Stream:	Internet Engineering Task Force (IETF)
RFC:	9813
BCP:	243
Category:	Best Current Practice
Published:	June 2025
ISSN:	2070-1721
Author:	A. DeKok
	InkBridge Networks

RFC 9813
Operational Considerations for RADIUS and TLS Pre-Shared Key (TLS-PSK)

Abstract

This document provides implementation and operational considerations for using TLS Pre-Shared Keys (TLS-PSKs) with RADIUS/TLS (RFC 6614) and RADIUS/DTLS (RFC 7360). The purpose of the document is to help smooth the operational transition from the use of RADIUS/UDP to RADIUS/TLS.

Status of This Memo

This memo documents an Internet Best Current Practice.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on BCPs is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9813.

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 Pre-Shared Keys (TLS-PSKs) with RADIUS. This document provides that missing guidance, and gives implementation and operational considerations.

To clearly distinguish the various secrets and keys, this document uses "shared secret" to mean "RADIUS shared secret", and "Pre-Shared Key (PSK)" to mean "secret keys that are used with TLS-PSK".

The purpose of the document is to help smooth the operational transition from the use of the insecure RADIUS/UDP to the use of the much more secure RADIUS/TLS. While using PSKs is often less preferable to using public or private keys, the operational model of PSKs follows the legacy RADIUS "shared secret" model. As such, it can be easier for implementers and operators to transition to TLS when that transition is offered as a series of small changes.

TLS-PSK is intended to be used in networks where the addresses of clients and servers are known, as with RADIUS/UDP. This situation is similar to the use case of RADIUS/UDP with shared secrets. TLS-PSK is not suitable for situations where clients dynamically discover servers, as there is no way for the client to dynamically determine which PSK should be used with a new server (or vice versa). In contrast, dynamic discovery [RFC7585] allows for a client or server to authenticate a previously unknown server or client, as the parties can be issued a certificate by a known Certification Authority (CA).

TLS-PSKs have the same issue of symmetric information between client and server: both parties know the secret key. A client could, in theory, pretend to be a server. In contrast, certificates are asymmetric, where it is impossible for the parties to assume the other's identity. Further discussion of this topic is contained in Section 4.3.

Unless it is explicitly called out that a recommendation applies to TLS or DTLS alone, each recommendation applies to both TLS and DTLS.

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.

External PSK
: A PSK (along with a related PSK Identity) that is administratively configured. That is, one that is external to TLS and is not created by the TLS subsystem.

Resumption PSK
: A PSK (along with a related PSK Identity) that is created by the TLS subsystem and/or application, for use with resumption.

3. Justification of PSK

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 that 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.

In terms of ongoing maintenance, it is generally simpler to maintain servers than clients. For one, there are many fewer servers than clients. Servers are also typically less resource constrained, and often run on general-purpose operating systems, where maintenance can be automated using widely available tools.

In contrast, clients are often numerous, resource constrained, and likely to be closed or proprietary systems with limited interfaces. As a result, it can be difficult to update these clients when a root CA expires. The use of TLS-PSK in such an environment may therefore offer management efficiencies.

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 that use TLS-PSK. Client-specific and server-specific issues are discussed in more detail later in this document.

4.1. Guidance for PSKs

We first give requirements for creating and managing PSKs, followed by usability guidance, and then a discussion of RADIUS shared secrets and their interaction with PSKs.

4.1.1. PSK Requirements

Reuse of a PSK in multiple versions of TLS (e.g., TLS 1.2 and TLS 1.3) is considered unsafe (see [RFC8446], Appendix E.7). Where TLS 1.3 binds the PSK to a particular key derivation function (KDF), 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. For this reason, an implementation **MUST NOT** use the same PSK for TLS 1.3 and for earlier versions of TLS. 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 versions less than TLS 1.3

[RFC9258] adds a KDF to the import interface of (D)TLS 1.3, which binds the externally provided PSK to the protocol version. That process is preferred to any trust-on-first-use (TOFU) mechanism. 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], Appendix 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 versions less than TLS 1.3.

Implementations **MUST** follow the directions of [RFC9257], Section 6 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 or more. As the PSKs are generally hashed before being used in TLS, the useful entropy of a PSK is limited by the size of the hash output. This output may be 256, 384, or 512 bits in length. Nevertheless, it is good practice for implementations to allow entry of PSKs of more than 64 octets, as the PSK may be in a form other than bare binary data. Implementations that limit the PSK to a maximum of 64 octets are likely to use PSKs that have much less than 512 bits of entropy. That is, a PSK with high entropy may be expanded via some construct (e.g., base32 as in the example below) in order to make it easier for people to interact with. Where 512 bits of entropy are input to an encoding construct, the output may be larger than 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, and PSKs **MUST** be random. The strength of the PSK is not determined by the length of the PSK, but instead by the number of bits of entropy that it contains. People are not good at creating data with high entropy, so a source of cryptographically secure random numbers **MUST** be used.

Where user passwords are generally intended to be remembered and entered by people on a regular basis, 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 [RFC8492]. Implementations **MUST** therefore support entry and storage of PSKs as undistinguished octets.

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

It may be tempting for servers to implement a TOFU policy with respect to clients. Such behavior is **NOT RECOMMENDED**. When servers receive a connection from an unknown client, they **SHOULD** log the PSK Identity, source IP address, and any other information that may be relevant. An administrator can then later look at the logs and determine the appropriate action to take.

4.1.2. Usability Guidance

PSKs in their purest form are opaque tokens, represented as an undistinguished series of octets. Where PSKs are expected to be managed automatically by scripted methods, this format is acceptable. However, in some cases it is necessary for administrators to share PSKs, in which case human-readable formats may be useful. Implementations **SHOULD** support entering PSKs as both binary data and via a human-readable form such as hex encoding.

Implementations **SHOULD** use a human-readable form of PSKs for interfaces that are intended to be used by people, and **SHOULD** allow for binary data to be entered via an application programming interface (API). Implementations **SHOULD** also allow for PSKs to be displayed in the hex encoding mentioned above, so that administrators can manually verify that a particular PSK is being used.

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

In order to guide implementers and administrators, we give example commands below that generate random PSKs from a locally secure source. While some commands may not work on some systems, one of the commands should succeed. The intent here is to document a concise and simple example of creating PSKs that are both secure and human-manageable. This document does not mandate that the PSKs follow this format or any other format.

```
openssl rand -base64 16

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

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

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

Only one of the above commands should be run; there is no need to run all of them. Each command reads 128 bits (16 octets) of random data from a secure source, and encodes it as printable and readable ASCII. This form of PSK will be accepted by any implementation that supports at least 32 octets for PSKs. Larger PSKs can be generated by changing the "16" number to a larger value. The above derivation assumes that the random source returns one bit of entropy for every bit of randomness that is returned. Sources failing that assumption are **NOT RECOMMENDED**.

4.1.3. Interaction Between PSKs and RADIUS 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 servers 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 no known use cases for using a PSK as a shared secret, or vice versa.

Implementations **MUST** reject configuration attempts that try to use the same value for the PSK and shared secret. To prevent administrative errors, implementations should not have interfaces that confuse PSKs and shared secrets or that 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. Similarly, using a "shared secret" field as a way for administrators to enter PSKs is bad practice. The PSK entry fields need to be labeled as being related to PSKs, and not to shared secrets.

4.2. PSK Identities

[RFC4279], Section 5.1 requires that PSK Identities be encoded in UTF-8 format. However, [RFC8446], Section 4.2.11 describes the "Pre-Shared Key Extension" and defines the ticket as an opaque string: "opaque identity<1..2^{16}-1>;". This PSK is then used in [RFC8446], Section 4.6.1 for resumption.

These definitions appear to be in conflict. This conflict is addressed in [RFC9257], Section 6.1.1, which discusses requirements for encoding and comparison of PSK Identities. Systems **MUST** follow the directions of [RFC9257], Section 6.1.1 when using or comparing PSK Identities for RADIUS/TLS. Implementations **MUST** follow the recommendations of [RFC8265] for handling PSK Identity strings.

In general, implementers should allow for external PSK Identities to follow [RFC4279] and be UTF-8, while PSK Identities provisioned as part of resumption are automatically provisioned, and therefore follow [RFC8446].

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 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.

It is up to administrators and implementations as to how they differentiate external PSK Identities from session resumption PSK Identities used in TLS 1.3 session tickets. While [RFC9257], Section 6.1.2 suggests the identities should be unique, it offers no concrete steps for how this differentiation may be done.

One approach could be to have externally provisioned PSK Identities contain an NAI such as what is described above, while session resumption PSK Identities contain large blobs of opaque, encrypted, and authenticated text. It should then be relatively straightforward to differentiate the two types of identities. One is UTF-8, the other is not. One is unauthenticated, the other is authenticated.

Servers **MUST** assign and/or track session resumption PSK Identities in a way that facilities the ability to distinguish those identities from externally configured PSK Identities, and that enables them to both find and validate the session resumption PSK. See Section 6.2.3 below for more discussion of issues around resumption.

A sample validation flow for TLS-PSK Identities could be performed via the following steps:

1. PSK Identities provided via an administration interface are enforced to be only UTF-8 on both client and server.
2. The client treats session tickets received from the server as opaque blobs.
3. When the server issues session tickets for resumption, the server ensures that they are not valid UTF-8.
4. One way to do this is to use stateless resumption with a forced non-UTF-8 key_name per [RFC5077], Section 4, such as by setting one octet to 0x00.
5. When receiving TLS, the server receives a Client-Hello containing a PSK, and checks if the identity is valid UTF-8:

   5.1. If yes, it searches for a preconfigured client that matches that identity.

      5.1.1. If the identity is found, it authenticates the client via PSK.

      5.1.2. Else, the identity is invalid, and the server closes the connection.

   5.2. If not, try resumption, which is usually handled by a TLS library.

      5.2.1. If the TLS library verifies the session ticket, then resumption has happened, and the connection is established.

      5.2.2. Else, the server ignores the session ticket, and performs a normal TLS handshake with a certificate.

This validation flow is only suggested. Other validation methods are possible.

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 that forms an injection attack for SQL, Lightweight Directory Access Protocol (LDAP), Representational State Transfer (REST), or shell meta characters; or it may contain embedded NUL octets that are incompatible with APIs that 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 that 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 that is rejected by these validation rules **MUST** cause the server to close the TLS connection.

The suggested validation rules for identities used outside of resumption are as follows:

- Identities **MUST** be checked to see if they have been provisioned as a resumption PSK. If so, then the session can be resumed, subject to any policies around resumption.
- 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. There is no purpose to allowing extremely long identities, and allowing them does little more than complicate implementations.
- Identities configured by administrators **SHOULD** be in UTF-8 format, and **SHOULD** be in the NAI format from [RFC7542]. While [RFC8446], Section 4.2.11 defines the PSK Identity as "opaque identity<1..2^{16}-1>", it is useful for administrators to manage human-readable identities in a recognizable format.

  This suggestion makes it easier to distinguish TLS-PSK Identities from TLS 1.3 resumption identities. It also allows implementations to more easily filter out unexpected or bad identities, and then to close inappropriate TLS connections.

It is **RECOMMENDED** that implementations extend these rules with any additional validation that is 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 that are composed of characters in the local language. The site could then reject identities that contain characters from other languages, even if those characters are valid UTF-8.

The purpose of these rules is to help administrators and implementers more easily manage systems using TLS-PSK, while also minimizing complexity and protecting from potential attackers' traffic. The rules follow a principle of "discard bad traffic quickly", which helps to improve system stability and performance.

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** support the ability 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 is also compatible with the practice of administrators who wish to reuse 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.

4.4. PSK Lifetimes

Unfortunately, [RFC9257] offers no guidance on PSK lifetimes other than to note in Section 4.2 that:

> Forward security may be achieved by using a PSK-DH mode or by using PSKs with short lifetimes.

It is **RECOMMENDED** that PSKs be rotated regularly. We offer no additional guidance on how often this process should occur. Changing PSKs has a non-zero cost. It is therefore up to administrators to determine how best to balance the cost of changing the PSK against the cost of a potential PSK compromise.

TLS-PSK **MUST** use modes such as PSK-DH or ECDHE_PSK [RFC5489] that provide forward secrecy. Failure to use such modes would mean that compromise of a PSK would allow an attacker to decrypt all sessions that had used that PSK.

As the PSKs are looked up by identity, the PSK Identity **MUST** also be changed when the PSK changes.

Servers **SHOULD** track when a connection was last received for a particular PSK Identity, and **SHOULD** automatically invalidate credentials when a client has not connected for an extended period of time. This process helps to mitigate the issue of credentials being leaked when a device is stolen or discarded.

5. Guidance for RADIUS Clients

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

For TLS 1.3, implementations **MUST** support the "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 [RFC8446], Section 9.1 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 "psk_dhe_ke" PSK key exchange mode.
  - For TLS 1.2 and earlier, use cipher suites that require ephemeral keying.

If a client initiated a connection using a PSK with TLS 1.3 by including the pre-shared key extension, it **MUST** close the connection if the server did not also select the pre-shared key to continue the handshake.

5.1. PSK Identities

This section offers advice on both requirements for PSK Identities and on usability.

5.1.1. PSK Identity Requirements

[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 PSKs is a new requirement for both Network Access Server (NAS) management interfaces and 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.

The historic methods of signing RADIUS packets have not yet been broken, but they are believed to be much less secure than modern TLS. Therefore, when a RADIUS shared secret is used to sign RADIUS/UDP or RADIUS/TCP packets, that shared secret **MUST NOT** be used with TLS-PSK. If the secrets were to be reused, then an attack on historic RADIUS cryptography could be trivially leveraged to decrypt TLS-PSK sessions.

With TLS-PSK, RADIUS/TLS clients **MUST** permit the configuration of a RADIUS server IP address or host name, because dynamic server lookups [RFC7585] can only be used if servers use certificates.

5.1.2. Usability Guidance

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".

6. Guidance for RADIUS Servers

In order to support clients with TLS-PSK, server implementations **MUST** allow the use of a pre-shared key (TLS-PSK) for RADIUS/TLS.

Systems that act as both client and server at the same time **MUST NOT** share or reuse PSK Identities between incoming and outgoing connections. Doing so would open up the systems to attack, as discussed in [RFC9257], Section 4.1.

For TLS 1.3, implementations **MUST** support the "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 [RFC8446], Section 9.1 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 "psk_dhe_ke" PSK key exchange mode.
  - For TLS 1.2 and earlier, use cipher suites that require ephemeral keying.

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.

6.1. Current Practices

RADIUS identifies clients by source IP address (see [RFC2865] and [RFC6613]) or by client certificate (see [RFC6614] and [RFC7585]). Neither of these approaches work for TLS-PSK. This section describes current practices and mandates behavior for servers that 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 Wi-Fi Access Point, 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. However, systems implementing RADIUS/TLS [RFC6614] and RADIUS/DTLS [RFC7360] **MUST** still use the shared secret as discussed in those specifications. Any PSK is only used by the TLS layer and has no effect on the RADIUS data that is being transported. That is, the RADIUS data transported in a TLS tunnel is the same no matter if client authentication is done via PSK or by client certificates. The encoding of the RADIUS data is entirely unaffected by the use (or not) of PSKs and client certificates.

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. When RADIUS servers do no source IP address filtering, it is easier for attackers to send malicious traffic to the server. An issue with a TLS library or even a TCP/IP stack could permit the attacker to gain unwarranted access. In contrast, when IP address filtering is done, attackers generally must first gain access to a secure network before attacking the RADIUS server.

Even where dynamic discovery [RFC7585] is not used, the use of TLS-PSK across unrelated organizations requires that those organizations share PSKs. Such sharing makes it easier for a client to impersonate a server, and vice versa. In contrast, when certificates are used, such impersonations are impossible. It is therefore **NOT RECOMMENDED** to use TLS-PSK across organizational boundaries.

When TLS-PSK is used in an environment where both client and server are part of the same organization, then impersonations only affect that organization. As TLS offers significant advantages over RADIUS/UDP, it is **RECOMMENDED** that organizations use RADIUS/TLS with TLS-PSK to replace RADIUS/UDP for all systems managed within the same organization. While such systems are generally located inside of private networks, there are no known security issues with using TLS-PSK for RADIUS/TLS connections across the public Internet.

If a client system is compromised, its complete configuration is exposed to the attacker. Exposing a client certificate means that the attacker can pretend to be the client. In contrast, exposing a PSK means that the attacker cannot only pretend to be the client, but can also pretend to be the server.

The main benefit of TLS-PSK, therefore, is that its operational processes are similar to that used for managing RADIUS/UDP, while gaining the increased security of TLS. However, it is still beneficial for servers to perform IP address filtering, in order to further limit their exposure to attacks.

6.2.1. IP Filtering

A server supporting this specification **MUST** perform IP address filtering on incoming connections. There are few reasons for a server to have a default configuration that allows connections from any source IP address.

A TLS-PSK server **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. It is **RECOMMENDED** that servers support IP address filtering even when TLS-PSK is not used.

The "allowed" network ranges could be implemented as a global list, or one or more network ranges could be tied to a client or clients. The intent here is to allow connections to be filtered by source IP address and to allow clients to be limited to a subset of network addresses. The exact method and representation of that filtering is up to an implementation.

Conceptually, the set of IP addresses and ranges, along with permitted clients and their credentials, form a logical "client table" that the server uses to both filter and authenticate clients. The client table should contain information such as allowed network ranges, PSK Identity and associated PSK, credentials for another TLS authentication method, or flags that indicate that the server should require a client certificate.

Once a server receives a connection, it checks the source IP address against the list of all allowed IP addresses or ranges in the client table. If none match, the connection **MUST** be rejected. That is, the connection **MUST** be from an authorized source IP address.

Once a connection has been established, the server **MUST NOT** process any application data inside of the TLS tunnel until the client has been authenticated. Instead, the server normally receives a TLS-PSK Identity from the client. The server then uses this identity to look up the client in the client table. If there is no matching client, the server **MUST** close the connection. The server then also checks if this client definition allows this particular source IP address. If the source IP address is not allowed, the server **MUST** close the connection.

Where the server does not receive TLS-PSK from the client, it proceeds with another authentication method such as client certificates. Such requirements are discussed elsewhere, most notably in [RFC6614] and [RFC7360].

An implementation may perform two independent IP address lookups: first to check if the connection is allowed at all, and second to check if the connection is authorized for this particular client. One or both checks may be used by a particular implementation. The two sets of IP addresses can overlap, and implementations **SHOULD** support that capability.

Depending on the implementation, one or more clients may share a list of allowed network ranges. Alternately, the allowed network ranges for two clients can overlap only partially, or not at all. All of these possibilities **MUST** be supported by the server implementation.

For example, a RADIUS server could be configured to accept connections from a source network of 192.0.2.0/24 or 2001:DB8::/32. The server could therefore discard any TLS connection request that 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.

As some clients may have dynamic IP addresses, it is possible for 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 **MUST** 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.

6.2.2. PSK Authentication

Once the source IP address 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].

If the PSK is not verified, then the server **MUST** close the connection. While TLS provides for fallback to other authentication methods such as client certificates, there is no reason for a client to be configured simultaneously with multiple authentication methods.

A client **MUST** use only one authentication method for TLS. An authentication method is either TLS-PSK, client certificates, or some other method supported by TLS.

That is, client configuration is relatively simple: use a particular set of credentials to authenticate to a particular server. While clients may support multiple servers and fail-over or load-balancing, that configuration is generally orthogonal to the choice of which credentials to use.

6.2.3. Resumption

It is **NOT RECOMMENDED** that servers enable resumption for sessions that use TLS-PSK. There are few practical benefits to supporting resumption and many complexities.

However, some systems will need to support both TLS-PSK and other TLS-based authentication methods such as certificates, while also supporting session resumption. It is therefore vital for servers to be able to distinguish the use case of TLS-PSK with preconfigured identities from TLS-PSK that is being used for resumptions.

The above discussion of PSK Identities is complicated by the use of PSKs for resumption in TLS 1.3. A server that receives a PSK Identity via TLS typically cannot query the TLS layer to see if this identity is for a resumed session (which is possibly for another TLS authentication method), or is instead a static pre-provisioned identity. This confusion complicates server implementations.

One way for a server to tell the difference between the two kinds of identities is via construction. Identities used for resumption can be constructed via a fixed format, such as what is recommended by [RFC5077], Section 4. A static pre-provisioned identity could be in the format of an NAI, as given in [RFC7542]. An implementation could therefore examine the incoming identity and determine from the identity alone what kind of authentication was being performed.

An alternative way for a server to distinguish the two kinds of identities is to maintain two tables. One table would contain static identities, as the logical client table described above. Another table could be the table of identities handed out for resumption. The server would then look up any PSK Identity in one table, and if it is not found, query the other one. Either an identity would be found in a table, in which case it can be authenticated, or the identity would not be found in either table, in which case it is unknown, and the server **MUST** close the connection.

As suggested in [RFC8446], TLS-PSK peers **MUST NOT** store resumption PSKs or tickets (and associated cached data) for longer than 604800 seconds (7 days), regardless of the PSK or ticket lifetime.

Since resumption in TLS 1.3 uses PSK Identities and keys, it is **NOT RECOMMENDED** to permit resumption of sessions when TLS-PSK is used. The use of resumption offers additional complexity with minimal additional benefits.

Where resumption is allowed with TLS-PSK, systems **MUST** cache data during the initial full handshake sufficiently enough to allow authorization decisions to be made during resumption. If the cached data cannot be retrieved securely, resumption **MUST NOT** be done. Instead, the system **MUST** perform a full handshake.

The data that needs to be cached is typically information such as the original PSK Identity, along with any policies associated with that identity.

Information from the original TLS exchange (e.g., the original PSK Identity) as well as related information (e.g., source IP addresses) may change between the initial full handshake and resumption. This change creates a "time-of-check time-of-use" (TOCTOU) security vulnerability. A malicious or compromised client could supply one set of data during the initial authentication and a different set of data during resumption, potentially allowing them to obtain access that they should not have.

If any authorization or policy decisions were made with information that has changed between the initial full handshake and resumption, and if changes may lead to a different decision, such decisions **MUST** be reevaluated. Systems **MUST** also reevaluate authorization and policy decisions during resumption, based on the information given in the new connection. Servers **MAY** refuse to perform resumption where the information supplied during resumption does not match the information supplied during the original authentication. If a safe decision is not possible, servers **MUST** instead continue with a full handshake.

6.2.4. Interaction with Other TLS Authentication Methods

When a server supports both TLS-PSK and client certificates, it **MUST** be able to accept authenticated connections from clients that may use either type of credentials, perhaps even from the same source IP address and at the same time. That is, servers are required to both authenticate the client and also to filter clients by source IP address. These checks both have to match in order for a client to be accepted.

7. Privacy Considerations

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

8. Security Considerations

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

Previous specifications discuss security considerations for TLS-PSK in detail. We refer the reader to Appendix E.7 of [RFC8446], [RFC9257], and [RFC9258]. Those documents are newer than [RFC6614] and [RFC7360], and therefore raise issues that were not considered during the initial design of RADIUS/TLS and RADIUS/DTLS.

Using TLS-PSK across the wider Internet for RADIUS can have different security considerations than for other protocols. For example, if TLS-PSK was for client/server communication with HTTPS, then having a PSK be exposed or broken could affect one user's traffic. In contrast, RADIUS contains credentials and personally identifiable information (PII) for many users. As a result, an attacker being able to see inside of a TLS-PSK connection for RADIUS would result in substantial amounts of PII being leaked, possibly including passwords.

When modes providing forward secrecy are used (e.g., ECDHE_PSK as seen in [RFC5489] and [RFC8442]), such attacks are limited to future sessions, and historical sessions are still secure.

9. IANA Considerations

This document has no IANA actions.

10. References

10.1. Normative References

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

[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>.

[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>.

[RFC8265] Saint-Andre, P. and A. Melnikov, "Preparation, Enforcement, and Comparison of Internationalized Strings Representing Usernames and Passwords", RFC 8265, DOI 10.17487/RFC8265, October 2017, <https://www.rfc-editor.org/info/rfc8265>.

[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>.

10.2. Informative References

[RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, DOI 10.17487/RFC5077, January 2008, <https://www.rfc-editor.org/info/rfc5077>.

[RFC5489] Badra, M. and I. Hajjeh, "ECDHE_PSK Cipher Suites for Transport Layer Security (TLS)", RFC 5489, DOI 10.17487/RFC5489, March 2009, <https://www.rfc-editor.org/info/rfc5489>.

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

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

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

[RFC8442] Mattsson, J. and D. Migault, "ECDHE_PSK with AES-GCM and AES-CCM Cipher Suites for TLS 1.2 and DTLS 1.2", RFC 8442, DOI 10.17487/RFC8442, September 2018, <https://www.rfc-editor.org/info/rfc8442>.

[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>.

[RFC8492] Harkins, D., Ed., "Secure Password Ciphersuites for Transport Layer Security (TLS)", RFC 8492, DOI 10.17487/RFC8492, February 2019, <https://www.rfc-editor.org/info/rfc8492>.

[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>.

[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>.

Acknowledgments

Thanks to the many reviewers in the RADEXT Working Group for positive feedback.

Author's Address

Alan DeKok
InkBridge Networks
Email: alan.dekok@inkbridge.io