Network Working Group
Internet-Draft
Intended status: Informational
Expires: October 22, 2017

M. Thomson
E. Rescorla
Mozilla

April 20, 2017

Unknown Key Share Attacks on uses of Transport Layer Security with the Session Description Protocol (SDP)
draft-thomson-mmusic-sdp-uks-00

Abstract

Unknown key-share attacks on the use of Datagram Transport Layer Security for the Secure Real-Time Transport Protocol (DTLS-SRTP) and its use with Web Real-Time Communications (WebRTC) identity assertions are described. Simple mitigation techniques are defined.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 2
2. Unknown Key-Share Attack 3
  2.1. Attack Overview 3
  2.2. Limits on Attack Feasibility 4
  2.3. Example 4
  2.4. Interactions with Key Continuity 6
3. Adding a Session Identifier 6
  3.1. The sdp_tls_id TLS Extension 7
4. WebRTC Identity Binding 8
  4.1. The webrtc_id_hash TLS Extension 8
5. Session Concatenation 9
6. Security Considerations 10
7. IANA Considerations 10
8. References 11
  8.1. Normative References 11
  8.2. Informative References 12
Appendix A. Acknowledgements 13
Authors' Addresses 13

1. Introduction

The use of Transport Layer Security (TLS) [RFC5246] with the Session Description Protocol (SDP) [RFC4566] is defined in [RFC4572]. Further use with Datagram Transport Layer Security (DTLS) [RFC6347] and the Secure Real-time Transport Protocol (SRTP) [RFC3711] is defined as DTLS-SRTP [RFC5763].

In these specifications, key agreement is performed using TLS or DTLS, with authentication being tied back to the session description (or SDP) through the use of certificate fingerprints. Communication peers check that a hash, or fingerprint, provided in the SDP matches the certificate that is used in the TLS or DTLS handshake. This is defined in [RFC4572].

The design in RFC 4572 relies on the integrity of the signaling channel. Certificate fingerprints are assumed to be provided by the communicating peers and carried by the signaling channel without being subject to modification. However, this design is vulnerable to an unknown key-share (UKS) attack where a misbehaving endpoint is able to advertise a key that it does not control. This leads to the creation of sessions where peers are confused about the identify of the participants.

An extension to TLS is defined that can be used to mitigate this attack.

A similar attack is possible with sessions that use WebRTC identity (see Section 5.6 of [I-D.ietf-rtcweb-security-arch]). This issue and a mitigation for it is discussed in more detail in Section 4.

2. Unknown Key-Share Attack

In an unknown key-share attack [UKS], a malicious participant in a protocol claims to control a key that is in reality controlled by some other actor. This arises when the identity associated with a key is not properly bound to the key.

In usages of TLS and DTLS that use SDP for negotiation, an endpoint is able to acquire the certificate fingerprint another entity. By advertising that fingerprint in place of one of its own, the malicious endpoint can cause its peer to communicate with a different peer, even though it believes that it is communicating with the malicious endpoint.

When the identity of communicating peers is established by higher-layer signaling constructs, such as those in SIP [RFC4474] or WebRTC [I-D.ietf-rtcweb-security-arch], this allows an attacker to bind their own identity to a session with any other entity.

By substituting the the fingerprint of one peer for its own, an attacker is able to cause a session to be established where one endpoint has an incorrect value for the identity of its peer. However, the peer does not suffer any such confusion, resulting in each peer involved in the session having a different view of the nature of the session.

This attack applies to any communications established based on the SDP "fingerprint" attribute [RFC4572].

2.1. Attack Overview

This vulnerability can be used by an attacker to create a session where there is confusion about the communicating endpoints.

A SIP endpoint or WebRTC endpoint that is configured to reuse a certificate can be attacked if it is willing to conduct two concurrent calls, one of which is with an attacker. The attacker can arrange for the victim to incorrectly believe that is calling the attacker when it is in fact calling a second party. The second party correctly believes that it is talking to the victim.

In a related attack, a single call using WebRTC identity can be attacked so that it produces the same outcome. This attack does not require a concurrent call.

2.2. Limits on Attack Feasibility

The use of TLS with SDP depends on the integrity of session signaling. Assuming signaling integrity limits the capabilities of an attacker in several ways. In particular:

1. An attacker can only modify the parts of the session signaling for a session that they are part of, which is limited to their own offers and answers.

2. No entity will complete communications with a peer unless they are willing to participate in a session with that peer.

The combination of these two constraints make the spectrum of possible attacks quite limited. An attacker is only able to switch its own certificate fingerprint for a valid certificate that is acceptable to its peer. Attacks therefore rely on joining two separate sessions into a single session.

The second condition is not necessary with WebRTC identity if the victim has or is configured with a target peer identity (this is defined in [WEBRTC]). Furthermore, any identity displayed by a browser could be different to the identity used by the application, since the attack affects the browser's understanding of the peer's identity.

2.3. Example

In this example, two outgoing sessions are created by the same endpoint. One of those sessions is initiated with the attacker, another session is created toward another honest endpoint. The attacker convinces the endpoint that their session has completed, and that the session with the other endpoint has succeeded.

```
     Norma                Mallory             Patsy
     (fp=N)                -----              (fp=P)
       |                     |                  |
       +---Offer1 (fp=N)--->|                  |
       +-----Offer2 (fp=N)-------------------->|
       |<--------------------Answer2 (fp=P)----+
       |<--Answer1 (fp=P)---+                  |
       |                     |                  |
       |======DTLS1====>(Forward)====DTLS1====>|
       |<=====DTLS2=====(Forward)<===DTLS2=====|
       |======Media1===>(Forward)====Media1===>|
       |<=====Media2====(Forward)<===Media2====|
       |                     |                  |
       |======DTLS2===========>(Drop)          |
       |                     |                  |
```

In this case, Norma is willing to conduct two concurrent sessions. The first session is established with Mallory, who falsely uses Patsy's certificate fingerprint. A second session is initiated between Norma and Patsy. Signaling for both sessions is permitted to complete.

Once complete, the session that is ostensibly between Mallory and Norma is completed by forwarding packets between Norma and Patsy. This requires that Mallory is able to intercept DTLS and media packets from Patsy so that they can be forwarded to Norma at the transport addresses that Norma associates with the first session.

The second session - between Norma and Patsy - is permitted to continue to the point where Patsy believes that it has succeeded. This ensures that Patsy believes that she is communicating with Norma. In the end, Norma believes that she is communicating with Mallory, when she is actually communicating with Patsy.

Though Patsy needs to believe that the second session is successful, Mallory has no real interest in seeing that session complete. Mallory only needs to ensure that Patsy does not abandon the session prematurely. For this reason, it might be necessary to permit the answer from Patsy to reach Norma to allow Patsy to receive a call completion signal, such as a SIP ACK. Once the second session completes, Mallory causes any DTLS packets sent by Norma to Patsy to be dropped.

For the attacked session to be sustained beyond the point that Norma detects errors in the second session, Mallory also needs to block any signaling that Norma might send to Patsy asking for the call to be abandoned. Otherwise, Patsy might receive a notice that the call is failed and thereby abort the call.

This attack creates an asymmetry in the beliefs about the identity of peers. However, this attack is only possible if the victim (Norma) is willing to conduct two sessions concurrently, and if the same certificate - and therefore SDP "fingerprint" attribute value - is used in both sessions.

2.4. Interactions with Key Continuity

Systems that use key continuity might be able to detect an unknown key-share attack if a session with the actual peer (i.e., Patsy in the example) was established in the past. Whether this is possible depends on how key continuity is implemented.

Implementations that maintain a single database of identities with an index on peer keys could discover that the identity saved for the peer key does not match the claimed identity. Such an implementation could notice the disparity between the actual keys (Patsy) and the expected keys (Mallory).

In comparison, implementations that first match based on peer identity could treat an unknown key-share attack as though their peer had used a newly-configured device. The apparent addition of a new device could generate user-visible notices (e.g., "Mallory appears to have a new device"). However, such an event is not always considered alarming; some implementations might silently save a new key.

3. Adding a Session Identifier

An attack on DTLS-SRTP is possible because the identity of peers involved is not established prior to establishing the call. Endpoints use certificate fingerprints as a proxy for authentication, but as long as fingerprints are used in multiple calls, they are vulnerable to attacks of the sort described.

The solution to this problem is to assign a new identifier to communicating peers. Each endpoint assigns their peer a unique identifier during call signaling. The peer echoes that identifier in the TLS handshake, binding that identity into the session. Including this new identity in the TLS handshake means that it will be covered by the TLS Finished message, which is necessary to authenticate it (see [SIGMA]). Validating that peers use the correct identifier then means that the session is established between the correct two endpoints.

This solution relies on the unique identifier given to DTLS sessions using the SDP "tls-id" attribute [I-D.ietf-mmusic-dtls-sdp]. This field is already required to be unique. Thus, no two offers or answers from the same client will have the same value.

A new "sdp_tls_id" extension is added to the TLS or DTLS handshake for connections that are established as part of the same call or real-time session. This carries the value of the "tls-id" attribute and provides integrity protection for its exchange as part of the TLS or DTLS handshake.

3.1. The sdp_tls_id TLS Extension

The "sdp_tls_id" TLS extension carries the unique identifier that an endpoint selects. The value includes the "tls-id" attribute from the SDP that the endpoint generated when negotiating the session.

The "extension_data" for the "sdp_tls_id" extension contains a SdpTlsId struct, described below using the syntax defined in [RFC5246]:

```
struct {
   opaque tls_id<20..255>;
} SdpTlsId;
```

The "tls_id" field of the extension includes the value of the "tls-id" SDP attribute as defined in [I-D.ietf-mmusic-dtls-sdp] (that is, the "tls-id-value" ABNF production). The value of the "tls-id" attribute is encoded using ASCII [RFC0020].

Where RTP and RTCP [RFC3550] are not multiplexed, it is possible that the two separate DTLS connections carrying RTP and RTCP can be switched. This is considered benign since these protocols are usually distinguishable. RTP/RTCP multiplexing is advised to address this problem.

The "sdp_tls_id" extension is included in a ClientHello and either ServerHello (for TLS and DTLS versions less than 1.3) or EncryptedExtensions (for TLS 1.3). In TLS 1.3, the "sdp_tls_id" extension MUST NOT be included in a ServerHello.

Endpoints MUST check that the "tls_id" parameter in the extension that they receive includes the "tls-id" attribute value that they received in their peer's session description. Comparison can be performed with either the decoded ASCII string or the encoded octets. An endpoint that receives a "sdp_tls_id" extension that is not identical to the value that it expects MUST abort the connection with a fatal "handshake_failure" alert.

An endpoint that is communicating with a peer that does not support this extension will receive a ClientHello, ServerHello or EncryptedExtensions that does not include this extension. An endpoint MAY choose to continue a session without this extension in

order to interoperate with peers that do not implement this specification.

In TLS 1.3, the "sdp_tls_id" extension MUST be sent in the EncryptedExtensions message.

4. WebRTC Identity Binding

The identity assertion used for WebRTC [I-D.ietf-rtcweb-security-arch] is bound only to the certificate fingerprint of an endpoint and can therefore be copied by an attacker along with any SDP "fingerprint" attributes.

The problem is compounded by the fact that an identity provider is not required to verify that the entity requesting an identity assertion controls the keys. Nor is it currently able to perform this validation. This is not an issue because verification is not a necessary condition for a secure protocol, nor would it be sufficient as established in [SIGMA].

A simple solution to this problem is suggested by [SIGMA]. The identity of endpoints is included under a message authentication code (MAC) during the cryptographic handshake. Endpoints are then expected to validate that their peer has provided an identity that matches their expectations.

In TLS, the Finished message provides a MAC over the entire handshake, so that including the identity in a TLS extension is sufficient to implement this solution. Rather than include a complete identity assertion - which could be sizeable - a collision-resistant hash of the identity assertion is included in a TLS extension. Peers then need only validate that the extension contains a hash of the identity assertion they received in signaling in addition to validating the identity assertion.

Endpoints MAY use the "sdp_tls_id" extension in addition to this so that two calls between the same parties can't be altered by an attacker.

4.1. The webrtc_id_hash TLS Extension

The "webrtc_id_hash" TLS extension carries a hash of the identity assertion that communicating peers have exchanged.

The "extension_data" for the "webrtc_id_hash" extension contains a WebrtcIdentityHash struct, described below using the syntax defined in [RFC5246]:

```
   struct {
      opaque assertion_hash<0..32>;
   } WebrtcIdentityHash;
```

A WebRTC identity assertion is provided as a JSON [RFC7159] object that is encoded into a JSON text. The resulting string is then encoded using UTF-8 [RFC3629]. The content of the "webrtc_id_hash" extension are produced by hashing the resulting octets with SHA-256 [FIPS180-2]. This produces the 32 octets of the assertion_hash parameter, which is the sole contents of the extension.

The SDP "identity" attribute includes the base64 [RFC4648] encoding of the same octets that were input to the hash. The "webrtc_id_hash" extension is validated by performing base64 decoding on the value of the SDP "identity" attribute, hashing the resulting octets using SHA-256, and comparing the results with the content of the extension.

Identity assertions might be provided by only one peer. An endpoint that does not produce an identity assertion MUST generate an empty "webrtc_id_hash" extension in its ClientHello. This allows its peer to include a hash of its identity assertion. An endpoint without an identity assertion MUST omit the "webrtc_id_hash" extension from its ServerHello or EncryptedExtensions message.

A peer that receives a "webrtc_id_hash" extension that is not equal to the value of the identity assertion from its peer MUST immediately fail the TLS handshake with an error. This includes cases where the "identity" attribute is not present in the SDP.

A "webrtc_id_hash" extension that is any length other than 0 or 32 is invalid and MUST cause the receiving endpoint to generate a fatal "decode_error" alert.

A peer that receives an identity assertion, but does not receive a "webrtc_id_hash" extension MAY choose to fail the connection, though it is expected that implementations that were written prior to the existence of this document will not support these extensions for some time.

In TLS 1.3, the "webrtc_id_hash" extension MUST be sent in the EncryptedExtensions message.

5. Session Concatenation

Use of session identifiers does not prevent an attacker from establishing two concurrent sessions with different peers and forwarding signaling from those peers to each other. Concatenating two signaling sessions creates a situation where both peers believe

that they are talking to the attacker when they are talking to each other.

Session concatention is possible at higher layers: an attacker can establish two independent sessions and simply forward any data it receives from one into the other. This kind of attack is prevented by systems that enable peer authentication such as WebRTC identity [I-D.ietf-rtcweb-security-arch] or SIP identity [RFC4474].

In the absence of any higher-level concept of peer identity, the use of session identifiers does not prevent session concatenation. The value to an attacker is limited unless information from the TLS connection is extracted and used with the signaling. For instance, a key exporter [RFC5705] might be used to create a shared secret or unique identifier that is used in a secondary protocol.

If a secondary protocol uses the signaling channel with the assumption that the signaling and TLS peers are the same then that protocol is vulnerable to attack. The identity of the peer at the TLS layer is not guaranteed to be the same as the identity of the signaling peer.

It is important to note that multiple connections can be created within the same signaling session. An attacker might concatenate only part of a session, choosing to terminate some connections (and optionally forward data) while arranging to have peers interact directly for other connections. It is even possible to have different peers interact for each connection. This means that the actual identity of the peer for one connection might differ from the peer on another connection.

Information extracted from a TLS connection therefore MUST NOT be used in a secondary protocol outside of that connection if that protocol relies on the signaling protocol having the same peers. Similarly, data from one TLS connection MUST NOT be used in other TLS connections even if they are established as a result of the same signaling session.

6. Security Considerations

This entire document contains security considerations.

7. IANA Considerations

This document registers two extensions in the TLS "ExtensionType Values" registry established in [RFC5246]:

- o The "sdp_tls_id" extension has been assigned a code point of TBD; it is recommended and is marked as "Encrypted" in TLS 1.3.
- o The "webrtc_id_hash" extension has been assigned a code point of TBD; it is recommended and is marked as "Encrypted" in TLS 1.3.

8. References

8.1. Normative References

[FIPS180-2]
Department of Commerce, National., "NIST FIPS 180-2, Secure Hash Standard", August 2002.

[I-D.ietf-mmusic-dtls-sdp]
Holmberg, C. and R. Shpount, "Using the SDP Offer/Answer Mechanism for DTLS", draft-ietf-mmusic-dtls-sdp-23 (work in progress), April 2017.

[I-D.ietf-rtcweb-security-arch]
Rescorla, E., "WebRTC Security Architecture", draft-ietf-rtcweb-security-arch-12 (work in progress), June 2016.

[RFC0020] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969, <http://www.rfc-editor.org/info/rfc20>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <http://www.rfc-editor.org/info/rfc3629>.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <http://www.rfc-editor.org/info/rfc3711>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <http://www.rfc-editor.org/info/rfc4566>.

[RFC4572] Lennox, J., "Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP)", RFC 4572, DOI 10.17487/RFC4572, July 2006, <http://www.rfc-editor.org/info/rfc4572>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[RFC5763] Fischl, J., Tschofenig, H., and E. Rescorla, "Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS)", RFC 5763, DOI 10.17487/RFC5763, May 2010, <http://www.rfc-editor.org/info/rfc5763>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

8.2. Informative References

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <http://www.rfc-editor.org/info/rfc3550>.

[RFC4474] Peterson, J. and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 4474, DOI 10.17487/RFC4474, August 2006, <http://www.rfc-editor.org/info/rfc4474>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <http://www.rfc-editor.org/info/rfc4648>.

[RFC5705] Rescorla, E., "Keying Material Exporters for Transport Layer Security (TLS)", RFC 5705, DOI 10.17487/RFC5705, March 2010, <http://www.rfc-editor.org/info/rfc5705>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[SIGMA] Krawczyk, H., "SIGMA: The 'SIGn-and-MAc'approach to authenticated Diffie-Hellman and its use in the IKE protocols", Annual International Cryptology Conference, Springer, pp. 400-425 , 2003.

[UKS] Blake-Wilson, S. and A. Menezes, "Unknown Key-Share Attacks on the Station-to-Station (STS) Protocol", Lecture Notes in Computer Science 1560, Springer, pp. 154-170 , 1999.

[WEBRTC] Bergkvist, A., Burnett, D., Narayanan, A., Jennings, C., and B. Aboba, "WebRTC 1.0: Real-time Communication Between Browsers", W3C WD-webrtc-30160531 , May 2016.

Appendix A. Acknowledgements

This problem would not have been discovered if it weren't for discussions with Sam Scott, Hugo Krawczyk, and Richard Barnes. A solution similar to the one presented here was first proposed by Karthik Bhargavan who provided valuable input on this document. Thyla van der Merwe assisted with a formal model of the solution. Adam Roach and Paul E. Jones provided useful review and input.

Authors' Addresses

Martin Thomson
Mozilla

Email: martin.thomson@gmail.com

Eric Rescorla
Mozilla

Email: ekr@rftm.com