RADEXT Working Group
Internet-Draft
Updates: 2865, 2866, 5176, 7585 (if approved)
Intended status: Standards Track
Expires: 26 November 2025

A. DeKok
InkBridge Networks

25 May 2025

Deprecating Insecure Practices in RADIUS
draft-ietf-radext-deprecating-radius-06

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. They have proven their utility as replacements for the previous 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 recent 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. We also discuss related security issues with RADIUS, and give recommendations 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

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 has been known for a long time. It 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 record of concerns about Access-Request packets spoofing was on the RADIUS working group mailing list [DATTACK] in 1998. There was substantial further discussions about the lack of integrity checks on the list over the next few years. The outcome of that process was the definition of Message-Authenticator as an optional HMAC-based attribute in [RFC2869], Section 5.14.

Unfortunately, the use of Message-Authenticator was made optional. This lack of integrity checks for Access-Request packets was deemed acceptable for some situations in [RFC2869], Section 7.1:

> Access-Request packets with a User-Password establish the identity of both the user and the NAS sending the Access-Request, because of the way the shared secret between NAS and RADIUS server is used.

That conclusion now appears to be incorrect. The text continues with an acknowledgment that:

> Access-Request packets with CHAP-Password or EAP-Message do not have a User-Password attribute, so the Message-Authenticator attribute should be used in access-request packets that do not have a User- Password, in order to establish the identity of the NAS sending the request.

This text was non-normative due to the lowercase 'should'. It appears that no implementation followed even this limited suggestion.

The packet forgery issue was further discussed in 2004 in [RFC3579], Section 4, and again in 2007 in [RFC5080], Section 2.2.2. That document suggested that implementations require the use of Message-Authenticator in order to prevent forgery:

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

To our knowledge, only one RADIUS server implemented even this limited suggestion. At the time of publication of [RFC5080], there was no consensus to require the use of Message-Authenticator in all Access-Request packets. If this recommendation had instead been made mandatory, then the recent BlastRADIUS attack [BLAST] would largely have been prevented.

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. The outcome of that review was the text in the remainder of [RFC6421], which created crypto-agility requirements for RADIUS. The main outcome of those requirements was not any change to RADIUS, but instead the definition of RADIUS/TLS in [RFC6614], and RADIUS/DTLS in [RFC7360]. The other outcome was a consensus that adding crypto-agility to RADIUS was likely not a good idea, and that standardizing RADIUS over TLS instead was a significantly better path forward.

While the RADIUS/TLS work is ongoing at the time of this writing, there are still a large number of sites using RADIUS/UDP. Those sites need to be supported and secured until they can migrate to TLS, while at the same time maintaining backwards compatibility.

To summarize, [RFC6151] is over a decade old as of the time of this writing. [RFC5080] is almost two decades old. The knowledge that Access-Request packets lack integrity checks is almost three decades old. Over that entire span of time, there has been no mandate to increase the security of Access-Request packets. This document provides that mandate.

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. We also discuss related security issues with RADIUS, and give many recommendations for practices which increase security and privacy.

1.1. RADIUS over the Internet

As the insecurity of MD5 has been well known for decades, RADIUS traffic over the Internet was historically secured with IPsec as described in [RFC3579], Section 4.2:

> To address the security vulnerabilities of RADIUS/EAP, implementations of this specification SHOULD support IPsec (RFC2401) along with IKE (RFC2409) for key management. IPsec ESP (RFC2406) with non-null transform SHOULD be supported, and IPsec ESP with a non-null encryption transform and authentication support SHOULD be used to provide per-packet confidentiality, authentication, integrity and replay protection. IKE SHOULD be used for key management.

The use of IPsec allowed RADIUS to be sent privately, and securely, across the Internet. However, experience showed that TLS was in many ways simpler for implementations and deployment than IPsec. While IPsec required operating system support, TLS was an application-space library. This difference, coupled with the wide-spread adoption of TLS for HTTPS, ensures that it was often easier for applications to use TLS than IPsec.

RADIUS/TLS [RFC6614] and RADIUS/DTLS [RFC7360] were then defined in order to meet the crypto-agility requirements of [RFC6421]. RADIUS/TLS has been in wide-spread use for about a decade, including eduroam [EDUROAM] [RFC7593], and more recently OpenRoaming [OPENROAMING] and [I-D.tomas-openroaming]. RADIUS/DTLS has seen less use across the public Internet, but it still has multiple implementations.

However, RADIUS/UDP is still widely used, even though it depends on MD5 and "ad hoc" constructions for security. The recent "BlastRADIUS" attack shows just how inadequate this dependency is. The BlastRADIUS attack is discussed in more detail below, in Section 5.

Even if we ignore the BlastRADIUS attack, problems with MD5 mean that a hobbyist attacker who can view RADIUS/UDP traffic can brute-force test all possible RADIUS shared secrets of eight characters in not much more than an hour. An more resourceful attacker (e.g. a nation-state) can check all much longer shared secrets with only modest expenditures. See Section 3.4 below for a longer discussion of this topic.

Determining the shared secret will also result in compromise of all passwords carried in the User-Password attribute. Even using CHAP-Password offers minimal protection, as the cost of cracking the underlying password is similar to the cost of cracking the shared secret. MS-CHAP ([RFC2433] and MS-CHAPv2 [RFC2759]) are significantly worse in security than PAP, as they can be completely broken with minimal resources, which Section 8.6 describes in more detail.

The use of Message-Authenticator does not change the cost of attacking the shared secret. The Message-Authenticator attribute is a later addition to RADIUS, and does does not replace the original MD5-based packet signatures. While that attribute therefore offers a stronger protection, it does not change the cost of attacking the shared secret. Moving to a stronger packet signatures (e.g. [RFC6218]) would still not fully address the issues with RADIUS, as the protocol still has privacy issues unrelated to the the security of packet authenticators.

That is, most attributes in RADIUS are sent in clear-text, and only a few attributes such as User-Password and Tunnel-Password have their contents hidden. Even the hidden attributes rely on "ad hoc" obfuscation methods using MD5, which have not been successfully attacked, but are not proven to be secure. Peoples locations can (and has) been accurately determined, and people have been tracked using location data sent insecurely across the Internet (Section 3.2).

The implications for security and individual safety are large, and negative.

These issues are only partly mitigated when the data carried within RADIUS use their own methods for increased security and privacy. For example, some authentication methods such EAP-TLS, EAP-TTLS, etc. allow for User-Name privacy and for more secure transport of passwords via the use of TLS. Some privacy can be gained through MAC address randomization, which can also limit device information identification to a particular manufacturer, instead of to a unique device.

However, these methods are not always used, or are not always available. Even if these methods were used ubiquitously, they do not protect all of the information which is publicly available over RADIUS/UDP or RADIUS/TCP transports. And even when TLS-based EAP methods are used, implementations have historically often skipped certificate validation, leading to password compromise ([SPOOFING]). In many cases, users were not even aware that the server certificate was incorrect or spoofed, which meant that there was no way for the user to detect that anything was wrong. Their passwords were simply handed to a spoofed server, with little possibility for the user to take any action to stop it.

1.2. Simply using IPsec or TLS is not enough

The use of a secure transport such as IPsec or TLS ensures complete privacy and security for all RADIUS traffic. An observer of encrypted traffic is limited to knowing rough activity levels of a client or server. That is, an observer can tell if there are a few users on a NAS, or many users on a NAS. All other information is hidden from all observers. Even with those limitations, it is not enough to say "use IPsec" and then move on to other issues. There are many issues which can only be addressed via an informed approach.

For example, it is possible for an attacker to record the session traffic, and later crack the TLS session key or IPsec parameters. This attack could comprise all traffic sent over that connection, including EAP session keys. If the cryptographic methods provide forward secrecy ([RFC7525], Section 6.3), then breaking one session provides no information about other sessions. As such, it is RECOMMENDED that all cryptographic methods used to secure RADIUS conversations provide forward secrecy. While forward secrecy will not protect individual sessions from attack, it will prevent attack on one session from being leveraged to attack other, unrelated, sessions.

AAA servers SHOULD minimize the impact of such attacks by using a total throughput or time based limit before replacing the session keys. The session keys can be replaced though a process of either re-keying the existing connection, or by opening a new connection and deprecating the use of the original connection. Note that if the original connection if closed before a new connection is open, it can cause spurious errors in a proxy environment.

The final attack possible in a AAA system is where one party in a AAA conversation is compromised or run by a malicious party. This attack is made more likely by the extensive use of RADIUS proxy forwarding chains. In that situation, every RADIUS proxy has full visibility into, and control over, the traffic it transports. The solution here is to minimize the number of proxies involved, such as by using Dynamic Peer Discovery, as defined in [RFC7585].

There are many security issues in addition to simply adding a secure transport. The rest of this document addresses those issues in detail.

1.3. Overview

The rest of this document begins a summary of issues with RADIUS, including showing just how trivial it is to crack RADIUS/UDP security. We then mandate the use of secure transport, and describe what that requirement means in practice. We give recommendations on how current systems can be migrated to using TLS. We give suggestions for increasing the security of existing RADIUS transports, including a discussion of the authentication protocols carried within RADIUS. We conclude with security and privacy considerations.

As IPsec has been discussed previously in the context of RADIUS, we do not discuss it more here, except to say it is an acceptable solution for securing RADIUS traffic. As the bulk of the current efforts are focused on TLS, this document likewise focuses on TLS. We note that all of the issues raised here about the RADIUS protocol also apply to IPsec transport. That is, when the application is not in charge of protocol security, the application is vulnerable to transport misconfigurations or attacks.

1.3.1. A Comment on Specifications

While this document tries to be comprehensive, it is necessarily imperfect. There may be issues which should have been included here, but which were missed due to oversight or accident. Any reader should be aware that there are good practices which are perhaps not documented in a specification, and bad behaviors which are likewise not forbidden. For example, documents such as [RFC5080] were written to both correct errors in earlier documents, and to address harmful behaviors which had been seen in practice.

These harmful behaviors can have a large impact both on security and on interoperability, even if they are not expressly forbidden in a specification.

There is a regrettable belief that a particular practice is "allowed" by a specification, simply because the specification does not forbid that practice. This belief is wrong. That is, a behavior which is not even mentioned in the specification cannot honestly be said to be "permitted" or "allowed" by that specification. The most charitable description would be that these behaviors are undefined, or at best, they are not forbidden.

By their very nature, documents include a small number of permitted, required, and/or forbidden behaviors. There are a much larger set of behaviors which are undefined. That is, behaviors which are neither permitted nor forbidden. Those behaviors may be good or bad, independent of what any specification says.

Outside of published specifications, there is also a large set of common practices and behaviors which have grown organically over time, but which have not been written into a specification. These practices have been found to be valuable by implementers and administrators. Deviations from these practices generally result in instabilities and incompatibilities between systems. As a result, implementers should exercise caution when creating new behaviors which have not previously been seen in the industry. Such behaviors are likely to cause problems, where there would have been no problems if common practices had instead been followed.

It is RECOMMENDED that implementations and administrators follow widely accepted practices which have been proven to work and to be secure, even if those practices are not written down in a public specification. Implementers SHOULD NOT create features which depend on undefined behavior; such features are very likely to be wrong.

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]

* TLS

  the Transport Layer Security protocol. Generally when we refer to TLS in this document, we are referring to RADIUS/TLS and/or RADIUS/DTLS.

* NAS

  Network Access Server, which is a RADIUS client.

In order to continue the terminology of [RFC2865], we describe 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. The reader is assured that no modern cryptographic methods are used with RADIUS/UDP.

3. Overview of issues with RADIUS

There are a large number of issues with RADIUS. The most serious is the BlastRADIUS vulnerability, which means that subject to some limitations, attackers can leverage MD5 known-prefix collisions to cause any user to be authenticated, and then be given any authorization. Multi-factor Authentication (MFA) systems can be bypassed, and the RADIUS server will in many cases not even be aware that an unauthorized user is on the network.

Another issue is that RADIUS sends most information "in the clear", with obvious privacy implications. Even if packets use Message-Authenticator for integrity checks, it is still possible for the average hobbyist who observes RADIUS traffic to perform brute-force attacks to crack even seemingly complex shared secrets.

There is no way to fix the RADIUS protocol to address all of these issues. The short-term fix is to require the use of Message-Authenticator for packet integrity checks. The long-term solution is to wrap the protocol in a secure transport, such as TLS or IPsec.

We address each of these issues in detail below.

3.1. The BlastRADIUS Vulnerability

The BlastRADIUS vulnerability is discussed in detail in [BLAST], and we only give a short summary here. We refer the reader to the original paper for a more complete description of the issue.

For the following description, we assume that we have texts "A", "B", "S". Following the use in [RFC2865], "+" denotes concatenation. The vulnerability then relies on the following property of MD5:

> If MD5(A) == MD5(B), then MD5(A + S) == MD5(B + S)

If an attacker is given text "A", and can find text "B" which has the same MD5 hash, then the attacker can perform a chosen prefix attack. Even if the attacker does not know text "S", if given M5(A + S), then the attacker can trivially calculate MD5(B + S): it has the same value.

In RADIUS, the Response Authenticator field [RFC2865], Section 3 is calculated via precisely this vulnerable construct:

> Response Authenticator = MD5(packet + secret)

The attacker can observe or predict a packet (e.g. Access-Reject), which is signed with a shared secret unknown to the attacker. The attacker can then find an Access-Accept with the same MD5 hash as the Access-Reject, and replace the Access-Reject with the Access-Accept using the Response Authenticator from the Access-Reject.

The client receives the packet, calculates MD5(Access-Accept + secret), verifies that the Response Authenticator is correct, and proceeds to follow the attackers instructions: give the user access, along with some authorization.

This process is the basic concept behind the BlastRADIUS vulnerability. We note that this attack does not expose the contents of the User-Password attribute. Instead, it bypasses all server-side authentication, and simply fools the client into accepting a forged response.

While this issue requires that an attacker be "on path" and be able to intercept and modify packets, the meaning of "on path" is often "the entire Internet". As such, this attack alone is sufficient reason to deprecate all uses of RADIUS/UDP and RADIUS/TCP.

3.2. Information is sent in Clear Text

With the exception of a few attributes such as User-Password, all RADIUS traffic is sent "in the clear" when using UDP or TCP transports. Even when TLS is used, all RADIUS traffic (including User-Password) is visible to proxies. The resulting data exposure has a large number of privacy issues. We refer to [RFC6973], and specifically to Section 5 of that document for detailed discussion, and to [RFC6973], Section 6 for recommendations on threat mitigations.

More discussion of location privacy is given in [RFC6280], which defines an "Architecture for Location and Location Privacy in Internet Applications". However, that work was too late to have any practical impact on the design of the RADIUS protocol, as [RFC5580] had already been published.

The effect is that any observer of non-TLS RADIUS traffic is able to obtain a substantial amount of personal identifiable information (PII) about users. The observer can tell who is logging in to the network, what devices they are using, where they are logging in from, and their approximate location (usually city). With location-based attributes as defined in [RFC5580], a user's location may be determined to within 15 or so meters outdoors, and with "meter-level accuracy indoors" [WIFILOC]. An observer can also use RADIUS accounting packets to determine how long a user is online, and to track a summary of their total traffic (upload and download totals).

When RADIUS/UDP is used across the public Internet, common Wi-Fi configurations allow the location of individuals to be tracked in real-time (usually 10 minute intervals), to within 15 meters. The user devices can be identified, and also tracked. Passwords can often be compromised by a resourceful attacker, or for MS-CHAP, by a hobbyist with a laptop. Even when the packets do not contain any [RFC5580] location information for the user, the packets usually contain the MAC address of the Wi-Fi access point. The MAC address and physical location of these devices are publicly available, and there are multiple services selling databases of this information.

These issues are not theoretical. Recently [BRIGGS] noted that:

> Overall, I think the above three examples are just the tip of the proverbial iceberg of SS7 and Diameter based location and monitoring exploits that have been used successfully against targeted people in the USA.

[BRIGGS] continues with a statement that there have been:

> ... numerous other exploits based on SS7 and Diameter that go beyond location tracking. Some of these involve issues like (1) the monitoring of voice and text messages, (2) the delivery of spyware to targeted devices, and (3) the influencing of U.S. voters by overseas countries using text messages.

While these comments apply to Diameter [RFC6733], the same location tracking and monitoring is also possible with RADIUS. There is every reason to believe that similar attacks on RADIUS have occurred, but are simply less publicized than similar attacks on Diameter.

The use of clear-text protocols across insecure networks is no longer acceptable. Using clear-text protocols in networks which are believed to be secure is not a significantly better solution. The correct solution is to use secure protocols, to minimize the amount of private data which is being sent, and to minimize the number of third parties who can see any traffic.

3.3. MD5 has been broken

Attacks on MD5 are summarized in part in [RFC6151]. The BlastRADIUS work substantially improved the speed of finding MD5 collisions, and those improvements are publicly available at [HASHCLASH].

While there have not been many other new attacks in the decade since [RFC6151] was published, that does not mean that further attacks do not exist. It is more likely that no one is looking for new attacks.

3.4. Cracking RADIUS shared secrets is not hard

The cost of cracking a a shared secret can only go down over time as computation becomes cheaper. The issue is made worse because of the way MD5 is used to authenticate RADIUS packets. The attacker does not have to calculate the hash over the entire packet, as the hash prefix can be calculated once, and then cached. The attacker can then begin the attack with that hash prefix, and brute-force only the shared secret portion.

At the time of writing this document, an "off the shelf" commodity computer can calculate at least 100M MD5 hashes per second. If we limit shared secrets to upper/lowercase letters, numbers, and a few "special" characters, we have 64 possible characters for shared secrets. Which means that for 8-character secrets, there are 2^48 possible combinations.

The result is that using consumer-grade machine, it takes approximately 32 days to brute-force the entire 8 octet / 64 character space for shared secrets. The problem is even worse when graphical processing units (GPUs) are used. A high-end GPU is capable of performing more than 64 billion hashes per second. At that rate, the entire 8 character space described above can be searched in approximately 90 minutes.

This is an attack which is feasible today for a hobbyist. Increasing the size of the character set raises the cost of cracking, but not enough to be secure. Increasing the character set to 93 characters means that the hobbyist using a GPU could search the entire 8 character space in about a day.

Increasing the length of the shared secret has a larger impact on the cost of cracking. For secrets ten characters long, the search space is approximately 2^60. One GPU can search a 64-character space in about six months. A 93 character space (2^65 complexity) would take approximately 24 years.

This brute-force attack is also trivially parallelizable. Nation-states have sufficient resources to deploy hundreds to thousands of systems dedicated to these attacks. That realization means that a "time to crack" of 24 years is simply expensive, but does not take much "wall clock" time. A thousand commodity CPUs are enough to reduce the crack time from 24 years to a little over a week.

Whether the above numbers are precise or only approximate is immaterial. These attacks will only get better over time. The cost to crack shared secrets will only go down over time.

If the shared secret is long, then "cracking" the secret is expensive. It is cheaper to perform the BlastRADIUS attack at a cost of approximately 2^53 per packet, and less than $100 in purchased CPU time. While cracking the shared secret would break all RADIUS packets using that secret, forging one packet is often enough to give the attacker administrator access to a NAS, where the shared secret is visible in the administration interface. The conclusion, then, is that increasing the security of the shared secret offers minimal protection when the Access-Request packets are unsigned.

Even if the shared secrets were enough to secure all RADIUS packets, administrators do not always derive shared secrets from secure sources of random numbers. The "time to crack" numbers given above are the absolute best case, assuming administrators follow best practices for creating secure shared secrets. For shared secrets created manually by a person, the search space is orders of magnitude smaller than the best case outlined above. Rather than brute-forcing all possible shared secrets, an attacker can create a local dictionary which contains common or expected values for the shared secret. Where the shared secret used by an administrator is in the dictionary, the cost of the attack can drop by multiple orders of magnitude.

Implementers and administrators SHOULD assume that a hobbyist attacker with modest resource can crack most shared secrets created by people in minutes, if not seconds.

Despite the ease of attacking MD5, it is still a common practice for some "cloud" and other RADIUS providers to send RADIUS/UDP packets over the Internet "in the clear". It is also common practice for administrators to use "short" shared secrets, and to use shared secrets created by a person, or secrets derived from a limited character set. Theses practice are easy to implement and follow, but they are highly insecure and MUST NOT be used.

Further requirements in shared secrets are given below in Section 7.1.

3.5. Tunnel-Password and CoA-Request packets

There are a number of security problems with the use Tunnel-Password attribute in CoA-Request and Disconnect-Request packets. A full explanation requires a review of the relevant specifications.

[RFC5176] Section 2.3 describes how to calculate the Request Authenticator field for these packets:

Request Authenticator

> In Request packets, the Authenticator value is a 16-octet MD5 [RFC1321] checksum, called the Request Authenticator. The Request Authenticator is calculated the same way as for an Accounting-Request, specified in [RFC2866].

Where [RFC2866] Section 3 says:

> The NAS and RADIUS accounting server share a secret. The Request Authenticator field in Accounting-Request packets contains a one-way MD5 hash calculated over a stream of octets consisting of the Code + Identifier + Length + 16 zero octets + request attributes + shared secret (where + indicates concatenation). The 16 octet MD5 hash value is stored in the Authenticator field of the Accounting-Request packet.

Taken together, these definitions mean that for CoA-Request packets, all attribute obfuscation is calculated with the Reply Authenticator being all zeroes. In contrast for Access-Request packets, the Request Authenticator is mandated to be 16 octets of random data. This difference reduces the security of the obfuscation.

For Tunnel-Password, [RFC5176] Section 3.6 allows it to appear in CoA-Request packets:

> ...
> Change-of-Authorization Messages
>
> | Request | ACK | NAK | # | Attribute |
> |---|---|---|---|---|
> | ... | | | | |
> | 0+ | 0 | 0 | 69 | Tunnel-Password (Note 5) |
> | ... | | | | |
>
> (Note 5) When included within a CoA-Request, these attributes represent an authorization change request. Where tunnel attributes are included within a successful CoA-Request, all existing tunnel attributes are removed and replaced by the new attribute(s).

However, [RFC2868] Section 3.5 says that Tunnel-Password is encrypted with the Request Authenticator:

> Call the shared secret S, the pseudo-random 128-bit Request Authenticator (from the corresponding Access-Request packet) R,

The assumption that the Request Authenticator is random data is true for Access-Request packets. That assumption is not true for CoA-Request packets.

That is, when the Tunnel-Password attribute is used in CoA-Request packets, the only source of randomness in the obfuscation is the salt, as defined in [RFC2868] Section 3.5;

> Salt
> The Salt field is two octets in length and is used to ensure the uniqueness of the encryption key used to encrypt each instance of the Tunnel-Password attribute occurring in a given Access-Accept packet. The most significant bit (leftmost) of the Salt field MUST be set (1). The contents of each Salt field in a given Access-Accept packet MUST be unique.

This chain of unfortunate definitions means that there is only 15 bits of entropy in the Tunnel-Password obfuscation (plus the secret). It is not known if this limitation makes it sufficiently easy for an attacker to determine the contents of the Tunnel-Password, as the obfuscated value still depends on the shared secret. However, such limited entropy cannot be a good thing, and it is one more reason to deprecate RADIUS/UDP and RADIUS/TCP.

Due to the above issues, implementations and new specifications SHOULD NOT use obfuscated attributes in CoA-Request or Disconnect-Request packets.

3.6. TLS-based EAP methods, RADIUS/TLS, and IPsec

The above analysis as to security and privacy issues focuses on RADIUS/UDP and RADIUS/TCP. These issues are partly mitigated through the use secure transports, but it is still possible for information to "leak".

When TLS-based EAP methods such as TTLS or PEAP are used, they still transport passwords inside of the TLS tunnel. It is possible for an authentication server to terminate the TLS tunnel, and then proxy the inner data over RADIUS/UDP. The design of both TTLS and PEAP make this process fairly trivial. The inner data for TTLS is in Diameter AVP format, which can be trivially transformed to RADIUS attributes. The inner data for PEAP is commonly EAP-MSCHAPv2, which can also be trivially transformed to bare EAP, or to MS-CHAPv2.

Similar issues apply to RADIUS/TLS and IPsec. A proxy receiving packets over IPsec terminates the secure tunnel, but then might forward the packets over an insecure transport protocol. While this process could arguably be seen as a misconfiguration issue, it is never the less possible due to the design of the RADIUS protocol. The design of RADIUS security is that it is "hop by hop", and there is no way for one "hop" to know anything about, or to control, the security of another "hop".

The only solution to either issue would be to create a new protocol which is secure by design. Unfortunately that path is not possible, and we are left with the recommendations contained in this document.

3.7. Other Issues

There are many other issues with RADIUS which are unrelated to security or privacy. As of the time of writing this document, those issues are being collated in [ISSUES]. The bulk of the problems noted in that Wiki are operational considerations, along with inconsistencies in the previous RADIUS specifications.

As the focus of this document is security, it does not address problems with the RADIUS protocol in general. For example, there are known problems with the RADIUS state machine. There are common practices which are secure but which are operationally expensive. RADIUS accounting is known to be inaccurate and often inconsistent.

Some of the issues noted in the above Wiki could potentially have security impact. For example, if a RADIUS server is not implemented correctly, an attacker can perform a resource exhaustion attack on it, and effectively take it offline. Proxies are subject to Denial of Service attacks even from trusted clients, because those clients originate packets at the request of untrusted and unknown users. Rate limiting for RADIUS requests is a poorly tested or documented process, and largely relies on mutual trust of administrators.

We hope that the above issues are addressed in a future document. This document focuses on security issues related to transport protocols and authentication protocols, which will have to be sufficient for now.

4. All short Shared Secrets have been compromised

Unless RADIUS packets are sent over a secure network (IPsec, TLS, etc.), administrators SHOULD assume that any shared secret of 8 characters or less has been compromised as soon as it is used. Administrators SHOULD assume that any shared secret of 10 characters or less has been compromised by an attacker with significant resources. Administrators SHOULD also assume that any private information (such as User-Password) which depends on such shared secrets has also been compromised.

To be perfectly clear: if a User-Password, or CHAP-Password, or MS-CHAP password has been sent over the Internet via RADIUS/UDP or RADIUS/TCP in the last decade, you should assume that the underlying password has been compromised.

5. The BlastRADIUS Attack

This section gives some more detail on the attack, so that the reader can be informed as to why this document makes particular recommendations.

An "on path" attacker can inject one or more Proxy-State attributes with special contents into an Access-Request packet. The Proxy-State attribute itself will not trigger any overflow or “out of bounds” issue with the RADIUS client or server. Instead, the contents of the attributes will allow the attacker to create an MD5 known-prefix collision when the server calculates the Response Authenticator. In effect, the attacker uses the RADIUS server, and its knowledge of the shared secret, to unknowingly authenticate packets which it has not created.

The behavior of the Proxy-State attribute is extremely useful to this attack. The attribute is defined in [RFC2865], Section 5.33 as an opaque token which is sent by a RADIUS proxy, and is echoed back by RADIUS servers. That is, the contents of the attribute are never examined or interpreted by the RADIUS server. Even better, testing shows that all RADIUS clients will simply ignore any unexpected Proxy-State attributes which they receive. Finally, implementations generally add Proxy-State to the end of response packets, which simplifies the attack.

This attribute is therefore ideally suited to an attackers purpose of injecting arbitrary data into packets, without that data affecting client or server behavior. The reasons for this behavior are outlined below in Section 5.6. While those reasons ended up being transient, the impact of those decisions has impacted RADIUS for decades.

While it is possible to use other attributes to achieve the same effect, the use of Proxy-State is simple, and is sufficient to trigger the issue. For example, it is theoretically possible to use the User-Name attribute for this attack, if it is echoed back in an Access-Accept, or even as part of the the contents of a Reply-Message in an Access-Accept. There is no much benefit in further researching that attack, as the mitigations for attacks using Proxy-State will also protect clients and servers from a similar attacks which use other attributes.

The injected data and resulting MD5 collision allows the attacker to modify the packet contents almost at will, and the client will still accept the modified packet as being authentic. The attack allows nearly arbitrary attributes to be added to the response. Those attributes are simply part of the MD5 collision calculation, and do not substantially impact the cost of that calculation.

We reiterate that since the RADIUS server can be convinced to authenticate packets using a prefix chosen by the attacker, there is no need for the attacker to know the shared secret. This attack succeeds no matter how secure the shared secret is, the only mitigation is to use Message-Authenticator or TLS.

The attack is implemented via the following steps, which are numbered the same as in the original paper.

1. The attacker requests network access from the RADIUS client (NAS). This action triggers the NAS to send an Access-Request packet to the RADIUS server.

2. The Access-Request is observed to obtain its contents, including the Request Authenticator field. The attacker prevents this packet from reaching the server until the MD5 collision data has been calculated.. The NAS will retransmit the packet one or more times after a delay, giving the attacker time to calculate the chosen prefix.

3. Some external resources are used to calculate an MD5 collision using the Request Authenticator, and the expected contents of an Access-Reject. As Access-Reject packets are typically empty (or can be observed), the expected packet contents are known in their entirety.

4. Once an MD5 collision is found, the resulting data is placed into one or more Proxy-State attributes in the previously seen Access-Request. The attacker then sends this modified Access-Request to the RADIUS server.

5. The RADIUS server responds with an Access-Reject, and includes the Proxy-State attributes from the modified Access-Request packets. The packet contains the malicious Proxy-State(s), along with a Response Authenticator which depends on both those malicious attributes, and the shared secret.

6. The attacker discards the original Access-Reject, and uses the chosen prefix data in the Proxy-State(s) to create a different (i.e. modified) response, such as an Access-Accept. Other authorization attributes such as VLAN assignment can also be add, modified, or deleted. This modified packet is sent to the NAS.

7. The NAS receives the modified Access-Accept, verifies that the Response Authenticator is correct, and gives the user access, along with the attackers desired authorization.

The result of this attack is a near-complete compromise of the RADIUS protocol. The attacker can cause any user to be authenticated. The attacker can give almost any authorization to any user.

While the above description uses Access-Reject responses, we reiterate that the root cause of the vulnerability is in the Access-Request packets. The attack will therefore succeed even if the server responds with Access-Accept, Access-Challenge, or Protocol-Error [RFC7930]. The vulnerability in Access-Challenge allows MFA to be bypassed, as the attacker simply replaces the Access-Challenge with an Access-Accept.

In addition to forging an Access-Accept for a user who has no credentials, the attacker can control the traffic of known and authenticated users. Many modern Broadband Network Gateways (BNG)s, Wireless LAN Controllers (WLCs), and Broadband Remote Access Servers (BRAS) support configuring a dynamic HTTP redirect using Vendor Specific Attributes (VSA)s. These VSAs are not protected by the shared secret, and could be injected into an Access-Accept in order to redirect a users traffic. The attacker could then set up a malicious website to launch Zero-Day/Zero-Click attacks, driving

subscribers to the website using an HTTP redirect. This issue is compounded by the fact that many devices perform automatic HotSpot 1.0 style walled garden discovery. The act of simply connecting to their home WiFi connect could be enough to compromise a subscriber's equipment.

The following subsections define mitigations which can be used to protect clients and servers from this attack when using RADIUS/UDP or RADIUS/TCP. However, we reiterate here, and in the rest of this document that the only long-term solution is to deprecate insecure transports entirely. In the long term, implementers SHOULD remove all uses of RADIUS/UDP and RADIUS/TCP from their products. Administrators SHOULD stop using RADIUS/UDP and RADIUS/TCP.

5.1. Root Cause of the Attack

Independent of any cryptographic vulnerability, there are a number of factors which led to this vulnerability being exploited.

A major factor is the continued use of MD5 for security, instead of mandating the use of an HMAC with Message-Authenticator. This change could have been made in [RFC2869] in the year 2000. A reason for not mandating Message-Authenticator was the issue of backwards compatibility. Unfortunately, issues which are not fixed only grow larger over time. The issue of backwards compatibility is significantly worse now than it was in the year 2000. History shows that a better approach is to fix issues immediately when they come up.

Another factor is the principle of "be conservative in what you do, be liberal in what you accept from others", often known as Postel's law, after John Postel. Looking at the Proxy-State attribute, it is intended for proxy to server signaling, and offers no other value for RADIUS clients. A NAS does not send Proxy-State in an Access-Request, and should not receive Proxy-State in an Access-Accept.

Reception of Proxy-State in an Access-Accept response is therefore a failure of signaling in the RADIUS protocol, and likely indicates a serious failure of configuration, implementation, or as seen in this case, an active attack. If all clients had discarded responses which contained unexpected Proxy-State attributes, then this attack could have been prevented.

With the benefit of experience, this specification errs on the side of security, while still allowing for backwards compatibility. It is not acceptable to maintain insecure practices simply because a small number of implementations or organizations find it difficult to upgrade. Insecure implementations or practices have a concrete cost

not only to the insecure organizations, but also to other organizations via secondary effects. When insecure organizations demand that others follow insecure practices continue due to perceived local costs, they are effectively offloading their costs onto everyone else. This practice both decreases security, and increases costs.

The solution here is to fix the protocol, and to mandate that everyone change to secure practices. Implementations and/or organizations which do not update their systems will either be insecure, or will be incompatible with secure practices.

5.2. Interaction with DTLS and TLS

The new behaviors described in this section apply only when UDP or TCP transport is used. Implementations MUST NOT use the mitigations outlined in this section when DTLS or TLS transport is used, as those transport protocols are not vulnerable to the BlastRADIUS attack.

However, clients and servers SHOULD include Message-Authenticator in Access-Request packets, and in responses to those Access-Requests, even then those packets are sent over TLS or DTLS transports. While the attribute serves no security purpose, we recommend including Message-Authenticator because of the behavior of some implementations. Some RADIUS proxies are known to automatically include Message-Authenticator in forwarded packets when the attribute is seen in the packet received from the client.

That is, such an implementation may receive packets over TLS transport, and then forward (or return) them over insecure transports such as UDP. If such an implementation receives a packet over TLS which contains Message-Authenticator, it will add the Message-Authenticator attribute to the packet which is sent over UDP. Including Message-Authenticator in packets sent over insecure transports is important for increasing RADIUS security.

It is therefore useful for systems to always include Message-Authenticator in Access-Request packets, even when TLS is used. While this practice does not increase the security of a particular TLS connection, it will increase the security of the larger RADIUS system.

The long term solution to the BlastRADIUS attack is to stop using insecure transport protocols, and switch to secure ones such as TLS. We recognize that switching to TLS transport may require a significant amount of effort. There is a substantial amount of work to perform in updating implementations, performing interoperability tests, changing APIs, changing user interfaces, and updating documentation. This effort cannot realistically be done in a short time frame.

There is therefore a need for a short-term action which can be implemented by RADIUS clients and servers which is both simple to do, and which is known to be safe. The recommendations in this section are known to protect implementations from the attack; to be simple to implement; and also to allow easy upgrade without breaking existing deployments.

5.3. Changes to RADIUS

There are a number of changes required 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 the partial mitigations, and therefore still perform the attack.

This section outlines the mitigation methods which protect RADIUS/UDP and RADIUS/TCP systems from this attack, along with the motivation for those methods.

We note that unless otherwise noted, the discussion here applies 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.

The mitigation methods outlined here allow systems to both protect themselves from the attack, while not breaking existing networks. There is no global “flag day” required for these changes. Systems which implement these recommendations are fully compatible with legacy RADIUS implementations, and can help protect those legacy implementations. However, when these mitigations are not implemented, systems are still vulnerable to the attack.

Note that in some network architectures, 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 a limited subset. We therefore mandate new behavior for all RADIUS clients and servers, while acknowledging that some organizations may choose to

not deploy all of the new functionality. For overall network security and good practice, we still recommend 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.

5.3.1. New Configuration Flags

We define new configuration flags for clients and servers. The behavior and meaning of these flags will be discussed below. Introducing these flags before discussing their meaning makes the subsequent discussion simpler and easier to understand.

The goal of these flags is to secure the RADIUS protocol without preventing client and server communication between legacy and upgraded systems. These flags instead allow a gradual migration process from legacy RADIUS, to fully secure RADIUS with all of the mitigations in place.

We mandate the following new configuration flags for RADIUS implementations when using UDP or TCP transport. These flags MUST be ignored when DTLS or TLS transport is used.

- Clients MUST have a per-server boolean configuration flag, which we call "require Message-Authenticator". The default value for this flag MUST be "false" in order to maintain compatibility with legacy servers.

- Servers MUST have a per-client boolean configuration flag, which we call "require Message-Authenticator". The default value for this flag MUST be "false" in order to maintain compatibility with legacy clients.

- Servers MUST have a per-client boolean configuration flag, which we call "limit Proxy-State". The default value for this flag MUST be "false" in order to maintain compatibility with 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 more specific setting defined.

The combination of global and more narrow configuration allows administrators to upgrade systems gradually, without requiring a "flag day" when everything changes on a global basis.

5.3.2. Clients and Access-Request

We mandate the following new behavior for RADIUS 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, so 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, as discussed below for responses to Access-Request (Access-Accept, etc.), the location of Message-Authenticator does matter. It is better to have consistent and clear messaging for addressing this attack, instead of having different recommendations for different kinds of packets.

All RADIUS servers will validate the Message-Authenticator attribute correctly when that attribute is received in a packet. We are not aware of any RADIUS servers which will reject or discard Access-Request packets if they unexpectedly contain a Message-Authenticator attribute.

This behavior has been enabled in the FreeRADIUS server for over a decade, and there have been no reports of interoperability problems. It is therefore safe for all clients to immediately implement this requirement.

However, many existing RADIUS clients do not 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 to both work with such systems by not breaking existing RADIUS deployments, while at the same time protecting them as much as practically possible.

5.3.3. Servers and Access-Request

We mandate the following new behavior for RADIUS servers:

> When receiving an Access-Request, 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 "require Message-Authenticator" is set to “false”, RADIUS 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 "require Message-Authenticator" is set to "false", RADIUS 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 5.3.4, below.
>
> If "require Message-Authenticator" is set to “true”, the server MUST examine the 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 the Message-Authenticator and then discard packets which fail this validation ([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 for authenticated packets.

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

When the "require Message-Authenticator" flag is set to "true", the server is protected from the BlastRADIUS attack. 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.

While servers must validate the contents of Message-Authenticator, we reiterate that they MUST NOT check the location of that attribute. There is no different meaning in RADIUS if Message-Authenticator is the first, second, or last attribute in a packet. Servers MUST accept a RADIUS packet as valid if it passes authentication checks, no matter where Message-Authenticator is located in the packet.

Unfortunately, there is no way for clients and servers to negotiate protocol-layer features in RADIUS/UDP or RADIUS/TCP. An implementation cannot determine if packets are discarded due to an attack, or if they are discarded due to a mismatched configuration between client and server. Implementations SHOULD therefore log the fact that the 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.

As a special case for debugging purposes, instead of discarding the packet, servers MAY instead send a Protocol-Error or Access-Reject response packet. This packet MUST contain a Message-Authenticator attribute as the first attribute in the packet, otherwise an attacker could turn 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 it this could cause the server to respond to forged Access-Request packets.

The purpose of this Protocol-Error packet 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 the 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.

We note that "Section 7.2 of [BLAST] has the following comment about the FreeRADIUS server, which has had this configuration option for clients since 2008:

> If support for these old clients is not required, enabling this option would make our attacks infeasible.

The next question is how to protect systems when legacy clients do not send Message-Authenticator.

5.3.4. Updated Servers and Legacy Clients

We mandate the following new behavior for RADIUS servers:

> When receiving an Access-Request which passes the above checks and 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 "limit Proxy-State" is set to "false", RADIUS 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 "limit Proxy-State" is set to "true", RADIUS 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 most 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, and the attack is essentially impossible without using Proxy-State in an Access-Request.

It is therefore safe 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 packet should be accepted and processed by the server.

On the other hand, if the packet cannot be authenticated via the use of 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, subject to the caveats and considerations described in the previous section for those responses.

5.3.5. Further Explanation and Comments

The "require Message-Authenticator" flag is needed in order to secure the RADIUS protocol. Once all Access-Request packets are required to contain a valid Message-Authenticator, the BlastRADIUS attack is impossible.

However, it may not be possible to upgrade all RADIUS clients. Some products may no longer be supported, or some vendors have gone out of business. Even if upgrades are available, the upgrade process may impact production networks, which has a cost. There is therefore a need for RADIUS servers to protect themselves from to the BlastRADIUS attack, while at the same time being compatible with legacy RADIUS client implementations.

Enabling the "limit Proxy-State" flag allows legacy (i.e. non-upgraded) clients to be used without substantially compromising on security. While it is theoretically possible to perform the BlastRADIUS attack via attributes other than Proxy-State, no such exploits are known at this time. Any such exploit would require that the server receive fields under the attackers control (e.g. User-Name), and echo them back in a response. Such attacks are only possible when the server is configured to echo back attacker-controlled data, which is not the default behavior for most servers.

As a result, these two flags allow the maximum amount of security while having the minimum disruption to operational networks. For the remaining attack vectors, is is RECOMMENDED that servers which echo back user-supplied data in responses do so only when the "require Message-Authenticator" flag is set to "true". If such user-supplied data is echoed back in responses when the "require Message-Authenticator" flag is set "false", then the BlastRADIUS attack is theoretically still possible, even though no exploit is currently available.

The two configuration flags on the server will protect the server even if clients have not been upgraded or been configured to be secure. The server configuration flags will not protect clients (NASes or proxies) from servers which have not been upgraded or been configured to be secure. More behavior changes to servers and clients are required.

5.3.6. Server Responses to Access-Request

We mandate the following behavior for servers when sending 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.

The client will validate the Response Authenticator ([RFC2865], Section 3) before accepting response, and the attacker is unable to modify the response due to the location of the Message-Authenticator. This behavior therefore protects one client to server hop, even if the server does not require Message-Authenticator in Access-Request packets, and even if the client does not examine or validate the contents of the Message-Authenticator.

When the Message-Authenticator is the last attribute in a packet (as was historically common in many implementations), the attacker can treat the Message-Authenticator as an unknown suffix, as with the shared secret. The attacker can then calculate the prefix as before, and have the RADIUS server authenticate the packet which contains the prefix.

The analysis is similar if the Message-Authenticator is in the middle of the packet, with attributes existing both before an after the Message-Authenticator. Attributes before the Message-Authenticator can be modified, discarded, or added, while attributes after the Message-Authenticator need to remain in the packet. We direct the reader to Section 7.2 of [BLAST] for a more complete description of these issues.

As a result, adding a Message-Authenticator anywhere other than as the first one will only mitigate the attack if the client implements the "require Message-Authenticator" flag, and has set that flag to "true". Since there is no feature negotiation in RADIUS, the server has no way of knowing the clients configuration, and therefore needs to behave as if the client has the most insecure configuration.

The location of the Message-Authenticator attribute is therefore critical to protect legacy clients which do not check for its existence or validate its contents. Many legacy clients do not send Message-Authenticator in Access-Request packets, and therefore are highly likely to not validate it in responses to those Access-Requests. Upgrading all of these clients may be difficult, or in some cases impossible. It is therefore important to have mitigation factors which protect those systems.

We note that Message-Authenticator has been defined for almost twenty-five (25) years, since [RFC2869]. All standards-compliant clients will validate Message-Authenticator; or if they do not

validate it, should ignore it. Since the publication of the original BlastRADIUS notification, it has become clear that some implementations do not behave as expected. That is, they discard packets which contain an unexpected Message-Authenticator attribute, even though that behavior is entirely unreasonable, and is not required by any existing standard.

There is very little to be done for such systems. The behavior mandated by this specification causes problems for such systems. The only way for RADIUS servers to be compatible with those systems is to never send Message-Authenticator in responses. However, doing so would open up significantly more systems to the BlastRADIUS attack.

In the end, the only safe solution is to declare that systems which discard packets containing Message-Authenticator are not compliant with the RADIUS specifications. We do not decrease the security of the RADIUS protocol in order to allow the continued existence of insecure and non-compliant implementations. In order to prevent such issues from happening in the future, we now define mandated behavior for unknown attributes in Section 8.12, below. In short, there is no reason for implementations to discard response packets, simply because they do not recognize an attribute contained therein.

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.

5.3.7. Clients Receiving Responses

We mandate the following new behavior for RADIUS clients:

> When receiving any response to an Access-Request packet (Access-Accept, Access-Challenge, Access-Reject, or Protocol-Error), clients MUST consult the value of 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 "require Message-Authenticator" is set to “false”, RADIUS 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 "require Message-Authenticator" is set to “true”, the client MUST examine the response packets for the existence of the Message-Authenticator attributes. 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. 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 for authenticated packets.

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

Unfortunately, the client cannot determine if a packet was discarded due to an active attack, or if it was discarded due to a mismatched configuration between client and server (e.g. mis-matched shared secret). The client SHOULD log the fact that the 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.

The above sections have now followed a complete path from client, to server, and back again. If each client to server hop is secured via the above methods, then by construction, the RADIUS protocol is no longer vulnerable to the BlastRADIUS attack.

5.3.8. Status-Server

While the 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.

5.4. Related Issues

This section contains discussions of issues related to the BlastRADIUS vulnerability which do not involve changes to the RADIUS protocol. It also explains why some solutions which may seem appealing are instead inadequate or inappropriate.

5.4.1. Documentation and Logging

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

Similarly, when either of the above flags cause a packet to be discarded, the RADIUS server SHOULD log a descriptive message (subject to rate limiting) about the problematic packet. This log is extremely valuable to administrators who wish to determine if anything is going wrong, and what to do about it.

5.4.2. Alternative Solutions

An alternative configuration flag with a similar effect to the “limit Proxy-State” flag could be one called “this client is a NAS, and will never send Proxy-State”. The intention for such a flag would be to clearly separate RADIUS proxies (which always send Proxy-State), from NASes (which will never send Proxy-State). When the flag is set for a client, the server could then discard Access-Request packets which contain Proxy-State. Alternatively, the server could also discard Proxy-State from all responses sent to that client.

Such a flag, however, depends on network topology, and fails to correct the underlying lack of packet authenticity and integrity. The flag may also work for one NAS, but it is likely to be incorrect if the NAS is replaced by a proxy. Where there are multiple different pieces of NAS equipment behind a NAT gateway, the flag is also likely to be correct for some packets, and incorrect for others.

Setting configuration flags by the desired outcome is preferable to setting flags which attempt to control network topology.

5.4.3. Non-Mitigations

It may be tempting to come up with other "ad hoc" solutions to this vulnerability. Such solutions are NOT RECOMMENDED, as they are likely to either break existing RADIUS deployments, or else they will not prevent the attack. The mitigations described in this document not only prevent the attack, they do so without affecting normal RADIUS operation. There is therefore no reason to use any other methods.

Other attempted mitigation factors are discussed in the "BlastRADIUS" document. For example, "BlastRADIUS" Section 7.4 explains why decreasing timeouts simply increases the cost of the attack without preventing it. Decreasing timeouts also can negatively affect normal traffic.

[BLAST] Section 7.7 explains why clients validating Proxy-State, or looking for unexpected Proxy-State does not protect from the attack. The attacker can likely just change the nature of the attack, and bypass those checks.

There is no reason to implement “ad hoc” solutions when a solution exists which has passed reviews by both the BlastRADIUS cryptographers, and by the IETF RADEXT working group. There is every reason to believe that cryptographic operations designed by experts and subject to rigorous peer review are better than random guesses made by programmers lacking relevant cryptographic and RADIUS experience.

Similarly, switching away from RADIUS to another protocol will not protect from the attack, as there is no other protocol which can replace RADIUS. No other protocol is supported by medium to low-end networking devices for end-user authentication, authorization, and accounting. Outside of situations where Diameter is used, the choice for nearly every use-case which controls network access is limited to one protocol: RADIUS.

Despite this reality, some "security" sites have recommended "securing" the network by switching to "alternative" protocols. Such recommendations are incorrect and inappropriate.

Diameter [RFC6733] is the closest protocol in functionality to RADIUS, but the Diameter use-case is applicable to large-scale telecommunications and internet service providers (ISPs). Support for Diameter is not available in equipment available to consumers or enterprises. As such, replacing RADIUS with Diameter is not an option.

Other proposals for protocols to replace RADIUS are even less effective. TACACS+ [RFC8907] has some overlap with RADIUS for administrator login to network devices, but it cannot be used outside of that limited scope. TACACS+ does not support 802.1X, end-user authentication, or end-user accounting. It is therefore impossible for an ISP or enterprise to replace RADIUS with TACACS+.

Kerberos [RFC4120] is also not a option. It is most generally used to authenticate applications, when the underlying system already has network access. Kerberos also does not support 802.1X, and does not support accounting.

The situation is much the same with any proposal to replace RADIUS with IPsec. While IPsec does authenticates devices prior to bringing up the VPN, those devices must already have network access. IPsec also requires that the end-user traffic be transported over the IPsec connection, where RADIUS does not transport any end-user traffic.

In conclusion, recommendations to use alternate protocols are, at best, misguided. We do not recommend following "security" advice which is based on a fundamental misunderstanding of networking protocols.

5.4.4. Network Operators

The most important outcome of this attack for network operators is that where possible, all RADIUS traffic should use TLS transport between client and server.

Methods other than IPsec to mitigate the attack are less secure, they still fail at adding privacy, and are therefore less useful. We recognize that not all networking equipment supports TLS transport, so we therefore give additional recommendations here which operators can follow to help mitigate the attack.

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.

It is RECOMMENDED that all RADIUS traffic be sent over a management VLAN. 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 performing this attack. We say “generally”, because security is limited by the least secure part of the network. If a network device has some unrelated vulnerability, then an attacker could exploit that vulnerability to gain access to the management network. The attacker would then be free to exploit this issue.

Only the use of TLS will prevent such attacks from being chained together.

Similarly, there are few reasons to use RADIUS/TCP. Any system which supports RADIUS/TCP likely also supports TLS, and that should be used instead.

Finally, any RADIUS/UDP or RADIUS/TCP traffic MUST NOT be sent over public networks such the Internet. This issue is discussed in more detail later in this document.

5.5. Limitations of the Mitigations

The above mitigations have some limitations. The design of the mitigations had to allow for backwards compatibility with legacy RADIUS systems, while still allowing for (but not requiring) wholesale network upgrades. There is a trade-off to be made between perfectly secure networks which are unusable, and networks which are usable but somewhat insecure. The mitigations outlined here create as much security as possible, while still not breaking existing networks.

The result is that there are situations where a network is functional, but insecure. This section outlines those limitations.

5.5.1. Vulnerable Systems

A RADIUS server is vulnerable to the attack if it does not require that all received Access-Request packets contain a Message-Authenticator attribute. This vulnerability exists for many common uses of Access-Request, including packets containing PAP, CHAP, MS-CHAP, or packets containing “Service-Type = Authorize-Only”. The vulnerability is also transitive. If any one RADIUS server in a proxy chain is vulnerable, then the attack can succeed, and the attacker can gain unauthenticated and/or unauthorized access.

Simply having the Message-Authenticator attribute present in Access-Request packets is not sufficient. In order to be protected, a server must require that the attribute is present, and discard packets where it is missing. Similarly, the client must also require that the attribute is present, and discard packets where it is missing.

The attack is fully mitigated only when both sides of the RADIUS conversation are updated and configured correctly.

5.5.2. Unaffected Systems

There are a number of systems which are not vulnerable to this attack. The most important ones are systems which only perform EAP authentication, such as with 802.1X / WPA Enterprise. The EAP over RADIUS protocol is defined in [RFC3579], Section 3.3 which states explicitly:

> If any packet type contains an EAP-Message attribute it MUST also contain a Message-Authenticator.

This requirement reiterates that of [RFC2869], Section 5.13, which defines EAP-Message and Message-Authenticator, but which does not get into details about EAP.

This requirement is enforced by all known RADIUS servers. As a result, when roaming federations such as eduroam use RADIUS/UDP, it is not possible for the attack to succeed.

Other roaming groups such as OpenRoaming require the use of TLS, and are not vulnerable. Other roaming providers generally use VPNs to connect disparate systems, and are also not vulnerable.

802.1X / WPA enterprise systems have an additional layer of protection, due to the use of the master session keys (MSK) which are derived from the EAP authentication method. These keys are normally carried in an Access-Accept, in the MS-MPPE-Recv-Key and MS-MPPE-Send-Key attributes, and are used to secure the link between the NAS and the supplicant. The contents of the attributes are obfuscated via the same method used for Tunnel-Password, and are not visible to an "on-path" attacker.

While an attacker can perhaps force an Access-Accept in some situations, or strip the Message-Authenticator from packets, it is not currently possible for an attacker to see, modify, or create the correct MSK for the EAP session. As a result, when 802.1X / WPA enterprise is used, even a successful attack on the Access-Accept packet would likely not result in the attacker obtaining network access.

5.5.3. The Weakest Link

RADIUS security is done on a “hop by hop” basis, which means that an attacker can take advantage of the weakest link in a proxy chain in order to attack other systems which have fully implemented the above mitigations. If the packets are passed through one or more proxies, then any one vulnerable proxy will still allow the attack to take place.

If proxies are used, then the weakest link in the proxy chain limits the security of the entire chain. That is, it does not matter if one hop implements RADIUS/TLS, if another hop implements RADIUS/UDP without sending or requiring Message-Authenticator.

Even worse, proxies have full control over packet contents. A malicious proxy can change a reject into an accept, and can add or delete any authorization attributes it desires. While proxies are generally part of a trusted network, there is every benefit in limiting the number of participants in the RADIUS conversation.

Proxy chains SHOULD therefore be avoided where possible, and [RFC7585] dynamic discovery should be used where possible. RADIUS clients and servers SHOULD also be configured with static IP addresses, and with static routes. This static configuration also protects the systems from DHCP related attacks where an attacker spoofs DHCP to cause clients or servers to route packets through the a system of the attackers choice.

5.6. Note on Proxy-State

As the BlastRADIUS paper points out in Appendix A:

> The presence of this attribute makes the protocol vulnerability much simpler to exploit than it would have been otherwise.

To see why Proxy-State has this particular design, we go back to the original discussion in May 1995 [MAY-1995]

> The RADIUS proxy may place any state information (subject to the length limitations of a RADIUS attribute) that it will need to transform a reply from its server into a reply to its client. This is typically the original authenticator, identifier, IP address and UDP port number of the proxy's RADIUS client.

There appear to be few, if any, RADIUS servers which implemented this suggestion. In part because later discussions note:

> This works only if the NAS is prepared to accept replies from a proxy server for a request issued to a different server.

This stateless proxy design has a number of additional issues, most notably violating the [RFC3539] "end-to-end" principle. It therefore negatively impacts the stability of a RADIUS proxy system.

This definition for Proxy-State later changed in [RFC2865], Section 5.33 to

> Usage of the Proxy-State Attribute is implementation dependent. A description of its function is outside the scope of this specification.

In practice, the utility of Proxy-State is limited to detecting proxy loops. Proxies can count the number of Proxy-State attributes in received packets, and if the total is more than some number, then a proxy loop is likely. We offer no advice on what to do if a proxy loop is detected, as RADIUS has no ability to signal protocol-layer errors.

It is likely that a "hop count" attribute would likely have been simpler to implement, but even in 1996, it was likely difficult to change the behavior of proxies due to multiple implementations.

5.7. Intrusion Detection Systems

Intrusion detection systems can be updated to detect and/or warn about the attack with the following rules. In the interests of brevity and generality, the rules are written as plain text.

1. Access-Request does not contain a Message-Authenticator attribute.

   Action: Warn the administrator that the system is vulnerable, and should be upgraded.

2. Access-Accept, Access-Reject, or Access-Challenge does not contain a Message-Authenticator attribute.

   Action: Warn the administrator that the system is vulnerable, and should be upgraded.

3. Access-Accept, Access-Reject, or Access-Challenge contains a Message-Authenticator attribute, but it is not the first attribute in the packet.

   Action: Warn the administrator that the system may be vulnerable, and should be upgraded.

4. Access-Request packet received by a RADIUS server contains Proxy-State, when the RADIUS client is a NAS.

   Action: Alert that an attack is likely taking place.

   Note that the check should be for packets received by the RADIUS server, and not for packets sent by the NAS. The attack involves packets being modified after they are sent by the NAS, and before they are received by the RADIUS server.

5. Access-Accept, Access-Reject, or Access-Challenge sent by a RADIUS server contain Proxy-State, when the RADIUS client is a NAS.

   Action: Alert that an attack is likely taking place.

Note that the check should be for packets sent by the RADIUS server, and not for packets received by the NAS. The attacker can modify packets to "hide" Proxy-State in another attribute, such as Vendor-Specific.

6. Any RADIUS traffic is sent over UDP or TCP transport, without IPsec or TLS.

   Action: Warn that the system uses deprecated transport protocols, and should be upgraded.

7. Any RADIUS traffic is sent external to the organization over UDP or TCP transport, without IPsec or TLS.

   Action: Warn that this is an insecure configuration, and can expose users private data, identities, passwords, locations, etc. to unknown attackers.

These rules should assist administrators with ongoing security and monitoring.

6. Deprecating Insecure Transports

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.

6.1. Deprecating RADIUS/UDP and RADIUS/TCP

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 TLS 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.

6.2. Mandating Secure transports

All systems which send RADIUS packets outside of secure networks MUST use either IPsec, RADIUS/TLS, or RADIUS/DTLS. For operational and security reasons, it is RECOMMENDED to use RADIUS/TLS or RADIUS/DTLS instead of IPsec.

Unlike (D)TLS, use of IPsec means that applications are generally 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: packets can be sent over an insecure link, and the RADIUS server is unaware of the failure of the security layer.

In contrast, (D)TLS gives the RADIUS application completely knowledge and control over transport-layer security. The failure cases around (D)TLS 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.

6.3. Crypto-Agility

The crypto-agility requirements of [RFC6421] are addressed 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 TLS or DTLS 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 TLS and DTLS 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.

We can now finalize the work began in [RFC6421]. This document updates [RFC2865] to state that any new RADIUS specification MUST NOT introduce new "ad hoc" cryptographic primitives to authenticate packets as was done with the Request / Response Authenticator, or to obfuscate attributes as was done with User-Password and Tunnel-Password. We allow legacy RADIUS-specific cryptographic methods existing as of the publication of this document to be used for historical compatibility. However, all new cryptographic work which is specific to the RADIUS protocol is forbidden.

We recognize that RADIUS/UDP will still be in use for many years, and that new standards may require some modicum of privacy. As the BlastRADIUS attack shows, RADIUS/UDP security is inadequate. The solution is not to fix RADIUS/UDP. The solution is to deprecate it entirely.

All new security and privacy requirements in RADIUS 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 forbidding 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. There would be no need for the RADIUS protocol to define any new cryptographic methods in order to transport this data.

Similarly, new specifications MAY define new attributes which use the obfuscation methods for User-Password as defined in [RFC2865] Section 5.2, or for Tunnel-Password as defined in [RFC2868] Section 3.5. However, due to the issues noted above in Section 3.5, the Tunnel-Password obfuscation method MUST NOT be used for packets other than Access-Request, Access-Challenge, and Access-Accept. If

the attribute needs to be send in another type of packet, then the protocol design is likely wrong, and needs to be revisited. It is again a difficult choice to forbid certain uses of the Tunnel-Password obfuscation method, but we believe that doing so is preferable to allowing sensitive data to be obfuscated with less security than the original design intent.

7. Migration Path and Recommendations

We recognize that it is difficult to upgrade legacy devices with new cryptographic protocols and user interfaces. The problem is made worse because of 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 takes significant time and effort to correct the deficiencies of all of them.

We therefore need to define a migration path to using 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. We also mandate the use of Message-Authenticator in all Access-Request packets for RADIUS/UDP and RADIUS/TCP. 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.

7.1. 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 node, RADIUS 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. Message-Authenticator

The Message-Authenticator attribute was defined in [RFC3579] Section 3.2. The "Note 1" paragraph at the bottom of [RFC3579] Section 3.2 required that Message-Authenticator be added to Access-Request packets when the EAP-Message as present, and suggested that it should be present in a few other situations. The BlastRADIUS attack has shown that these recommendations are inadequate.

While the text in Section 5 goes into detail about the mitigations, we summarize them here. Please see the previous section for a longer and more detailed explanation of the flags and their behavior.

> RADIUS clients MUST include the Message-Authenticator in all Access-Request packets when UDP or TCP transport is used.
>
> RADIUS clients and servers MUST have a boolean flag which we call "require Message-Authenticator".
>
> When set to "true", implementations receiving Access-Request, Access-Accept, Access-Challenge, Access-Reject, and Protocol-Error packets which do not contain Message-Authenticator MUST silently discard those packets.
>
> Servers MUST have a boolean flag which we call "limit Proxy-State".
>
> This flag MUST be examined when the "require Message-Authenticator" flag is set to "false".
>
> When set to "true", servers receiving an Access-Request packet which contains Proxy-State but no Message-Authenticator MUST discard that packet.

In contrast, when TLS-based transports are used, the Message-Authenticator attribute serves no purpose, and could be omitted, even when the Access-Request packet contains an EAP-Message attribute. However, implementations SHOULD include it, even if it servers no immediate purpose. As noted earlier, including Message-Authenticator can increase the security of legacy proxies which do not implement the BlastRADIUS mitigations.

Servers receiving Access-Request packets over TLS-based transports SHOULD NOT silently discard a packet if it is missing a Message-Authenticator attribute. However, if the Message-Authenticator attribute is present, it still MUST be validated as discussed in [RFC7360].

7.3. 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].

7.4. Guidelines for Administrators

The above text provides guidelines for implementers. We also need to provide guidelines for administrators as to how, and when, to configure the above flags. The guidelines provided in this section are suggestions only. Administrators are free to take other actions as they see fit.

However, 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 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 (4) 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 receiving 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.

8. Increasing the Security of RADIUS

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.

8.1. Minimizing 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.

8.1.1. 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.

8.2. User-Password Visibility

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.

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

Organizations MAY still use User-Password attributes within their own systems, for reasons which we will explain below in Section 8.7.

8.3. Delaying Access-Rejects

Anyone can cause a NAS to send Access-Request packets at will, simply by attempting to requesting network access, or login permissions from the NAS. If this login process is not rate-limited, it can be abused by an attacker to perform dictionary attacks.

In order to prevent these brute-force attacks, servers which originate Access-Reject packets MUST enforce a minimum delay between reception of the Access-Request, and transmission of a corresponding Access-Reject. This delay SHOULD be configurable. Experience shows that values between one (1) second and ten (10) seconds work well in practice.

Systems which simply proxy Access-Reject packets MUST NOT add any artificial delays to those packets. Doing so would result in delays accumulating across a chain of proxies.

Servers SHOULD also add a small random jitter to the delay for a particular packet, in order to better protect themselves from timing attacks.

8.4. 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).

8.5. 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] 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.

8.5.1. 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.

8.5.2. 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. Administrators SHOULD limit the source IP of allowed connections. Server SHOULD filter received packets 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.

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.

8.6. Do Not Use MS-CHAP

MS-CHAP (v1 in [RFC2433] and v2 in [RFC2759]) have major design flaws, and should not be used outside of a secure tunnel such as PEAP or TTLS. As MS-CHAPv1 is less commonly used, the discussion in this section will focus on MS-CHAPv2, but the same analysis applies to MS-CHAPv1.

MS-CHAP has been broken since 2004, as seen in [ASLEAP]. While the attack there mentions LEAP, the same attack applies to MS-CHAP. This information was apparently insufficiently clear in the [ASLEAP] attack, as most implementations still support MS-CHAP, and no previous standard has deprecated it.

The attack relies on a vulnerability in the protocol design in [RFC2759], Section 8.4. In that section, the response to the MS-CHAP challenge is calculated via three DES operations, which are based on

the 16-octet NT-Hash form of the password. However, the DES operation requires 7 octet keys, so the 16-octet NT-Hash cannot be divided evenly into the 21 octets of keys required for the DES operation.

The solution in [RFC2759] Section 8.4 is to use the first 7 octets of the NT-Hash for the first DES key, the next 7 octets for the second DES key, leaving only 2 octets for the final DES key. The final DES key is padded with zeros. This construction means that an attacker who can observe the MS-CHAP2 exchange only needs to perform 2^16 DES operations in order to determine the final 2 octets of the original NT-Hash.

If the attacker has a database which correlates known passwords to NT-Hashes, then those two octets can be used as an index into that database, which returns a subset of candidate hashes. Those hashes are then checked via brute-force operations to see if they match the original MS-CHAPv2 data.

This process lowers the complexity of cracking MS-CHAP by nearly five orders of magnitude as compared to a brute-force attack. The attack has been demonstrated using databases which contain tens to hundreds of millions of passwords. On a consumer-grade machine, the time required for such an attack to succeed is on the order of tens of milliseconds.

While this attack does require a database of known passwords, such databases are easy to find online, or to create locally from generator functions. Passwords created manually by people are notoriously predictable, and are highly likely to be found in a database of known passwords. In the extreme case of strong passwords, they will not be found in the database, and the attacker is still required to perform a brute-force dictionary search.

The result is that MS-CHAP has significantly lower security than PAP. When the MS-CHAP data is not protected by TLS, it is visible to everyone who can observe the RADIUS traffic. Attackers who can see the MS-CHAP traffic can therefore obtain the underlying NT-Hash with essentially zero effort, as compared to cracking the RADIUS shared secret. In contrast, the User-Password attribute is obfuscated with data derived from the Request Authenticator and the shared secret, and that method has not yet been successfully attacked.

Implementers and administrators MUST therefore treat MS-CHAP and MS-CHAPv2 as being equivalent in security 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. MS-CHAP offers little benefit over PAP, and has many drawbacks as discussed here, and in the next section.

As MS-CHAP can be trivially broken by an observer, this document therefore mandates that MS-CHAP or MS-CHAPv2 authentication data carried in RADIUS MUST NOT be sent in situations where the that data is visible to an observer. MS-CHAP or MS-CHAPv2 authentication data MUST NOT be sent over RADIUS/UDP or RADIUS/TCP.

As MS-CHAP offers no practical benefits over PAP and has many downsides, MS-CHAP authentication SHOULD NOT be used even when the transport protocol is secure, as with IPsec or RADIUS over TLS.

Existing RADIUS client implementations SHOULD deprecate the use of MS-CHAPv1 and MS-CHAPv2. Clients SHOULD forbid new configurations from enabling MS-CHAP authentication. New RADIUS clients MUST NOT implement the attributes used for MS-CHAPv1 and MS-CHAPv2 authentication (MS-CHAP-Challenge and MS-CHAP-Response).

8.7. Password Visibility and Storage

An attacker may choose to ignore the wire protocol entirely, and bypass all of the issues described earlier in this document. An attacker could instead focus on the database which holds user credentials such as account names and passwords. At the time of this writing, databases such as [PWNED] claim to have records of over twelve billion user accounts which have been compromised. User databases are therefore highly sought-after targets.

The attack discussed in this section is dependent on vulnerabilities with the credential database, and does not assume an attacker can see or modify RADIUS traffic. As a result, issues raised here apply equally well when TTLS, PEAP, or RADIUS/TLS are used. The success of the attack depends only on how the credentials are stored in the database. Since the choice of authentication method affects the way credentials are stored in the database, the security of that dependency needs to be discussed and explained.

Some organizations may desire to increase the security of their network by avoiding PAP, and using CHAP or MS-CHAP, instead. These attempts are misguided. If simple password-based methods must be used, in almost all situations, the security of the network as a whole is increased by using PAP in preference to CHAP or MS-CHAP. The reason is found through a straightforward risk analysis, which we explain in more detail below.

8.7.1. PAP Security Analysis

When PAP is used, the RADIUS server sees a clear-text password from the user, and compares that password to credentials which have been stored in a user database. The credentials stored in the database can be salted and/or hashed in a form which is commonly referred to as being in "crypt"ed form. The RADIUS server takes the users clear-text password, performs the same "crypt" transformation, and then compares the two "crypt"ed passwords.

Any compromise the RADIUS server will result in that clear-text password leaking. However, in most cases, the clear-text password is available only in the memory of the RADIUS server application (i.e. not "on the wire"), and then only for a short period of time. An attacker who desires to obtain passwords for all users would have to wait for all users to log in, which can take a substantial amount of time. During that time, an administrator may discover the breach, and resolve the issue.

When PAP is used, the credentials in the database are stored securely "at rest", presuming that the administrator only stores "crypt"ed credentials. Any compromise of the database results in the disclosure of minimal information to the attacker. That is, an attacker cannot easily obtain the clear-text passwords from the compromised database.

The result is that the user passwords are visible in clear-text only for a short time, and then only on the RADIUS server. The security of this system is not as good as seen with EAP-pwd [RFC5931] for example, but it is not terrible.

While the obfuscation method used for the User-Password attribute has not been shown to be insecure, it has not been proven to be secure. The obfuscation method depends on calculating MD5(secret + Request Authenticator), which has a few helpful properties for an attacker. The cost of brute-forcing short secrets is not large, Section 3.4 discusses that cost in detail. Even for longer secrets which are humanly generated, the MD5 state for hashing the secret can be pre-calculated and stored on disk. This process is relatively inexpensive, even for billions of possible shared secrets. The Request Authenticator can then be added to each pre-calculated state via brute-force, and compared to the obfuscated User-Password data.

The MD5 digest is 16 octets long, and many passwords are shorter than that. This difference means that the final octets of the digest are placed into the User-Password attribute without modification. The result is that a brute-force attack does not need to decode the User-Password and see if the decoded password "looks reasonable". Instead, the attacker simply needs to compare the final octets of the calculated digest with the final octets of the User-Password attribute. The result is a signal which indicates with high probability that the guessed secret is correct.

The only protection from this attack is to ensure that the secret is long, and derived from a cryptographically strong pseudo-random number generator. Section 7.1 discusses these issues in more detail.

8.7.2. CHAP and MS-CHAP Password Storage

In contrast with PAP, when CHAP or MS-CHAP is used, those methods do not expose a clear-text password to the RADIUS server, but instead a hashed transformation of it. That hash output is in theory secure even if an attacker can observe it. While CHAP is still believed to be secure, MS-CHAP is not secure, as we saw earlier in Section 8.6. For the purposes of this section, we will focus on the construct of "hashed passwords", and will ignore any attacks specific to MS-CHAP. We will also note that EAP-MD5 [RFC3748], Section 5.4 is essentially CHAP, and has the same security analysis.

The hash transformations for CHAP and MS-CHAP depend on a random challenge. The intent was to increase security, but their construction makes strong requirements on the form in which user credentials are stored.

The process for performing CHAP and MS-CHAP is inverted from the process for PAP. Using similar terminology as above for illustrative purposes, the "hash"ed passwords are carried in the CHAP method, and are sent to the server. The server must obtain the clear-text (or NT hashed) password from the database, and then perform the "hash"

operation on the password from the database. The two "hash"ed passwords are then compared as was done with PAP. This inverted process decreases system security substantially.

When CHAP or MS-CHAP are used, all of credentials are stored as clear-text (or clear-text equivalent) in the database, all of the time. Even if the database contents are encrypted, the decryption keys are necessarily accessible to the application which reads that database. Any compromise of the application means that the entire database can be immediately read and exfiltrated as a whole. The attacker then has complete access to all user identities, and all associated clear-text passwords.

It should go without saying that having an attacker obtain all clear-text passwords is more of an issue than having the same attacker obtain "crypt"ed passwords. Similarly, it is more secure for a RADIUS server to have access to some clear-text passwords, some of the time, rather than having access to all of the clear-text passwords, all of the time.

8.7.3. On-the-wire User-Password versus CHAP-Password

There is one more security myth which should be put to rest about PAP versus CHAP. There is a common belief that CHAP is more secure, because passwords are sent "in the clear" via the User-Password attribute. This belief is false.

The User-Password attribute is obfuscated when it is sent in an Access-Request packet, using keyed MD5 and the shared secret, as defined in [RFC2865], Section 5.2. At the time of this writing, no attack better than brute force has been found which allows an attacker to reverse this obfuscation.

There have been claims that it is preferable to use CHAP-Password as it does not "send the password in clear-text". This preference is based on a misunderstanding of how CHAP-Password and User-Password attributes are calculated.

The CHAP-Password attribute depends on the hash of a visible Request Authenticator (or CHAP-Challenge) and the users password. The obfuscated User-Password depends on the same Request Authenticator, and on the RADIUS shared secret. For an attacker, the difference between the two calculations is minimal. They can both be attacked with similar amounts of effort, as they use similar constructs. As a result, any security analysis which makes the claim that "User-Password insecure because it uses MD5" ignores the fact that the CHAP-Password attribute is constructed through substantially the same method.

An attacker who can observe the CHAP-Password and CHAP-Challenge can also perform an off-line dictionary attack on the observed values. The complexity of cracking CHAP-Password is similar to that noted above for cracking RADIUS packets, which was discussed above in Section 3.4. The difference between the two attacks is that the shared secrets are more likely to be secure than passwords for an end-user.

An attacker who can crack one users password can gain network access as that user, or even administrator access to network devices. In contrast, an attacker who can crack the shared secret can gain network access as any user, and perform any authorization. The result is that it is more valuable to crack shared secrets, even if the underlying attack process is essentially the same.

8.7.4. PAP vs CHAP Conclusions

A careful security analysis shows that for all of PAP, CHAP, and MS-CHAP, the RADIUS server must at some point have access to the clear-text version of the password. As a result, there is minimal difference in risk exposure between the different authentication methods if a RADIUS server is compromised.

However, when PAP is used, the user credentials can be stored securely "at rest" in a database, while such secure storage is impossible with CHAP and MS-CHAP. There is therefore a substantial difference in risk exposure between the different authentication methods, with PAP offering substantially higher security due to its ability to secure passwords at rest via the "crypt" construct mentioned above.

In contrast, CHAP is highly insecure, as any database compromise results in the immediate exposure of the clear-text passwords for all users. The security of MS-CHAP is best described as near zero, independent of any database compromise. This makes MS-CHAP the worst of all possible choices.

This security difference is shown not just in the [PWNED] database, but also in attacks on RADIUS systems [EXPLOIT], where attackers identified a vulnerable RADIUS system, and then:

- utilized SQL commands to dump the credentials [T1555], which contained both clear-text and hashed passwords for user and administrative accounts.

The attack proceeded to leverage those passwords to gain more permissions:

> Having gained credentials from the RADIUS server, PRC state-sponsored cyber actors used those credentials with custom automated scripts to authenticate to a router via Secure Shell (SSH), execute router commands, and save the output.

This attack is only possible when systems store clear-text passwords.

The result is that 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. 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 above.

That being said, other authentication methods such as EAP-TLS [RFC9190] and EAP-pwd [RFC5931] 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.

8.8. 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.

8.9. Eliminating Proxies

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.

8.10. Accounting Is Imperfect

The use of RADIUS/UDP for accounting means that accounting is inherently unreliable. Unreliable accounting means that different entities in the network can have different views of accounting traffic. These differences can have multiple impacts, including incorrect views of who is on the network, to disagreements about financial obligations. These issues are discussed in substantial detail in [RFC2975], and we do not repeat those discussions here. We do, however, summarize a few key issues. Sites which use accounting SHOULD be aware of the issues raised in [RFC2975], and the limitations of the suggested solutions.

Using a reliable transport such as RADIUS/TLS makes it more likely that accounting packets are delivered, and that acknowledgments to those packets are received. Reducing the number of proxies means that there are fewer disparate systems which need to have their accounting data reconciled. Using non-volatile storage for accounting packets means that a system can reboot with minimal loss of accounting data. Using interim accounting updates means that transient network issues or data losses can be corrected by later updates.

As RADIUS does not provide for end-to-end signaling or transport, using RADIUS/TLS provides for reliable transport only when the client originating the accounting traffic is connected directly to the server which records it. If there are instead one or more proxies involved, the proxies increase overall unreliability.

Systems which perform accounting are also subject to significant operational loads. Wheres authentication and authorization may use multiple packets, those packets are sent at session start, and then never again. In contrast, accounting packets can be sent for the lifetime of a session, which may be hours or even days. There is a large cost to receiving, processing, and storing volumes of accounting data.

However, even with all of the above concerns addressed, accounting is still imperfect. The obvious way to increase the accuracy of accounting data is to increase the rate at which interim updates are sent, but doing so also increases the load on the servers which process the accounting data. At some point, the trade-off of cost versus benefit becomes negative.

There is no perfect solution here. Instead, there are simply a variety of imperfect trade-offs.

8.10.1. Incorrect Accounting Data

Even if all accounting packets were delivered and stored without error, there is no guarantee that the contents of those packets are in any way reasonable. The Wireless Broadband Alliance RADIUS Accounting Assurance [WBA] group has been investigating these issues. While the results are not yet public, a presentation was made at IETF 118 in the RADEXT working group [RADEXT118].

The data presented indicated that the WBA saw just about every possible counter attribute in RADIUS accounting packets as containing data which was blatantly wrong or contradictory. One example is extremely short sessions which have impossibly large amounts of data being downloaded. Other accounting packets allege that large amounts of data were downloaded via octet counters, while at the same time claiming negligible packet counters, leading to absurdly large packet sizes.

The only conclusion from this analysis is that RADIUS clients act as if it is better to produce incorrect accounting data rather than to produce no data at all. This failure to follow reasonable practices is expensive. In effect, vendors have offset their costs to produce quality data onto their customers, who have to take difficult and uncertain steps in order to sanitize or verify the confusing data which vendors provide in accounting packets.

It should go without saying that accounting systems need to produce correct data. However, [RFC2865] makes no requirement that the accounting data transported in RADIUS is correct, or is even vaguely realistic. We therefore state here that systems which produce accounting data MUST generate correct, accurate, and reasonably precise data. Vendors of networking equipment SHOULD test their systems to verify that the data they produce is accurate.

8.11. 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. Specifically, some implementations misunderstood the requirement (from the BlastRADIUS mitigations) that Message-Authenticator is sent as the first attribute in responses to Access-Request packets. Despite the mandate 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 defining attribute order (quoted below) does not cover all possible cases:

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.

We add a missing case here.

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 unexpected to the implementation, but which is 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.

8.12. Unknown Attributes

An 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, we must admit that it is not explicitly forbidden, either. This document corrects that failure.

Unknown attributes are defined to be 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.

We note this recommendation is to "ignore" these attributes, and not to discard the encapsulating packet. Instead of ignoring unknown attributes, some implementations instead discard those packets. This behavior is incorrect, and leads to interoperability issues and network problems.

With limited exceptions, implementations MUST NOT discard packets if they receive attributes with an unknown Type. We update [RFC2865] to require that 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 requirements is CoA-Request and Disconnect-Request packets, as discussed in [RFC8559], Section 4.3.2.

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.

And also 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.

A client is not required to implement all possible authorizations which can be sent in an Access-Accept. We therefore extend the above scenarios to packets which contain unknown Types. A client SHOULD treat Access-Accepts with no known or supported authorizations as though an Access-Reject had been received instead.

This requirement for unknown Types is already met by most, if not all, RADIUS implementations. That is, experience has shown that discarding packets for arbitrary reasons causes problems. Existing

implementations have largely chosen to follow reasonable practices, and the recommendation here simply documents that wide-spread practice.

9. Practical Suggestions

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.

MS-CHAP can be cracked with minimal effort. This information has been known 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.

10. 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.

11. 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.

11.1. 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.

11.2. 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.

12. 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]

13. 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.

14. 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.

15. References

15.1. Normative References

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

15.2. 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., <https://www.blastradius.fail/pdf/radius.pdf>.

[BRIGGS] Briggs, K., "Comments on the FCC's Public Notice DA 24-308 on SS7 and Diameter Vulnerabilities", n.d., <https://www.fcc.gov/ecfs/document/10427582404839/1>.

[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>.

[EXPLOIT] Agency, A. C. D., "People's Republic of China State-Sponsored Cyber Actors Exploit Network Providers and Devices", n.d., <https://www.cisa.gov/news-events/cybersecurity-advisories/aa22-158a>.

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

[HASHCLASH] Stevens, M., "Project HashClash - MD5 & SHA-1 cryptanalytic toolbox", n.d., <https://github.com/cr-marcstevens/hashclash>.

[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. A., and S. Gundavelli, "WBA OpenRoaming Wireless Federation", Work in Progress, Internet-Draft, draft-tomas-openroaming-05, 15 April 2025, <https://datatracker.ietf.org/doc/html/draft-tomas-openroaming-05>.

[ISSUES] RADEXT, "Issues and Fixes 2", n.d., <https://github.com/radext-wg/issues-and-fixes-2/wiki>.

[MAY-1995] O’Dell, M., "Proxy-State radius extension to support stateless proxies", n.d., <http://ftp.cerias.purdue.edu/pub/doc/network/radius/archive/ietf-radius.9506>.

[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/>.

[PWNED] Hunt, T., "Have I been Pwned", n.d., <https://haveibeenpwned.com/>.

[RADEXT118]
Alliance, W. B., "RADIUS Accounting Assurance at IETF 118", n.d., <https://youtu.be/wwmYSItcQt0?t=3953>.

[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>.

[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>.

[RFC2975] Aboba, B., Arkko, J., and D. Harrington, "Introduction to Accounting Management", RFC 2975, DOI 10.17487/RFC2975, October 2000, <https://www.rfc-editor.org/rfc/rfc2975>.

[RFC3539] Aboba, B. and J. Wood, "Authentication, Authorization and Accounting (AAA) Transport Profile", RFC 3539, DOI 10.17487/RFC3539, June 2003, <https://www.rfc-editor.org/rfc/rfc3539>.

[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>.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <https://www.rfc-editor.org/rfc/rfc3748>.

[RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, DOI 10.17487/RFC4120, July 2005, <https://www.rfc-editor.org/rfc/rfc4120>.

[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>.

[RFC6280] Barnes, R., Lepinski, M., Cooper, A., Morris, J., Tschofenig, H., and H. Schulzrinne, "An Architecture for Location and Location Privacy in Internet Applications", BCP 160, RFC 6280, DOI 10.17487/RFC6280, July 2011, <https://www.rfc-editor.org/rfc/rfc6280>.

[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>.

[RFC6733] Fajardo, V., Ed., Arkko, J., Loughney, J., and G. Zorn, Ed., "Diameter Base Protocol", RFC 6733, DOI 10.17487/RFC6733, October 2012, <https://www.rfc-editor.org/rfc/rfc6733>.

[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>.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7525, DOI 10.17487/RFC7525, May 2015, <https://www.rfc-editor.org/rfc/rfc7525>.

[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>.

[RFC8907] Dahm, T., Ota, A., Medway Gash, D.C., Carrel, D., and L. Grant, "The Terminal Access Controller Access-Control System Plus (TACACS+) Protocol", RFC 8907, DOI 10.17487/RFC8907, September 2020, <https://www.rfc-editor.org/rfc/rfc8907>.

[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>.

[SPOOFING] Cudbard-Bell, A., "Wi-Fi Spoofing for Fun and Profit", n.d., <https://networkradius.com/articles/2021/08/04/wifi-spoofing.html>.

[WBA] Alliance, W. B., "RADIUS Accounting Assurance", n.d., <https://wballiance.com/radius-accounting-assurance/>.

[WIFILOC] Alliance, W.-F., "Accurate indoor location with Wi-Fi connectivity", n.d., <https://www.wi-fi.org/discover-wi-fi/wi-fi-location>.

Author's Address

Alan DeKok
InkBridge Networks
Email: aland@inkbridgenetworks.com