

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 6 June 2021

T. Pauly
Apple Inc.
D. Schinazi
Google LLC
C.A. Wood
Cloudflare
3 December 2020

TLS Ticket Requests **draft-ietf-tls-ticketrequests-07**

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.

Discussion Venues

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

Source for this draft and an issue tracker can be found at <https://github.com/tlswg/draft-ietf-tls-ticketrequest>.

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 6 June 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the [Trust Legal Provisions](#) and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	2
1.1.	Requirements Language	3
2.	Use Cases	3
3.	Ticket Requests	4
4.	IANA Considerations	6
5.	Performance Considerations	6
6.	Security Considerations	6
7.	Acknowledgments	7
8.	References	7
8.1.	Normative References	7
8.2.	Informative References	7
	Authors' Addresses	8

[1.](#) Introduction

As as described in [[RFC8446](#)], TLS servers vend clients an arbitrary number of session tickets at their own discretion in NewSessionTicket messages. There are at least three limitations with this design.

First, servers vend some (often hard-coded) number of tickets per connection. Some server implementations return a different default number of tickets for session resumption than for the initial connection that created the session. No static choice, whether fixed, or resumption-dependent is ideal for all situations.

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 parallel 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 for avoiding ticket reuse. See [\[RFC8446\]](#), [Appendix C.4](#) for discussion of ticket reuse risks.

Third, all tickets in the client's possession ultimately derive from some initial connection. Especially when the client was initially authenticated with a client certificate, that session may need to be refreshed from time to time. Consequently, a server may periodically force a new connection even when the client presents a valid ticket. When that happens, it is possible that any other tickets derived from the same original session are equally invalid. A client avoids a full handshake on subsequent connections if it replaces all stored tickets with new ones obtained from the just performed full handshake. The number of tickets the server should vend for a new connection may therefore need to be larger than the number for routine resumption.

This document specifies a new TLS extension - "ticket_request" - that clients can use to express their desired number of session tickets. Servers can use this extension as a hint for 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 of (D)TLS.

[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 improve performance, a client may open parallel connections. To avoid ticket reuse, the client may use distinct tickets on each connection. 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 such cases, clients may use more than one ticket while racing

connection attempts in order to establish one successful connection. Having multiple tickets equips clients with enough tickets to initiate connection racing while avoiding ticket re-use and ensuring that their cache of tickets does not empty during such races. Moreover, as some servers may implement single-use tickets, distinct tickets prevent premature ticket invalidation by racing.

- * 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 do not intend to resume a connection by sending a ticket request with count of zero.

3. Ticket Requests

As discussed in [Section 1](#), clients may want different numbers of tickets for new or resumed connections. Clients may indicate to servers their desired number of tickets to receive on a single connection, in the case of a new or resumed 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 new_session_count;  
    uint8 resumption_count;  
} ClientTicketRequest;
```

new_session_count The number of tickets desired by the client if the server chooses to negotiate a new connection.

resumption_count The number of tickets desired by the client if the server is willing to resume using a ticket presented in this ClientHello.

A client starting a new connection SHOULD set `new_session_count` to the desired number of session tickets and `resumption_count` to 0. Once a client's ticket cache is primed, a `resumption_count` of 1 is a good choice that allows the server to replace each ticket with a new ticket, without over-provisioning the client with excess tickets. However, clients which race multiple connections and place a separate ticket in each will ultimately end up with just the tickets from a single resumed session. In that case, clients can send a `resumption_count` equal to the number of connections they are attempting in parallel. (Clients which send a `resumption_count` less than the number of parallel connection attempts might end up with zero tickets.)

When a client presenting a previously obtained ticket finds that the server nevertheless negotiates a new connection, the client SHOULD assume that any other tickets associated with the same session as the presented ticket are also no longer valid for resumption. This includes tickets obtained during the initial (new) connection and all tickets subsequently obtained as part of subsequent resumptions. Requesting more than one ticket in cases when servers complete a new connection helps keep the session cache primed.

Servers SHOULD NOT send more tickets than requested for the connection type selected by the server (new or resumed connection). Moreover, servers SHOULD place a limit on the number of tickets they are willing to send, whether for new or resumed connections, to save resources. Therefore, the number of `NewSessionTicket` messages sent will typically be the minimum of the server's self-imposed limit and the number requested. Servers MAY send additional tickets, typically using the same limit, if the tickets that are originally sent are somehow invalidated.

A server which supports and uses a client `"ticket_request"` extension MUST also send the `"ticket_request"` extension in the `EncryptedExtensions` message. It contains the following structure:

```
struct {  
    uint8 expected_count;  
} ServerTicketRequestHint;
```

`expected_count` The number of tickets the server expects to send in this connection.

Servers MUST NOT send the `"ticket_request"` extension in any handshake message, including `ServerHello` or `HelloRetryRequest` messages. A client MUST abort the connection with an `"illegal_parameter"` alert if the `"ticket_request"` extension is present in any server handshake 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 `ClientTicketRequest` 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, EE", and "Recommended" column being set to "Y".

5. Performance Considerations

Servers can send tickets in `NewSessionTicket` messages any time after the server `Finished` message (see [\[RFC8446\]](#); [Section 4.6.1](#)). A server which chooses to send a large number of tickets to a client can potentially harm application performance if the tickets are sent before application data. For example, if the transport connection has a constrained congestion window, ticket messages could delay sending application data. To avoid this, servers should prioritize sending application data over tickets when possible.

6. 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 cached tickets after some reasonable amount of time that mimics the session ticket encryption key rotation period. Specifically, as specified in [Section 4.6.1 of \[RFC8446\]](#), clients MUST NOT cache tickets for longer than 7 days.

In some cases, a server may send `NewSessionTicket` messages immediately upon sending the server `Finished` message rather than waiting for the client `Finished`. If the server has not verified the client's ownership of its IP address, e.g., with the TLS Cookie extension (see [\[RFC8446\]](#); [Section 4.2.2](#)), an attacker may take advantage of this behavior to create an amplification attack proportional to the count value toward a target by performing a (DTLS) key exchange over UDP with spoofed packets. Servers SHOULD limit the number of `NewSessionTicket` messages they send until they have verified the client's ownership of its IP address.

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.

7. 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. Viktor Dukhovni contributed text allowing clients to send multiple counts in a ticket request.

8. References

8.1. Normative References

- [I-D.ietf-tls-dtls13]
Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", Work in Progress, Internet-Draft, [draft-ietf-tls-dtls13-39](http://www.ietf.org/internet-drafts/draft-ietf-tls-dtls13-39.txt), 2 November 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-tls-dtls13-39.txt>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](https://www.rfc-editor.org/info/rfc2119), [RFC 2119](https://www.rfc-editor.org/info/rfc2119), DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in [RFC 2119](https://www.rfc-editor.org/info/rfc2119) Key Words", [BCP 14](https://www.rfc-editor.org/info/rfc8174), [RFC 8174](https://www.rfc-editor.org/info/rfc8174), 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](https://www.rfc-editor.org/info/rfc8446), DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.

8.2. Informative References

- [RFC8305] Schinazi, D. and T. Pauly, "Happy Eyeballs Version 2: Better Connectivity Using Concurrency", [RFC 8305](https://www.rfc-editor.org/info/rfc8305), 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", Work in Progress, Internet-Draft, [draft-ietf-taps-impl-08](http://www.ietf.org/internet-drafts/draft-ietf-taps-impl-08.txt), 2

November 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-taps-impl-08.txt>>.

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
Cloudflare
101 Townsend St
San Francisco,
United States of America

Email: caw@heapingbits.net

