Workgroup: RADEXT Working Group
Internet-Draft: draft-ietf-radext-tls-psk-03
Published: 24 August 2023
Intended Status: Informational
Expires: 25 February 2024
Authors: A. DeKok
FreeRADIUS

RADIUS and TLS-PSK

Abstract

This document gives implementation and operational considerations for using TLS-PSK with RADIUS/TLS (RFC6614) and RADIUS/DTLS (RFC7360).

About This Document

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

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

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/.

Source for this draft and an issue tracker can be found at https://github.com/freeradius/radext-tls-psk.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 25 February 2024.

Table of Contents

1. Introduction

The previous specifications "Transport Layer Security (TLS) Encryption for RADIUS" [RFC6614] and "Datagram Transport Layer Security (DTLS) as a Transport Layer for RADIUS" [RFC7360] defined how (D)TLS can be used as a transport protocol for RADIUS. However, those documents do not provide guidance for using TLS-PSK with

RADIUS. This document provides that missing guidance, and gives implementation and operational considerations.

2. Terminology

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

3. History

TLS deployments usually rely on certificates in most common uses. However, we recognize that it may be difficult to fully upgrade client implementations to allow for certificates to be used with RADIUS/TLS and RADIUS/DTLS. These upgrades involve not only implementing TLS, but can also require significant changes to administration interfaces and application programming interfaces (APIs) in order to fully support certificates.

For example, unlike shared secrets, certificates expire. This expiration means that a working system using TLS can suddenly stop working. Managing this expiration can require additional notification APIs on RADIUS clients and servers which were previously not required when shared secrets were used.

Certificates also require the use of certification authorities (CAs), and chains of certificates. RADIUS implementations using TLS therefore have to track not just a small shared secret, but also potentially many large certificates. The use of TLS-PSK can therefore provide a simpler upgrade path for implementations to transition from RADIUS shared secrets to TLS.

4. General Discussion of PSKs and PSK Identities

Before we define any RADIUS-specific use of PSKs, we must first review the current standards for PSKs, and give general advice on PSKs and PSK identities.

The requirements in this section apply to both client and server implementations which use TLS-PSK. Client-specific and server-specific issues are discussed in more detail later in this document.

4.1. Requirements on PSKs

Reuse of a PSK in multiple versions of TLS (e.g. TLS 1.2 and TLS 1.3) is considered unsafe ([RFC8446] Section E.7). Where TLS 1.3 binds the PSK to a particular key derivation function, TLS 1.2 does not. This binding means that it is possible to use the same PSK in

different hashes, leading to the potential for attacking the PSK by comparing the hash outputs. While there are no known insecurities, these uses are not known to be secure, and should therefore be avoided.

[RFC9258] adds a key derivation function to the import interface of (D)TLS 1.3, which binds the externally provided PSK to the protocol version. In particular, that document:

> ... describes a mechanism for importing PSKs derived from external PSKs by including the target KDF, (D)TLS protocol version, and an optional context string to ensure uniqueness. This process yields a set of candidate PSKs, each of which are bound to a target KDF and protocol, that are separate from those used in (D)TLS 1.2 and prior versions. This expands what would normally have been a single PSK and identity into a set of PSKs and identities.

An implementation MUST NOT use the same PSK for TLS 1.3 and for earlier versions of TLS. This requirement prevents reuse of a PSK with multiple TLS versions, which prevents the attacks discussed in [RFC8446] Section E.7. The exact manner in which this requirement is enforced is implementation-specific. One possibility is to have two different PSKs. Another possibility is to forbid the use of TLS 1.3, or to forbid the use of TLS versions less than TLS 1.3.

It is RECOMMENDED that systems follow the directions of [RFC9257] Section 4 for the use of external PSKs in TLS. That document provides extremely useful guidance on generating and using PSKs.

Implementations MUST support PSKs of at least 32 octets in length, and SHOULD support PSKs of 64 octets. Implementations MUST require that PSKs be at least 16 octets in length. That is, short PSKs MUST NOT be permitted to be used.

Administrators SHOULD use PSKs of at least 24 octets, generated using a source of cryptographically secure random numbers. Implementers needing a secure random number generator should see [RFC8937] for for further guidance. PSKs are not passwords, and administrators should not try to manually create PSKs.

Passwords are generally intended to be remembered and entered by people on a regular basis. In contrast, PSKs are intended to be entered once, and then automatically saved in a system configuration. As such, due to the limited entropy of passwords, they are not acceptable for use with TLS-PSK, and would only be acceptable for use with a password-authenticated key exchange (PAKE) TLS method.

We also incorporate by reference the requirements of Section 10.2 of [RFC7360] when using PSKs.

In order to guide Implementers, we give an example script below which generates random PSKs. While the script is not portable to all possible systems, the intent here is to document a concise and simple method for creating PSKs which are both secure, and humanly manageable.

```
#!/usr/bin/env perl use MIME::Base32; use Crypt::URandom(); print
join('-', unpack("(A4)*", lc
encode_base32(Crypt::URandom::urandom(12)))), "\n";
```

This script reads 96 bits (12 octets) of random data from a secure source, encodes it in Base32, and then formats it to be more humanly manageable. The generated keys are of the form "2nw2-4cfi-nicw-3g2i-5vxq". This form of PSK will be accepted by any implementation which supports at least 24 octets for PSKs. Larger PSKs can be generated by passing larger values to the "urandom()" function.

4.1.1. Interaction between PSKs and Shared Secrets

Any shared secret used for RADIUS/UDP or RADIUS/TLS MUST NOT be used for TLS-PSK.

It is RECOMMENDED that RADIUS clients and server track all used shared secrets and PSKs, and then verify that the following requirements all hold true:

- no shared secret is used for more than one RADIUS client
- no PSK is used for more than one RADIUS client
- no shared secret is used as a PSK

Note that the shared secret of "radsec" given in [RFC6614] can be used across multiple clients, as that value is mandated by the specification. The intention here is to recommend best practices for administrators who enter site-local shared secrets.

There may be use-cases for using one shared secret across multiple RADIUS clients. There may similarly be use-cases for sharing a PSK across multiple RADIUS clients. Details of the possible attacks on reused PSKs are given in [RFC9257] Section 4.1.

There are few, if any, use-cases for using a PSK as a shared secret, or vice-versa.

Implementations SHOULD NOT provide user interfaces which allow both PSKs and shared secrets to be entered at the same time. There is too much of a temptation for administrators to enter the same value in both fields, which would violate the limitations given above. Implementations MUST NOT use a "shared secret" field as a way for administrators to enter PSKs. The PSK entry fields MUST be labeled as being related to PSKs, and not to shared secrets.

4.2. PSK Identities

It is RECOMMENDED that systems follow the directions of [RFC9257] Section 6.1.1 for the use of external PSK Identities in TLS. Note that the PSK identity is sent in the clear, and is therefore visible to attackers. Where privacy is desired, the PSK identity could be either an opaque token generated cryptographically, or perhaps in the form of a Network Access Identifier (NAI) [RFC7542], where the "user" portion is an opaque token. For example, an NAI could be "68092112@example.com". If the attacker already knows that the client is associated with "example.com", then using that domain name in the PSK identity offers no additional information. In contrast, the "user" portion needs to be both unique to the client and private, so using an opaque token there is a more secure approach.

Implementations MUST support PSK Identities of 128 octets, and SHOULD support longer PSK identities. We note that while TLS provides for PSK identities of up to 2^16-1 octets in length, there are few practical uses for extremely long PSK identities.

4.2.1. Security of PSK Identities

We note that the PSK identity is a field created by the connecting client. Since the client is untrusted until both the identity and PSK have been verified, both of those fields MUST be treated as untrusted. That is, a well-formed PSK Identity is likely to be in UTF-8 format, due to the requirements of [RFC4279] Section 5.1. However, implementations MUST support managing PSK identities as a set of undistinguished octets.

It is not safe to use a raw PSK Identity to look up a corresponding PSK. The PSK may come from an untrusted source, and may contain invalid or malicious data. For example, the identity may have incorrect UTF-8 format; or it may contain data which forms an injection attack for SQL, LDAP, REST or shell metacharacters; or it may contain embedded NUL octets which are incompatible with APIs which expect NUL terminated strings. The identity may also be up to 65535 octets long.

As such, implementations MUST validate the identity prior to it being used as a lookup key. When the identity is passed to an

external API (e.g. database lookup), implementations MUST either escape any characters in the identity which are invalid for that API, or else reject the identity entirely. The exact form of any escaping depends on the API, and we cannot document all possible methods here. However, a few basic validation rules are suggested, as outlined below. Any identity which is rejected by these validation rules SHOULD cause the server to close the TLS connection.

The suggested validation rules are as follows:

* Identities longer than a fixed maximum SHOULD be rejected. The limit is implementation dependent, but SHOULD NOT be less than 128, and SHOULD NOT be more than 1024.

* Identities which are not in UTF-8 format SHOULD be rejected. This includes any identity with embedded control characters, NUL octets, etc.

* Where the NAI format is expected, identities which are not in NAI format SHOULD be rejected

It is RECOMMENDED that implementations extend these rules with any additional validation which are found to be useful. For example, implementations and/or deployments could both generate PSK identities in a particular format for passing to client systems, and then also verify that any received identity matches that format. For example, a site could generate PSK identities which are composed of characters in the local language. The site could then reject identities which contain characters from other languages, even if those characters are valid UTF-8.

4.3. PSK and PSK Identity Sharing

While administrators may desire to share PSKs and/or PSK identities across multiple systems, such usage is NOT RECOMMENDED. Details of the possible attacks on reused PSKs are given in [RFC9257] Section 4.1.

Implementations MUST be able to configure a unique PSK and PSK identity for each possible client-server relationship. This configuration allows administrators desiring security to use unique PSKs for each such relationship. This configuration also allows administrators to re-use PSKs and PSK Identities where local policies permit.

Implementations SHOULD warn administrators if the same PSK identity and/or PSK is used for multiple client-server relationships.

5. Guidance for RADIUS Clients

Client implementations MUST allow the use of a pre-shared key (TLS-PSK) for RADIUS/TLS. The client implementation can then expose a user interface flag which is "TLS yes / no", and then also fields which ask for the PSK identity and PSK itself.

For TLS 1.3, Implementations MUST "psk_dhe_ke" Pre-Shared Key Exchange Mode in TLS 1.3 as discussed in [RFC8446] Section 4.2.9 and in [RFC9257] Section 6. Implementations MUST implement the recommended cipher suites in [RFC9325] Section 4.2 for TLS 1.2, and in [RFC9325] Section 4.2 for TLS 1.3.

5.1. PSK Identities

[RFC6614] is silent on the subject of PSK identities, which is an issue that we correct here. Guidance is required on the use of PSK identities, as the need to manage identities associated with PSK is a new requirement for NAS management interfaces, and is a new requirement for RADIUS servers.

RADIUS systems implementing TLS-PSK MUST support identities as per [RFC4279] Section 5.3, and MUST enable configuring TLS-PSK identities in management interfaces as per [RFC4279] Section 5.4.

Due to the security issues described above, RADIUS shared secrets cannot safely be used as TLS-PSKs. Therefore in order to prevent confusion between shared secrets and TLS-PSKs, management interfaces and APIs need to label PSK fields as "PSK" or "TLS-PSK", rather than as "shared secret".

RADIUS/TLS clients MUST still permit the configuration of a RADIUS server IP address or host name. Where [RFC7585] dynamic server lookups are used, that specification only makes provisions for servers to use certificates. Since there is no way to determine which PSK to use for a connection to a particular server, then TLS-PSK cannot be used with [RFC7585] dynamic lookups.

6. Guidance for RADIUS Servers

The following section(s) describe guidance for RADIUS server implementations and deployments. We first give an overview of current practices, and then extend and/or replace those practices for TLS-PSK.

Implementations MUST support the recommended cipher suites in [RFC9325] Section 4.2 for TLS 1.2, and in [RFC9325] Section 4.2 for

TLS 1.3. In order to future-proof these recommendations, we give the following recommendations:

* Implementations SHOULD use the "Recommended" cipher suites listed in the IANA "TLS Cipher Suites" registry,

  - for TLS 1.3, use the use "psk_dhe_ke" PSK key exchange mode,

  - for TLS 1.2 and earlier, use ciphersuites which require ephemeral keying.

6.1. Current Practices

RADIUS identifies clients by source IP address ([RFC2865] and [RFC6613]) or by client certificate ([RFC6614] and [RFC7585]). Neither of these approaches work for TLS-PSK. This section describes current practices and mandates behavior for servers which use TLS-PSK.

A RADIUS/UDP server is typically configured with a set of information per client, which includes at least the source IP address and shared secret. When the server receives a RADIUS/UDP packet, it looks up the source IP address, finds a client definition, and therefore the shared secret. The packet is then authenticated (or not) using that shared secret.

That is, the IP address is treated as the clients identity, and the shared secret is used to prove the clients authenticity and shared trust. The set of clients forms a logical database "client table", with the IP address as the key.

A server may be configured with additional site-local policies associated with that client. For example, a client may be marked up as being a WiFi Access Point, or a VPN concentrator, etc. Different clients may be permitted to send different kinds of requests, where some may send Accounting-Request packets, and other clients may not send accounting packets.

6.2. Practices for TLS-PSK

We define practices for TLS-PSK by analogy with the RADIUS/UDP use-case, and by extending the additional policies associated with the client. The PSK identity replaces the source IP address as the client identifier. The PSK replaces the shared secret as proof of client authenticity and shared trust.

In order to securely support dynamic source IP addresses for clients, we also require that servers limit clients based on a network range. The alternative would be to suggest that RADIUS

servers allow any source IP address to connect and try TLS-PSK, which could be a security risk.

In most situations a RADIUS server does not need to allow connections from the entire Internet. Where such connections are required, as with [RFC7585] or with a roaming consortium, TLS-PSK MUST NOT be used. It is significantly easier for an attacker to crack a PSK than to forge a client certificate.

For example, a RADIUS server could be configured to be accept connections from a source network of 192.0.2.0/24. The server could therefore discard any TLS connection request which comes from a source IP address outside of that network. In that case, there is no need to examine the PSK identity or to find the client definition. Instead, the IP source filtering policy would deny the connection before any TLS communication had been performed.

RADIUS servers need to be able to limit certain PSK identifiers to certain network ranges or IP addresses. That is, if a NAS is known to have a dynamic IP address within a particular subnet, the server should limit use of the NASes PSK to that subnet. This filtering can therefore help to catch configuration errors.

As some clients may have dynamic IP addresses, it is possible for a one PSK identity to appear at different source IP addresses over time. In addition, as there may be many clients behind one NAT gateway, there may be multiple RADIUS clients using one public IP address. RADIUS servers need to support multiple PSK identifiers at one source IP address.

That is, a server needs to support multiple different clients within one network range, multiple clients behind a NAT, and one client having different IP addresses over time. All of those use-cases are common and necessary.

The following section describes these requirements in more detail.

6.2.1. Requirements for TLS-PSK

A server supporting this specification MUST be configurable with a set of "allowed" network ranges from which clients are permitted to connect. Any connection from outside of the allowed range(s) MUST be rejected before any PSK identity is checked.

Once a connection has been made from a permitted source, the server MUST use the PSK identity as the logical identifier for a RADIUS client instead of the IP address as was done with RADIUS/UDP. The PSK identity is then looked up in the local "client table" which was described above.

Servers implementing this specification SHOULD also be able to associate an IP range or ranges for each client. Any connection from outside the allowed range(s) MUST be rejected, even if the PSK identity is known, and before the PSK is verified.

Note that this lookup is independent from the "allowed" network ranges which are checked when the TLS connection is made. The two range limitations can overlap completely, or only partially. In addition, the allowed network range(s) for any two clients may overlap partially, completely, or not at all. All of these possibilities MUST be supported by the server implementation. That is, it is possible for multiple clients to have the same allowed source IP address or range.

Once the source IP has been verified to be allowed for this particular client, the server authenticates the TLS connection via the PSK taken from the client definition. If the PSK is verified, the server then accepts the connection, and proceeds with RADIUS/TLS as per [RFC6614].

This process satisfies all of the requirements of the previous section.

Finally, if a RADIUS server does not recognize the PSK identity or if the identity is not permitted to use PSK, then the server MAY proceed with a certificate-based handshake. Since TLS 1.3 [RFC8446] uses PSK for resumption, another use-case is that the PSK identity used for resumption may be rejected, in which case a full handshake is performed.

7. Privacy Considerations

We make no changes over [RFC6614] and [RFC7360].

8. Security Considerations

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

9. IANA Considerations

There are no IANA considerations in this document.

RFC Editor: This section may be removed before final publication.

10. Acknowledgements

Thanks to the many reviewers in the RADEXT working group for positive feedback.

11. Changelog

*00 - initial version

*01 - update examples

12. References

12.1. 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/info/rfc8174>.

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

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

[RFC4279] Eronen, P., Ed. and H. Tschofenig, Ed., "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, DOI 10.17487/RFC4279, December 2005, <https://www.rfc-editor.org/info/rfc4279>.

[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/info/rfc7585>.

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

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

[RFC9258] Benjamin, D. and C. A. Wood, "Importing External Pre-Shared Keys (PSKs) for TLS 1.3", RFC 9258, DOI 10.17487/RFC9258, July 2022, <https://www.rfc-editor.org/info/rfc9258>.

12.2. Informative References

[RFC6613]

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

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

[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/info/rfc7360>.

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

[RFC8937] Cremers, C., Garratt, L., Smyshlyaev, S., Sullivan, N., and C. Wood, "Randomness Improvements for Security Protocols", RFC 8937, DOI 10.17487/RFC8937, October 2020, <https://www.rfc-editor.org/info/rfc8937>.

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

[RFC9325] Sheffer, Y., Saint-Andre, P., and T. Fossati, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 9325, DOI 10.17487/RFC9325, November 2022, <https://www.rfc-editor.org/info/rfc9325>.

Author's Address

Alan DeKok
FreeRADIUS

Email: aland@freeradius.org