rtcweb
Internet-Draft
Intended status: Standards Track
Expires: September 29, 2012

M. Thomson
Skype

March 28, 2012

Using Datagram Transport Layer Security (DTLS) For Interactivity Connectivity Establishment (ICE) Connectivity Checking: ICE-DTLS draft-thomson-rtcweb-ice-dtls-00

Abstract

Interactivity Connectivity Establishment (ICE) connectivity checking using the Datagram Transport Layer Security (DTLS) handshake is described. The DTLS handshake provides sufficient information to identify valid candidates and establish consent.

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 http://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 September 29, 2012.

Copyright Notice

Table of Contents

1. Introduction

User experience with real time applications depends greatly on low latency. At session setup time, optimizing the total number of network round trips between a user action and the receipt of media is key to improving user experience.

Interactivity Connectivity Establishment (ICE) [RFC5245] performs a crucial function in establishing a functional transport flow between peers in the presence of network address translation (NAT) middleboxes. Performing the complete ICE procedure can add significant additional latency to session setup.

A complete ICE connectivity check and candidate nomination requires - at best - one additional round trip. This assumes aggressive nomination and all the associated drawbacks. Without aggressive nomination, two round trips are added.

A transport session that is secured with Datagram Transport Layer Security (DTLS) [RFC6347] requires at least two additional round trips to establish once ICE negotiation completes.

A method is described that removes the need for blocking ICE connectivity checks and nominations with only minimal additional state overhead at each peer. This allows a secured session setup without additional latency. In addition, the negative consequences of aggressive nomination are avoided.

Peers identify candidates using information encoded in the DTLS cookie. This avoids the need for a DTLS HelloVerifyRequest and corresponding round trip.

Continuing consent to receive media in the session is verified using the DTLS heartbeat extension [RFC6520].

An extension to the Session Description Protocol (SDP) [RFC4566] identifies candidates that support this method.

2. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [RFC2119] and indicate requirement levels for compliant implementations.

The term "server" and "client" are used in this document to refer to

the peers that act in the DTLS server and client roles. This is distinct from the caller and callee who are participants in the media session.

3. ICE using DTLS

The DTLS ClientHello and ServerHello messages can be used as a replacement connectivity check request.

3.1. Without Optimization

A complete session setup using ICE and DTLS optimistically requires between 5 and 7 round trips to complete. From the instant that the caller initiates a call:

1. Caller gathers server reflexive and relay candidates.

2. Caller signals call creation. Callee signals response, callee sends first connectivity check - a Session Traversal Utilities for NAT (STUN) [RFC5389] Binding request.

3. Caller responds to connectivity check, creates own connectivity check. Callee responds to connectivity check.

4. Caller creates an ICE nomination (a STUN Binding request with USE-CANDIDATE set). Callee responds to nomination. Callee sends DTLS ClientHello.

5. Caller sends a DTLS HelloVerifyRequest with a cookie. Callee re-sends the ClientHello with the cookie.

6. Caller sends a DTLS ServerHello and associated messages. Callee sends a DTLS Finished and ChangeCipherSpec.

7. Caller validates the Finished message. Caller can begin media transmission. Caller sends a Finished message. Callee validates the Finished message. Callee can begin media transmission.

This sequence assumes that no packets are lost or blocked during setup. It also assumes that the callee creates the DTLS ClientHello in order to save half a round trip, which probably doesn't correspond with established practice.

The delay imposed by acquiring consent for the call from the callee potentially dwarfs any delays from session setup. Low latency setup is most applicable to pre-authorized sessions and situations where an automated system is able to rapidly accept calls.

3.2. With Optimization

Using the DTLS handshake messages for connectivity checking takes far fewer round trips in the best case:

1. Caller gathers server reflexive and relay candidates.

2. Caller signals call creation, indicating support for ICE-DTLS. Callee signals response. Callee sends DTLS ClientHello including a specially formatted cookie.

3. Caller sends a DTLS ServerHello and associated messages. Callee sends a DTLS Finished and ChangeCipherSpec.

4. Caller validates the Finished message. Caller can begin media transmission. Caller sends a Finished message. Callee validates the Finished message. Callee can begin media transmission.

3.3. Connectivity Check and Nomination

Validating the DTLS handshake requires that peers retain copies of the handshake messages. A (DTLS) client that sends multiple ClientHello messages in place of connectivity checks MUST retain the contents of each of those messages in order to later successfully complete the DTLS handshake.

This requires additional state retention - especially at the (DTLS) server - when multiple connectivity checks are sent and received. This information does not add significantly to the state burden imposed by ICE. Assuming that the client does not alter the configuration for each session, the Random and Cookie fields will vary in every ClientHello.

Continuing the handshake past the *Hello messages indicates that the candidate pair in use has been nominated. Once a Finished message has been received for any session, any state retained for other candidates within the same ICE negotiation MAY be discarded and any sessions abandoned.

3.4. ICE Controller Selection

This process implies that the peer in the DTLS client role is acting as the ICE controller. Since both peers need to send packets in order to create the session, a mechanism for determining which of the two clients is the controller is necessary.

A DTLS peer that receives a ClientHello when it has one of its own ClientHello messages outstanding continues to send. A peer that

receives a message is more likely to be able to respond to those messages on the same transport flow than it is to successfully send its own messages. Based on the receipt of a message alone, there is no way to tell if the other peer has successfully received any other packets. Therefore, the peer creates the ServerHello and associated messages in response.

A peer that receives a ServerHello when its own ServerHello is outstanding must choose whether to end the handshake, or whether it is the controller. Only the ICE controller is able to continue the handshake.

Unless the ICE controller is selected by other means, the ICE controller is the peer that has the largest numeric public key value, taken from the DTLS Certificate message.

3.5. DTLS Cookie Handling

The HelloVerifyRequest message in DTLS is used to determine that a client is genuine and is able to receive packets. This allows a server to avoid allocating state for a client based on the receipt of an arbitrary packet.

Where a signaling relationship already exists between client and server, the cookie exchange provides less utility. However, the cookie still provides protection against receipt of spoofed ClientHello packets.

Populating the DTLS cookie with information from the signaling allows a server to determine that a packet is genuine without requiring a round trip for confirmation. Using the ICE username fragment and password ensures that no additional state is required to use this method.

The DTLS cookie includes information from the signaling, chosen by the DTLS server, plus a HMAC [RFC2104]. The HMAC that ensures that access to packets does not enable the sending of spoofed ClientHello messages.

The value of the Cookie is formed using a method similar to the ICE short term credential mechanism. The ICE user is formed by concatenating the username fragment from the peer, a colon (':') and the local username fragment. The HMAC is calculated using the ICE password as the key, with the text being a concatenation of the Random value from the DTLS ClientHello and the ICE user. No other information from the ClientHello is authenticated.

The Cookie is calculated as follows:

```
ice_user = ice_ufrag_peer | ':' | ice_ufrag_local

Cookie = HMAC[hash](ice_pwd_peer, Random | ice_user) | ice_user
```

... where '|' implies concatenation. This method is roughly equivalent to the one used by ICE when forming STUN packets.

Hash agility is achieved by signaling the hash to use. Implementations MUST support SHA-1 [RFC3174]. See Section 6 for an example.

Multiple hash algorithms MAY be signaled to indicate that multiple different cookies are accepted.

In order to support ICE usernames longer than 12 octets or hash algorithms other than SHA-1, DTLS 1.2 [RFC6347] MUST be supported. DTLS 1.2 expands the size of the cookie field to 255 octets.

4. Indicating Continuing Consent to Receive

An important security concern for the web context is that a media sender has a means of checking that a media receiver consents to receive that media. Since consent can be revoked, a regular check is necessary to ensure that media is not unwanted.

The DTLS heartbeat extension [RFC6520] provides a means of signaling liveness of a DTLS session. A successful response also indicates that the receiver consents to the continuing receipt of data.

In order to support this feature, peers MUST use the heartbeat extension and MUST NOT send peer_not_allowed_to_send in the handshake.

5. Parallel STUN

The main drawback of this optimization is that it is not possible to gather peer reflexive addresses using the connectivity check. Performing a STUN Binding request in parallel to the DTLS handshake allows a client to gather a peer reflexive address without additional latency.

6. Signaling in SDP

The Session Description Protocol (SDP) [RFC4566] can be used to signal support for this feature.

The "ice-dtls" attribute is set to the textual name of the hash function used in the HMAC. Valid values are taken from the IANA registry of Hash Function Textual Names [IANAHashText], with multiple values separated by spaces.

7. Security Considerations

The Random field in the DTLS handshake provides more entropy than the corresponding field in STUN (224 bits vs. 96 bits). Both use an equivalent HMAC method. This makes guessing the correct ClientHello considerably harder than guessing the correct STUN Binding request.

The state maintained by peers using the DTLS handshake is increased marginally over what is required to perform ICE. Each peer is required to retain all the messages in the DTLS handshake in order to correctly form the Finished message. Since each peer also authenticates every ClientHello, only hosts with access to signaling are able to create state in this fashion.

State accumulation can be limited using the same methods recommended for the STUN Amplification Attack (Section 18.5.2 in [RFC5245]).

8. IANA Considerations

[[Note to IANA/RFC Editor: Please replace instance of RFCXXXX with the number of the published RFC and remove this note.]]

This specification defines a new SDP 'ice-dtls' attribute following the procedures of Section 8.2.4 of [RFC4566].

Contact Name: Martin Thomson, martin.thomson@gmail.com

Attribute Name: ice-dtls

Long Form: ice-dtls

Type of Attribute: session- or media-level

Charset Considerations: The attribute is not subject to the charset attribute.

Purpose: This attribute indicates that DTLS is supported as an alternative mechanism for ICE connectivity checking and consent. The values of the attribute indicate the hash functions that can be used to calculate the value of the DTLS cookie.

Appropriate Values: A whitespace-separated list of values taken from the IANA registry of Hash Function Textual Names [IANAHashText].

9. Acknowledgments

Cullen Jennings provided the initial idea for this optimization.

10. References

10.1. Normative References

[IANAHashText] Internet Asigned Numbers Authority, "IANA registry of Hash Function Textual Names".

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3174] Eastlake, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, September 2001.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC6520] Seggelmann, R., Tuexen, M., and M. Williams, "Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension", RFC 6520, February 2012.

10.2. Informative References

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

Author's Address

Martin Thomson
Skype
3210 Porter Drive
Palo Alto, CA 94304
US

Phone: +1 650-353-1925
Email: martin.thomson@gmail.com