Behave
Internet-Draft
Intended status: Standards Track
Expires: September 5, 2012

Muthu A M. Perumal
D. Wing
R. Ram Mohan
Cisco Systems
H. Kaplan
Acme Packet
March 4, 2012

STUN Usage for Consent Freshness
draft-muthu-behave-consent-freshness-00

Abstract

This document describes a STUN usage that enables WebRTC implementations to verify the peer consent for continuing to receive traffic on a candidate pair ICE is using for a media component after session establishment. Verification of peer consent is necessary to ensure that a malicious JavaScript cannot use the browser as a platform for launching attacks. This form of consent verification also serves the purpose of refreshing NAT bindings.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

Consent verification is the mechanism using which WebRTC implementations can verify the peer consent for receiving traffic on candidate media transport addresses. This has two parts

1. Verifying peer consent for receiving traffic on candidate media transport addresses at session establishment.
2. Verifying peer consent for continuing to receive traffic on candidate media transport addresses after session establishment.

WebRTC implements are required to perform STUN connectivity checks at session establishment as part of ICE procedures [RFC5245]. This takes care of the first part of the consent verification described above.

After session establishment ICE requires STUN Binding indications to be used for refreshing NAT bindings for a candidate pair ICE is using for a media component. Since a STUN Binding indication does not evoke a response, it cannot be used for the second part of the consent verification describes above.

This document defines a new STUN method, Consent and describes a STUN usage based on STUN Consent request/response to enable verifying peer consent for continuing to receive traffic on a candidate pair ICE is using for a media component after session establishment. This usage also serves the purpose of refreshing NAT bindings for that candidate pair.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Definitions

Consent Freshness: It is the mechanism of verifying peer consent for continuing to receive traffic on a candidate pair ICE is using for a media component after ICE has concluded. This document uses completion of session establishment synonymous with the conclusion of ICE.

Transport Address: The combination of an IP address and port number (such as a UDP or TCP port number).

4. Design Considerations

As described in the Introduction, STUN indications are not suitable for performing consent freshness. Hence, performing consent freshness requires the use of STUN request/response.

ICE requires the usage of message integrity with STUN using its short-term credential mechanism. The need for this mechanism goes beyond just security and is required for the correct operation of the ICE connectivity check procedures; without message integrity the connectivity checks can yield false positives, as described in Appendix B section B.4 of the ICE specification. However, this problem is not applicable for consent freshness, since consent freshness is performed only after ICE concludes.

One of the reasons for ICE choosing STUN Binding indications for keepalives is because Binding indication allows integrity to be disabled, allowing for better performance. This is useful for large-scale endpoints, such as PSTN gateways and SBCs as described in Appendix B section B.10 of the ICE specification.

STUN requires the 96 bits transaction ID to be uniformly and randomly chosen from the interval 0 .. 2**96-1, and be cryptographically random. This is deemed sufficient for consent freshness from a security perspective.

Considering these aspects, STUN request/response without the message integrity and short/long-term credential mechanisms have been chosen for consent freshness in this document. In addition, in order to ensure that this STUN usage does not leave an existing ICE implementation broken, this document chooses to define a new STUN method, Consent to be used for consent freshness.

5. STUN Consent Method

The STUN message type field from the STUN specification [RFC5389] is shown below

```
                      2  3  4  5  6  7  8  9  A  B  C  D  E  F

                     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+
                     |M |M |M |M |M |C |M |M |M |C |M |M |M |M |
                     |b |a |9 |8 |7 |1 |6 |5 |4 |0 |3 |2 |1 |0 |
                     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Figure 1: Format of STUN Message Type Field

Here the bits in the message type field are shown as most significant (Mb) through least significant (M0). C1 and C0 represent a 2-bit encoding of the class. Mb through M0 represent a 12-bit encoding of the method. The Consent method is encoded as 0b000000000010.

A Consent request has class=0b00 (request) and method=0b000000000010 (Consent) and is encoded into the first 16 bits of the STUN header as 0x0002.

A Consent success response has class=0b10 (success response) and method=0b000000000010 (Consent) and is encoded into the first 16 bits of the STUN header as 0x0102.

A Consent error response has class=0b11 (error response) and method=0b000000000010 (Consent) and is encoded into the first 16 bits of the STUN header as 0x0112.

A Consent indication has class=0b01 (indication) and method=0b000000000010 (Consent) and is encoded into the first 16 bits of the STUN header as 0x0012.

6. STUN Consent Method Processing

Processing of the STUN Consent method is similar to the processing of the STUN Binding method except that the procedures pertaining to the message integrity and short/long-term credential mechanisms are not applicable. In particular, the USERNAME and MESSAGE-INTEGRITY attributes are not included in a Consent request or response.

6.1. Generating a Consent Request

TBD

6.2. Receiving a Consent Request

TBD

6.3. Generating a Consent Response

TBD

6.4. Receiving a Consent Response

TBD

7. Performing Consent Freshness

The consent freshness described here is performed using STUN Consent request/response after ICE concludes. The FINGERPRINT mechanism MUST be used for consent freshness.

7.1. Generating Consent Freshness Request

A STUN agent generates a consent freshness request by constructing a STUN Consent request. If there has been no STUN Consent request generated on a candidate pair ICE is using for a media component for Tc seconds, the agent MUST generate a consent freshness request.

If the transaction fails because all retransmissions of the request time out or an ICMP error or a Consent failure received, then the agent checks if Tm seconds have elapsed since the transaction began. If Tm seconds have elapsed, then consent freshness is considered to have failed in the direction from the agent to its peer. Otherwise, the agent retries the Consent request after Tc seconds.

If the second transaction also fails, then the agent checks if Tm seconds have elapsed since the first transaction began. If Tm seconds have elapsed, then consent freshness is considered to have failed in the direction from the agent to its peer. Otherwise, the agent retries the Consent request again after Tc seconds.

If the third transaction also fails, then consent freshness is considered to have failed in the direction from the agent to its peer.

If consent freshness fails either because of Tm seconds elapsing or the all retries exhausting, the agent MUST stop sending traffic on that candidate pair. However, the agent MAY continue to receive traffic from the peer. At this point, if a consent freshness initiated from the peer succeeds, the agent MAY initiate a consent freshness request. If this consent freshness request succeeds, the agent MAY start sending traffic to the peer on this candidate pair.

Both Tc and Tm SHOULD be configurable. Tc SHOULD have a default of

15 seconds and MUST NOT be configured to less than 15 seconds. Tm SHOULD have a default of 30 seconds.

8. Examples

The examples in this section show how consent freshness requests are retried with Tc and Tm set to their default values.

8.1. Example 1

Suppose the consent freshness requests generated by an agent evoke an ICM error almost immediately every time. Then the agent will retry the requests at times 15 seconds and 30 seconds approximately before concluding the consent freshness to have failed in the direction from the agent to its peer.

8.2. Example 2

Suppose the STUN RTO (Retransmission TimeOut) is 500 ms. Then an agent initiating consent freshness will retransmit the request at 500 ms, 1500 ms, 3500 ms, 7500 ms, 15500 ms, and 31500 ms. If the agent has not received a response after 39500 ms, the agent will consider the transaction to have timed out, as described in section 7.2.1 of the STUN specification. At this point, since Tm seconds have elapsed since the transaction began, the agent will consider the consent freshness to have failed in the direction from the agent to its peer.

9. Generating Consent Freshness Response

A STUN agent receiving a consent freshness request for a candidate pair ICE is using for a media component MUST generate a STUN Consent response.

10. SDP Extension for Consent Freshness

ICE provides the a=ice-options SDP attribute for defining new ICE extensions in a backward compatible manner. This document defines a new ICE extension "consent" to negotiate the consent freshness usage described in this document.

An offerer that supports ICE and wishes to perform the consent freshness as described in this document MUST include the a=ice-options:consent session level SDP attribute in the SDP offer.

An answerer that agrees to perform consent freshness as described in

this document MUST include the a=ice-options:consent session level SDP attribute in the SDP answer.

If the SDP offer does not contain the a=ice-options:consent session level SDP attribute, the SDP answer MUST NOT contain the a=ice-options:consent session level SDP attribute.

11. Interaction with Keepalives used for Refreshing NAT Bindings

An implementation that uses the consent freshness described in this document has no need to also perform the keepalives described in ICE [RFC5245] or RTP keepalive [RFC6263], as they both force recurring messages to be sent over the UDP port used by RTP. Thus, an implementation that uses the consent freshness described in this document SHOULD NOT also do the keepalives described in ICE [RFC5245] or RTP keepalives [RFC6263] for the UDP port used for RTP.

The RTCP port, if different from the RTP port, does not need consent freshness (does it?) and continues to use the keepalives described in ICE [RFC5245] or RTP keepalives [RFC6263] to refresh NAT bindings.

12. Open Items

1. This document describes a consent freshness mechanism for an ICE usage. Is there a need for consent freshness even when ICE is not used (certainly not for WebRTC where ICE is mandatory)?
2. If the RTCP port is different from the RTP port, does the RTCP port need consent freshness? It looks unlikely given that RTCP is typically rate limited.

13. Security Considerations

TBD

14. IANA Considerations

TBD

15. Acknowledgement

TBD

16. Normative References

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

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

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

[RFC6263] Marjou, X. and A. Sollaud, "Application Mechanism for Keeping Alive the NAT Mappings Associated with RTP / RTP Control Protocol (RTCP) Flows", RFC 6263, June 2011.

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

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

Authors' Addresses

Muthu Arul Mozhi Perumal
Cisco Systems
Cessna Business Park
Sarjapur-Marathahalli Outer Ring Road
Bangalore, Karnataka 560103
India

Email: mperumal@cisco.com

Dan Wing
Cisco Systems
821 Alder Drive
Milpitas, California 95035
USA

Email: dwing@cisco.com

Ram Mohan R
Cisco Systems
Cessna Business Park
Sarjapur-Marathahalli Outer Ring Road
Bangalore, Karnataka 560103
India

Email: rmohanr@cisco.com

Hadriel Kaplan
Acme Packet

Email: hkaplan@acmepacket.com