of Message-Authenticator with validated contents, then the existence (or not) of Proxy-State does not matter; the packets are authentic, and can therefore be accepted and processed by the server.

On the other hand, if the packet cannot be authenticated by validating its Message-Authenticator, then the existence of an unexpected Proxy-State is suspicious, and the packet should be discarded.

As with the previous section, servers SHOULD log a message when packets are discarded due to this flag. Servers MAY also send an error response as discussed above, subject to the caveats and considerations described in the previous section for those responses.

After a server receives an Access-Request and processes it, it needs to send a response. The next step is to ensure that an upgraded server can protect legacy clients.

4.5. Server Responses to Access-Request

The following behavior is mandated for RADIUS/UDP and RADIUS/TCP servers, when they send responses to Access-Request packets:

> Servers MUST add Message-Authenticator as the first attribute in all responses to Access-Request packets. That is, all Access-Accept, Access-Reject, Access-Challenge, and Protocol-Error packets. The attribute MUST be the first one in the packet, immediately after the 20 octet packet header.

Adding Message-Authenticator as the first attribute means that for the purposes of MD5 known prefix attacks, the unknown suffix begins with the Message-Authenticator, and continues for the remainder of the packet. The attacker is therefore unable to leverage the attack using a known prefix, and the vulnerability is mitigated.

As it is difficult to upgrade both clients and servers simultaneously, we also need a method to protect clients when the server has not been updated. That is, clients cannot depend on the Message-Authenticator existing in response packets. Clients need to take additional steps to protect themselves, independent of any server updates.

4.6. Clients Receiving Responses

The following new behavior is mandated for RADIUS/UDP and RADIUS/TCP clients:

> When receiving any response to an Access-Request packet (Access-Accept, Access-Challenge, Access-Reject, or Protocol-Error), clients MUST consult the "require Message authenticator" flag prior to accepting the packet for processing. This flag MUST NOT be consulted for responses to other types of request packets.
>
> If the "require Message-Authenticator" flag is set to “false”, clients MUST follow legacy behavior for validating and enforcing the existence of Message-Authenticator in response packets. For example, enforcing the requirement that all packets containing EAP-Message also contain a Message-Authenticator attributes, but otherwise accepting and validating the Message-Authenticator attribute if it is present, while taking no action if the attribute is missing.
>
> If the "require Message-Authenticator" flag is set to “true”, the client MUST examine the response packets for the existence of the Message-Authenticator attribute. Response packets which do not contain Message-Authenticator MUST be silently discarded. This check MUST be done before the Response Authenticator or Message-Authenticator has been verified. No further processing of discarded packets should take place.

> The client MUST validate the contents of the Message-Authenticator and discard packets which fail this validation ([RFC2869], Section 5.14).
>
> Clients MUST NOT discard a packet based on the location of the Message-Authenticator attribute. If Message-Authenticator passes validation, then the packet is authentic and it has not been modified. The location of Message-Authenticator within the packet does not matter for authenticated packets.

When the response is discarded, the client MUST behave as if no response was received. That is, any retransmission timers MUST NOT be modified as a result of receiving a packet which is silently discarded.

Unfortunately, the client cannot determine if invalid packets are being discarded due to an ongoing attack, or if they are being discarded due to a mismatched configuration between client and server (e.g. a mismatched shared secret). The client SHOULD log the fact that the packet was discarded (with rate limits) in order to inform the administrator either that an attack is underway, or that there is a configuration mismatch between client and server.

The above discussions have now followed the complete path from client to server and back again. If each client to server hop is secured via the above mitigations, then by extension, all systems using RADIUS/UDP or RADIUS/TCP will be protected from the BlastRADIUS attack.

4.7. Status-Server

While the BlastRADIUS attack works only for Access-Request packets, Access-Accept or Access-Reject can also be sent in response to Status-Server packets ([RFC5997]). In order to simplify client implementations, we mandate the following new behavior with respect to Status-Server:

> Servers MUST follow the above recommendations relating to Message-Authenticator when sending Access-Accept, Access-Challenge, or Access-Reject packets, even if the original request was Status-Server.

This requirement ensures that clients can examine responses independent of any requests. That is, a client can perform a simple verification pass of response packets prior to doing any more complex correlation of responses to request.

We note that [RFC5997], Section 3 states:

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

As a result, compliant implementations of [RFC5997] do not need to change their behavior with respect to sending or receiving Status-Server packets: they are already protected against the BlastRADIUS attack.

4.8. Documentation and Logging

It is RECOMMENDED that all RADIUS implementations document the behavior of these flags in detail, including how they help protect against this attack. An informed administrator is more likely to engage in secure practices.

Similarly, when any of the above flags cause a packet to be discarded, the system SHOULD log a descriptive message (subject to rate limiting) about the problematic packet. This log is extremely valuable to administrators who wish to determine exactly what is going wrong, and what actions can be taken to correct the issue.

5. New Requirements on Clients and Servers

This section defines a number of updates to the RADIUS protocol which address interoperability issues. While these updates do not directly increase the security of the protocol, they correct implementation errors which have caused RADIUS ssytems to be fragile.

5.1. Attribute Location and Ordering

While [RFC2865], Section 5 states that attribute ordering does not matter, some implementations would discard packets attributes were not received in a particular order chosen by the implementer. In one such case, some implementations misunderstood the BlastRADIUS mitigations which required that Message-Authenticator be sent as the first attribute in responses to Access-Request packets. Despite the communicated requirement that clients do not check the location of Message-Authenticator, non-compliant implementations would discard valid and authentic Access-Request packets where Message-Authenticator was not the first attribute. This behavior is not appropriate.

The [RFC2865], Section 5 text which discusses attribute order (quoted below) does not cover all possible cases. The previous requirement is:

If multiple Attributes with the same Type are present, the order of Attributes with the same Type MUST be preserved by any proxies. The order of Attributes of different Types is not required to be preserved. A RADIUS server or client MUST NOT have any dependencies on the order of attributes of different types. A RADIUS server or client MUST NOT require attributes of the same type to be contiguous.

This specification adds the following requirement to cover a situation which was not covered in that text.

A RADIUS client or server MUST NOT have dependencies on the order or location of a particular attribute. A RADIUS client or server MUST NOT discard otherwise valid packets which have attributes in an order which is unexpected to the implementation, but which is valid by the above rules.

For example, if Message-Authenticator passes validation, then the packet is authentic and it has not been modified. The location of Message-Authenticator within the packet does not matter for authenticated packets. If can be the first, second, or last attribute, without any difference in meaning or security.

5.2. Unknown Attributes

Another outcome of the BlastRADIUS mitigations was the discovery that some implementations would discard packets which contained an attribute that they did not recognize. While this behavior is not explicitly permitted by previous specifications, it is not explicitly forbidden, either. This document corrects that failure.

Unknown attributes are defined as attributes which are well-formed, but which are not recognized by the implementation. Processing of unknown attributes is discussed in [RFC2866], Section 5:

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

This specification adds the following requirement to cover a situation which was not covered in that text.

RADIUS client and server implementations MUST ignore Attributes with an unknown Type. Those attributes MUST be treated in the same manner as an "Invalid Attribute" which is defined in [RFC6929], Section 2.8. The only exception to the above requirement is CoA-Request and Disconnect-Request packets, as discussed in [RFC8559], Section 4.3.2.

For all situations other than the ones discussed in [RFC8559], Section 4.3.2, implementations MUST NOT discard a packet if it contains an attribute with an unknown Type.

This behavior is secure, so long as implementations follow some additional guidance for Access-Accept packets. This guidance follows logically from existing text in [RFC2865], Section 4.4 for similar situations with Access-Challenge:

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

Additional requirements are given for Service-Type in [RFC2865], Section 5.6:

> A NAS is not required to implement all of these service types, and MUST treat unknown or unsupported Service-Types as though an Access-Reject had been received instead.

It is not practical to require that a RADIUS client implement all possible authorizations which can be sent in an Access-Accept. This specification adds the following requirement to cover a situation which was not covered by the above text.

A RADIUS client MUST treat Access-Accepts with no known or supported authorizations as though an Access-Reject had been received instead.

This requirement is already met by most RADIUS implementations. That is, experience has shown that discarding packets for arbitrary reasons causes problems. Implementations have largely chosen to follow reasonable practices, and the recommendation here codifies existing practices.

5.3. Obfuscated Attributes

The content obfuscation methods such as User-Password have not been proven to be insecure. However, they have also not been proven to be secure. As such, outside of limited situations, all existing obfuscation methods are deprecated or forbidden. The reader is directed to [I-D.dekok-radext-review-radius] for descriptions of the known security issues for those obfuscation methods.

5.3.1. User-Password obfuscation method

The User-Password obfuscation method was defined in [RFC2865], Section 5.2, and has been used for other attributes such as MS-CHAP-MPPE-Keys ([RFC2548], Section 2.4.1).

Specifications MAY define new attributes which use this obfuscation method. Specifications MAY allow these attribute in Access-Request, but MUST NOT allow them in any other packet type.

Implementations MAY allow these attributes in Access-Request. Implementations MUST NOT allow these attributes in other packet types.

5.3.2. Tunnel-Password obfuscation method

The Tunnel-Password obfuscation method was defined in [RFC2868], Section 3.5, and has been used for other attributes such as MS-MPPE-Send-Key ([RFC2548], Section 2.3.2).

Specifications MAY define new attributes which use this obfuscation method. Specifications MAY allow these attributes in Access-Accept, but MUST NOT allow them in any other packet type.

Implementations MAY MAY allow these attributes in Access-Accept. Implementations MUST NOT allow these attributes in packets types other than Access-Accept or CoA-Request.

5.3.3. Other obfuscation methods

Other attribute obfuscation methods have been defined by implementors. One common one is used for Ascend-Send-Secret and Ascend-Recv-Secret vendor-specific attributes. These methods have not had any cryptographic analysis, and are therefore inappropriate for use.

As noted above in Section 3.4.1, specifications MUST NOT define any new attribute obfuscation methods. This prohibition includes defining attributes which use the above Ascend-Send-Secret obfuscation method, as there is no specification which describes how that method works.

5.4. Rate Limiting

The design of network access means that anyone can cause a NAS to send Access-Request packets at will, simply by attempting to gain network access. If this process is not rate-limited, it can be abused by an attacker to perform dictionary or denial of service (DoS) attacks.

Real-world corner cases can also turn into effective DoS attacks on the RADIUS infrastructure. For example, if a large area of a city loses power and then regains it, the RADIUS server may see hundreds of thousands of authentication attempts within a short period of time.

A naive RADIUS client implementation will simply send an Access-Request packet for every authentication attempt, without limit. The cost to the RADIUS server to process these packets is likely significantly higher than the cost to the RADIUS client to generate them.

Poor client implementations can compound this problem when they are configured to send Accounting-Request packets. Some clients do not implement jitter in retransmissions, as suggested by [RFC5080], Section 2.2.1. The result is that when many Access-Request packets are grouped together in time, the resulting Accounting-Request packets are also grouped together. This grouping leads to the RADIUS server seeing periods of low activity, followed by sudden spikes of traffic.

Other client implementations implement a single fixed retransmission timer for all accounting traffic. That is, they have one timer which fires at a fixed interval (e.g. 10 minutes). When that timer fires, the client sends Accounting-Request packets for all active sessions. This practice may result in enormous spikes of traffic, where up to hundreds of thousands of packets are sent in only a few seconds.

Such practices create network instability, and need to be avoided.

5.4.1. Mandating Retransmission Timers

RADIUS clients MUST implement either the retransmission algorithm defined in [RFC5080], Section 2.2.1, or an equivalent one which offers similar functionality including jitter and exponential backoff.

This behavior was only recommended in [RFC5080], Section 2.2.1. Many implementations chose to ignore that recommendation, which created network instabilities as noted in the previous section.

The cost of these issues are not trivial, and there is therefore no reason to permit such behavior to continue.

5.4.2. Attacks by Unauthenticated Devices

As unauthenticated devices can cause a RADIUS client to send Access-Request packets, the RADIUS ecosystem is subject to a DoS attack unless those packets are rate limited. A misconfigured or poorly implemented device can simply restart the authentication process if it does not receive a response within a tiny time window. A malicious device can also send authentication attempts as quickly as possible.

Anecdotal evidence from network operators indicates that this is a real-world problem. Due to either poor device implemrentation or active attacks, operators sometimes see traffic spikes from individual devices which send thousands of packets a second.

Similarly, many devices are known to immediately reauthenticate after receiving an authentication failure. That issue is addressed in Section 5.4.4, below. In the interest of preventing unauthenticated devices from causing DoS attacks on the RADIUS infrastructure, it is necessary to mandate new behavior for RADIUS clients.

5.4.3. Rate Limiting Access-Request

Client implementations which control network access for unauthenticated devices MUST rate limit the number of Access-Request packets that they originate. This requirement does not apply to proxies, as they do not directly control network access for unauthenticated devices.

Implementations SHOULD implement rate limits separately for each of individual device, network port, and globally across the client. Network security and stability is improved when a client limits network access to malicious or misconfigured devices. The alternative is for the client to contribute to the problem by simply forwarding the problematic traffic. Since the cost to servers can be large, this forwarding effectively amplifies the attack.

For individual devices or wired network ports, authentication requests can be limited to once per second. Longer rate limit windows are likely to be humanly noticable, while shorter times are likely to still have negative impact on the RADIUS ecosystem.

These rate limits will not affect well-behaved devices. Rate limiting malicious devices will only have a positive effects.

For global rate limits, implementations SHOULD provide a configuration option which can be set by an administrator. Enforcing this limit in overload situations may cause some devices to fail

authentication for a short period of time, until they retry. However, if authentications are not globally rate limited, the RADIUS server will likely be overloaded, and the devices will be unable to authenticate even when there are no global rate limits.

It is difficult to offer specific advice for setting global rate limits on sending Access-Request packets. Networks can vary widely in performance and capability. A local RADIUS server may have low latency, and could be capable of processing tens or hundreds of thousands of requests per second. A remote RADIUS server at the end of a long proxy chain may have high latency, and could therefore process a much smaller number of requests per second.

One possible rate limiting method is to use a method similar to the TCP window size. A client could limit the number of sent Access-Requests which have not yet seen an Access-Accept or an Access-Reject. If there are too many outstanding Access-Requests, then the server is deemed to be overloaded. The client then waits for existing sessions to finish (or time out) before sending Access-Requests for new sessions.

Other rate limiting methods are possible, but are not discussed here. This topic is the subject of ongoing research.

5.4.4. Delaying Access-Rejects

Many devices are known to immediately reauthenticate after receiving an authentication failure. While these devices are behaving poorly and not maliciously, the effect on the RADIUS systems is similar.

In order to prevent rejected devices from reauthenticating immediately, servers MUST be able to delay Access-Reject packets. Servers SHOULD enforce a minimum delay between reception of the Access-Request and transmission of any corresponding Access-Reject. This delay SHOULD be configurable. Experience shows that values of about one (1) second work well in practice.

This delay can be enforced by any RADIUS server, including a proxy. Howerver, for proxies, this delay MUST NOT be additive. That is, proxies MUST NOT add a fixed delay to Access-Reject packets. If multiple servers in a chain of proxies were to each add a delay, the delays woud be cumultative, and therefore problematic.

Instead, proxies need only to enforce a minimum delay between Access-Request and Access-Reject.

Servers SHOULD also add a small random jitter to any preconfigured delay, in order to better protect themselves from timing attacks.

A NAS does not need add delays when rejecting a device. Instead, it should send the reject to the device immediately. As discussed above, a NAS should rate limit authentication requests from a device or port. This rate limiting is likely to be easier for the NAS to implement than delaying rejects, and will have much the same effect.

6. Migrating Away from Insecure Transports

It can be difficult to upgrade legacy devices with new cryptographic protocols and user interfaces. The problem is made worse due to the volume of RADIUS devices which are in use. The exact number is unknown, and can only be approximated. Our best guess is that at the time of this writing there are millions of devices supporting RADIUS/UDP in daily use. It will take significant time and effort to correct the deficiencies of all of them.

This section therefore documents a migration path from RADIUS/UDP to secure transports. In the following sections, we give a number of migration steps which could each be done independently. We recommend increased entropy for shared secrets. Finally, where [RFC6614] Section 2.3 makes support for TLS-PSK optional, we suggest that RADIUS/TLS and RADIUS/DTLS implementations SHOULD support TLS-PSK.

6.1. Network Operators

It is RECOMMENDED that all RADIUS traffic be sent over a logically or physically separate management network. This recommendation should be followed even if TLS transport is used. There is no reason to mix user traffic and management traffic on the same network.

Using a management network for RADIUS traffic will generally prevent anyone other than trusted administrators from attacking RADIUS. We say “generally”, because security is limited by the least secure part of the network. If a network device has a vulnerability, then an attacker could exploit that vulnerability in order to gain access to the management network. The attacker would then be free to exploit the RADIUS infrastructure.

As noted above, it is RECOMMENDED that all RADIUS traffic use TLS transport between client and server, even when the local network is believed to be secure. While IPSec is useful to connect disparate sites across untrusted networks, it is still useful to use TLS transport to secure RADIUS traffic. A defense in depth strategy helps to protect the network from both active attacks, and from accidental changes which decrease network security.

All networking equipment should be physically secure. There is no reason to have critical portions of networking infrastructure physically accessibly to the public. Where networking equipment must be in public areas (e.g. access points), that equipment SHOULD NOT have any security role in the network. Instead, any network security validation or enforcement SHOULD be done by separate equipment which is in a physically secure location.

Similarly, the use of RADIUS/TCP in any circumstances is NOT RECOMMENDED. Any system which supports RADIUS/TCP is also likely to support TLS, and that SHOULD be used instead.

6.2. Recommending TLS-PSK

Given the insecurity of RADIUS/UDP, the absolute minimum acceptable security is to use strong shared secrets. However, administrator overhead for TLS-PSK is not substantially higher than for shared secrets, and TLS-PSK offers significantly increased security and privacy.

It is therefore RECOMMENDED that implementations support TLS-PSK. In some cases TLS-PSK is preferable to certificates. It may be difficult for RADIUS clients to upgrade all of their interfaces to support the use of certificates, and TLS-PSK more closely mirrors the historical use of shared secrets, with similar operational considerations.

Additional implementation and operational considerations for TLS-PSK are given in [I-D.ietf-radext-tls-psk].

7. Practices to Increase RADIUS Security and Privacy

While we still permit the use of UDP and TCP transports in secure environments, there are opportunities for increasing the security of RADIUS when those transport protocols are used. The amount of personal identifiable information (PII) sent in packets should be minimized. Information about the size, structure, and nature of the visited network should be omitted or anonymized. The choice of authentication method also has security and privacy impacts.

The recommendations here for increasing the security of RADIUS transports also applies when TLS is used. TLS transports protect the RADIUS packets from observation by from third-parties. However, TLS does not hide the content of RADIUS packets from intermediate proxies, such as ones uses in a roaming environment. As such, the best approach to minimizing the information sent to proxies is to minimize the number of proxies which see the RADIUS traffic, and to minimize the amount of PII which is sent.

Implementers and administrators need to be aware of all of these issues, and then make the best choice for their local network which balances their requirements on privacy, security, and cost. Any security approach based on a simple "checklist" of "good / bad" practices is likely to result in decreased security as compared to an end-to-end approach which is based on understanding the issues involved.

7.1. Use Long and Complex Shared Secrets

[RFC2865] Section 3 says:

> It is preferred that the secret be at least 16 octets. This is to ensure a sufficiently large range for the secret to provide protection against exhaustive search attacks. The secret MUST NOT be empty (length 0) since this would allow packets to be trivially forged.

This recommendation is no longer adequate, so we strengthen it here.

RADIUS implementations MUST support shared secrets of at least 32 octets, and SHOULD support shared secrets of 64 octets. Implementations MUST warn administrators that the shared secret is insecure if it is 12 octets or less in length.

Administrators SHOULD use shared secrets of at least 24 octets, generated using a source of secure random numbers. Any other practice is likely to lead to compromise of the shared secret, user information, and possibly of the entire network.

Creating secure shared secrets is not difficult. The following figure outlines four separate ways to create shared secrets.

```
openssl rand -base64 16

dd if=/dev/urandom bs=1 count=16 | base64

dd if=/dev/urandom bs=1 count=16 | base32

dd if=/dev/urandom bs=1 count=16 |
    (hexdump -ve '/1 "%02x"' && echo)
```

Only one of the above commands should be run, as they are functionally equivalent. Each command reads 128 bits (16 octets) of random data from a secure source, and encodes it as printable / readable ASCII. This form of PSK will be accepted by any implementation which supports at least 32 octets for PSKs. Larger PSKs can be generated by changing the "16" number in the command to a

larger value. The above derivation assumes that the random source returns one bit of entropy for every bit of randomness which is returned. Sources failing that assumption are NOT RECOMMENDED.

Given the simplicity of creating strong secrets, there is no excuse for using weak shared secrets with RADIUS. The management overhead of dealing with complex secrets is less than the management overhead of dealing with compromised networks.

Over all, the security analysis of shared secrets is similar to that for TLS-PSK. It is therefore RECOMMENDED that implementers manage shared secrets with same the practices which are recommended for TLS-PSK, as defined in [RFC8446] Section E.7 and [RFC9257] Section 4.

On a practical note, implementers SHOULD provide tools for administrators to help them create and manage secure shared secrets. The cost to do so is minimal for an implementer. Providing such tools can further enable and motivate administrators to use secure practices.

7.2. Use Constant Time Comparisons

Both clients and servers SHOULD use constant-time operations to compare received versus calculated values which depend on secret information. If comparison operations are stopped as soon as a difference is seen, an attacker could using timing attacks to determine the correct underlying values, even without seeing them. A constant-time operation instead compares the entire value, accumulating the result along the way. Only when the entire value has been examined does the comparison return a "match" or "no-match" result.

Constant-time operations SHOULD be used for the Request Authenticator and Response Authenticator fields. Constant time comparisons SHOULD be used for attributes which directly contain secret values (e.g. User-Password), or are derived from secret values (e.g. CHAP-Password, and Message-Authenticator).

7.3. Limit the use of User-Password

The design of RADIUS means that when proxies receive Access-Request packets, the clear-text contents of the User-Password attribute are visible to the proxy. Despite various claims to the contrary, the User-Password attribute is never sent "in the clear" over the network. Instead, the password is protected by TLS (RADIUS/TLS) or via the obfuscation methods defined in [RFC2865], Section 5.2. However, the nature of RADIUS means that each proxy must first undo the password obfuscation of [RFC2865], and then re-do it when sending

the outbound packet. As such, the proxy has the clear-text password visible to it, and stored in its application memory.

It is therefore possible for every intermediate proxy to snoop and record all User-Name and User-Password values which they see. This exposure is most problematic when the proxies are administered by an organization other than the one which operates the home server. Even when all of the proxies are operated by the same organization, the temporary existence of clear-text passwords on multiple machines is a security risk.

It is therefore NOT RECOMMENDED for organizations to send the User-Password attribute in packets which are sent outside of the organization. If RADIUS proxying is necessary, another authentication method which provides for end-to-end security of user information SHOULD be used, such as EAP-TLS, TTLS, or PEAP.

Organizations MAY still use User-Password attributes within their own systems.

Client and server implementations MUST use secure programming techniques to wipe passwords and other sensitive data from memory when they are no longer needed.

7.4. Use PAP in preference to CHAP and MS-CHAP

When the system as a whole is taken into account, the risk of password compromise is substantially less with PAP than with CHAP or MS-CHAP. The full reasons are outlined in [I-D.dekok-radext-review-radius] an Section 3.3.

It is therefore RECOMMENDED that administrators use PAP in preference to CHAP or MS-CHAP. It is also RECOMMENDED that administrators store passwords "at rest" in a secure form (salted, hashed), as with the "crypt" format discussed elsewhere: TBD - add ref to review document.

That being said, other authentication methods such as EAP-TLS [RFC9190] do not expose clear-text passwords to the RADIUS server or any intermediate proxy. Thor methods therefore lower the risk of password exposure even more than using PAP. It is RECOMMENDED that administrators avoid password-based authentication methods where at all possible.

7.5. Use EAP Where Possible

If more complex authentication methods are needed, there are a number of EAP methods which can be used. These methods variously allow for the use of certificates (EAP-TLS), or passwords (EAP-TTLS [RFC5281], PEAP [I-D.josefsson-pppext-eap-tls-eap])) and EAP-pwd [RFC5931].

We also note that the TLS-based EAP methods which transport passwords also hide the passwords from intermediate RADIUS proxies, which also increases security.

Finally, password-based EAP methods still send PAP / CHAP / MS-CHAP inside of the TLS tunnel. As such, the security of a home server which checks those passwords is subject to the analysis above about PAP versus CHAP, along with the issues of storing passwords in a database.

7.6. Clients need to Implement Exponential Backoff

RADIUS client retransmission behavior is defined in [RFC5080], Section 2.2.2. That specification notes:

> Some existing RADIUS clients implement excessively aggressive retransmission behavior, utilizing default retransmission timeouts of one second or less without support for congestive backoff. When deployed at a large scale, these implementations are susceptible to congestive collapse.

Despite that specification being almost two decades old, many clients still follow the inappropriate behavior quoted above. Perhaps unsurprisingly, those implementations failures continue to result in many issues, including contributing to congestive collapse.

As a result, where [RFC5080], Section 2.2.2 suggests that clients implement these behaviors, this specification now requires that clients MUST implement the jitter and congestive backoff algorithm defined in [RFC5080], Section 2.2.2.

7.7. Minimize the use of Proxies

The design of RADIUS means that even when RADIUS/TLS is used, every intermediate proxy has access to all of the information in each packet. The only way to secure the network from such observers is to minimize the use of proxies.

Where it is still necessary to use intermediate proxies such as with eduroam ([EDUROAM], [RFC7593]) and OpenRoaming ([OPENROAMING]), it is RECOMMENDED to use EAP methods instead of bare PAP, CHAP, or MS-CHAP.

If passwords are used, they can be can be protected from being seen by proxies via TLS-based EAP methods such as EAP-TTLS or PEAP. Passwords can also be omitted entirely from being sent over the network, as with EAP-TLS [RFC9190] or EAP-pwd [RFC5931].

In many cases, however, the existence of proxies is to either due contractual obligations, or to a need to solve "N by M" connection problems. A centralized proxy system can often simplify overall network management and maintenance.

7.7.1. Eliminate Proxies Where Possible

The best way to avoid malicious proxies is to eliminate proxies entirely. The use of dynamic peer discovery ([RFC7585]) means that the number of intermediate proxies is minimized.

However, the server on the visited network still acts as a proxy between the NAS and the home network. As a result, all of the above analysis still applies when [RFC7585] peer discovery is used. There is an intermediate system which may have access to passwords or PII. The only solution is using end-to-end security for AAA, which would involve a completely new protocol.

7.7.2. There is no RADIUS Routing Protocol

While [RFC7585] allows for a client to connect directly to a server, that configuration is not always used. Historically, RADIUS systems implemented realm [RFC7542] roaming, where multiple visited networks were connected to multiple home via chains of intermediate proxies [RFC2194]. As there is no RADIUS routing protocol to control realm forwarding through these proxies, there is therefore no way to automatically determine which realms are routable, or how best to route packets for known realms.

The outcome of this limitation is that all such realm routing rules are largely configured statically, manually, and individually on multiple systems. This process can be automated within one administrative system, but it is open to mistakes or abuse in multi-system networks.

In RADIUS, each proxy which sees traffic is completely trusted. It can modify, filter, or record any packets which transit the proxy. This ability means that a proxy can engage in a large number of negative behaviors. For example, a proxy could forge Access-Request packets for realms which it knows about, and potentially perform dictionary attacks on home networks. A proxy could also alter or invent data in Accounting-Request packets, in order to defraud a home server of revenue. A proxy could also observe Accounting-Request traffic, and use the obtained information to forge Disconnect-Request packets.

Proxies can also inject traffic for realms which do not normally transit the proxy. Without a routing protocol, there is no way for a home server to automatically control which set of realms is allowed to be sent from a particular client. There is also no general way for a proxy to signal that a particular Access-Request or Accounting-Request is non-routable: it must be either rejected or discarded.

Visited sites also have no control over proxies past the ones that they have relationships with. Subsequent proxies are completely unknown, and unknowable to the visited network. Despite these systems being completely unknown, they are completely trusted due to limitations in the RADIUS protocol.

That is, there is no fine-grained way for a visited or home network to limit which intermediary systems see traffic for their realms, or what traffic can be seen by those systems. While these filtering rules can be manually documented as seen in [FILTER], this process is error-prone, and fragile.

Administrators should be aware of the above issues: fraud, forgery, and filtering are all possible in a "trusted" RADIUS ecosystem.

Historically, these issues do not appear to have been widely exploited. The most common defense against these attacks is to limit RADIUS relationships to entities which share a contractual relationship. This relationship can be direct between clients, servers, and proxies. This relationship can also be indirect, as when multiple organizations are members of a shared consortium such as eduroam.

Implementations therefore SHOULD provide methods by which routing information can be tied to particular clients and to particular home servers. Implementations SHOULD allow packets to be filtered by some combination of realm and client or home server. Administrators SHOULD take advantage of these filters to double-check that received traffic is coming from the expected sources, and contains the expected realms.

7.7.3. Dynamic Discovery and Filtering

When [RFC7585] dynamic discovery is used, intermediate proxy hops are avoided. There are a number of possible attacks here, though [RFC7585], Section 5 largely limits its discussion to rate limiting of connections.

A client which supports dynamic discovery of home servers still has to perform filtering on NAI realms before doing any lookups. When no filtering takes place, an attacker can cause a RADIUS client to do DNS lookups for arbitrary domains, and then cause it to connect to arbitrary servers. As there is no RADIUS routing protocol, there is no general way for a client to determine which realms are part of a particular organization, and are thus permitted for dynamic DNS lookups.

Organizations relying on dynamic discovery SHOULD have some way of automatically sharing which realms are valid, and which are not. There are a number of possibilities here, and choosing the best one is up to each individual organization.

Clients supporting dynamic discovery SHOULD require that servers use certificates from a private Certification Authority (CA). Clients MUST NOT automatically accept server certificates rooted from public CAs (e.g. as is done for web servers). Instead, clients MUST be configurable to use only a limited set of CAs. The default list of accepted CAs SHOULD be empty.

Similarly, servers SHOULD require that clients use certificates from a private Certification Authority (CA). Servers MUST NOT accept client certificates rooted from a public CA.

Servers which accept connections from dynamic discover are necessarily open to the Internet.

TBD - care should be taken. Where possible, IP filtering is used. Do NAI filtering, etc.

Administrators SHOULD limit the source IP of allowed connections. Server SHOULD filter packets received by NAI, and close connections when the NAIs in incoming packets do not match the NAI(s) that the server expects. This mismatch indicates either a misconfigured or malicious client.

Both clients and servers can send any data inside of a TLS tunnel. Implementations SHOULD take care to treat the data inside of a TLS tunnel as a potential source of attacks.

Where multiple realms resolve to the same destination IP address, implementations MAY send packets for multiple realms across a connection to that IP address. Clients SHOULD use SNI to indicate which realm they are connecting to. Servers SHOULD present a certificate for the requested realm, instead of using a shared or "hosting" certificate which is owned by the hosting provider, and is used by multiple realms. Such certificate sharing decreases security, and increases operational costs.

TBD: Let’s say that the RADIUS or EAP server certificate for the "example.com" is instead from the hosting company "example.org". When your company changes hosting providers, you will need to re-provision every system with a new server certificate. If instead the server certificate is for your domain, no re-provisioning is necessary.

Where systems do not have a pre-defined list of allowed realms, implementations MUST support negative caching. That is, if the lookup for a particular realm fails, or a connection to that realm fails, then the implementation needs to cache that negative result for a period of time. This cache needs to be examined prior to any new lookup or connection being made. If there is an entry in the negative cache, then the server MUST skip the lookup or connection attempt, and instead return an immediate error. This negative cache time SHOULD be configurable.

Other attacks are possible. If there are implementation bugs in a clients TLS library, an attacker could use dynamic discovery to cause the client to connect to a malicious server, and then use the server to attack the client. A malicious server could also slow down its TCP connection to engage client resources for extended periods of time. This process could even be done even before any TLS credentials are exchanged.

In general, [RFC7585] dynamic discovery is substantially different from normal application protocols which use TLS. There is substantial attack surface added by an unknown, and unauthenticated user who can cause a RADIUS client to connect to arbitrary systems under an attacker control. Dynamic discovery should be used with care, and only with substantial amounts of filtering on the NAI realms which are allowed, and only with stringent limits on the number of lookups, connection attempts, open connections, etc.

7.8. Use Rate Limiting

TBD:

7.9. Minimize Personal Identifiable Information

One approach to increasing RADIUS privacy is to minimize the amount of PII which is sent in packets. Implementers of RADIUS products and administrators of RADIUS systems SHOULD ensure that only the minimum necessary PII is sent in RADIUS.

Where possible, identities should be anonymized (e.g. [RFC7542] Section 2.4). The use of anonymized identities means that the the Chargeable-User-Identifier [RFC4372] should also be used. Further discussion on this topic is below.

Device information SHOULD be either omitted, or randomized. e.g. MAC address randomization could be used on end-user devices. The details behind this recommendation are the subject of ongoing research and development. As such, we do not offer more specific recommendations here.

Information about the visited network SHOULD be replaced or anonymized before packets are proxied outside of the local organization. The attribute Operator-NAS-Identifier [RFC8559] can be used to anonymize information about NASes in the local network.

Location information ([RFC5580] SHOULD either be omitted, or else it SHOULD be limited to the broadest possible information, such as country code. For example, [I-D.tomas-openroaming] says:

> All OpenRoaming ANPs MUST support signaling of location information

This location information is required to include at the minimum the country code. We suggest the country code SHOULD also be the maximum amount of location information which is sent over third-party networks.

7.9.1. Creating Chargeable-User-Identity

Where the Chargeable-User-Identity (CUI) [RFC4372] is used, it SHOULD be unique per session. This practice will help to maximize user privacy, as it will be more difficult to track users across multiple sessions. Due to additional constraints which we will discuss below, we cannot require that the CUI change for every session.

What we can do is to require that the home server MUST provide a unique CUI for each combination of user and visited network. That is, if the same user visits multiple networks, the home server MUST provide different CUIs to each visited network for that user. The CUI MAY be the same across multiple sessions for that user on one particular network. The CUI MAY be the same for multiple devices used by that user on one particular network.

We note that the MAC address is likely the same across multiple user sessions on one network. Therefore changing the CUI offers little additional benefit, as the user can still be tracked by the unchanging MAC address. Never the less, we believe that having a unique CUI per session can be useful, because there is ongoing work on increasing user privacy by allowing more MAC address randomization. If we were to recommend that the CUI remain constant across multiple sessions, that would in turn negate much of the effort being put into MAC address randomization.

One reason to have a constant CUI value for a user (or user devices) on one network is that network access providers may need to enforce limits on simultaneous logins. Network providers may also need to correlate user behavior across multiple sessions in order to track and prevent abuse. Both of these requirements are impossible if the CUI changes for every user session.

The result is that there is a trade-off between user privacy and the needs of the local network. While perfect user privacy is an admirable goal, perfect user privacy may also allow anonymous users to abuse the visited network. The network would then likely simply refuse to provide network access. Users may therefore have to accept some limitations on privacy, in order to obtain network access.

Although the CUI contents are not directly related to security, we still give recommendations for creating and managing of the CUI. We believe that these recommendations will help implementers satisfy the preceding requirements, while not imposing undue burden on the implementations.

In general, the simplest way to track CUIs long term is to associate the CUI to user identity in some kind of cache or database. This association could be created at the tail end of the authentication process, and before any accounting packets were received. This association should generally be discarded after a period of time if no accounting packets are received. If accounting packets are received, the CUI to user association should then be tracked along with the normal accounting data.

The above method for tracking CUI works no matter how the CUI is generated. If the CUI can be unique per session, or it could be tied to a particular user identity across a long period of time. The same CUI could also be associated with multiple devices.

Where the CUI is not unique for each session, the only minor issue is the cost of the above method is that the association is stored on a per-session basis when there is no need for that to be done. Storing the CUI per session means that is it possible to arbitrarily change how the CUI is calculated, with no impact on anything else in the system. Designs such as this which decouple unrelated architectural elements are generally worth the minor extra cost.

For creating the CUI, that process should be done in a way which is scalable and efficient. For a unique CUI per user, implementers SHOULD create a value which is unique both to the user, and to the visited network. There is no reason to use the same CUI for multiple visited networks, as that would enable the tracking of a user across multiple networks.

Before suggesting a method for creating the CUI, we note that [RFC4372] Section 2.1 defines the CUI as being of data type 'string' ([RFC8044] Section 3.5). [RFC4372] Section 2.1 further suggests that the value of the CUI is interpreted as an opaque token, similar to the Class attribute ([RFC2865] Section 5.25). Some organizations create CUI values which use the Network Access Identifier (NAI) format as defined in [RFC7542]. This format can allow the home network to be identified to the visited network, where the User-Name does not contain a realm. Such formats SHOULD NOT be used unless all parties involved have agreed to this behavior.

The CUI SHOULD be created via a construct similar to what is given below, where "+" indicates concatenation:

CUI = HASH(Visited Network Data + User Identifier + Key)

This construct has the following functional parameters.

HASH

A cryptographic hash function. It is RECOMMENDED to use an HMAC instead of a hash function.

Visited Network Data

Data which identifies the visited network.

This data could be the Operator-Name attribute ([RFC5580] Section 4.1).

User Identifier

The site-local user identifier. For tunneled EAP methods such as PEAP or TTLS, this could be the user identity which is sent inside of the TLS tunnel.

Key

A secret known only to the local network. The key is generally a large random string. It is used to help prevent dictionary attacks on the CUI.

Where the CUI needs to be constant across multiple user sessions or devices, the key can be a static value. It is generated once by the home network, and then stored for use in further CUI derivations.

Where the CUI needs to be unique per session, the above derivation SHOULD still be used, except that the "key" value will instead be a random number which is different for each session. Using such a design again decouples the CUI creation from any requirement that it is unique per session, or constant per user. That decision can be changed at any time, and the only piece which needs to be updated is the derivation of the "key" field. In contrast, if the CUI is generated completely randomly per session, then it may be difficult for a system to later change that behavior to allow the CUI to be constant for a particular user.

If an NAI format is desired, the hash output can be converted to printable text, truncated if necessary to meet length limitations, and then an "@" character and a realm appended to it. The resulting text string is then in NAI form.

We note that the above recommendation is not invertible. That is, given a particular CUI, it is not possible to determine which visited network or user identifier was used to create it. If it is necessary to use the CUI to look up a user, the home network needs to store the full set of CUI values which a user has been assigned.

If this tracking is too complex for a network, it is possible to create the CUI via an invertible encryption process as follows:

CUI = ENCRYPT(Key + Visited Network Data + User Identifier)

This construct has the following functional parameters.

ENCRYPT

A cryptographically secure encryption function.

Key

The encryption key. Note that the same key must not be used for more both hashing and encryption.

Visited Network Data

Data which identifies the visited network.

This data could be the Operator-Name attribute ([RFC5580] Section 4.1).

User Identifier

The site-local user identifier. For tunneled EAP methods such as PEAP or TTLS, this could be the user identity which is sent inside of the TLS tunnel.

However, it is RECOMMENDED that HMAC based methods are used instead of methods based on reversible encryption.

The intent is for CUI to leak as little information as possible, and ideally be different for every session. However, business agreements, legal requirements, etc. may mandate different behavior. The intention of this section is not to mandate complete CUI privacy, but instead to clarify the trade-offs between CUI privacy and business realities.

8. Privacy Considerations

The primary focus of this document is addressing privacy and security considerations for RADIUS.

Deprecating insecure transport for RADIUS, and requiring secure transport means that personally identifying information is no longer sent "in the clear". As noted earlier in this document, such information can include MAC addresses, user identifiers, and user locations.

In addition, this document suggests ways to increase privacy by minimizing the use and exchange of PII.

9. Security Considerations

The primary focus of this document is addressing privacy and security considerations for RADIUS.

Deprecating insecure transports for RADIUS, and requiring secure transports, means that many historical security issues with the RADIUS protocol are mitigated.

We reiterate the discussion above that any security analysis must be done on the system as a whole. It is not reasonable to put an expensive lock on the front door of a house while leaving the window next to it open, and then somehow declare the house to be "secure". Any approach to security based on a simple checklist is at best naive, and more truthfully is deeply misleading. At worst, such practices will decrease security by causing people to follow false security practices, and to ignore real security practices.

Implementers and administrators need to be aware of the issues raised in this document. They can then make the best choice for their local network which balances their requirements on privacy, security, and cost. Only informed choices will lead to the best security.

9.1. Historical Considerations

The BlastRADIUS vulnerability is the result of RADIUS security being a low priority for decades. Even the recommendation of [RFC5080], Section 2.2.2 that all clients add Message-Authenticator to all Access-Request packets was ignored by nearly all implementers. If that recommendation had been followed, then the BlastRADIUS vulnerability notification would have been little more than "please remember to set the require Message-Authenticator flag on all RADIUS servers."

For MS-CHAP, it has not previously been deprecated for similar reasons, even though it has been proven to be insecure for decades. This continued use of MS-CHAP has likely resulted in the leaking of many users clear-text passwords.

9.2. Practical Implications

This document either deprecates or forbids methods and behaviors which have been common practice for decades. While insecure practices have been viewed as tolerable, they are no longer acceptable.

10. IANA Considerations

IANA is instructed to update the RADIUS Types registry, and the "Values for RADIUS Attribute 101, Error-Cause Attribute" sub-registry with the following addition:

Value,Description,Reference
510,Missing Message-Authenticator,[THIS-DOCUMENT]

11. Acknowledgements

Thanks to the many reviewers and commenters for raising topics to discuss, and for providing insight into the issues related to increasing the security of RADIUS. In no particular order, thanks to Margaret Cullen, Alexander Clouter, and Josh Howlett.

Many thanks to Nadia Heninger and the rest of the BlastRADIUS team, along with Heikki Vatiainen, for extensive discussions and feedback about the issue.

The author is deeply indebted to the late Bernard Aboba for decades of advice and guidance.

12. Changelog

* 01 - added more discussion of IPsec, and move TLS-PSK to its own document,

* 02 - Added text on Increasing the Security of Insecure Transports

* 03 - add text on CUI. Add notes on PAP vs CHAP security

* 04 - add text on security of MS-CHAP. Rearrange and reword many sections for clarity.

* 05 - Rework title to deprecating "insecure practices". Clarifications based on WG feedback.

* 00 - adoption by WG.

* 01 - review from Bernard Aboba. Added discussion on accounting, clarified and re-arranged text. Added discussion of server behavior for missing Message-Authenticator

* 02 - BlastRADIUS updates.

* 03 - add delay Access-Reject, constant-time comparison, no routing protocol. Updated the text significantly and made it more consistent with the BlastRADIUS recommendations. Add "updates" other RFCs.

* 04 - updates with review from Fabian Mauchle

* 05 - merge in spelling fixes from Andrew Wood. Update and rewrite BlastRADIUS mitigations to make them clearer. Add section describing processes administrators can use to upgrade their networks.

* 06 - updates and clarifications based on reviews.

* 07 - move "review" text into draft-dekok-radext-review-radius

13. References

13.1. Normative References

[I-D.dekok-radext-review-radius]
DeKok, A., "A Review of RADIUS Security and Privacy", Work in Progress, Internet-Draft, draft-dekok-radext-review-radius-00, 6 November 2025, <https://datatracker.ietf.org/doc/html/draft-dekok-radext-review-radius-00>.

[I-D.ietf-radext-radiusdtls-bis]
Rieckers, J., Cullen, M., and S. Winter, "RadSec: RADIUS over Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", Work in Progress, Internet-Draft, draft-ietf-radext-radiusdtls-bis-15, 23 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-radext-radiusdtls-bis-15>.

[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/rfc/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/rfc/rfc2865>.

[RFC6421] Nelson, D., Ed., "Crypto-Agility Requirements for Remote Authentication Dial-In User Service (RADIUS)", RFC 6421, DOI 10.17487/RFC6421, November 2011, <https://www.rfc-editor.org/rfc/rfc6421>.

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

13.2. Informative References

[ASLEAP] Wright, J., "asleap - recovers weak LEAP and PPTP passwords", n.d., <https://github.com/joswr1ght/asleap>.

[BLAST] Goldberg, S , et al, "RADIUS/UDP Considered Harmful", n.d., <33rd USENIX Security Symposium (USENIX Security 24), 2024, pp. 7429 - 7446.>.

[DATTACK] DeKok, A., "CHAP and Shared Secret", n.d., <https://www.ietf.org/ietf-ftp/ietf-mail-archive/radius/1998-11.mail>.

[EDUROAM] eduroam, "eduroam", n.d., <https://eduroam.org>.

[FILTER] Committee, J. I. S., "Filtering of Invalid Realms", n.d., <https://community.jisc.ac.uk/library/janet-services-documentation/filtering-invalid-realms>.

[I-D.ietf-radext-tls-psk] DeKok, A., "Operational Considerations for RADIUS and TLS-PSK", Work in Progress, Internet-Draft, draft-ietf-radext-tls-psk-12, 21 January 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-radext-tls-psk-12>.

[I-D.josefsson-pppext-eap-tls-eap] Palekar, A., Josefsson, S., Simon, D., and G. Zorn, "Protected EAP Protocol (PEAP) Version 2", Work in Progress, Internet-Draft, draft-josefsson-pppext-eap-tls-eap-10, 21 October 2004, <https://datatracker.ietf.org/doc/html/draft-josefsson-pppext-eap-tls-eap-10>.

[I-D.tomas-openroaming]
Tomas, B., Grayson, M., Canpolat, N., Cockrell, B., and S. Gundavelli, "WBA OpenRoaming Wireless Federation", Work in Progress, Internet-Draft, draft-tomas-openroaming-07, 23 January 2026, <https://datatracker.ietf.org/doc/html/draft-tomas-openroaming-07>.

[KAMATH] Palekar, R. H. and A., "Microsoft EAP CHAP Extensions", June 2007.

[MD5-1996] group, I. R. W., "MD5 Key recovery attack", n.d., <https://www.ietf.org/ietf-ftp/ietf-mail-archive/radius/1998-02>.

[OPENROAMING]
Alliance, W. B., "OpenRoaming: One global Wi-Fi network", n.d., <https://wballiance.com/openroaming/>.

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

[RFC2194] Aboba, B., Lu, J., Alsop, J., Ding, J., and W. Wang, "Review of Roaming Implementations", RFC 2194, DOI 10.17487/RFC2194, September 1997, <https://www.rfc-editor.org/rfc/rfc2194>.

[RFC2433] Zorn, G. and S. Cobb, "Microsoft PPP CHAP Extensions", RFC 2433, DOI 10.17487/RFC2433, October 1998, <https://www.rfc-editor.org/rfc/rfc2433>.

[RFC2548] Zorn, G., "Microsoft Vendor-specific RADIUS Attributes", RFC 2548, DOI 10.17487/RFC2548, March 1999, <https://www.rfc-editor.org/rfc/rfc2548>.

[RFC2759] Zorn, G., "Microsoft PPP CHAP Extensions, Version 2", RFC 2759, DOI 10.17487/RFC2759, January 2000, <https://www.rfc-editor.org/rfc/rfc2759>.

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

[RFC2868] Zorn, G., Leifer, D., Rubens, A., Shriver, J., Holdrege, M., and I. Goyret, "RADIUS Attributes for Tunnel Protocol Support", RFC 2868, DOI 10.17487/RFC2868, June 2000, <https://www.rfc-editor.org/rfc/rfc2868>.

[RFC2869] Rigney, C., Willats, W., and P. Calhoun, "RADIUS Extensions", RFC 2869, DOI 10.17487/RFC2869, June 2000, <https://www.rfc-editor.org/rfc/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/rfc/rfc3579>.

[RFC4372] Adrangi, F., Lior, A., Korhonen, J., and J. Loughney, "Chargeable User Identity", RFC 4372, DOI 10.17487/RFC4372, January 2006, <https://www.rfc-editor.org/rfc/rfc4372>.

[RFC5080] Nelson, D. and A. DeKok, "Common Remote Authentication Dial In User Service (RADIUS) Implementation Issues and Suggested Fixes", RFC 5080, DOI 10.17487/RFC5080, December 2007, <https://www.rfc-editor.org/rfc/rfc5080>.

[RFC5176] Chiba, M., Dommety, G., Eklund, M., Mitton, D., and B. Aboba, "Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS)", RFC 5176, DOI 10.17487/RFC5176, January 2008, <https://www.rfc-editor.org/rfc/rfc5176>.

[RFC5281] Funk, P. and S. Blake-Wilson, "Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)", RFC 5281, DOI 10.17487/RFC5281, August 2008, <https://www.rfc-editor.org/rfc/rfc5281>.

[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/rfc/rfc5580>.

[RFC5931] Harkins, D. and G. Zorn, "Extensible Authentication Protocol (EAP) Authentication Using Only a Password", RFC 5931, DOI 10.17487/RFC5931, August 2010, <https://www.rfc-editor.org/rfc/rfc5931>.

[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/rfc/rfc5997>.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <https://www.rfc-editor.org/rfc/rfc6151>.

[RFC6218] Zorn, G., Zhang, T., Walker, J., and J. Salowey, "Cisco Vendor-Specific RADIUS Attributes for the Delivery of Keying Material", RFC 6218, DOI 10.17487/RFC6218, April 2011, <https://www.rfc-editor.org/rfc/rfc6218>.

[RFC6613] DeKok, A., "RADIUS over TCP", RFC 6613, DOI 10.17487/RFC6613, May 2012, <https://www.rfc-editor.org/rfc/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, <https://www.rfc-editor.org/rfc/rfc6614>.

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

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <https://www.rfc-editor.org/rfc/rfc6973>.

[RFC7360] DeKok, A., "Datagram Transport Layer Security (DTLS) as a Transport Layer for RADIUS", RFC 7360, DOI 10.17487/RFC7360, September 2014, <https://www.rfc-editor.org/rfc/rfc7360>.

[RFC7542] DeKok, A., "The Network Access Identifier", RFC 7542, DOI 10.17487/RFC7542, May 2015, <https://www.rfc-editor.org/rfc/rfc7542>.

[RFC7585] Winter, S. and M. McCauley, "Dynamic Peer Discovery for RADIUS/TLS and RADIUS/DTLS Based on the Network Access Identifier (NAI)", RFC 7585, DOI 10.17487/RFC7585, October 2015, <https://www.rfc-editor.org/rfc/rfc7585>.

[RFC7593] Wierenga, K., Winter, S., and T. Wolniewicz, "The eduroam Architecture for Network Roaming", RFC 7593, DOI 10.17487/RFC7593, September 2015, <https://www.rfc-editor.org/rfc/rfc7593>.

[RFC7930] Hartman, S., "Larger Packets for RADIUS over TCP", RFC 7930, DOI 10.17487/RFC7930, August 2016, <https://www.rfc-editor.org/rfc/rfc7930>.

[RFC8018] Moriarty, K., Ed., Kaliski, B., and A. Rusch, "PKCS #5: Password-Based Cryptography Specification Version 2.1", RFC 8018, DOI 10.17487/RFC8018, January 2017, <https://www.rfc-editor.org/rfc/rfc8018>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC8559] DeKok, A. and J. Korhonen, "Dynamic Authorization Proxying in the Remote Authentication Dial-In User Service (RADIUS) Protocol", RFC 8559, DOI 10.17487/RFC8559, April 2019, <https://www.rfc-editor.org/rfc/rfc8559>.

[RFC9190] Preu Mattsson, J. and M. Sethi, "EAP-TLS 1.3: Using the Extensible Authentication Protocol with TLS 1.3", RFC 9190, DOI 10.17487/RFC9190, February 2022, <https://www.rfc-editor.org/rfc/rfc9190>.

[RFC9257] Housley, R., Hoyland, J., Sethi, M., and C. A. Wood, "Guidance for External Pre-Shared Key (PSK) Usage in TLS", RFC 9257, DOI 10.17487/RFC9257, July 2022, <https://www.rfc-editor.org/rfc/rfc9257>.

Appendix A. Best Practice Checklist

In the interest of simplifying the above explanations, this section provides a short-form checklist of recommendations. Following this checklist does not guarantee that RADIUS systems are secure from all possible attacks. However, systems which do not follow this checklist are likely to be vulnerable to known attacks, and are therefore less secure than they could be.

- Do not use RADIUS/UDP or RADIUS/TCP across the wider Internet

  Exposing user identifiers, device identifiers, and locations is a privacy and security issue.

- Avoid RADIUS/UDP or RADIUS/TCP in other networks, too.

  It can take time to upgrade equipment, but the long-term goal is to entirely deprecate RADIUS/UDP.

- Implement the BlastRADIUS mitigations

Both Implementers and administrators should implement the mitigations in order to secure Access-Request packets.

- Use strong shared secrets

Shared secrets should be generated from a cryptographically strong pseudo-random number generator. They should contain at least 128 bits of entropy. Each RADIUS client should have a unique shared secret.

- Minimize the use of RADIUS proxies.

More proxies means more systems which could be compromised, and more systems which can see private or secret data.

- Do not proxy from secure to insecure transports

If user information (credentials or identities) is received over a secure transport (IPsec, RADIUS/TLS, TLS-based EAP method), then proxying the protected data over RADIUS/UDP or RADIUS/TCP degrades security and privacy.

- Prefer EAP authentication methods to non-EAP methods.

EAP authentication methods are better at hiding user credentials from observers.

- For EAP, use anonymous outer identifiers

There are few reasons to use individual identities for EAP. Identifying the realm is usually enough.

[RFC7542] Section 2.4 recommends that "@realm" is preferable to "anonymous@realm", which is in turn preferable to "user@realm".

- Prefer using PAP over CHAP or MS-CHAP.

PAP allows for credentials to be stored securely "at rest" in a user database. CHAP and MS-CHAP do not.

- Do not use MS-CHAP outside of TLS-based EAP methods such as PEAP or TTLS.

MS-CHAP can be cracked with minimal effort. The attack has been available for two decades.

- Store passwords in "crypt"ed form

Where is is necessary to store passwords, use systems such as PBKDF2 ([RFC8018].

- Regularly update to the latest cryptographic methods.

TLS 1.0 with RC4 was acceptable at one point in time. It is no longer acceptable. Similarly, the current cryptographic methods will at some point will be deprecated, and replaced by updated methods. Upgrading to recent cryptographic methods should be a normal part of operating a RADIUS server.

- Regularly deprecate older cryptographic methods.

Administrators should actively deprecate the use of older cryptographic methods. If no system is using older methods, then those methods should be disabled or removed entirely. Leaving old methods enabled makes the server more vulnerable to attacks.

- Send the minimum amount of information which is needed,.

Where proxying is used, it is a common practice is to simply forward all of the information from a NAS to other RADIUS servers. Instead, the proxy closest to the NAS should filter out any attributes or data which are not needed by the "next hop" proxies, or by the home server.

Appendix B. BlastRADIUS Mitigations

B.1. Implementor Checklist

The following list outlines the requirements on client implementations, and references the prior sections which contain the normative text. The intent is to give readers a short checklist which lets them quickly validate that their implementations are correct. While the following list does not contain normative text (in order to avoid potential conflict or confusion), the reader should follow the references below to verify that the behavior described below is truly normative.

* clients include Message-Authenticator in all Access-Request packets, Section 4.2

  - clients can place Message-Authenticator as the first attribute in all Access-Request packets, but this placement is not required for security.

* clients validate the contents of Message-Authenticator in all packets that they receive, [RFC2869], Section 5.14

* clients do not check the location of Message-Authenticator in any response packet that they receive, Section 4.6

* clients do not discard packets which contain unknown attributes, Section 5.2

* clients implement a boolean configuration flag "require Message-Authenticator", Section 4.1

  - If set to "false", clients do not take any additional steps.

  - if set to "true", clients discard all responses to Access-Request packets which do not contain Message-Authenticator. This discard happens before the Response Authenticator or Message-Authenticator are validated.

The following list outlines requirements on server implementations, with the same explanations and caveats given above for the list of requirements on client implementations.

* servers validate the contents of Message-Authenticator in all packets that they receive, Section 4.3

* server do take check the location of Message-Authenticator in any request packet that they receive, Section 4.5

* servers do not discard packets which contain unknown attributes, Section 5.2

* servers implement a boolean configuration flag "require Message-Authenticator", Section 4.1

  - If set to "false", servers implement checks for the "limit Proxy-State" flag.

  - if set to "true", servers discard all Access-Request packets which do not contain a Message-Authenticator attribute. This discard happens before the Request Authenticator or Message-Authenticator are validated. Servers then do not implement the checks for the "limit Proxy-State" flag.

* servers implement a boolean configuration flag "limit Proxy-State", Section 4.1 and Section 4.4.

  - servers check this flag only when the "require Message-Authenticator" flag is set to "false".

  - If set to "false", servers take no further action.

- If set to "true", servers discard all Access-Request packets which do not contain Message-Authenticator, and which also contain one or more Proxy-State attributes. This discard happens before the Request Authenticator or Message-Authenticator are validated.

* servers include Message-Authenticator in all responses to Access-Request packets, Section 4.5

* servers include Message-Authenticator in all Access-Accept, Access-Reject, Access-Challenge, and Protocol-Error packets, Section 4.5 and Section 4.7

* servers place Message-Authenticator as the first attribute in all responses to Access-Request packets, and in all Access-Accept, Access-Reject, and Access-Challenge packets, Section 4.5.

Appendix C. Administrator Upgrade Process

The preceding sections define requirements for client and server implementations which address the BlastRADIUS attack. It is useful to also provide guidelines for administrators as to how, and when, to set the new configuration flags. The guidelines provided in this section are a suggestion only. Administrators are free to take other actions as they see fit.

The guidelines provided here are known to provide minimal outages while upgrading complex systems. As such, it is RECOMMENDED that administrators follow the steps outlined here, in order, so that RADIUS systems can be upgraded with minimal impact to operational networks.

1. Administrators SHOULD upgrade servers before upgrading clients. There are many fewer clients than servers, and upgrading servers can often protect clients which are not upgraded.

2. Administrators SHOULD configure servers to set "limit Proxy-State" to "true" for all clients which are NASes. i.e. clients which are not proxies.

3. Administrators of servers which proxy packets SHOULD verify that all "next hop" proxies have been upgraded, and that they return Message-Authenticator in all responses to Access-Request packets.

4. Once step (3) has been validated, administrators SHOULD configure their proxy so that the outgoing client configuration sets the "require Message-Authenticator" flag to "true".

5. Administrators of servers which receive proxied packets (i.e. packets not from a NAS) SHOULD configure the server to set the the "require Message-Authenticator" flag to "true" for each client which is an upgraded proxy.

Once the above five steps are followed, the network should be secure, and any client upgrade and configuration can be done over time.

For client upgrades, administrators can proceed with the following steps:

1. Administrators SHOULD upgrade clients individually, i.e. one at a time. Upgrading multiple clients at the same time is NOT RECOMMENDED.

2. Once a client has been upgraded, administrators SHOULD verify that it sends Message-Authenticator in all Access-Request packets.

3. Once step (2) has been validated, administrators SHOULD configure each server that receives packets from that client to set the "require Message-Authenticator" flag to "true" for that client.

4. If a server has been updated, administrators SHOULD verify that it sends Message-Authenticator as the first attribute in all responses to Access-Request packets.

5. Once step (4) has been validated, administrators SHOULD configure each client receiving packets from that server to set the "require Message-Authenticator" flag to "true" for that server.

Once all of the above steps are followed for all clients and servers, the network is secure from the BlastRADIUS attack.

Author’s Address

Alan DeKok
InkBridge Networks
Email: aland@inkbridgenetworks.com