RADEXT Working Group
Internet-Draft
Intended status: Informational
Expires: 16 September 2026

A. DeKok
InkBridge Networks

15 March 2026

A Review of RADIUS Security and Privacy
draft-dekok-radext-review-radius-01

Abstract

This document provides a comprehensive review of security issues related to the RADIUS Protocol. This review motivates the changes to RADIUS security which are made in [I-D.ietf-radext-deprecating-radius].

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-dekok-radext-review-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/review-radius.git.

Status of This Memo

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

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

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

This Internet-Draft will expire on 16 September 2026.

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, 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). As much of the protocol data is sent in clear-text, packets can leak information about use names, devices, and locations.

This document provides a comprehensive review of RADIUS security and privacy. The discussion here motivates the changes to RADIUS security which are made in [I-D.ietf-radext-deprecating-radius]. In order to simplify the protocol changes for implementers, this historical review is a separate document from the protocol changes. While this document contains some operational recommendations, it does not change the RADIUS protocol.

1.1. History of RADIUS Security

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.

It appears that 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]. Another outcome was the 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.

RadSec has now been standardized in [I-D.ietf-radext-radiusdtls-bis]. That document standardizes TLS and DTLS transporst for RADIUS, which gives implementors and operators a way to secure the RADIUS protocol.

As for RADIUS/UDP and RADIUS/TCP, they still depend on MD5 for all security. The insecurity of MD5 was noted in [RFC6151], which is over a decade old as of the time of publication of this document. The recommendation to use Message-Authenticator in [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 explains why that mandate is now being made in [I-D.ietf-radext-deprecating-radius].

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 explains why insecure uses of RADIUS need to be deprecated.

[I-D.ietf-radext-deprecating-radius] mandates the use of secure practices in RADIUS, including the use of (D)TLS via [I-D.ietf-radext-radiusdtls-bis].

1.2. RADIUS/UDP over the Internet is insecure

Since 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 simpler than IPSec in many ways simpler for implementations and deployments. 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 such as RADIUS 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 in eduroam [EDUROAM] [RFC7593], and more recently in 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 details of the BlastRADIUS attack are discussed in more detail below, in Section 4.

1.3. RADIUS/UDP Has Security and Privacy Problems

Even if we ignore the BlastRADIUS attack, problems with MD5 mean that a hobbyist attacker who can view RADIUS/UDP traffic can brute-force check all possible RADIUS shared secrets of eight characters in not much more than an hour. A more resourceful attacker (e.g. a nation-state) can check all much longer shared secrets with only modest expenditures. See Section 3.6 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. Attacks on MS-CHAP are described below in Section 5.1.1.

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 provides stronger protection for packets, 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 the Authenticator field.

Most attributes in RADIUS are sent in clear-text, with only a few attributes such as User-Password and Tunnel-Password have their contents hidden. The hidden attributes rely on "ad hoc" obfuscation methods using MD5, which have not been successfully attacked, but have 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 3.4).

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.

1.4. 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. When the cryptographic methods provide forward secrecy ([RFC7525], Section 6.3), then breaking one session provides no information about other sessions.

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 more security issues in addition to the need for a secure transport. The rest of this document discusses those issues in detail.

1.5. Overview of this document

This document begins with a summary of issues with RADIUS, including showing just how trivial it is to crack RADIUS/UDP security. We then explain why mandating a secure transport is necessary, 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.

1.5.1. 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 in some readers that a particular practice is "allowed" by a specification, simply because the practice is not forbidden. 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 perhaps 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 are unconstrained by the specification, and therefore may be good or bad.

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 formally written down. These practices have been found to be valuable by implementers and administrators. Deviations from these practices generally results in instabilities and incompatibilities between systems. As such, 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 been followed instead.

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

* 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 is inconsistent with historic RADIUS practices. The reader is assured that no modern cryptographic methods are used with RADIUS/UDP.

3. Security 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 in many cases the RADIUS server will not even be aware that an unauthorized user is on the network.

Another issue is that RADIUS sends most information (but not passwords) "in the clear", with obvious privacy implications. Publicly available data includes information such as names, MAC addresses, locations, etc.

As for authenticating the packets themselves, even if Message-Authenticator is used for integrity checks, an average hobbyist who observes RADIUS traffic can 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 in [I-D.ietf-radext-deprecating-radius], which requires the use of Message-Authenticator to authenticate Access-Request packets, and responses to them. The long-term solution is in [I-D.ietf-radext-radiusdtls-bis], which wraps the protocol in a secure transport.

With the benefit of experience, [I-D.ietf-radext-deprecating-radius] errs on the side of security, while still allowing for backwards compatibility. It is not acceptable to permit insecure practices in the RADIUS protocol 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.

We address these issues in more detail below.

3.1. The BlastRADIUS Vulnerability

The BlastRADIUS vulnerability was first described in [BLAST]. This section gives a short summary of why RADIUS is vulnerable to this attack. Section 4, below, gives a longer explanation as to how the attack works, and why the mitigations defined in [I-D.ietf-radext-deprecating-radius] protect from the attack. The reader is referred to [BLAST] for a comprehensive description of the attack.

The discussion below assumes that there exist plain texts "A", "B", "S". Following the practice of [RFC2865], we use "+" to denote concatenation. The vulnerability then relies on the following property of MD5:

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

This construction menas that 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. The attack works even if the attacker does not know text "S". That is, 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 generally observe or predict an Access-Reject packet, as they are generally empty. Each valid Access-Reject is signed with a shared secret unknown to the attacker. With sufficient work, the attacker can create an Access-Accept which has the same MD5 hash as the Access-Reject. The attacker then replaces the Access-Reject with this Access-Accept, using the Response Authenticator from the Access-Reject.

The client will then receive the packet, calculate MD5(Access-Accept + secret), and verify that the Response Authenticator is correct. The client will then follow the attackers instructions: give the user access, along with some authorization.

This chosen prefix attack is root cause behind the BlastRADIUS vulnerability.

We note also that this attack does not expose the contents of the User-Password attribute. Instead, the attack simply bypasses all server-side authentication, and fools the client into accepting a forged response.

While this attack 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, the existence of this attack alone is sufficient reason to deprecate all uses of RADIUS/UDP and RADIUS/TCP.

3.2. Failed Attempts to Improve RADIUS Security

Independent of any cryptographic vulnerability, there are a number of factors which contributed to the ongoing failure to improve RADIUS security.

A major factor is the continued use of MD5 for security, instead of mandating the use of an HMAC via Message-Authenticator. This change could have been made in [RFC2869] in the year 2000. The stated reason there for not mandating Message-Authenticator was the issue of backwards compatibility. Unfortunately, experience shows that issues which are not fixed just grow larger over time. The issue of backwards compatibility is significantly worse now than it was in the year 2000.

The issue of unauthenticated Access-Request packets was raised again in [RFC5080], Section 2.2.2, and again was widely ignored. If vendors had implemented this recommendation in 2007, then the BlastRADIUS attack would have been impossible.

3.3. Failures of the Protocol State Machine

Another contributing factor to the BlastRADIUS vulnerability 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. This principle means that a client will accept packets that are well-formed, but which contain invalid signaling. Specifically, the Proxy-State attribute is intended for signalling between a proxy and a "next hop" server. It offers no value for RADIUS clients. A NAS

which originates packets therefore does not send Proxy-State in an Access-Request, and should also not receive Proxy-State in any response packets.

If a NAS does receive Proxy-State in a response, where the request did not contain Proxy-State, this is arguably a violation of the protocol state machine. It would be useful if such a packet would either trigger a warning message, or instead be rejected entirely.

That is, when a NAS sees Proxy-State in an Access-Accept, that is a failure of signaling in the RADIUS protocol. It indicates either a serious failure of configuration, implementation, or as seen in this case, an active attack. If the specifications had instructed clients to discard responses which contained unexpected Proxy-State attributes, then this attack would have been prevented.

3.4. Most information is sent in Clear Text

Even ignoring security issues, the RADIUS protocol has fundamental problems with privacy.

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 [RFC6973], Section 5 for detailed discussion, and to [RFC6973], Section 6 for recommendations on threat mitigations.

When RADIUS/UDP or RADIUS/TCP is used across the public Internet, common configurations allow the location of individuals to be tracked in real-time (usually 10 minute intervals), to within a small physical location. The users devices can be identified, and also tracked. 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 the user device and often W-Fi access points are publicly available. There are multiple services selling databases of Wi-Fi access point location.

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 published too late to have any practical impact on the design of location information attributes, as [RFC5580] had already been published.

The effect of these design decisions 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).

These issues are not theoretical. Recently, [BRIGGS] noted for the Diameter [RFC6733] protocol 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 mention only Diameter, 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.

3.5. 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 instead that no one is looking for new attacks.

3.6. 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 a consumer-grade machine, it can take about 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 trivially parallelizable. Nation-states have sufficient resources to deploy hundreds to thousands of systems dedicated to these attacks. That reality means that a "time to crack" of 24 years means "24 CPU years", and does not mean "wall clock" time. A thousand commodity CPUs are enough to reduce the crack time from 24 years to a little over a week. This attack is feasible for any organisation with a modest amount of resources.

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, and different trade-offs occur. Rather than cracking the secret, 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 likely enough to give the attacker administrator access to a NAS, where the shared secret is likely to be 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. It is also common practice for administrators to use "short" shared secrets, and to use shared secrets created by a person, or to use secrets that are derived from a limited character set. Theses practice are simple to implement and to follow, but they are highly insecure, and do not provide adequate security. Any system using these practices is vulnerable to all of the issues which are outlined in this document.

[I-D.ietf-radext-deprecating-radius] gives suggestions for how strong shared secrets can be created.

3.7. CoA-Request packets might leak Tunnel-Password contents

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 RADIUS shared secret). It is not currently 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.

Due to the above issues, the use obfuscated attributes in CoA-Request or Disconnect-Request packets needs to be avoided.

3.8. Secure Transports Can Still Leak Information

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 for proxies even when RADIUS/TLS and IPsec are used. A proxy which receives packets over IPsec will terminate the IPSec tunnel, but it 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. As RADIUS security is "hop by hop", there is no way for one "hop" to know anything about, or to control, the security of another "hop".

The use of channel binding [RFC6677] does not prevent these attack, as they made by parties within the trusted RADIUS system. Instead, channel binding protects from an unrelated attack by an untrusted third party.

One solution to the above issues would be to create a new protocol which is secure by design, but that solution is not practical.

3.9. All short Shared Secrets have been compromised

As a result of the above analysis, administrators can assume that any shared secret of 8 characters or less has been compromised as soon as it is used in RADIUS/UDP or RADIUS/TCP. Administrators can assume that any shared secret of 10 characters or less has been compromised by an attacker with significant resources. Administrators can also assume that all private information (such as User-Password or Tunnel-Password) which depend on such shared secrets have also been compromised.

As the analysis in [](#user-password} below shows, a strong shared secret protects the contents of User-Password, no matter how weak the end-users password is. As such it is acceptable (for a short time) for RADIUS/UDP to continue to carry User-Password and Tunnel-Password, but only when a strong shared secret is used.

Using a strong shared secret is only a partial protection from known attacks. RADIUS can carry end-user credentials such as CHAP-Password and MS-CHAP which are not protected with the shared secret. Many of those credentials have been compromised, too.

3.10. Many End-User Passwords Have Been Compromised

The analyis in Section 5.1.1 below shows how the construction of MS-CHAP can allow attackers to determine the underyling password in milliseconds. Section 5.1.2 also shows how a different attack on CHAP-Password can recover the underlying password in milliseconds. This section explains the effects of those attacks.

As the security of CHAP-Password and MS-CHAP depends only on the strength of the end-users password, those methods can be safe only if the user chooses a strong password. That is, a password which is 10 characters or more, and which is derived from a cryptograpically strong pseudo-random number generator. This requirement is unlikely to be met by a large percentage of end users. As such, the use of CHAP-Password and MS-CHAP needs to be deprecated.

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

4. The BlastRADIUS Attack

This section gives more details on the BlastRADIUS attack, so that the reader can be informed as to why [I-D.ietf-radext-deprecating-radius] makes its recommendations. In the interest of simplicity for implementers, {I-D.ietf-radext-deprecating-radius}} omits all explanation of the attack. That document also gives minimal explanation for each of the protocol changes. This document contains the full details instead.

The attack depends on a few related factors. If any one of these factors are not present, the attack is not possible. These factors are outlined below:

* The Access-Request packets are not authenticated, and can therefore be modified without detection.

* The use of MD5 within RADIUS is subject to known prefix attacks.

* The improvements to MD5 collisions in [HASHCLASH] make the attack feasible.

* The structure and behavior of Proxy-State makes it the perfect vector for an attacker to inject the "MD5 garbage" ([BLAST]) which is needed to force the MD5 collision.

The attack works by having An "on path" attacker who modifies an Access-Request packet, and injects one or more Proxy-State attributes with special contents. 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 allows 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 known 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 5.3. While those reasons were transient and decades in the past, the impact of those decisions has continued to impact RADIUS until the present.

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, so long as 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 against the attack for RADIUS systems to use TLS, or to require that packets contain a is valid Message-Authenticator.

4.1. Detailed Description of the Attack

The specific details of the attack are outlined below, as steps which are numbered the same as in the original paper ([BLAST]).

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. An external resource is used to calculate an MD5 collision using the Request Authenticator, along with 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 "MD5 garbage" 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 added, 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-total 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 leverages Access-Reject responses, we reiterate that the root cause of the vulnerability is the unauthenticated Access-Request packets. The attack will therefore succeed even if the server responds with Access-Accept, Access-Challenge, or Protocol-Error [RFC7930]. The ability for the attacker to avoid 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 in any way, 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.

[I-D.ietf-radext-deprecating-radius] defines mitigations which will 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 need to remove all uses of RADIUS/UDP and RADIUS/TCP from their products. Administrators need to stop using RADIUS/UDP and RADIUS/TCP.

4.2. Mitigating the Attack

While [I-D.ietf-radext-deprecating-radius] defines the mitigations that are mandated for clients and servers, we give a summary description of those mandates here for clarity. These descriptions are not normative, and readers are instructed to refer to [I-D.ietf-radext-deprecating-radius] for the full list of normative changes to RADIUS.

Clients

Clients are required to include Message-Authenticator as the first attribute in all Access-Request packets.

Clients are required to have a new boolean configuration flag for each server, called "require Message-Authenticator".

When this flag is set to "false", client behavior remains unchanged from legacy RADIUS.

When this flag is set to "true", clients discard all responses to Access-Request packets which do not contain a Message-Authenticator attribute.

Clients still need to validate the contents of Message-Authenticator when it is present. Clients also need to accept valid and authenticated responses, no matter where the Message-Authenticator is located in the response.

Servers

Servers are required to include Message-Authenticor as the first attribute in all responses to Access-Request packets.

Servers are required to have a new boolean configuration flag for each client, called "require Message-Authenticator".

When this flag is set to "false", their behavior remains unchanged from legacy RADIUS, except that the "limit Proxy-State" flag below is also checked.

When this flag is set to "true", clients discard all Access-Request packets which do not contain a Message-Authenticator attribute.

Servers still need to validate the contents of Message-Authenticator when it is present. Servers also need to accept valid and authenticated Access-Requests, no matter where the Message-Authenticator is located in the request.

Servers are required to have a new boolean configuration flag for each client, called "limit Proxy-State".

When this flag is set to "false", server behavior remains unchanged from legacy RADIUS.

When this flag is set to "true", servers discard all Access-Request packets that contain a Proxy-State attribute.

4.3. Why the Mitigiations Work

This section explains the rationale for the mitigations defined by [I-D.ietf-radext-deprecating-radius].

Adding Message-Authenticator as the first attribute in packets means that the first attribute contains data which is impossible for the attacker to predict. That is, 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 prevented.

When this change is made on clients, it is necessary to prevent the attack, but it is not sufficient. When a server does not require that Access-Request packets contain Message-Authenticator, an attacker can simply remove it from the Access-Request. The attack can then proceed, as the server will receive, process, and respond to, an unauthenticated Access-Request packet.

In contrast, when both clients and servers requires that packets contain a valid Message-Authenticator, the BlastRADIUS attack is impossible. Therefore the "require Message-Authenticator" flag is needed on both clients and servers in order to secure the RADIUS protocol. In order to enable compatibility with legacy systems, this protocol change must be enabled by a configuration flag.

4.3.1. Protecting Clients

A client is fully protected from the attack if it requires that all responses to Access-Request contain a Message-Authenticator, and it validates the contents of Message-Authenticator. The client is also protected when the server sends Message-Authenticator as the first attribute in all responses to Access-Request packets.

That server behavior secures one client to server connection, 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. As noted above, this location of the Message-Authenticator ensures that the unknown suffix is the entire packet, and the attack is impossible.

In contrast, 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 itself as an unknown suffix, as it does with the shared secret. The attacker can then proceed with the attack, with no additional effort.

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 [BLAST] Section 7.2 for a more complete description of these issues.

In short, the only solution which mitigates the attack is that servers need to place Message-Authenticator as the first attribute in all responses to Access-Request packets.

4.3.2. Protecting Servers and Proxies

Ugrading all client equipment can be difficult, if only because there are many more clients than servers. Some client products may no longer be supported, or the relevant vendor may have gone out of business. Even if upgraded software images are available, the upgrade process may impact production networks, which has a cost. As a result, any mitigations must work even when clients have not been updated.

A server is vulnerable to the attack when it proxies packets, even if it adds Message-Authenticator is added as the first attribute in responses to all Access-Request packets. Due to the limitations of RADIUS, a proxy has no way of knowing whether or not a "next hop" RADIUS server has been upgraded. It therefore has to protect itself from attacks when it is the only upgraded party in a RADIUS proxy chain.

In this scenario, a legacy client sends Access-Request packets to an upgraded proxy, which in turn forwards the packets to a legacy next hhop server. Responses from the next hop server are sent back to the proxy, and then to the client.

Upgrading the proxy will protect only the responses from the proxy to the client. The attacker can still modify packets from the client to the proxy, or it can modify all request and response packets that are sent between the proxy and next hop server. The result is that the upgraded server is still vulnerable to the attack.

The "limit Proxy-State" flag allows servers to detect and prevent attacks when Access-Request packets do not contain Message-Authenticator. This configuration is only necessary when the server is a proxy. When the server enables the "limit Proxy-State" flag, legacy clients to be used without substantially compromising security.

The proxy is likely to still be vulnerable to attacks on the link between itself and the next hop server. However, the proxy can use the client "require Message-Authenticator" flag defined above to protect itself. Even when the proxy cannot set that flag, the link between the proxy and the next hop server is much more likely to be protected via TLS or IPSec than the link between the client and proxy.

In addition, it is generally easier to upgrade servers than clients. The focus of the mitigations, therefore, has been on securing the link between clients and servers, not between proxy servers.

4.3.3. Other Attributes

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 those fields back in a response. Such attacks are therefore only possible when the server is configured to echo back attacker-controlled data, which is not their default behavior.

As a result, the configuration flags described above in Section 4.2 allow the maximum amount of security while adding 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 their "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 server configuration flags will protect it 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.

4.3.4. Requirements for Full Mitigation

The attack will only be mitigated in either of the following two circumstances:

1. The client implements the "require Message-Authenticator" flag, and has set that flag to "true",

2. The server places Message-Authenticator as the first attribute in all responses to Access-Request packets.

Since RADIUS has no feature negotiation, the server has no way of knowing whether or not the client has been configured securely. The only remaining choice then for server behavior then, is the second item. [I-D.ietf-radext-deprecating-radius] therefore mandates that all RADIUS servers send Message-Authenticator as the first attribute in all responses to Access-Request packets. This change is the simplest possible fix to the RADIUS protocol which will protect systems from the attack.

4.4. 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 defined in [I-D.ietf-radext-deprecating-radius] 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. In addition, there are situations where existing client implementations are not compatible with the mitigations. This section outlines those limitations.

4.4.1. 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 entire chain is vulnerable. The attack can proceed on the vulnerable systems, and the attacker can gain unauthenticated and/or unauthorized access to any systems which depend on that proxy chain.

Similarly, 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 must also discard packets which are missing it. Similarly, the client must also require that the attribute is present, and discard packets which are missing oit.

Similarly, clients are vulnerable when they do not require that all responses to Access-Request packets contain Message-Authenticator. Note that clients which validate Message-Authenticator are not

vulnerable even if Message-Authenticator is the last attribute in a response. The HMAC construct of Message-Authenticator makes the attack impossible in that situation.

The requirement on servers to place Message-Authenticator as the first attribute in all responses to Access-Request is there only to protect legacy clients which do not validate Message-Authenticator. There is no need for a client to discard responses where the Message-Authenticator is valid, but is also not the first attribute. Such behavior is incorrect, and will cause interoperability problems.

4.4.2. 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 [EDUROAM] use RADIUS/UDP to transport EAP, the attack is not possible.

Other roaming groups such as OpenRoaming [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 could perhaps force an Access-Accept in some situations where EAP is used, 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.

4.5. Implementations with Incorrect Mitigations

This section summarizes the various implementation issues, and the recommended fixes to them. While this section does not contain normative text, it refers to normative requirements in [I-D.ietf-radext-deprecating-radius]. This summary is necessary because multiple implementations failed to follow the normative requirements of that document. Instead, those systems either implemented behavior which was forbidden by the normative text, or else failed to implement behavior which was mandated by the normative text.

It is therefore necessary to reiterate to the reader that the normative text in [I-D.ietf-radext-deprecating-radius] is, in fact, normative, and that the mandates of the normative text need to be respected. The reader should understand that any non-normative text in this specification does not over-ride the clear mandates of the normative text in [I-D.ietf-radext-deprecating-radius].

The following list outlines the problems seen, in no particular order.

* Some implementations discard packets which contain Message-Authenticator.

* Some implementations discard responses where the Message-Authenticator is not first, in violation of [RFC2865], Section 5. It should be reiterated that the requirement in [I-D.ietf-radext-deprecating-radius] "Server Responses" to place Message-Authenticator first is a requirement on the server, and is not a requirement on the client.

4.5.1. Discarding Packets with Message-Authenticator is Wrong

Nearly all clients which do not validate Message-Authenticator are known to accept responses which contain it, due to the provisions of [RFC2866], Section 5:

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

These RADIUS clients are compatible with the protocol change outlined in this document. We note also that Message-Authenticator has been defined for almost twenty-five (25) years, since [RFC2869], so there are few reasons for equipment to not support it.

Since the publication of the original BlastRADIUS notification, it has become known that some implementations do not behave as expected. That is, instead of ignoring an unexpected Message-Authenticator attribute, they discard all responses with contain Message-Authenticator. That behavior is entirely unreasonable, and is not required by any standard.

The unfortunate reality is that the only way that RADIUS servers could be compatible with such systems is for them to never send Message-Authenticator in responses. However, doing so would open up significantly more systems to the BlastRADIUS attack. As such, there is no attempt made to be compatible with implementations that fail to implement RADIUS correctly.

The only way to secure those systems is to upgrade them. Failing that, the administrators of those systems will need to accept the fact that their systems are vulnerable.

The solution adopted by [I-D.ietf-radext-deprecating-radius] is to declare that clients or servers which discard packets containing Message-Authenticator are not compliant with the RADIUS specifications. It is not acceptable to decrease the security of the RADIUS protocol in order to be compatible with insecure and non-compliant implementations. That specification attempts to prevent such issues from happening in the future, by mandating behavior for unknown attributes in [I-D.ietf-radext-deprecating-radius] "Unknown Attributes". There is no reason for an implementation to discard response a packet simply because it does not recognize an attribute in the packet.

4.5.2. Checking the location of Message-Authenticator is Wrong

Nothing in any previous RADIUS specification requires attributes to be placed in any particular location in a packet. Nothing in any previous RADIUS specification requires implementations to discard packets which contain unrecognized attributes.

Further, the construction of Message-Authenticator allows for a RADIUS implementation to authenticate packets (other than Access-Request), even if the Message-Authenticator is not validated.

This issue is addressed in [I-D.ietf-radext-deprecating-radius], Part TBD, which clarifies and extends the requirements on attribute ordering and location. [I-D.ietf-radext-deprecating-radius], Part TBD also clarifies and extends the requirements on receiving unknown attributes.

4.6. Less Effective Mitigations

There was substantial discussion around the design and effectiveness of the mitigations defined in [I-D.ietf-radext-deprecating-radius]. This section outlines some obvious mitigations which were considered and rejected. As protocol design is subject to a complex series of trade-offs, it is useful to explain what those alternative mitigations are, and why they were rejected.

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.

Using configuration flags which control the desired outcome is preferable to using flags which depend on network topology that is outside of the control of clients and servers.

4.7. Non-Mitigations

It may be tempting to come up with other "ad hoc" solutions to this vulnerability which are simpler than the ones outlined in [I-D.ietf-radext-deprecating-radius]. Such solutions are likely to either break existing RADIUS deployments, or else they will not protect systems from the attack. The mitigations described in [I-D.ietf-radext-deprecating-radius] not only prevent the attack, they do so without negatively affecting normal RADIUS operation. There is therefore no reason to use any other methods.

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

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

There is therefore no reason to implement “ad hoc” solutions when a solution exists which has passed reviews by both the BlastRADIUS cryptographers, and by the relevant RADIUS experts. 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 who lack the relevant cryptographic and RADIUS experience.

4.7.1. Switch to Other Protocols is Not Appropriate

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 rarely present in equipment which is 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 any "security" advice which is based on a fundamental misunderstanding of networking protocols.

4.7.2. Intrusion Detection Rules

Intrusion detection systems can be updated to detect and/or warn about the BlastRADIUS 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.

4.8. Summary

The RADIUS protocol as defined in [RFC2865] is vulnerable to an attack due to Access-Request packets being entirely unauthenticated. This issue has been known and ignored for decades. It was first raised as a vulnerability in 1998 [DATTACK], and a fix was rejected in [RFC2869]. A practicl fix was suggested in 2007 in [RFC5080], Section 2.2.2, but it took until [I-D.ietf-radext-deprecating-radius] before a fix was mandated, That mandate only occurred because an exploit was demonstrated in 2024, in [BLAST].

5. Other RADIUS Problems

Independent of the above security and privacy issues, there are a large number of other problems with the RADIUS protocol, and with the historic practices around the use of RADIUS. This section discusses those problems.

5.1. Authentication Methods

There are a number of problems with authentication methods that are transported in RADIUS attributes. Even independent of security issues with a particular authentication method, the choice of authentication method can in fact decrease security, as noted below in Section 5.2.

5.1.1. Attacks on MS-CHAP

MS-CHAP (v1 in [RFC2433] and v2 in [RFC2759]) have major design flaws, and are not suitable for use outside of a secure tunnel such as PEAP, TEAP, 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 and no previous sp=ecification has deprecated MS-CHAP. As a result, most implementations still support 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 was 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 to returns a small subset (1/65536) of candidate hashes. Those hashes are then checked via brute-force operations to see if they match the original MS-CHAPv2 data. Limiting the number of candidate hashes allows the attacker to use greatly increase the size of precalculated hashes, with minimal additional cost.

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 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 requires 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 data can therefore obtain the underlying NT-Hash with essentially zero effort as compared to cracking the RADIUS shared secret. This attack means that from a cryptographic perspective, using MS-CHAP is essentially the same as sending passwords in clear-text.

5.1.2. CHAP-Password

This section describes a viable attack on CHAP, which does not appear to have been published before.

The contents of CHAP-Password are calculated using MD5 to hash an identifier, a password, and a challenge. From [RFC1994]:

> The Response Value is the one-way hash calculated over a stream of octets consisting of the Identifier, followed by (concatenated with) the "secret", followed by (concatenated with) the Challenge Value.

That is, the CHAP-Password contents are MD5(ID + password + challenge). While this construction is different from the one used to sign the Authenticator fields, attacks on CHAP-Password have substantially the same cost as for cracking the RADIUS shared secret (Section 3.6).

That is, checking all 8 character passwords from a 93 character set is possible for a hobbyist in about day.

The attack is made easier by the human practice of creating insecure passwords. An attacker can create dictionaries of hashes of potential passwords. For CHAP-Password, this also means a 256 times increase in dictionary size, due to the need to calculate the prefix of MD5(ID + password). However, that calculation is done once, and the results then stored on disk. The only ongoing cost is the need for more disk space.

Once the attacker sees a CHAP-Password, the ID field is used to select one of 256 dictionaries, and then every password hash in that dictionary extended with the challenge, and checked against the contents of CHAP-Password. As these pre-calculated dictionaries generally contain hundreds of millions or low billions of passwords, that number bounds the total number of hashes which need to be checked.

As noted above in Section 3.6, a high-end retail GPU is capable of performing more than 64 billion hashes per second. Which means that most CHAP-Password attributes can be turned into the equivalent clear-text passwords in sub-second time. The main gating factor in this attack is the time taken to stream billions of candidate hashes from disk to the GPU.

This attack means that from a cryptographic perspective, using CHAP-Password is essentially the same as sending passwords in clear-text.

5.1.3. User-Password

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 3.6 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 particular attack is to ensure that the secret is long, and is derived from a cryptographically strong pseudo-random number generator. To put it more clearly, if the RADIUS packet is secure due to the use of a strong shared secret, then the User-Password attribute is also secure.

5.1.4. EAP

There are too many EAP methods for them to be discussed here in any detail. Instead, we can make a few simple observations:

* EAP-MD5 is essentially the same as CHAP-Password, and shares the same security analysis. EAP-MD5 is not suitable for use outside of a secure tunnel such as PEAP, TEAP, or TTLS.

* EAP-MSCHAPv2 (any variant) is essentially the same as MS-CHAP, and shares the same security analysis. EAP-MSCHAPv2 is not suitable for use outside of a secure tunnel such as PEAP, TEAP, or TTLS.

* TLS-based EAP methods (e.g. TTLS, PEAP, etc.) benefit from the security of TLS, and are therefore secure.

* Other EAP methods are not discussed here.

5.1.5. Summary

There are no known security issues with User-Password, or with many EAP methods. In contrast, MS-CHAP and CHAP-Password are insecure, and can only be used safely within a TLS tunnel.

While the "on the wire" encoding of User-Password is secure, the passwords still have to be stored somewhere, typically in a database. The next section explains how the interaction between authentication methods and password storage methods can increase, or decrease, security of the system as a whole.

5.2. Password Visibility and Storage

An attacker can ignore the wire protocol entirely, and bypass all of the issues described earlier in this document. One such attack is to focus on the database that 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.

5.2.1. PAP Security Analysis

When PAP is used, the User-Password is obfuscated "on the wire", but the RADIUS server sees a clear-text password from the user. The server then compares that password to credentials which have been stored in a user database, and either accepts or rejects the user.

In many cases, 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 can take the users clear-text password, performs the same "crypt" transformation, and then compares the two "crypt"ed passwords.

Any compromise of the RADIUS server can result in the compromise of clear-text passwords for users. 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.

Storing passwords securely "at rest" is significantly more secure than storing clear-text passwords in a database, even when PAP authentication is used in RADIUS.

5.2.2. 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. The design goal of those methods was to make the hash output secure even if an attacker can observe it. However, as noted above, those methods have been shown to be insecure.

For the purposes of this section, we will ignore the previous attacks, and instead focus on the implications of using these hashing methods.

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.

Critically, when CHAP or MS-CHAP are used, all credentials must be 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 passwords in a "crypt"ed form. Similarly, it is more secure for a RADIUS server to have limited clear-text passwords (i.e. some of the time), rather than having unlimited access to all of the clear-text passwords, all of the time.

5.2.3. 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. Presuming that the user password has similar complexity to the shared secret, they can both be attacked with similar amounts of effort. 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.

The difference in security between the two methods is due instead to the practice of people creating their own insecure password, versus a RADIUS administrator creating a strong shared secret. The CHAP-Password contents depends only on the strength of the users chosen password. The User-Password contents instead depends on both the RADIUS shared secret, and on the users password. As such, even though their constructs are roughly similar, in practice User-Password is significantly more secure than CHAP-Password.

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 attacks are similar.

5.2.4. 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 this 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 or MS-CHAP are highly insecure, as any database compromise results in the immediate exposure of the clear-text passwords for all users. The security of those methods are best described as near zero, independent of any database compromise. The construction of MS-CHAP makes it easier to crack than CHAP-Password, which 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. Administrators should therefore prefer PAP over CHAP or MS-CHAP. Administrators should also 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 to any intermediate proxy. Those methods therefore lower the risk of password exposure even more than using PAP.

The conclusion is that administrators should avoid password-based authentication methods where at all possible. If passwords ahve to be used, wrap them in TLS (e.g. TTLS or PEAP). Where TLS cannot be used, prefer User-Password to CHAP-Password or MS-CHAP.

5.2.5. 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 RadSec, 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.

5.3. 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.

6. Other Protocol Failures

There are many other issues with RADIUS which are not directly related to security or privacy, but still have negative affects on security, privacy, and on operation of the protocol. At of the time of writing, those issues are being collated in [ISSUES].

Although the focus of this document is a review of RADIUS security, it is still important to discuss problems with the protocol in general. For example, there is implicitly a RADIUS state machine which correlates multiple types of packets, but that state machine is not defined anywhere. There are common practices which are secure but which are operationally expensive. RADIUS accounting is known to be inaccurate and is often inconsistent, as seen in [WBA-ACCT]

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.

6.1. 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 number of imperfect trade-offs.

6.1.1. 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 on the topic was made at IETF 118 in the RADEXT working group [WBA-ACCT].

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 some 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 for network operators. 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.

7. Privacy Considerations

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

The use of insecure transport protocols for RADIUS means that personally identifying information is 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.

8. Security Considerations

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

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."

Similarly, the MS-CHAP, was not officially deprecated, 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 the clear-text passwords for many users.

9. 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 that issue.

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

10. Changelog

* 00 - copy from draft-ietf-radext-deprecating-radius, and edit to contain only historical review contents.

* 01 - word smithing and updated analysis of CHAP-Password.

11. References

11.1. Normative References

[BCP14] 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>.

[I-D.ietf-radext-deprecating-radius] DeKok, A., "Deprecating Insecure Practices in RADIUS", Work in Progress, Internet-Draft, draft-ietf-radext-deprecating-radius-08, 6 November 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-radext-deprecating-radius-08>.

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

[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>.

11.2. 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>.

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

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

[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/>.

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

[RFC1994] Simpson, W., "PPP Challenge Handshake Authentication Protocol (CHAP)", RFC 1994, DOI 10.17487/RFC1994, August 1996, <https://www.rfc-editor.org/rfc/rfc1994>.

[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>.

[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>.

[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>.

[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>.

[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>.

[RFC6677] Hartman, S., Ed., Clancy, T., and K. Hoeper, "Channel-Binding Support for Extensible Authentication Protocol (EAP) Methods", RFC 6677, DOI 10.17487/RFC6677, July 2012, <https://www.rfc-editor.org/rfc/rfc6677>.

[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>.

[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>.

[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>.

[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>.

[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/>.

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

[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: alan.dekok@inkbridge.io