

Internet Engineering Task Force (IETF)  
Request for Comments: 7675  
Category: Standards Track  
ISSN: 2070-1721

M. Perumal  
Ericsson  
D. Wing  
Cisco Systems, Inc.  
R. Ravindranath  
T. Reddy  
Cisco Systems  
M. Thomson  
Mozilla  
October 2015

## Session Traversal Utilities for NAT (STUN) Usage for Consent Freshness

### Abstract

To prevent WebRTC applications, such as browsers, from launching attacks by sending traffic to unwilling victims, periodic consent to send needs to be obtained from remote endpoints.

This document describes a consent mechanism using a new Session Traversal Utilities for NAT (STUN) usage.

### Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in [Section 2 of RFC 5741](#).

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at <http://www.rfc-editor.org/info/rfc7675>.

## Copyright Notice

Copyright (c) 2015 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 (<http://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  |
| 2. Applicability . . . . .                     | 3  |
| 3. Terminology . . . . .                       | 3  |
| 4. Design Considerations . . . . .             | 4  |
| 5. Solution . . . . .                          | 4  |
| 5.1. Expiration of Consent . . . . .           | 5  |
| 5.2. Immediate Revocation of Consent . . . . . | 6  |
| 6. DiffServ Treatment for Consent . . . . .    | 7  |
| 7. DTLS Applicability . . . . .                | 7  |
| 8. Security Considerations . . . . .           | 7  |
| 9. References . . . . .                        | 8  |
| 9.1. Normative References . . . . .            | 8  |
| 9.2. Informative References . . . . .          | 8  |
| Acknowledgements . . . . .                     | 9  |
| Authors' Addresses . . . . .                   | 10 |

## 1. Introduction

To prevent attacks on peers, endpoints have to ensure the remote peer is willing to receive traffic. Verification of peer consent before sending traffic is necessary in deployments like WebRTC to ensure that a malicious JavaScript cannot use the browser as a platform for launching attacks. This is performed both when the session is first established to the remote peer using Interactive Connectivity Establishment (ICE) [[RFC5245](#)] connectivity checks, and periodically for the duration of the session using the procedures defined in this document.

When a session is first established, ICE implementations obtain an initial consent to send by performing STUN connectivity checks. This document describes a new STUN usage with exchange of request and

response messages that verifies the remote peer's ongoing consent to receive traffic. This consent expires after a period of time and needs to be continually renewed, which ensures that consent can be terminated.

This document defines what it takes to obtain, maintain, and lose consent to send. Consent to send applies to a single 5-tuple. How applications react to changes in consent is not described in this document. The consent mechanism does not update the ICE procedures defined in [RFC5245].

Consent is obtained only by full ICE implementations. An ICE-lite agent (as defined in Section 2.7 of [RFC5245]) does not generate connectivity checks or run the ICE state machine. Hence, an ICE-lite agent does not generate consent checks and will only respond to any checks that it receives. No changes are required to ICE-lite implementations in order to respond to consent checks, as they are processed as normal ICE connectivity checks.

## 2. Applicability

This document defines what it takes to obtain, maintain, and lose consent to send using ICE. Sections 4.4 and 5.3 of [WebRTC-SA] further explain the value of obtaining and maintaining consent.

Other applications that have similar security requirements to verify peer consent before sending non-ICE packets can use the consent mechanism described in this document. The mechanism of how applications are made aware of consent expiration is outside the scope of the document.

## 3. 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].

**Consent:** The mechanism of obtaining permission from the remote endpoint to send non-ICE traffic to a remote transport address. Consent is obtained using ICE. Note that this is an application-level consent; no human intervention is involved.

**Consent Freshness:** Maintaining and renewing consent over time.

**Transport Address:** The remote peer's IP address and UDP or TCP port number.

#### 4. Design Considerations

Although ICE requires periodic keepalive traffic to keep NAT bindings alive (see [Section 10 of \[RFC5245\]](#) and also [\[RFC6263\]](#)), those keepalives are sent as STUN Indications that are send-and-forget, and do not evoke a response. A response is necessary for consent to continue sending traffic. Thus, we need a request/response mechanism for consent freshness. ICE can be used for that mechanism because ICE implementations are already required to continue listening for ICE messages, as described in [Section 10 of \[RFC5245\]](#). STUN binding requests sent for consent freshness also serve the keepalive purpose (i.e., to keep NAT bindings alive). Because of that, dedicated keepalives (e.g., STUN Binding Indications) are not sent on candidate pairs where consent requests are sent, in accordance with [Section 20.2.3 of \[RFC5245\]](#).

When Secure Real-time Transport Protocol (SRTP) is used, the following considerations are applicable. SRTP is encrypted and authenticated with symmetric keys; that is, both sender and receiver know the keys. With two party sessions, receipt of an authenticated packet from the single remote party is a strong assurance the packet came from that party. However, when a session involves more than two parties, all of whom know each other's keys, any of those parties could have sent (or spoofed) the packet. Such shared key distributions are possible with some Multimedia Internet KEYing (MIKEY) [\[RFC3830\]](#) modes, Security Descriptions [\[RFC4568\]](#), and Encrypted Key Transport (EKT) [\[EKT\]](#). Thus, in such shared keying distributions, receipt of an authenticated SRTP packet is not sufficient to verify consent.

The mechanism proposed in the document is an optional extension to the ICE protocol; it can be deployed at one end of the two-party communication session without impact on the other party.

#### 5. Solution

Initial consent to send traffic is obtained using ICE [\[RFC5245\]](#). An endpoint gains consent to send on a candidate pair when the pair enters the Succeeded ICE state. This document establishes a 30-second expiry time on consent. 30 seconds was chosen to balance the need to minimize the time taken to respond to a loss of consent with the desire to reduce the occurrence of spurious failures.

ICE does not identify when consent to send traffic ends. This document describes two ways in which consent to send ends: expiration of consent and immediate revocation of consent, which are discussed in the following sections.

### 5.1. Expiration of Consent

A full ICE implementation obtains consent to send using ICE. After ICE concludes on a particular candidate pair and whenever the endpoint sends application data on that pair consent is maintained following the procedure described in this document.

An endpoint **MUST NOT** send data other than the messages used to establish consent unless the receiving endpoint has consented to receive data. Connectivity checks that are paced as described in [Section 16 of \[RFC5245\]](#), and responses to connectivity checks are permitted. That is, no application data (e.g., RTP or Datagram Transport Layer Security (DTLS)), can be sent until consent is obtained.

Explicit consent to send is obtained and maintained by sending a STUN binding request to the remote peer's transport address and receiving a matching, authenticated, non-error STUN binding response from the remote peer's transport address. These STUN binding requests and responses are authenticated using the same short-term credentials as the initial ICE exchange.

Note: Although TCP has its own consent mechanism (TCP acknowledgements), consent is necessary over a TCP connection because it could be translated to a UDP connection (e.g., [\[RFC6062\]](#)).

Consent expires after 30 seconds. That is, if a valid STUN binding response has not been received from the remote peer's transport address in 30 seconds, the endpoint **MUST** cease transmission on that 5-tuple. STUN consent responses received after consent expiry do not re-establish consent and may be discarded or cause an ICMP error.

To prevent expiry of consent, a STUN binding request can be sent periodically. To prevent synchronization of consent checks, each interval **MUST** be randomized from between 0.8 and 1.2 times the basic period. Implementations **SHOULD** set a default interval of 5 seconds, resulting in a period between checks of 4 to 6 seconds. Implementations **MUST NOT** set the period between checks to less than 4 seconds. This timer is independent of the consent expiry timeout.

Each STUN binding request for consent **MUST** use a new STUN transaction identifier, as described in [Section 6 of \[RFC5389\]](#). Each STUN binding request for consent is transmitted once only. A sender therefore cannot assume that it will receive a response for every consent request, and a response might be for a previous request (rather than for the most recently sent request).

An endpoint SHOULD await a binding response for each request it sends for a time based on the estimated round-trip time (RTT) (see [Section 7.2.1 of \[RFC5389\]](#)) with an allowance for variation in network delay. The RTT value can be updated as described in [\[RFC5389\]](#). All outstanding STUN consent transactions for a candidate pair MUST be discarded when consent expires.

To meet the security needs of consent, an untrusted application (e.g., JavaScript or signaling servers) MUST NOT be able to obtain or control the STUN transaction identifier, because that enables spoofing of STUN responses, falsifying consent.

To prevent attacks on the peer during ICE restart, an endpoint that continues to send traffic on the previously validated candidate pair during ICE restart MUST continue to perform consent freshness on that candidate pair as described earlier.

While TCP affords some protection from off-path attackers ([\[RFC5961\]](#), [\[RFC4953\]](#)), there is still a risk an attacker could cause a TCP sender to send forever by spoofing ACKs. To prevent such an attack, consent checks MUST be performed over all transport connections, including TCP. In this way, an off-path attacker spoofing TCP segments cannot cause a TCP sender to send once the consent timer expires (30 seconds).

An endpoint does not need to maintain consent if it does not send application data. However, an endpoint MUST regain consent before it resumes sending application data. In the absence of any packets, any bindings in middleboxes for the flow might expire. Furthermore, having one peer unable to send is detrimental to many protocols. Absent better information about the network, if an endpoint needs to ensure its NAT or firewall mappings do not expire, this can be done using keepalive or other techniques (see [Section 10 of \[RFC5245\]](#) and see [\[RFC6263\]](#)).

After consent is lost, the same ICE credentials MUST NOT be used on the affected 5-tuple again. That means that a new session, or an ICE restart, is needed to obtain consent to send on the affected candidate pair.

## 5.2. Immediate Revocation of Consent

In some cases, it is useful to signal that consent is terminated rather than relying on a timeout.

Consent for sending application data is immediately revoked by receipt of an authenticated message that closes the connection (e.g., a Transport Layer Security (TLS) fatal alert) or receipt of a valid

and authenticated STUN response with error code Forbidden (403). Note however that consent revocation messages can be lost on the network, so an endpoint could resend these messages, or wait for consent to expire.

Receipt of an unauthenticated message that closes a connection (e.g., TCP FIN) does not indicate revocation of consent. Thus, an endpoint receiving an unauthenticated end-of-session message SHOULD continue sending media (over connectionless transport) or attempt to re-establish the connection (over connection-oriented transport) until consent expires or it receives an authenticated message revoking consent.

Note that an authenticated Secure Real-time Transport Control Protocol (SRTP) BYE does not terminate consent; it only indicates the associated SRTP source has quit.

## 6. DiffServ Treatment for Consent

It is RECOMMENDED that STUN consent checks use the same Diffserv Codepoint markings as the ICE connectivity checks described in [Section 7.1.2.4 of \[RFC5245\]](#) for a given 5-tuple.

Note: It is possible that different Diffserv Codepoints are used by different media over the same transport address [[WebRTC-QoS](#)]. Such a case is outside the scope of this document.

## 7. DTLS Applicability

The DTLS applicability is identical to what is described in [Section 4.2 of \[RFC7350\]](#).

## 8. Security Considerations

This document describes a security mechanism, details of which are mentioned in [Sections 4.1 and 4.2 of \[RFC7350\]](#). Consent requires 96 bits transaction ID defined in [Section 6 of \[RFC5389\]](#) to be uniformly and randomly chosen from the interval  $0 \dots 2^{96}-1$ , and be cryptographically strong. This is good enough security against an off-path attacker replaying old STUN consent responses. Consent Verification to avoid attacks using a browser as an attack platform against machines is discussed in [Sections 3.3 and 4.2 of \[WebRTC-SEC\]](#).

The security considerations discussed in [\[RFC5245\]](#) should also be taken into account.

## 9. References

### 9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", [RFC 5245](#), DOI 10.17487/RFC5245, April 2010, <<http://www.rfc-editor.org/info/rfc5245>>.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", [RFC 5389](#), DOI 10.17487/RFC5389, October 2008, <<http://www.rfc-editor.org/info/rfc5389>>.

### 9.2. Informative References

- [EKT] Mattsson, J., McGrew, D., and D. Wing, "Encrypted Key Transport for Secure RTP", Work in Progress, [draft-ietf-avtcore-srtp-ekt-03](#), October 2014.
- [RFC3830] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and K. Norrman, "MIKEY: Multimedia Internet KEYing", [RFC 3830](#), DOI 10.17487/RFC3830, August 2004, <<http://www.rfc-editor.org/info/rfc3830>>.
- [RFC4568] Andreassen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", [RFC 4568](#), DOI 10.17487/RFC4568, July 2006, <<http://www.rfc-editor.org/info/rfc4568>>.
- [RFC4953] Touch, J., "Defending TCP Against Spoofing Attacks", [RFC 4953](#), DOI 10.17487/RFC4953, July 2007, <<http://www.rfc-editor.org/info/rfc4953>>.
- [RFC5961] Ramaiah, A., Stewart, R., and M. Dalal, "Improving TCP's Robustness to Blind In-Window Attacks", [RFC 5961](#), DOI 10.17487/RFC5961, August 2010, <<http://www.rfc-editor.org/info/rfc5961>>.



- [RFC6062] Perreault, S., Ed. and J. Rosenberg, "Traversal Using Relays around NAT (TURN) Extensions for TCP Allocations", [RFC 6062](#), DOI 10.17487/RFC6062, November 2010, <<http://www.rfc-editor.org/info/rfc6062>>.
- [RFC6263] Marjou, X. and A. Sollaud, "Application Mechanism for Keeping Alive the NAT Mappings Associated with RTP / RTP Control Protocol (RTCP) Flows", [RFC 6263](#), DOI 10.17487/RFC6263, June 2011, <<http://www.rfc-editor.org/info/rfc6263>>.
- [RFC7350] Petit-Huguenin, M. and G. Salgueiro, "Datagram Transport Layer Security (DTLS) as Transport for Session Traversal Utilities for NAT (STUN)", [RFC 7350](#), DOI 10.17487/RFC7350, August 2014, <<http://www.rfc-editor.org/info/rfc7350>>.
- [WebRTC-QoS]  
Dhesikan, S., Jennings, C., Druta, D., Jones, P., and J. Polk, "DSCP and other packet markings for RTCWeb QoS", Work in Progress, [draft-ietf-tsvwg-rtcweb-qos-04](#), July 2015.
- [WebRTC-SA]  
Rescorla, E., "[WebRTC Security Architecture](#)", Work in Progress, [draft-ietf-rtcweb-security-arch-11](#), March 2015.
- [WebRTC-SEC]  
Rescorla, E., "[Security Considerations for WebRTC](#)", Work in Progress, [draft-ietf-rtcweb-security-08](#), February 2015.

#### Acknowledgements

Thanks to Eric Rescorla, Harald Alvestrand, Bernard Aboba, Magnus Westerlund, Cullen Jennings, Christer Holmberg, Simon Perreault, Paul Kyzivat, Emil Ivov, Jonathan Lennox, Inaki Baz Castillo, Rajmohan Banavi, Christian Groves, Meral Shirazipour, David Black, Barry Leiba, Ben Campbell, and Stephen Farrell for their valuable inputs and comments. Thanks to Christer Holmberg for doing a thorough review.

## Authors' Addresses

Muthu Arul Mozhi Perumal  
Ericsson  
Ferns Icon  
Doddanekundi, Mahadevapura  
Bangalore, Karnataka 560037  
India  
Email: muthu.arul@gmail.com

Dan Wing  
Cisco Systems, Inc.  
170 West Tasman Drive  
San Jose, California 95134  
United States  
Email: dwing@cisco.com

Ram Mohan Ravindranath  
Cisco Systems  
Cessna Business Park  
Sarjapur-Marathahalli Outer Ring Road  
Bangalore, Karnataka 560103  
India  
Email: rmohanr@cisco.com

Tirumaleswar Reddy  
Cisco Systems  
Cessna Business Park, Varthur Hobli  
Sarjapur Marathalli Outer Ring Road  
Bangalore, Karnataka 560103  
India  
Email: tiredddy@cisco.com

Martin Thomson  
Mozilla  
650 Castro Street, Suite 300  
Mountain View, California 94041  
United States  
Email: martin.thomson@gmail.com