Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 7, 2020

T. Pauly
Apple Inc.
D. Schinazi
Google LLC
C. Wood
Apple Inc.

November 04, 2019

TLS Ticket Requests
draft-ietf-tls-ticketrequests-04

Abstract

TLS session tickets enable stateless connection resumption for clients without server-side, per-client state. Servers vend an arbitrary number of session tickets to clients, at their discretion, upon connection establishment. Clients store and use tickets when resuming future connections. This document describes a mechanism by which clients can specify the desired number of tickets needed for future connections. This extension aims to provide a means for servers to determine the number of tickets to generate in order to reduce ticket waste, while simultaneously priming clients for future connection attempts.

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 May 7, 2020.

Copyright Notice

Table of Contents

1. Introduction

As per [RFC5077], and as described in [RFC8446], TLS servers vend clients an arbitrary number of session tickets at their own discretion in NewSessionTicket messages. There are two limitations with this design. First, servers choose some (often hard-coded) number of tickets vended per connection. Second, clients do not have a way of expressing their desired number of tickets, which can impact future connection establishment. For example, clients can open multiple TLS connections to the same server for HTTP, or race TLS connections across different network interfaces. The latter is especially useful in transport systems that implement Happy Eyeballs [RFC8305]. Since clients control connection concurrency and resumption, a standard mechanism for requesting more than one ticket is desirable.

This document specifies a new TLS extension - "ticket_request" - that can be used by clients to express their desired number of session tickets. Servers can use this extension as a hint of the number of

NewSessionTicket messages to vend. This extension is only applicable to TLS 1.3 [RFC8446], DTLS 1.3 [I-D.ietf-tls-dtls13], and future versions thereof.

1.1. Requirements Language

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 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Use Cases

The ability to request one or more tickets is useful for a variety of purposes:

- Parallel HTTP connections: To minimize ticket reuse while still improving performance, it may be useful to use multiple, distinct tickets when opening parallel connections. Clients must therefore bound the number of parallel connections they initiate by the number of tickets in their possession, or risk ticket re-use.

- Connection racing: Happy Eyeballs V2 [RFC8305] describes techniques for performing connection racing. The Transport Services Architecture implementation from [TAPS] also describes how connections can race across interfaces and address families. In cases where clients have early data to send and want to minimize or avoid ticket re-use, unique tickets for each unique connection attempt are useful. Moreover, as some servers may implement single-use tickets (and even session ticket encryption keys), distinct tickets will be needed to prevent premature ticket invalidation by racing.

- Connection priming: In some systems, connections can be primed or bootstrapped by a centralized service or daemon for faster connection establishment. Requesting tickets on demand allows such services to vend tickets to clients to use for accelerated handshakes with early data. (Note that if early data is not needed by these connections, this method SHOULD NOT be used.

Fresh handshakes SHOULD be performed instead.)

- Less ticket waste: Currently, TLS servers use application-specific, and often implementation-specific, logic to determine how many tickets to issue. By moving the burden of ticket count to clients, servers do not generate wasteful tickets. As an example, clients might only request one ticket during resumption. Moreover, as ticket generation might involve expensive computation, e.g., public key cryptographic operations, avoiding waste is desirable.

- Decline resumption: Clients can indicate they have no intention of resuming connections by sending a ticket request with count of zero.

3. Ticket Requests

Clients can indicate to servers their desired number of tickets for a single connection via the following "ticket_request" extension:

```
enum {
    ticket_request(TBD), (65535)
} ExtensionType;
```

Clients MAY send this extension in ClientHello. It contains the following structure:

```
struct {
    uint8 count;
} TicketRequestContents;
```

count The number of tickets desired by the client.

A supporting server MAY use TicketRequestContents.count when determining how many NewSessionTicket messages to send to a requesting client, and SHOULD place a limit on the number of tickets sent. The number of NewSessionTicket messages sent SHOULD be the minimum of the server's self-imposed limit and TicketRequestContents.count.

Servers that support ticket requests MUST NOT echo "ticket_request" in the EncryptedExtensions message. A client MUST abort the connection with an "illegal_parameter" alert if the "ticket_request" extension is present in the EncryptedExtensions message.

If a client receives a HelloRetryRequest, the presence (or absence) of the "ticket_request" extension MUST be maintained in the second ClientHello message. Moreover, if this extension is present, a client MUST NOT change the value of TicketRequestContents.count in the second ClientHello message.

4. IANA Considerations

IANA is requested to Create an entry, ticket_request(TBD), in the existing registry for ExtensionType (defined in [RFC8446]), with "TLS 1.3" column values being set to "CH", and "Recommended" column being set to "Yes".

5. Security Considerations

Ticket re-use is a security and privacy concern. Moreover, clients must take care when pooling tickets as a means of avoiding or amortizing handshake costs. If servers do not rotate session ticket encryption keys frequently, clients may be encouraged to obtain and use tickets beyond common lifetime windows of, e.g., 24 hours. Despite ticket lifetime hints provided by servers, clients SHOULD dispose of pooled tickets after some reasonable amount of time that mimics the ticket rotation period.

Servers that do not enforce a limit on the number of NewSessionTicket messages sent in response to a "ticket_request" extension could leave themselves open to DoS attacks, especially if ticket creation is expensive.

6. Acknowledgments

The authors would like to thank David Benjamin, Eric Rescorla, Nick Sullivan, Martin Thomson, Hubert Kario, and other members of the TLS Working Group for discussions on earlier versions of this draft.

7. References

7.1. Normative References

[I-D.ietf-tls-dtls13]
Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", draft-ietf-tls-dtls13-33 (work in progress), October 2019.

[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>.

[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>.

[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>.

7.2. Informative References

[RFC8305] Schinazi, D. and T. Pauly, "Happy Eyeballs Version 2: Better Connectivity Using Concurrency", RFC 8305, DOI 10.17487/RFC8305, December 2017, <https://www.rfc-editor.org/info/rfc8305>.

[TAPS] Brunstrom, A., Pauly, T., Enghardt, T., Grinnemo, K., Jones, T., Tiesel, P., Perkins, C., and M. Welzl, "Implementing Interfaces to Transport Services", draft-ietf-taps-impl-04 (work in progress), July 2019.

Authors' Addresses

Tommy Pauly
Apple Inc.
One Apple Park Way
Cupertino, California 95014
United States of America

Email: tpauly@apple.com

David Schinazi
Google LLC
1600 Amphitheatre Parkway
Mountain View, California 94043
United States of America

Email: dschinazi.ietf@gmail.com

Christopher A. Wood
Apple Inc.
One Apple Park Way
Cupertino, California 95014
United States of America

Email: cawood@apple.com