

TRAM Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

A. Johnston
Avaya
J. Uberti
Google
J. Yoakum
K. Singh
Avaya
October 19, 2015

An Origin Attribute for the STUN Protocol
draft-ietf-tram-stun-origin-06

Abstract

STUN, or Session Traversal Utilities for NAT, is a protocol used to assist other protocols traverse Network Address Translators or NATs. This specification defines an ORIGIN attribute for STUN that can be used in similar ways to the HTTP header field of the same name. WebRTC browsers utilizing STUN and TURN would include this attribute which would provide servers with additional information about the STUN and TURN requests they receive. This specification defines the usage of the STUN ORIGIN attribute for web, SIP, and XMPP contexts.

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 April 21, 2016.

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
1.1. Requirements Language	4
2. STUN ORIGIN attribute	4
2.1. Origin Matching Rules	5
2.2. STUN Usage	6
2.3. TURN Usage	6
2.4. NAT Behavior Discovery Usage	6
2.5. ICE Usage	6
2.6. Media Keep-Alive Usage	6
2.7. SIP Keep-Alive Usage	6
2.8. Multiple Origins	7
3. IANA Considerations	7
4. Security Considerations	7
5. Implementation Status	8
6. Acknowledgements	10
7. References	10
7.1. Normative References	10
7.2. Informative References	11
Authors' Addresses	12

1. Introduction

STUN, or Session Traversal Utilities for NAT [RFC5389], is a protocol used to assist other protocols traverse Network Address Translators or NATs. TURN, or Traversal Using Relays around NAT [RFC5766], is a STUN extension that allows endpoints to acquire a relayed address for media flows. It is most commonly used in conjunction with ICE, Interactive Connectivity Establishment [RFC5245], which is used to establish peer-to-peer flows between endpoints through NATs and firewalls.

STUN defines three authentication modes, depending on the STUN usage. For STUN binding requests sent between peers, such as for ICE connectivity checks, a short term authentication method is recommended. Each peer contributes random strings which are exchanged over signaling and used to authenticate the connectivity checks. For TURN, a usage of STUN used to acquire and refresh relay addresses, a long term authentication method is recommended. This

authentication is similar to SIP Digest [RFC3261], which involves an authentication challenge for each request. A server, upon receipt of a TURN request, generates an authentication challenge that includes a realm and nonce. The client resends the TURN request supplying a user name and password based on the realm indicated by the server. For a STUN binding request sent to a STUN server, no authentication is recommended, as generating the response is less work for a server than the server utilizing the short term or long term authentication approach. The server responds statelessly, so the only resources used are those to process each request, which are minimal.

WebRTC, Web Real-Time Communications, adds peer-to-peer real-time, interactive voice and video media capabilities and data channels to browsers [I-D.ietf-rtcweb-overview] without a plugin or download, and allows web developers to access this functionality using JavaScript API calls [WebRTC-API]. WebRTC includes STUN, TURN, and ICE client functionality built into browsers. For a session established between two browsers, if either browser is behind a NAT, a STUN server is necessary. Public STUN servers are currently available and a web application can suggest a particular STUN server be used. In cases where a particular combination of NAT mapping and filtering and/or firewalling is present, a TURN server is needed to establish a peer connection. In this case, TURN credentials need to be available to the browser for the long term authentication approach. A TURN server for WebRTC might serve a number of different domains and realms.

From the perspective of the web application provider, providing service for a number of different domains and realms, it is useful to know something about the source of the STUN request when processing the request. For a web application provider STUN or TURN server, the server will have no idea which web pages or sites are sending binding requests to the service. In conventional applications, the SOFTWARE attribute would provide some identifying information to the service, but that no longer works when the browser is the application. For a web application provider TURN server, the TURN server does not know which realm to include in an authentication challenge.

In the web world, HTTP requests have the concept of origin. The origin of a web page, as defined in [RFC6454], is defined by the URI's scheme, host or IP address, and port portions. The HTTP Origin header field inserted by the web browser carries this information and is useful information for servers that receive HTTP requests generated via JavaScript. For example, Cross Origin Resource Sharing, CORS, allows an HTTP server to serve HTTP requests from multiple origins.

This specification proposes extending the origin concept to STUN requests. STUN requests generated by a web browser would include the

origin of the HTTP page that is initiating the Peer Connection. Using this extra information, a STUN server could use the origin to determine which STUN binding requests to respond to, reducing the load on a STUN server. Using this information, a TURN server could use the origin to determine which realm to include in the authentication challenge. A TURN server can also use the origin information for logging and analytics, and also as additional information after authentication for providing service. This specification also defines an origin for SIP and XMPP users of STUN and TURN.

An important use case that the STUN Origin helps solve is the operation of a multi-tenanted TURN server (i.e. a TURN server that serves multiple, perhaps tens of thousands of different domains). The problem associated with this use case is described in paragraph 6 of Section 4 of [RFC7376]. While it is possible for a TURN server to use the same authentication credentials across many domains, a more likely (and more manageable) scenario is to have separate credentials for each domain, and hence a different realm for each domain. With the TURN server configured with a mapping between a domain (conveyed in the Origin) and the realm string (to be used in the authentication challenge), a single TURN URI could be used across all domains, and the resulting JavaScript code would be portable.

Note that the origin information is most useful as a hint in initial STUN and TURN requests as received by a server. However, origin information still has value throughout the session even after authentication for logging and other purposes.

The following sections of this document define the STUN ORIGIN attribute and define its usage.

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 RFC 2119 [RFC2119].

2. STUN ORIGIN attribute

This specification defines how to apply the web origin concept and syntax of [RFC6454] to the STUN protocol.

This specification defines a new Attribute to the STUN protocol [RFC5389]. The attribute is called ORIGIN and uses the syntax defined in Section 15 of [RFC5389]. The number used for this in the type field is 0x802F, chosen in the comprehension optional range.

The value of ORIGIN is a variable-length value. It MUST contain a UTF-8 [RFC3629] encoded sequence of characters. Senders MAY include multiple ORIGIN attributes in a request, and receivers MUST support parsing and receiving multiple ORIGIN attributes. HTTP origins are less than 267 characters (the maximum 253 character domain name plus 8 characters for the URI scheme plus 5 characters for the port number).

For a web browser (HTTP User Agent), the contents of the ORIGIN attribute is the unicode-serialization of an origin defined in Section 6.1 of [RFC6454]. The origin value included is the same as the Origin header field for an HTTP request generated from the web page that is creating the Peer Connection. It does not include any string terminating (\x00) character in the serialization.

For a SIP User Agent [RFC3261] using STUN and TURN, the ORIGIN attribute is set to be the URI of the registrar server used by the User Agent (i.e. the Request-URI of a REGISTER method). This is the full Request-URI component of the SIP ABNF defined in [RFC3261]. For example, a SIP user "sip:bob@biloxi.example.com" might register with the Request-URI of "sip:registrar.biloxi.example.com".

For an XMPP client [RFC6120] using STUN and TURN, the ORIGIN attribute is an XMPP URI [RFC5122] representing the full domainpart of the client's Jabber ID (JID) as defined in the ABNF of [RFC6122]; for example, if the client's JID is "juliet@im.example.com/balcony" then the ORIGIN attribute would be "xmpp:im.example.com".

Other contexts can define a usage of the ORIGIN attribute to use an appropriate URI or URL.

If an ORIGIN attribute is not present in a request, it is up to the server how to handle the request. For example, it could assume a default Origin.

2.1. Origin Matching Rules

For any usage of ORIGIN, the following rules MUST be followed to ensure privacy. An ORIGIN attribute MUST NOT be included in a STUN or TURN request if the fully qualified domain name (FQDN) of the Origin does not match the FQDN of the STUN or TURN URI which was used to reach the STUN or TURN server. That is, only if the domain of the Origin is the same as the origin of the STUN or TURN server is the Origin information sent. The FQDN comparison does not include ports or URI schemes. For example, a web origin from `http://foo.example.com:8080` would match a TURN URI of `turn:foo.example.com`. A SIP origin of `sips:bar.example.org:5061` would match a STUN URI of `stun:bar.example.org:999`.

2.2. STUN Usage

For STUN requests sent without authentication to a STUN server (i.e. STUN binding requests sent to a STUN server), the STUN client **MUST** include the ORIGIN attribute if it matches per the rules of Section 2.1. Otherwise, the ORIGIN attribute is omitted.

A STUN server can derive additional information for logging and analytics about the request through the ORIGIN attribute, such as the source of the request. For example, an enterprise STUN server might only reply to STUN binding requests from certain domains.

2.3. TURN Usage

TURN [RFC5766] Allocate requests **MUST** include ORIGIN if it matches per the rules of Section 2.1. Otherwise, the ORIGIN attribute is omitted. For all other TURN requests, including the Send method, the use of the ORIGIN attribute is **NOT RECOMMENDED** as it provides no value.

A TURN server can use the ORIGIN attribute to determine which REALM to include in the authentication challenge. A TURN server can also use the ORIGIN attribute after authentication to provide appropriate service.

2.4. NAT Behavior Discovery Usage

For the NAT Behavior Discovery Usage in [RFC5780], the rules for STUN usage apply.

2.5. ICE Usage

ICE [RFC5245] connectivity checks **MUST NOT** include the ORIGIN attribute. Including the ORIGIN attribute leaks information about the site used by one peer to the other peer in the session.

2.6. Media Keep-Alive Usage

For media keep-alive STUN requests described in Section 20 of [RFC5245], the rules for ICE usage apply.

2.7. SIP Keep-Alive Usage

For SIP keep-alive STUN requests described in [RFC5626], the ORIGIN attribute **MUST NOT** be included. No valid use cases for the ORIGIN attribute have been identified to date, and as a result the ORIGIN attribute will be ignored.

2.8. Multiple Origins

Multiple Origins for HTTP Requests are described in Section 7.2 of [RFC6454]. Multiple origins can occur when the same resource is fetched by multiple origins at the same time (e.g. multiple tabs, windows, frames, etc.). In the context of WebRTC, it doesn't make sense for a STUN binding or TURN allocation to be shared across origins (e.g. Peer Connections). Based on their definitions, multiple SIP and XMPP origins also do not apply here. Therefore, if a STUN request contains multiple origins, the first origin MUST be used and the remaining origins ignored.

3. IANA Considerations

This specification, if approved, adds a new value to the IANA "STUN Attributes Registry" created by [RFC5389]. The ORIGIN attribute value is 0x802F.

4. Security Considerations

The security considerations of [RFC6454] apply to this extension. Servers using the information present in the STUN ORIGIN attribute need to realize that this attribute could be set arbitrarily by a non-browser client or modified by an intermediary. The method proposed in this document is not meant to replace existing STUN authentication mechanisms but to provide additional information to the server for logging and analytics and how to handle the request after authentication.

Just as browsers do not allow a web application to set the HTTP Origin header field via JavaScript, web browsers (HTTP User Agents) MUST NOT allow a web application to set the STUN ORIGIN attribute through JavaScript.

While the STUN MESSAGE-INTEGRITY attribute can provide integrity protection for all attributes present in a STUN request, MESSAGE-INTEGRITY is not present in the initial STUN message sent. As a result, an ORIGIN attribute could be modified or removed from a STUN request without the server knowing. DTLS or TLS transport SHOULD be used when integrity protection for the ORIGIN attribute is important.

The STUN ORIGIN attribute has privacy implications. As a result, TLS or DTLS transport SHOULD be used. If STUN requests containing the origin attribute are not encrypted with TLS or DTLS, an on-path attacker could determine this information by inspecting STUN messages between the STUN client and STUN server. This information is often available in other messages sent by the browser, such as DNS or HTTP requests. However, in cases where secure HTTP is used, including the

ORIGIN attribute over an unencrypted transport could leak this information. In cases where privacy is paramount, but TLS or DTLS transport is not available, the ORIGIN attribute MAY be omitted.

The STUN ORIGIN attribute also has privacy implications in that the origin information is shared with a STUN or TURN server which otherwise might not know this information. This information could be used to track usage of real-time communication services. A STUN or TURN server will always know the public IP address of each user, but the ORIGIN attribute provides more information about which service or provider is being used. The particular STUN and TURN servers used are usually selected by the real-time communications service provider (i.e. the web provider for WebRTC or the SIP or XMPP service provider). In addition, they are usually also run by the same provider, or by a trusted partner, especially for TURN. However, a service or provider using an untrusted third party STUN or TURN server needs to recognize that the operator of the third party STUN or TURN server will learn the identity of the service or provider through this extension. In order to prevent the leakage of usage of information, the Origin and STUN or TURN URI matching rules defined in Section 2.1 MUST be used. These rules ensure that the inclusion of the ORIGIN attribute does not provide any new information to the operator of the STUN or TURN server than they already know.

5. Implementation Status

Note to RFC Editor: Please remove this entire section prior to publication, including the reference to RFC 6982.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC6982]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC6982], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

Two proof-of-concept implementations have been created in support of this proposed standard. One provides a WebRTC enabled browser that includes the appropriate STUN ORIGIN Attribute with the Origin insight known to the browser in STUN/TURN messages sent to servers. The other provides an example of a multiple realms capable TURN server that takes advantage of Origin insight provided in the STUN ORIGIN Attribute.

A Chrome browser implementation has been created by Graham Yoakum and Ryan Yoakum (Skobalt LLC) and is freely licensed under the standard terms of the open source Chromium and WebRTC projects. This proof-of-concept version of the Google Chrome browser (nicknamed 'Chromeo') sends Origin insight in STUN and TURN messages using the proposed new STUN ORIGIN attribute with a value of 0x802F (as initially proposed, however that value is easily changed in a single line of code). 'Chromeo' includes a Chrome flag to enable and disable this unique feature (and is by default disabled to prevent any non-intentional use of this feature until the standard is finalized). This implementation is based on is draft-johnston-tram-stun-origin-02.

Coordinated changes to both the WebRTC and Chromium open source projects have been formally submitted for consideration. The two submitted change lists together implement the complete browser proof-of-concept. 'Chromeo' has been built for Linux and STUN protocol behavior has been verified using WireShark traces illustrating that proper STUN Origin attributes are being included in STUN/TURN messages sent by the browser to servers (screen captures of STUN messages illustrating the Origin attribute and content are available).

The WebRTC and Chromium open source projects can be found at:
<http://www.webrtc.org/> and <http://www.chromium.org/>

Google can choose to accept or modify the changes proposed for Chrome and other browser vendors can access and take advantage of the publicly available WebRTC and Chromium open source submissions as desired. Hopefully this will enable browsers to quickly implement STUN Origin enhancements.

A multiple realms capable advanced open source Origin enabled TURN server (named 'Coturn') has been created by Oleg Moskalenko and is freely licensed under the New BSD license. This reference implementation and proof-of-concept provides a clone (a spin-off) of the rfc5766-turn-server project adding Origin-based multiple realms support.

'Coturn' is backward-compatible with rfc5766-turn-server project but the code is more complex and it uses a different (also more complex)

database structure. It is the intent to add all IETF TRAM TURN server related capabilities to this project as they mature. 'Coturn' is publicly available and can be found at: <https://github.com/coturn/coturn/>

6. Acknowledgements

Thanks to John Selbie, Tirumaleswar Reddy, Simon Perreault, Marc Petit-Huguenin, Andy Hutton, and Oleg Moskalkenko for their feedback and reviews. Special thanks to Graham Yoakum and Ryan Yoakum of Skobalt LLC and Oleg Moskalkenko of rfc5766-turn-server project for contributing open source proof-of-concept implementations for a Chrome web browser and a multiple realms capable TURN server, quickly demonstrating feasibility.

7. References

7.1. 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>>.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <<http://www.rfc-editor.org/info/rfc3261>>.
- [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>>.
- [RFC5122] Saint-Andre, P., "Internationalized Resource Identifiers (IRIs) and Uniform Resource Identifiers (URIs) for the Extensible Messaging and Presence Protocol (XMPP)", RFC 5122, DOI 10.17487/RFC5122, February 2008, <<http://www.rfc-editor.org/info/rfc5122>>.
- [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>>.
- [RFC5626] Jennings, C., Ed., Mahy, R., Ed., and F. Audet, Ed., "Managing Client-Initiated Connections in the Session Initiation Protocol (SIP)", RFC 5626, DOI 10.17487/RFC5626, October 2009, <<http://www.rfc-editor.org/info/rfc5626>>.
- [RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, DOI 10.17487/RFC5766, April 2010, <<http://www.rfc-editor.org/info/rfc5766>>.
- [RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, DOI 10.17487/RFC6120, March 2011, <<http://www.rfc-editor.org/info/rfc6120>>.
- [RFC6122] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Address Format", RFC 6122, DOI 10.17487/RFC6122, March 2011, <<http://www.rfc-editor.org/info/rfc6122>>.
- [RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <<http://www.rfc-editor.org/info/rfc6454>>.
- [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>>.

7.2. Informative References

- [I-D.ietf-rtcweb-overview] Alvestrand, H., "Overview: Real Time Protocols for Browser-based Applications", draft-ietf-rtcweb-overview-14 (work in progress), June 2015.
- [RFC5780] MacDonald, D. and B. Lowekamp, "NAT Behavior Discovery Using Session Traversal Utilities for NAT (STUN)", RFC 5780, DOI 10.17487/RFC5780, May 2010, <<http://www.rfc-editor.org/info/rfc5780>>.

- [RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, DOI 10.17487/RFC6982, July 2013, <<http://www.rfc-editor.org/info/rfc6982>>.
- [RFC7376] Reddy, T., Ravindranath, R., Perumal, M., and A. Yegin, "Problems with Session Traversal Utilities for NAT (STUN) Long-Term Authentication for Traversal Using Relays around NAT (TURN)", RFC 7376, DOI 10.17487/RFC7376, September 2014, <<http://www.rfc-editor.org/info/rfc7376>>.
- [WebRTC-API] Bergkvist, A., Burnett, D., Jennings, C., and A. Narayanan, "WebRTC 1.0: Real-time Communication Between Browsers", W3C Working Draft <http://www.w3.org/TR/webrtc/>, 2013, <<http://www.w3.org/TR/2013/WD-webrtc-20130910/>>.

Authors' Addresses

Alan Johnston
Avaya
St. Louis, MO
USA

Email: alan.b.johnston@gmail.com

Justin Uberti
Google
Kirkland, WA
USA

Email: justin@uberti.name

John Yoakum
Avaya
Cary, NC
USA

Email: yoakum@avaya.com

Kundan Singh
Avaya
San Francisco, CA
USA

Email: kundan10@gmail.com

TRAM
Internet-Draft
Intended status: Standards Track
Expires: April 10, 2016

P. Patil
T. Reddy
D. Wing
Cisco
October 8, 2015

TURN Server Auto Discovery
draft-ietf-tram-turn-server-discovery-05

Abstract

Current Traversal Using Relays around NAT (TURN) server discovery mechanisms are relatively static and limited to explicit configuration. These are usually under the administrative control of the application or TURN service provider, and not the enterprise, ISP, or the network in which the client is located. Enterprises and ISPs wishing to provide their own TURN servers need auto discovery mechanisms that a TURN client could use with no or minimal configuration. This document describes three such mechanisms for TURN server discovery.

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 April 10, 2016.

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. Terminology	3
3. Discovery Procedure	3
4. Discovery using Service Resolution	4
4.1. Retrieving Domain Name	4
4.1.1. DHCP	5
4.1.2. From own Identity	5
4.2. Resolution	5
5. DNS Service Discovery	6
5.1. mDNS	7
6. Discovery using Anycast	8
7. Deployment Considerations	8
7.1. Mobility and Changing IP addresses	9
8. IANA Considerations	9
8.1. Anycast	9
9. Security Considerations	9
9.1. Service Resolution	10
9.2. DNS Service Discovery	10
9.3. Anycast	11
10. Acknowledgements	11
11. References	11
11.1. Normative References	11
11.2. Informative References	12
Appendix A. Change History	13
A.1. Change from draft-patil-tram-serv-disc-00 to -01	14
A.2. Change from draft-ietf-tram-turn-server-discovery-01 to 02	14
Authors' Addresses	14

1. Introduction

TURN [RFC5766] is a protocol that is often used to improve the connectivity of Peer-to-Peer (P2P) applications (as defined in section 2.7 of [RFC5128]). TURN allows a connection to be established when one or both sides are incapable of a direct P2P connection. It is an important building block for interactive, real-time communication using audio, video, collaboration etc.

While TURN services are extensively used today, the means to auto discover TURN servers do not exist. TURN clients are usually

explicitly configured with a well known TURN server. To allow TURN applications to operate seamlessly across different types of networks and encourage the use of TURN without the need for manual configuration, it is important that there exists an auto discovery mechanism for TURN services. Web Real-Time Communication (WebRTC) [I-D.ietf-rtcweb-overview] usages and related extensions, which are mostly based on web applications, need this immediately.

This document describes three discovery mechanisms, so as to maximize opportunity for discovery, based on the network in which the TURN client finds itself. The three discovery mechanisms are:

- o A resolution mechanism based on straightforward Naming Authority Pointer (S-NAPTR) resource records in the Domain Name System (DNS). [RFC5928] describes details on retrieving a list of server transport addresses from DNS that can be used to create a TURN allocation.
- o DNS Service Discovery
- o A mechanism based on anycast address for TURN.

In general, if a client wishes to communicate using one of its interfaces using a specific IP address family, it SHOULD query the TURN server(s) that has been discovered for that specific interface and address family. How to select an interface and IP address family, is out of the scope of this document.

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. Discovery Procedure

A TURN client that implements the auto discovery algorithm uses the following mechanisms for discovery:

1. Local Configuration : Local or manual TURN configuration i.e., TURN servers configured at the system level. For example, in case of Web Real-Time Communication (WebRTC) [I-D.ietf-rtcweb-overview] usages and related extensions, which are based on web applications, a Java Script specified TURN server MUST be considered as local configuration. An implementation MAY give the user an opportunity (e.g., by means of configuration file options or menu items) to specify a TURN server for each address family.

2. Service Resolution : The TURN client attempts to perform TURN service resolution using the host's DNS domain.
3. DNS SD: DNS Service Discovery.
4. Anycast : Send TURN allocate request to the assigned TURN anycast request for each combination of interface and address family.

Not all TURN servers may be discovered using NAPTR records or DNS SD; Similarly, not all TURN servers may support anycast. For best results, a client SHOULD implement all discovery mechanisms described above.

The document does not prescribe a strict order that a client must follow for discovery. An implementation may choose to perform all the above steps in parallel for discovery OR choose to follow any desired order and stop the discovery procedure if a mechanism succeeds.

On hosts with more than one interface or address family (IPv4/v6), the TURN server discovery procedure has to be performed for each combination of interface and address family. A client MAY optionally choose to perform the discovery procedure only for a desired interface/address combination if the client does not wish to discover a TURN server for all combinations of interface and address family.

4. Discovery using Service Resolution

This mechanism is performed in two steps:

1. A DNS domain name is retrieved for each combination of interface and address family.
2. Retrieved DNS domain names are then used for S-NAPTR lookups as per [RFC5928]. Further DNS lookups may be necessary to determine TURN server IP address(es).

4.1. Retrieving Domain Name

A client has to determine the domain in which it is located. The following sections provide two possible mechanisms to learn the domain name, but other means of retrieving domain names may be used, which are outside the scope of this document e.g. local configuration.

Implementations may allow the user to specify a default name that is used if no specific name has been configured.

4.1.1. DHCP

DHCP can be used to determine the domain name related to an interface's point of network attachment. Network operators may provide the domain name to be used for service discovery within an access network using DHCP. Sections 3.2 and 3.3 of [RFC5986] define DHCP IPv4 and IPv6 access network domain name options to identify a domain name that is suitable for service discovery within the access network. [RFC2132] defines the DHCP IPv4 domain name option; While this option is less suitable, it may still be useful if the options defined in [RFC5986] are not available.

For IPv6, the TURN server discovery procedure MUST try to retrieve DHCP option 57 (OPTION_V6_ACCESS_DOMAIN). If no such option can be retrieved, the procedure fails for this interface. For IPv4, the TURN server discovery procedure MUST try to retrieve DHCP option 213 (OPTION_V4_ACCESS_DOMAIN). If no such option can be retrieved, the procedure SHOULD try to retrieve option 15 (Domain Name). If neither option can be retrieved the procedure fails for this interface. If a result can be retrieved it will be used as an input for S-NAPTR resolution.

4.1.2. From own Identity

For a TURN client with an understanding of the protocol mechanics of calling applications, the client may wish to extract the domain name from its own identity i.e canonical identifier used to reach the user.

Example

```
SIP    : 'sip:alice@example.com'
JID    : 'alice@example.com'
email  : 'alice@example.com'
```

'example.com' is retrieved from the above examples.

The means to extract the domain name may be different based on the type of identifier and is outside the scope of this document.

4.2. Resolution

Once the TURN discovery procedure has retrieved domain names, the resolution mechanism described in [RFC5928] is followed. An S-NAPTR lookup with 'RELAY' application service and the desired protocol tag is made to obtain information necessary to connect to the authoritative TURN server within the given domain.

In the example below, for domain 'example.net', the resolution algorithm will result in IP address, port, and protocol tuples as follows:

```
example.net.
  IN NAPTR 100 10 "" RELAY:turn.udp "" example.net.

example.net.
  IN NAPTR 100 10 S RELAY:turn.udp "" _turn._udp.example.net.

_turn._udp.example.net.
  IN SRV    0 0 3478 a.example.net.

a.example.net.
  IN A      192.0.2.1
```

Order	Protocol	IP address	Port
1	UDP	192.0.2.1	3478

If no TURN-specific S-NAPTR records can be retrieved, the discovery procedure fails for this domain name (and the corresponding interface and IP protocol version). If more domain names are known, the discovery procedure may perform the corresponding S-NAPTR lookups immediately. However, before retrying a lookup that has failed, a client MUST wait a time period that is appropriate for the encountered error (NXDOMAIN, timeout, etc.).

5. DNS Service Discovery

DNS-based Service Discovery (DNS-SD) [RFC6763] and Multicast DNS (mDNS) [RFC6762] provide generic solutions for discovering services available in a local network. DNS-SD/ mDNS define a set of naming rules for certain DNS record types that they use for advertising and discovering services. PTR records are used to enumerate service instances of a given service type. A service instance name is mapped to a host name and a port number using a SRV record. If a service instance has more information to advertise than the host name and port number contained in its SRV record, the additional information is carried in a TXT record.

Section 4.1 of [RFC6763] specifies that a service instance name in DNS-SD has the following structure:

```
<Instance> . <Service> . <Domain>
```

The <Domain> portion specifies the DNS sub-domain where the service instance is registered. It may be "local.", indicating the mDNS local domain, or it may be a conventional domain name such as "example.com.". The <Service> portion of the TURN service instance name MUST be "_turnserver._udp", "_turnserver._tcp".

The <Instance> portion is a DNS label, containing UTF-8-encoded text [RFC5198], limited to 63 octets in length. It is meant to be a user-friendly description of the service instance, suitable for a menu-like user interface display. Thus it can contain any characters including spaces, punctuation, and non-Latin characters as long as they can be encoded in UTF-8.

For example, TURN server advertises the following DNS records :

```
_turnserver._udp.local. PTR example.com._turnserver._udp.local.
```

```
example.com._turnserver._udp.local. SRV 0 0 5030 example-turn-  
server.local.
```

```
example-turn-server.local. A 192.168.1.2
```

In addition to the service instance name, IP address and the port number, DNS-SD provides a way to publish other information pertinent to the service being advertised. The additional data can be stored as name/value attributes in a TXT record with the same name as the SRV record for the service. Each name/value pair within the TXT record is preceded by a single length byte, thereby limiting the length of the pair to 255 bytes (See Section 6 of [RFC6763] and Section 3.3.14 of [RFC1035] for details).

5.1. mDNS

A TURN client tries to discover the TURN servers being advertised in the site by multicasting a PTR query "_turnserver._udp.local." or "_turnserver._tcp.local" or the TURN server can send out gratuitous multicast DNS answer packets whenever it starts up, wakes from sleep, or detects a change in network configuration. TURN clients receive these gratuitous packet and cache the information contained in it.

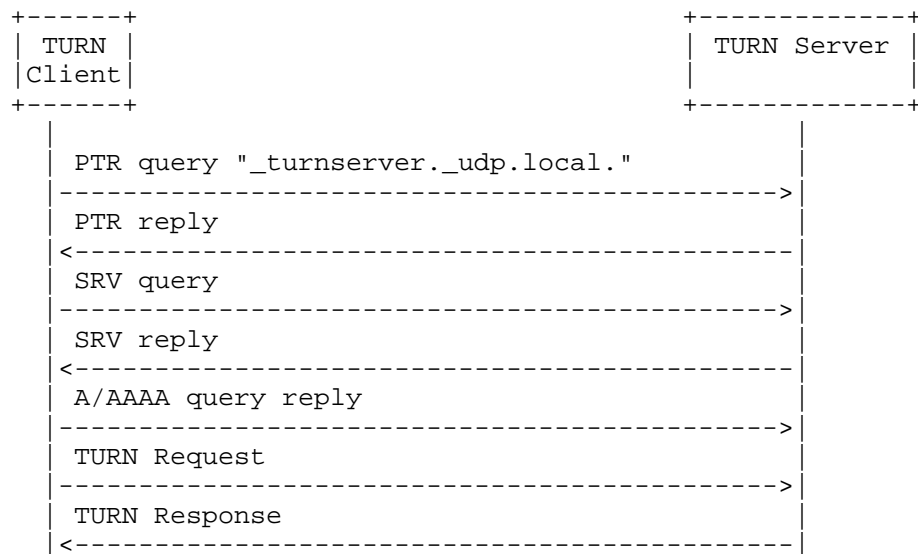


Figure 1: TURN Server Discovery using mDNS

6. Discovery using Anycast

IP anycast can also be used for TURN service discovery. A packet sent to an anycast address is delivered to the "topologically nearest" network interface with the anycast address. Using the TURN anycast address, the only two things that need to be deployed in the network are the two things that actually use TURN.

When a client requires TURN services, it sends a TURN allocate request to the assigned anycast address. The TURN anycast server responds with a 300 (Try Alternate) error as described in [RFC5766]; The response contains the TURN unicast address in the ALTERNATE-SERVER attribute. For subsequent communication with the TURN server, the client uses the responding server's unicast address. This has to be done because two packets addressed to an anycast address may reach two different anycast servers. The client, thus, also needs to ensure that the initial request fits in a single packet. An implementation may choose to send out every new request to the anycast address to learn the closest TURN server each time.

7. Deployment Considerations

7.1. Mobility and Changing IP addresses

A change of IP address on an interface may invalidate the result of the TURN server discovery procedure. For instance, if the IP address assigned to a mobile host changes due to host mobility, it may be required to re-run the TURN server discovery procedure without relying on earlier gained information. New requests should be made to the newly learned TURN servers learned after TURN discovery re-run. However, if an earlier learned TURN server is still accessible using the new IP address, procedures described for mobility using TURN defined in [I-D.wing-tram-turn-mobility] can be used for ongoing streams.

8. IANA Considerations

8.1. Anycast

IANA should allocate an IPv4 and an IPv6 well-known TURN anycast address. 192.0.0.0/24 and 2001:0000::/48 are reserved for IETF Protocol Assignments, as listed at

<http://www.iana.org/assignments/iana-ipv4-special-registry/> and

<http://www.iana.org/assignments/iana-ipv6-special-registry/>

9. Security Considerations

Clients can use TURN servers provided by the local network or by the access network without authenticating with the TURN server. It is recommended that clients use (D)TLS with network provided TURN servers to validate the TURN server and prevent man-in-middle attacks. A TURN client may use the following techniques to validate a TURN server:

- o For certificate-based authentication, a pre-populated trust anchor store [RFC6024] allows a TURN client to perform path validation for the server certificate obtained during the (D)TLS handshake. If the client used a domain name to discover the TURN server, that domain name also provides a mechanism for validation of the TURN server. The client MUST use the rules and guidelines given in section 6 of [RFC6125] to validate the TURN server identity.
- o For TURN servers that don't have a certificate trust chain (e.g., because they are on a home network or a corporate network), a configured list of TURN servers can contain the Subject Public Key Info (SPKI) fingerprint of the TURN servers. The public key is used for the same reasons HTTP pinning [RFC7469] uses the public key.

- o Raw public key-based authentication, as defined in [RFC7250], could also be used to authenticate a TURN server.
- o For opportunistic privacy, analogous to SMTP opportunistic encryption [RFC7435] one does not require privacy, but one desires privacy when possible. With opportunistic privacy, a client might learn of a TLS-enabled TURN server from an untrusted source and may not be able to validate the TLS certificate. These choices maximize availability and performance, but they leave the client vulnerable to on-path attacks that remove privacy. Opportunistic privacy can be used by any current client, but it only provides guaranteed privacy when there are no on-path active attackers.

9.1. Service Resolution

The primary attack against the methods described in this document is one that would lead to impersonation of a TURN server. An attacker could attempt to compromise the S-NAPTR resolution. Security considerations described in [RFC5928] are applicable here as well.

In addition to considerations related to S-NAPTR, it is important to recognize that the output of this is entirely dependent on its input. An attacker who can control the domain name can also control the final result. Because more than one method can be used to determine the domain name, a host implementation needs to consider attacks against each of the methods that are used.

If DHCP is used, the integrity of DHCP options is limited by the security of the channel over which they are provided. Physical security and separation of DHCP messages from other packets are commonplace methods that can reduce the possibility of attack within an access network; alternatively, DHCP authentication [RFC3188] can provide a degree of protection against modification. When using DHCP discovery, clients are encouraged to use unicast DHCP INFORM queries instead of broadcast queries which are more easily spoofed in insecure networks.

9.2. DNS Service Discovery

Since DNS-SD is just a specification for how to name and use records in the existing DNS system, it has no specific additional security requirements over and above those that already apply to DNS queries and DNS updates. For DNS queries, DNS Security Extensions (DNSSEC) [RFC4033] should be used where the authenticity of information is important. For DNS updates, secure updates [RFC2136][RFC3007] should generally be used to control which clients have permission to update DNS records.

For mDNS, in addition to what has been described above, a principal security threat is a security threat inherent to IP multicast routing and any application that runs on it. A rogue system can advertise that it is a TURN server. Discovery of such rogue systems as TURN servers, in itself, is not a security threat if there is a means for the TURN client to authenticate and authorize the discovered TURN servers.

9.3. Anycast

In a network without any TURN server that is aware of the TURN anycast address, outgoing TURN requests could leak out onto the external Internet, possibly revealing information.

Using an IANA-assigned well-known TURN anycast address enables border gateways to block such outgoing packets. In the default-free zone, routers should be configured to drop such packets. Such configuration can occur naturally via BGP messages advertising that no route exists to said address.

Sensitive clients that do not wish to leak information about their presence can set an IP TTL on their TURN requests that limits how far they can travel into the public Internet.

10. Acknowledgements

The authors would like to thank Simon Perrault, Paul Kyzivat, Troy Shields, Eduardo Gueiros, Ted Hardie, Bernard Aboba and Karl Stahl for their review and valuable comments. Thanks to Adam Roach for his detailed review and suggesting DNS Service Discovery as an additional discovery mechanism.

11. References

11.1. Normative References

- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<http://www.rfc-editor.org/info/rfc1035>>.
- [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>>.
- [RFC2132] Alexander, S. and R. Droms, "DHCP Options and BOOTP Vendor Extensions", RFC 2132, DOI 10.17487/RFC2132, March 1997, <<http://www.rfc-editor.org/info/rfc2132>>.

- [RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <<http://www.rfc-editor.org/info/rfc2136>>.
- [RFC3007] Wellington, B., "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, DOI 10.17487/RFC3007, November 2000, <<http://www.rfc-editor.org/info/rfc3007>>.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <<http://www.rfc-editor.org/info/rfc4033>>.
- [RFC5198] Klensin, J. and M. Padlipsky, "Unicode Format for Network Interchange", RFC 5198, DOI 10.17487/RFC5198, March 2008, <<http://www.rfc-editor.org/info/rfc5198>>.
- [RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, DOI 10.17487/RFC5766, April 2010, <<http://www.rfc-editor.org/info/rfc5766>>.
- [RFC5928] Petit-Huguenin, M., "Traversal Using Relays around NAT (TURN) Resolution Mechanism", RFC 5928, DOI 10.17487/RFC5928, August 2010, <<http://www.rfc-editor.org/info/rfc5928>>.
- [RFC5986] Thomson, M. and J. Winterbottom, "Discovering the Local Location Information Server (LIS)", RFC 5986, DOI 10.17487/RFC5986, September 2010, <<http://www.rfc-editor.org/info/rfc5986>>.
- [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <<http://www.rfc-editor.org/info/rfc6762>>.
- [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <<http://www.rfc-editor.org/info/rfc6763>>.

11.2. Informative References

- [I-D.ietf-rtcweb-overview] Alvestrand, H., "Overview: Real Time Protocols for Browser-based Applications", draft-ietf-rtcweb-overview-14 (work in progress), June 2015.

- [I-D.wing-tram-turn-mobility]
Wing, D., Patil, P., Reddy, T., and P. Martinsen,
"Mobility with TURN", draft-wing-tram-turn-mobility-03
(work in progress), May 2015.
- [RFC3188] Hakala, J., "Using National Bibliography Numbers as
Uniform Resource Names", RFC 3188, DOI 10.17487/RFC3188,
October 2001, <<http://www.rfc-editor.org/info/rfc3188>>.
- [RFC5128] Srisuresh, P., Ford, B., and D. Kegel, "State of Peer-to-
Peer (P2P) Communication across Network Address
Translators (NATs)", RFC 5128, DOI 10.17487/RFC5128, March
2008, <<http://www.rfc-editor.org/info/rfc5128>>.
- [RFC6024] Reddy, R. and C. Wallace, "Trust Anchor Management
Requirements", RFC 6024, DOI 10.17487/RFC6024, October
2010, <<http://www.rfc-editor.org/info/rfc6024>>.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and
Verification of Domain-Based Application Service Identity
within Internet Public Key Infrastructure Using X.509
(PKIX) Certificates in the Context of Transport Layer
Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March
2011, <<http://www.rfc-editor.org/info/rfc6125>>.
- [RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J.,
Weiler, S., and T. Kivinen, "Using Raw Public Keys in
Transport Layer Security (TLS) and Datagram Transport
Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250,
June 2014, <<http://www.rfc-editor.org/info/rfc7250>>.
- [RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection
Most of the Time", RFC 7435, DOI 10.17487/RFC7435,
December 2014, <<http://www.rfc-editor.org/info/rfc7435>>.
- [RFC7469] Evans, C., Palmer, C., and R. Sleevi, "Public Key Pinning
Extension for HTTP", RFC 7469, DOI 10.17487/RFC7469, April
2015, <<http://www.rfc-editor.org/info/rfc7469>>.

Appendix A. Change History

[Note to RFC Editor: Please remove this section prior to
publication.]

A.1. Change from draft-patil-tram-serv-disc-00 to -01

- o Added IP address (Section 4.1.2) and Own identity (4.1.3) as new means to obtain domain names
- o New Section 4.2.1 SOA (inspired by draft-kist-alto-3pdisc)
- o 300 (Try Alternate) response for Anycast

A.2. Change from draft-ietf-tram-turn-server-discovery-01 to 02

- o Removed sections that describe reverse IP lookup
- o Added DNS Service Discovery as an additional discovery mechanism

Authors' Addresses

Prashanth Patil
Cisco Systems, Inc.
Bangalore
India

Email: praspati@cisco.com

Tirumaleswar Reddy
Cisco Systems, Inc.
Cessna Business Park, Varthur Hobli
Sarjapur Marathalli Outer Ring Road
Bangalore, Karnataka 560103
India

Email: tiredy@cisco.com

Dan Wing
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, California 95134
USA

Email: dwing@cisco.com

TRAM
Internet-Draft
Intended status: Standards Track
Expires: March 19, 2016

T. Reddy
S. Nandakumar
D. Wing
Cisco
B. Williams
Akamai, Inc.
September 16, 2015

Metadata discovery for third party authorized TURN session
draft-reddy-tram-token-metadata-01

Abstract

The operator of the TURN server might want to have fine grained control on the clients usage of the server resources for providing features such as limiting the bandwidth usage, number of allocations and so on. This document proposes a generic mechanism for the operator to introspect the access token to retrieve any policy restrictions imposed by the authorization server on the TURN server resources assigned to the client.

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 March 19, 2016.

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. Terminology	3
3. Introspection Request	3
4. Introspection Response	4
5. INTROSPECTION_ENDPOINT Attribute	5
6. Notifications from Introspection Endpoint	5
7. Example usage with WebRTC	6
8. Alternate Approach	8
9. Security Considerations	9
10. IANA Considerations	9
10.1. JSON Web Token Claims	9
10.2. Well-Known 'introspection' URI	10
10.3. STUN attribute	10
11. Acknowledgements	10
12. References	10
12.1. Normative References	10
12.2. Informative References	11
Authors' Addresses	11

1. Introduction

The TURN protocol [RFC5766] is used to setup a relay service (via a TURN Server) to exchange traffic (real time media, data) between peers when direct peer-to-peer connection is not otherwise possible. Due to the costs associated with operating a relay service, it is important to constrain resource usage. For example, the operator might want to limit the number of allocations or bandwidth.

[RFC7635] allows clients to obtain OAuth2.0 access token (of type 'Assertion') authorized by a Authorization Server to access a given TURN server. On receiving such a token, the TURN server validates the token to grant or reject access to the session resources. However, having a token doesn't provide any control for the operator of the TURN server restrict the server's resources. This specification proposes using the mechanism defined in [I-D.ietf-oauth-introspection] to query OAuth2.0 authorization server to determine resource restrictions for this token.

The rest of the document is organized as follows. Section 3 provides procedure for querying the OAuth2.0 Introspection Endpoint and

Section 4 shows the introspection response with the parameters identifying the policy controls associated with the access token.

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].

This document defines the following terms:

Access Token: OAuth 2.0 access token.

Token Introspection: The act of inquiring about the current state of an OAuth 2.0 token through use of the network protocol defined in this document.

Introspection Endpoint: The OAuth 2.0 endpoint through which the token introspection operation is accomplished. The Introspection Endpoint could be a WebRTC server.

3. Introspection Request

For introspecting the meta-information associated with the access token, the TURN server shall execute the procedures defined in Section 2.1 of [I-D.ietf-oauth-introspection].

```
POST {scheme}://{host}:{port}/.well-known/introspection
Accept: application/json
Content-Type: application/x-www-form-urlencoded
```

```
{
  "token" : "string"
  "token_type_hint" : "string"
}
```

token REQUIRED. This parameter is defined in [I-D.ietf-oauth-introspection]. The access token is conveyed by the TURN client to the TURN server as discussed in Section 3.1 of [RFC7635].

token_type_hint OPTIONAL. This parameter is defined in [I-D.ietf-oauth-introspection]. The token type MUST be set to 'access_token' defined in [RFC7009]. If the token type is not 'access_token', the server rejects the request with a 400 (Bad Request) error.

Following is a non-normative example request showcasing the introspection request for a given access token.

```
POST /introspect HTTP/1.1
Host: server.example.com
Accept: application/json
Content-Type: application/x-www-form-urlencoded

{
  "token" : "2YotnFZFEjrlzCsicMWpAA"
  "token_type_hint" : "access_token"
}
```

4. Introspection Response

The OAuth2.0 Introspection Endpoint on recognizing the token, responds with a JSON object [RFC7159] in "application/json" format with the following members.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "active" : "boolean",
  "scope" : "string",
  "max_upstream_bandwidth" : "unsigned integer",
  "max_downstream_bandwidth" : "unsigned integer",
  "max_allocations" : "unsigned integer",
  "lifetime" : "unsigned integer",
}
```

active REQUIRED. This parameter is defined in [I-D.ietf-oauth-introspection].

scope OPTIONAL. This parameter is defined in [I-D.ietf-oauth-introspection]. For this specification, the scope MUST be 'stun'.

max_upstream_bandwidth REQUIRED. The value of this parameter is an 64 bit unsigned integer that represents the maximum upstream bandwidth permitted for the token in kilobits per second (1 kilobit = 1024 bits).

max_downstream bandwidth REQUIRED. The value of this parameter is an 64 bit unsigned integer that represents the maximum

downstream bandwidth permitted for the token in kilobits per second (1 kilobit = 1024 bits).

max_allocations: REQUIRED. 16 bit unsigned integer defining maximum number of allocations that is allowable for the given access token.

lifetime: REQUIRED: The lifetime of the access token, in seconds.

NOTE: Future specifications are allowed to define further top-level members as mandated by the use-cases.

Following is a non-normative example response:

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
  "active" : true,
  "scope" : "stun",
  "upstream-bandwidth" : 4096,
  "downstream-bandwidth" : 4096,
  "max-allocations" : 1,
}
```

5. INTROSPECTION_ENDPOINT Attribute

This attribute is used by the TURN client to inform the TURN server the FQDN of Introspection Endpoint.

The TURN server establishes an HTTPS connection with the indicated server and sends the above-described communications to that server. The INTROSPECTION_ENDPOINT attribute is a comprehension-optional attribute (see Section 15 from [RFC5389]).

TBD: An alternate approach is to convey the FQDN in the token itself.

6. Notifications from Introspection Endpoint

Introspection Endpoint can send unsolicited responses to notify updates to the metadata associated with the token to the TURN server using HTTP/2 server push mechanism. Examples where such notifications are desired are:

- o The Introspection Endpoint can signal the TURN server to revoke the access token after the call is terminated by setting lifetime to zero.

- o When the call switches from audio to video, the Introspection Endpoint notifies the increased bandwidth to the TURN server.

7. Example usage with WebRTC

Below diagram shows a flow where a WebRTC client uses the procedures discussed in [RFC7635] to obtain a OAuth 2.0 access token from the WebRTC server. The TURN Server queries the Introspection Endpoint to determine the metadata associated with the token. Steps 7, 8 and 9 are done using the procedures mentioned in this document.

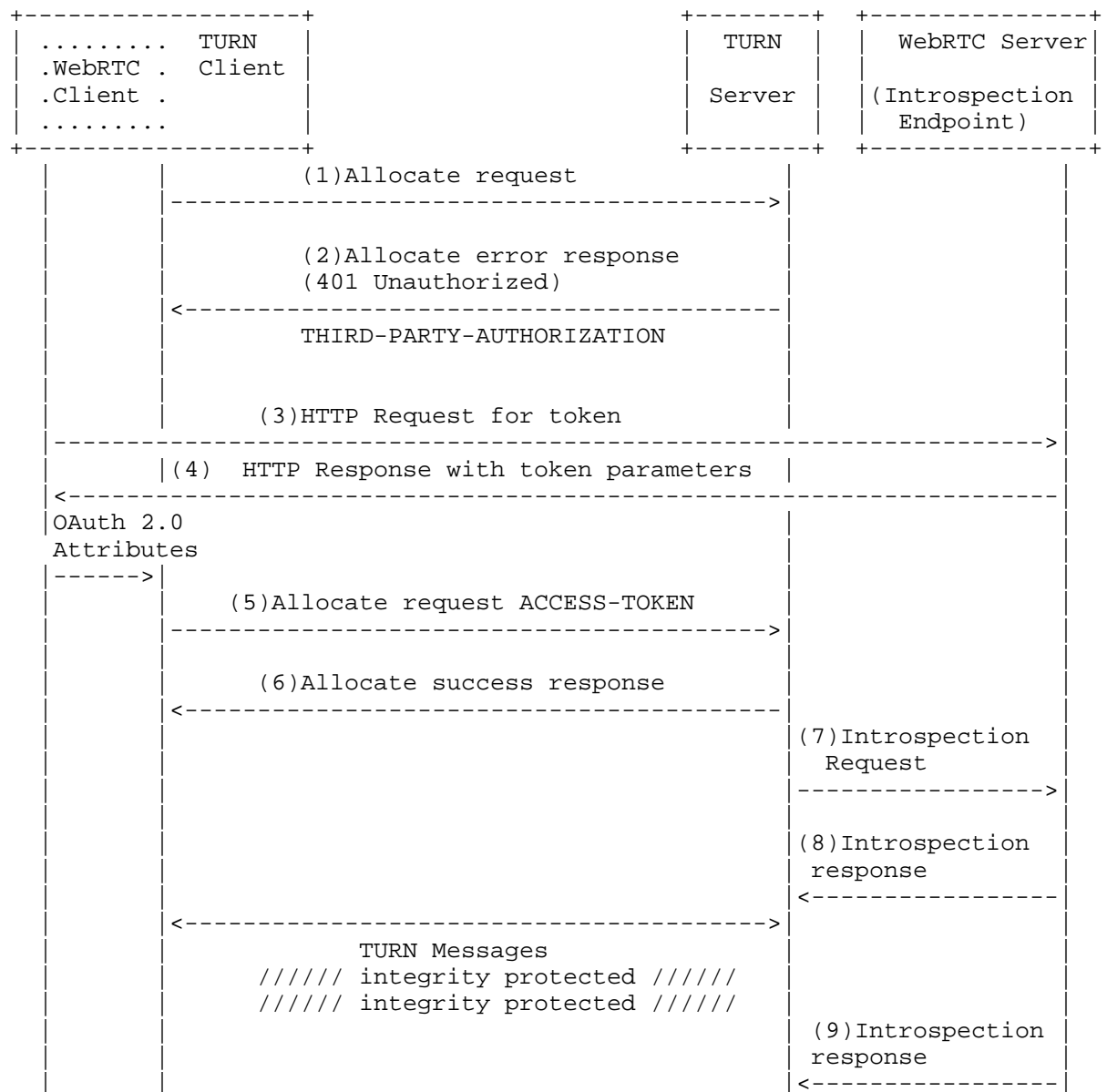


Figure 1: Metadata discovery for TURN session

8. Alternate Approach

An alternate approach considered by the authors makes use of the access token itself to deliver metadata related to the TURN authentication request. Standard STUN TLV encoded attributes are used to communicate additional metadata associated with the token. Such attributes can be used to define the maximum bandwidth utilization allowed for allocations associated with the token, the maximum number of distinct concurrent allocations, etc.

To include STUN attributes within the body of the access token, the authorization server simply appends them to the access token's plaintext immediately after the lifetime field. The variable length list of attributes **MUST** consume all of the additional plaintext data space within the body of the access token. No explicit option length value is required or provided.

In order for inclusion of attributes within the plaintext to work correctly in the absence of an explicit length field, one of two things must be true: either the receiver must be able to reliably determine the correct content length from the output of the decryption operation, or the receiver must be able to reliably differentiate between padding bytes and data bytes within the token. AES-128-GCM is an example of the first case, and AEAD mode AES-CBC-HMAC-SHA256 is an example of the second case.

Before parsing the optional data within the access token, the TURN server **MUST** first perform all token validation required by [RFC7635]. If any of the specified validation checks fail, the TURN server **MUST NOT** attempt to parse optional attributes.

To interpret the optional attributes within an access token, the TURN server first calculates the amount of option space included in the plaintext by subtracting the size of the base payload data (14 bytes + key_length) from the total payload size. It then interprets the data in the option space as STUN TLV formatted attributes. While parsing the option space, the TURN server **MUST** apply the same validations to the access token's attributes that it would have applied if the attributes had been included in the outer STUN header (e.g. Verify the data format and value types). If any such validation checks fail, the TURN server **MUST** reject the STUN request with an error response 401 (Unauthorized).

The following STUN attributes are defined by this document for inclusion in the access token: TBD. Additional attributes may be defined for this purpose in future specifications.

The primary benefits of this method of metadata distribution are:

- o It does not impose additional requirements on the Introspection Endpoint for out of band communication with the TURN server.
- o Communicating with the Introspection Endpoint may increase the latency associated with TURN allocation request handling.

The primary shortcomings of this method of metadata distribution are:

- o Needs a larger TURN packet to accommodate the token. For example, inclusion of the four fields defined above (max_upstream_bandwidth, max_downstream_bandwidth, max_allocations, and lifetime) would increase the token size by around 38 bytes, depending upon whether the AEAD algorithm requires padding.
- o The Introspection Endpoint cannot notify the TURN server of changes to the metadata associated with the token.

[NOTE: Backward compatibility with [RFC7635] requires discussion, but it should not be a major issue if the dynamic option space calculation method described is considered acceptable.]

[NOTE: The authors are seeking feedback from the working group on the relative merits of this approach versus the "Introspection Endpoint" approach. Which should we attempt to move forward? Or does each one have enough merit that we should try to advance both?]

9. Security Considerations

The Security Considerations and Privacy Considerations of [I-D.ietf-oauth-introspection] apply to this document.

10. IANA Considerations

10.1. JSON Web Token Claims

This specification requests IANA to register the following values into the IANA JSON Web Token Claims registry for JWT Claim Names.

- o Claim Name: "max_upstream_bandwidth"
- o Claim Description: Maximum limit of upstream bandwidth
- o Change Controller: IESG
- o Specification Document(s): Section 4 of [[this document]].
- o Claim Name: "max_downstream_bandwidth"
- o Claim Description: Maximum limit of downstream bandwidth
- o Change Controller: IESG
- o Specification Document(s): Section 4 of [[this document]].

- o Claim Name: "max_allocations"
- o Claim Description: Maximum number of allocations
- o Change Controller: IESG
- o Specification Document(s): Section 4 of [[this document]].

10.2. Well-Known 'introspection' URI

This memo registers the 'introspection' well-known URI in the Well-Known URIs registry as defined by [RFC5785].

URI suffix: introspection

Change controller: IETF

Specification document(s): This document

Related information: None

10.3. STUN attribute

[Paragraphs below in braces should be removed by the RFC Editor upon publication]

[IANA is requested to add the following attributes to the STUN attribute registry [iana-stun], the INTROSPECTION_ENDPOINT attribute requires that IANA allocate a value in the "STUN attributes Registry" from the comprehension-optional range (0x8000-0xBFFF)].

This document defines the INTROSPECTION_ENDPOINT attribute, described in Section 5. IANA has allocated the comprehension-optional codepoint TBD for this attribute.

11. Acknowledgements

Authors would like to thank Ram Mohan for comments and review.

12. References

12.1. Normative References

[I-D.ietf-oauth-introspection]

Richer, J., "OAuth 2.0 Token Introspection", draft-ietf-oauth-introspection-11 (work in progress), July 2015.

[iana-stun]

IANA, , "IANA: STUN Attributes", April 2011, <<http://www.iana.org/assignments/stun-parameters/stun-parameters.xml>>.

- [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>>.
- [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>>.
- [RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, DOI 10.17487/RFC5766, April 2010, <<http://www.rfc-editor.org/info/rfc5766>>.
- [RFC7635] Reddy, T., Patil, P., Ravindranath, R., and J. Uberti, "Session Traversal Utilities for NAT (STUN) Extension for Third-Party Authorization", RFC 7635, DOI 10.17487/RFC7635, August 2015, <<http://www.rfc-editor.org/info/rfc7635>>.

12.2. Informative References

- [RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <<http://www.rfc-editor.org/info/rfc5785>>.
- [RFC7009] Lodderstedt, T., Ed., Dronia, S., and M. Scurtescu, "OAuth 2.0 Token Revocation", RFC 7009, DOI 10.17487/RFC7009, August 2013, <<http://www.rfc-editor.org/info/rfc7009>>.
- [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>>.

Authors' Addresses

Tirumaleswar Reddy
Cisco Systems, Inc.

Email: tiredy@cisco.com

Suhas Nandakumar
Cisco Systems, Inc.

Email: snandaku@cisco.com

Dan Wing
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, California 95134
USA

Email: dwing@cisco.com

Brandon Williams
Akamai, Inc.
8 Cambridge Center
Cambridge, MA 02142
USA

Email: brandon.williams@akamai.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 12, 2014

B. Williams
Akamai
T. Reddy
Cisco
June 10, 2014

Peer-specific Redirection for Traversal Using Relays around NAT (TURN)
draft-williams-peer-redirect-01

Abstract

This specification describes a peer-specific redirection method that allows the TURN server to redirect a client for the purpose of improving communication with a specific peer without negatively affecting communication with other peers.

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 December 12, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

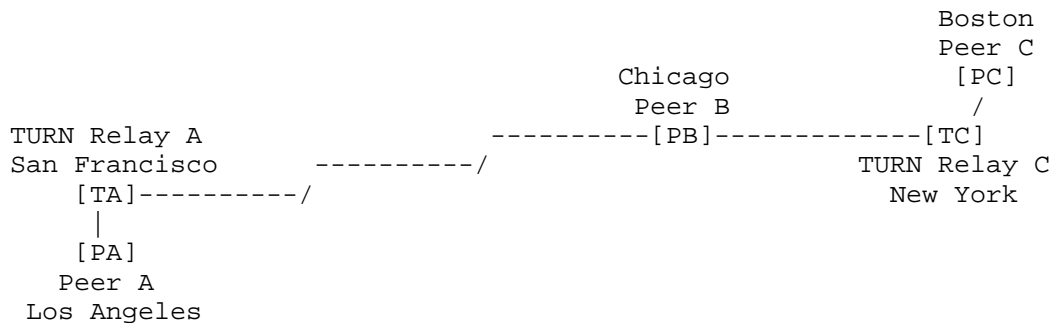
This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this 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	3
2. Terminology	4
3. Peer-specific Server Redirect Mechanism	4
3.1. Attribute Usage	4
3.2. Sending a CreatePermission or ChannelBind Request	6
3.2.1. The CHECK-ALTERNATE Attribute	6
3.2.2. The XOR-OTHER-ADDRESS attribute	7
3.3. Receiving a CreatePermission or ChannelBind Request	7
3.4. Receiving a CreatePermission or ChannelBind Error Response	8
3.5. Receiving a CreatePermission or ChannelBind Success Response	8
4. Security Considerations	9
4.1. CHECK-ALTERNATE Flood	9
4.2. Unsolicited or Invalid ALTERNATE-SERVER	10
5. IANA Considerations	11
6. References	11
6.1. Normative References	11
6.2. Informative References	11
Authors' Addresses	12

1. Introduction

A Traversal Using Relay around NAT (TURN) [RFC5766] service provider may provide multiple candidate TURN servers for use by a host, but it is not possible to determine which candidate TURN server will provide the best performance until both peers have been identified. In addition, the best TURN server to use for one peer may be different than the best TURN server to use for another peer. For optimum relay performance, it is desirable to select the TURN server based on the peer to which data is to be relayed. Consider the following example:



When Peer B wishes to communicate with either Peer A or Peer C, it performs a DNS lookup and discovers TURN Relay C, the nearest of the candidate TURN servers. Peer B then sends a TURN Allocate request to TURN Relay C to determine the reflexive and relay candidates to offer. After the reflexive candidate has been chosen, Peer B sends a ChannelBind request to TURN Relay C to establish a channel for communication with the peer. If Peer C is the remote peer, the existing allocation will perform reasonably well, but if Peer A is the remote peer, the latency for relayed packets will be nearly twice as long as if TURN Relay A had been selected as the relay candidate. The problem is worse if Peer B wishes to communicate with both Peer A and Peer C, since there is no single relay candidate that would provide optimum performance for both Peer A and Peer C.

If TURN Relay C and TURN Relay A are part of a common TURN service, it would be possible for TURN Relay C to determine that TURN Relay A will provide optimal service for communication between Peer B and Peer A. This allows the TURN service to redirect just the data channel between Peer A and Peer B to TURN relay A, thus providing optimal performance for both relay channels.

The Session Traversal Utilities for NAT (STUN) protocol [RFC5389] defines an ALTERNATE-SERVER mechanism with which a server can redirect a client to another server by replying to a request message with an error response with error code 300 (Try Alternate). The TURN

protocol describes error code 300 as one of the possible error codes for an Allocate error response.

This specification describes an additional use of the ALTERNATE-SERVER STUN attribute for TURN that allows the TURN server to redirect a client for the purpose of improving communication with a specific peer without negatively affecting communication with other peers. The client application indicates the nature of the desired response, which allows the client to treat the alternate server selection as either a requirement or a suggestion. This flexibility gives the client the option to choose the best way for the Interactive Connectivity Establishment (ICE) protocol [RFC5245] to respond (e.g. discarding the existing relay candidate for communication with this peer versus evaluating the two candidate servers using ICE connectivity checks and selecting the best one).

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. Peer-specific Server Redirect Mechanism

This specification describes two new uses of the existing STUN ALTERNATE-SERVER attribute. In the first case, the ALTERNATE-SERVER attribute is included with either a CreatePermission error response or a ChannelBind error response. In the second case, the ALTERNATE-SERVER attribute is included with either a CreatePermission success response or a ChannelBind success response.

This specification also defines two new comprehension-optional STUN attributes: CHECK-ALTERNATE and XOR-OTHER-ADDRESS. The CHECK-ALTERNATE attribute is used by the client to request that the server perform peer-specific redirection. The XOR-OTHER-ADDRESS is used by the client to provide an alternate peer address for location identification in the event that the XOR-PEER-ADDRESS attribute in the CreatePermission or ChannelBind request is not expected to reliably serve this purpose.

3.1. Attribute Usage

When sending a CreatePermission or a ChannelBind request, the CHECK-ALTERNATE STUN attribute allows a TURN client to indicate support for peer-specific server redirection. To maintain backward compatibility with [RFC5766] compliant TURN servers that do not support peer-

specific redirection, this attribute is defined as comprehension-optional, which allows a TURN server that does not support peer-specific redirection to ignore the attribute. To maintain backward compatibility with [RFC5766] compliant TURN clients that do not support peer-specific redirection, a TURN server only sends the ALTERNATE-SERVER attribute in CreatePermission and ChannelBind responses when the CHECK-ALTERNATE STUN attribute was present in the request. This prevents transmission of the ALTERNATE-SERVER attribute in cases where the receiving client might not consider the usage legitimate.

The CHECK-ALTERNATE STUN attribute's value indicates the expected server response type: error or success. This capability to declare the expected response type allows TURN client implementers greater flexibility during session establishment. For example, a TURN client implementer may wish to maintain the smallest number of permissions possible during session establishment in order keep the internal client implementation simple, in which case an error response would be desirable. On the other hand, a TURN client implementer may wish to optimize for faster session establishment by continuing to use a sub-optimal allocation while setting up the new one, in which case a success response would be desirable. This second case could be achieved with an error response if the client were to send a second request without the CHECK-ALTERNATE attribute, but such an approach would require an extra RTT.

The XOR-OTHER-ADDRESS STUN attribute allows the TURN client to provide an alternate peer address that can be used by the server to identify the network geographic location of the peer when performing the peer-specific redirection check. Use of this attribute is only necessary if the XOR-PEER-ADDRESS already contained in the CreatePermission or ChannelBind request does not adequately serve this purpose, which should only be true when both peers require a TURN relay for end-to-end data flow. In this case, the TURN CreatePermission or ChannelBind request will provide the peer's TURN relay address as the XOR-PEER-ADDRESS value. If the RTT between the peer and its TURN relay server is very small, the TURN relay address might still be an appropriate address to use for the peer-specific redirection check. As the RTT grows, the TURN relay address will become less suitable for this purpose. For this reason, it is generally the case that the peer's public address (i.e. its host or reflexive address) is a better indication of its network geographic location than its TURN relay address.

Even in cases where both peers require a TURN relay, a typical ICE protocol implementation will give higher candidate priority to the peer's host and reflexive addresses, which means that the first CreatePermission or ChannelBind request will provide the peer's

public address as the XOR-PEER-ADDRESS value and no XOR-OTHER-ADDRESS attribute is necessary. However, although ICE recommends this priority, it does not require it, and so the first request may contain the peer's TURN relay address. With such an implementation, the XOR-OTHER-ADDRESS attribute allows the client to provide the peer's reflexive address in a request that populates the XOR-PEER-ADDRESS attribute with the peer's relay address.

3.2. Sending a CreatePermission or ChannelBind Request

A client that supports peer-specific server redirection and desires such redirection to be performed MUST include the CHECK-ALTERNATE attribute in the first CreatePermission or ChannelBind request when that request is expected to form a new permission or binding. A client MUST NOT include the CHECK-ALTERNATE attribute in a CreatePermission or ChannelBind request that is intended to extend the lifetime of an existing permission or binding.

Peer-specific server redirection is only supported for requests that include a single XOR-PEER-ADDRESS attribute. When forming a CreatePermission request with multiple XOR-PEER-ADDRESS attributes, the client MUST NOT include the CHECK-ALTERNATE attribute.

When the CreatePermission or ChannelBind request includes the CHECK-ALTERNATE attribute, the client MAY also include an XOR-OTHER-ADDRESS attribute with a value appropriate for the above described purpose. The XOR-OTHER-ADDRESS attribute SHOULD NOT be included in the request if its value will be identical to the request's XOR-PEER-ADDRESS attribute.

3.2.1. The CHECK-ALTERNATE Attribute

When forming a CHECK-ALTERNATE attribute, the STUN Type is TBD-CA. This type is in the comprehension-optional range, which means that STUN agents can safely ignore the attribute if they do not understand it.

The CHECK-ALTERNATE attribute takes a 1-byte Value, which means that the Length is 1 and 3 bytes of padding are required after the Value. The format of the Value is:

```

0
0 1 2 3 4 5 6 7
+---+---+---+---+---+
|E|      RFFU      |
+---+---+---+---+---+
```

The Value contains a single 1-bit flag:

E: If 1, the server is requested to send a Try Alternate (300) error response when redirection is expected. If 0, the server is request to include an ALTERNATE-SERVER attribute in the success response for the request.

The other 7 bits of the attribute's value must be set to zero on transmission and ignored on reception.

3.2.2. The XOR-OTHER-ADDRESS attribute

When forming an XOR-OTHER-ADDRESS attribute, the STUN Type is TBD-XOA. This type is in the comprehension-optional range, which means that STUN agents can safely ignore the attribute if they do not understand it.

The XOR-OTHER-ADDRESS value specifies an address and port suitable for identification of the peer's network geographic location. It is encoded in the same way as XOR-MAPPED-ADDRESS [RFC5389].

3.3. Receiving a CreatePermission or ChannelBind Request

When a server receives a CreatePermission or ChannelBind request that includes a CHECK-ALTERNATE attribute, it processes as per the TURN specification [RFC5766] plus the specific rules mentioned here.

The server checks the following:

- o If the CHECK-ALTERNATE attribute is not recognized, ignore the attribute because its type indicates that it is comprehension-optional. This should be the existing behavior.
- o If the message is a CreatePermission request with multiple XOR-PEER-ADDRESS attributes, ignore the CHECK-ALTERNATE attribute if present.
- o If peer-specific redirection is not supported by the server, ignore the attribute.
- o If the associated permission or binding already exists, ignore the attribute.

If none of the above causes the attribute to be ignored and no other cause for sending an error response has been found, the server attempts to identify an alternate server that will provide better performance for the session. When an XOR-OTHER-ADDRESS attribute is found in the request message, the server SHOULD use this address for peer location identification. Otherwise, the server SHOULD use the address provided in the XOR-PEER-ADDRESS attribute.

If no alternate server is identified, the server replies with a success response that does not include an ALTERNATE-SERVER attribute.

If an alternate server is identified and the client requested an error response for redirection, the server rejects the request with a 300 (Try Alternate) error. No new permission or binding is generated on the server in this case.

If an alternate server is identified and the client did not request an error response for redirection, the server creates the permission or binding. The server then replies to the request with a success response, including an ALTERNATE-SERVER attribute in the message.

3.4. Receiving a CreatePermission or ChannelBind Error Response

If the client receives a CreatePermission or ChannelBind error response with error code 420 (Unknown Attribute) and CHECK-ALTERNATE is listed in the UNKNOWN-ATTRIBUTE attribute of the message, the client SHOULD retransmit the original request without the CHECK-ALTERNATE attribute. This case is not expected due to the use of a comprehension-optional attribute type.

If the client receives a CreatePermission or ChannelBind error response with error code 300 (Try Alternate), the client SHOULD attempt to form an allocation to the TURN server indicated in the ALTERNATE-SERVER attribute.

If the alternate server responds to the Allocate request with a success response, the client SHOULD attempt to form a new permission or binding using the new allocation from the alternate server. The CreatePermission or ChannelBind request to the alternate server MAY include a CHECK-ALTERNATE attribute but SHOULD NOT request redirection via an error response. This helps to avoid the possibility of redirection loops.

If the alternate server responds to the Allocate request with an error response, the client MAY resend the original CreatePermission or ChannelBind request, either without the CHECK-ALTERNATE attribute or with a CHECK-ALTERNATE attribute that does not request an error response.

See Section 4 below for discussion of how the client should respond when receiving a Try Alternate error response that was not requested.

3.5. Receiving a CreatePermission or ChannelBind Success Response

If the client receives a CreatePermission or ChannelBind success response, it proceeds with processing according to the TURN

specification [RFC5766]. If the message does not include an ALTERNATE-SERVER attribute, no additional processing is required.

If the success response includes an ALTERNATE-SERVER attribute, the client SHOULD attempt to form an allocation to the TURN server indicated in the ALTERNATE-SERVER attribute.

If the alternate server responds to the Allocate request with a success response, the client SHOULD attempt to form a new permission or binding using the new allocation from the alternate server. The CreatePermission or ChannelBind request to the alternate server MAY include a CHECK-ALTERNATE attribute with either attribute value. If this is done, care should be taken in the client implementation to recognize and avoid redirection loops.

While waiting for the new allocation and permission or binding to form via the indicated alternate server, the client SHOULD use the original permission or binding from the request that included the CHECK-ALTERNATE attribute. In this way, peer-specific redirection without an error response can be considered a "hint" that allows the client to establish an alternate path and test its quality before switching to it.

See Section 4 below for discussion of how the client should respond when receiving an ALTERNATE-SERVER attribute that was not requested.

4. Security Considerations

This section considers attacks that are possible in a TURN deployment through the specified protocol extension, and discusses how they are mitigated by mechanisms in the protocol or recommended practices in the implementation.

The specified mechanism affects the use of TURN CreatePermission request messages, ChannelBind request messages, and their respective success and error response messages. Each of these TURN message types requires the MESSAGE-INTEGRITY STUN attribute, which limits attacks that attempt to make use of the specified mechanism to authenticated clients and servers.

4.1. CHECK-ALTERNATE Flood

A compromised TURN client could send a large number of CreatePermission or ChannelBind request messages, which would drive increased load on the TURN server. The CHECK-ALTERNATE attribute does not make such an attack more likely, though it could make it possible to increase the impact of such an attack due to the

additional load associated with determining whether an alternate server should be used by the client. The TURN server MAY be configured to ignore the CHECK-ALTERNATE attribute under some conditions in order to limit the associated load. The conditions under which it is appropriate for a TURN server to ignore the CHECK-ALTERNATE attribute are implementation dependent.

4.2. Unsolicited or Invalid ALTERNATE-SERVER

A compromised TURN server could send the "Try Alternate" error code in response to a request message that did not contain the CHECK-ALTERNATE attribute or where the value of the attribute did not request an error response. For client connectivity, this is no worse than any other error response code that could be sent. No matter what the error response code may be, the client is unable to relay data to the remote peer. The client MUST ignore the ALTERNATE-SERVER attribute in error responses when the CHECK-ALTERNATE attribute was not included in the associated request. The client SHOULD ignore the ALTERNATE-SERVER attribute in error responses when the CHECK-ALTERNATE attribute was included in the associated request if the attribute value did not request an error response. The client MAY discontinue use of the associated TURN allocation when an unsolicited Try Alternate error is received.

A compromised TURN server could send an ALTERNATE-SERVER attribute in a success response message for a request message that did not contain the CHECK-ALTERNATE attribute. The client MUST ignore the ALTERNATE-SERVER attribute in success responses when the CHECK-ALTERNATE attribute was not included in the associated request message. The client SHOULD ignore the ALTERNATE-SERVER attribute in success responses when the CHECK-ALTERNATE attribute was included in the associated request if the attribute value requested an error response. The client MAY discontinue use of the associated TURN allocation when an unsolicited ALTERNATE-SERVER attribute is received.

A compromised TURN server could send an invalid ALTERNATE-SERVER attribute value in either an error or a success response message, where the value refers to an unaffiliated TURN server to which the sending TURN server is not allowed to redirect traffic. Such an attack is already allowed by the use of Try Alternate errors in response to Allocate request messages. Use of the ALTERNATE-SERVER attribute in the context of peer-specific redirection does not make such an attack more likely, though it could make it possible to increase the scale of such an attack by allowing multiple ALTERNATE-SERVER attributes to each client, one per requested permission or binding. A client SHOULD ignore all future ALTERNATE-SERVER attributes received from the TURN server after an authentication

failure with any server identified via an ALTERNATE-SERVER attribute. A client MAY discontinue use of the associated TURN allocation after an authentication failure with any server identified via an ALTERNATE-SERVER attribute.

5. IANA Considerations

[Paragraphs below in braces should be removed by the RFC Editor upon publication]

[The CHECK-ALTERNATE attribute requires that IANA allocate a value in the "STUN attributes Registry" from the comprehension-optional range (0x8000-0xFFFF), to be replaced for TBD-CA throughout this document]

This document defines the CHECK-ALTERNATE STUN attribute, described in Section 3.2.1. IANA has allocated the comprehension-optional codepoint TBD-CA for this attribute.

[The XOR-OTHER-ADDRESS attribute requires that IANA allocate a value in the "STUN attributes Registry" from the comprehension-optional range (0x8000-0xFFFF), to be replaced for TBD-XOA throughout this document]

This document defines the XOR-OTHER-ADDRESS STUN attribute, described in Section 3.2.2. IANA has allocated the comprehension-optional codepoint TBD-XOA for this attribute.

6. References

6.1. Normative References

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

6.2. Informative References

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

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

[RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using

Relays around NAT (TURN): Relay Extensions to Session
Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.

Authors' Addresses

Brandon Williams
Akamai, Inc.
8 Cambridge Center
Cambridge, MA 02142
USA

Email: brandon.williams@akamai.com

Tirumaleswar Reddy
Cisco Systems, Inc.
Cessna Business Park, Varthur Hobli
Sarjapur Marathalli Outer Ring Road
Bangalore, Karnataka 560103
India

Email: tiredy@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 16, 2016

B. Williams
Akamai
J. Uberti
Google
October 14, 2015

Ufrag Permissions for Traversal Using Relays around NAT (TURN)
draft-williams-tram-ufrag-permission-00

Abstract

When using a TURN relay, ICE connectivity checks require an explicit permission or channel binding to be established for each peer address to be checked. This requires the answerer to send its candidate addresses to the offerer via the rendezvous server, which can impose a latency penalty when the rendezvous server is centrally located. This document defines a new type of TURN permission that will allow any ICE connectivity check message that contains the offerer's ufrag value to be accepted on a relay address for delivery over the associated TURN tunnel.

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 April 16, 2016.

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.

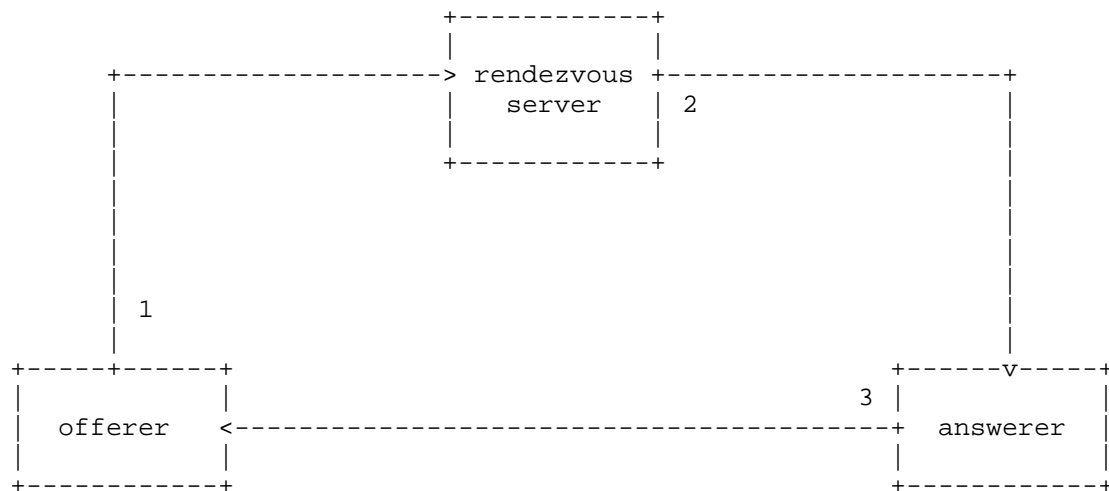
Table of Contents

1. Introduction	2
2. Terminology	4
3. Ufrag Permission Usage	5
3.1. Forming a CreatePermission Request	6
3.2. Processing a CreatePermission Request	6
3.3. Server Backward Compatibility	6
3.4. Processing a ChannelBind Request	7
3.5. Processing a Message on the Relay Transport Address	7
3.6. Processing a Data Indication	7
4. ICE Interactions	7
5. Security Considerations	8
6. IANA Considerations	8
7. Acknowledgements	8
8. Normative References	8
Authors' Addresses	9

1. Introduction

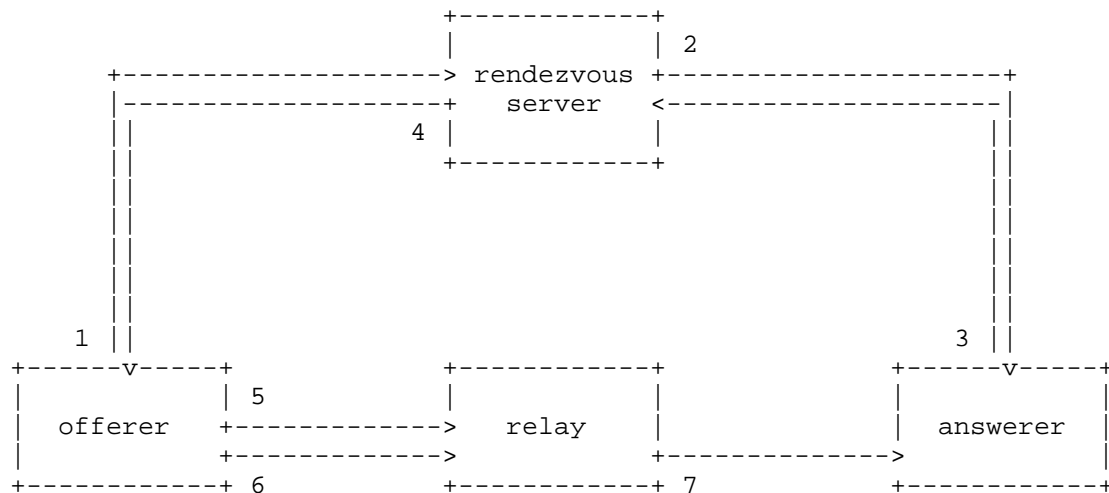
Interactive Connectivity Establishment (ICE) [RFC5245] provides a connectivity checking mechanism that peers can use to determine how to communicate directly with each other (e.g. which network layer protocol to use, which network address and transport port to use, etc.). The peers gather their sets of candidate addresses and exchange them via a rendezvous server using an offer/answer protocol. After gathering the addresses, the peers then send connectivity checks between address pairs to find a suitable local/remote address pair to use for communication.

Successful direct connectivity checks between the peers is the simplest scenario.



The offerer sends an offer with its candidate addresses to the rendezvous server (1), the rendezvous server forwards the offer to the answerer (2), and the answerer is able to send a connectivity check directly to the offerer (3) at the same time that it sends its answer back to the offerer via the rendezvous server.

Successful connectivity checks for a relayed candidate with Traversal Using Relays around NAT (TURN) [RFC5766] is more complicated and time consuming, partially due to the requirement for the local peer to explicitly notify the relay server about every permitted remote address.



In this case, the offerer first issues an allocation request to the relay server (not pictured) before sending an offer that includes the assigned relay address to the rendezvous server (1), which forwards the offer to the answerer (2). If the answerer sends a connectivity check to the relay address immediately, the relay will reject the message because there is no permission established for the answerer's address yet. Instead, the answerer must send its answer along with its candidate list to the rendezvous server (3), which relays the answer to the offerer (4). Only now can the offerer send a permission request to the relay (5) and then send a connectivity check message to the relay (6) to be forwarded to the answerer (7). Since the answer must be delivered before the necessary TURN permissions can be established, successful connectivity checks via the offerer's relay require an extra half round trip time via the rendezvous server as compared to direct host-to-host connectivity checks. This could be a significant penalty in the common case of a remotely located rendezvous server.

The latency penalty for the relay use case could be mitigated by permitting all ICE connectivity check messages to be delivered by the relay, regardless of whether there is an active permission for the sender. However, doing so would mean that use of the relay opens up the client to potential attacks from anywhere on the Internet. TURN permissions limit the risk by requiring the attacker to first discover an address associated with an active permission. This may be trivial to accomplish for an attacker who is on-path between the answerer and the relay, but would be more difficult for an off-path attacker.

When ICE is in use, the offer and answer messages each contain an ice-ufrag value, which is used in connectivity check messages as part of the username for Session Traversal Utilities for NAT (STUN) [RFC5389]. This document describes a new TURN permission type that allows any ICE connectivity check message to be relayed to TURN client if it has the expected remote ufrag (RFRAG) value in the STUN username attribute. This mechanism allows ICE connectivity checks to the offerer's relayed candidate to succeed without having to wait for the answer to arrive at the offerer, while at the same time continuing to require an attacker to learn some information about an active permission in order to construct packets that will be accepted by the relay for delivery to the client.

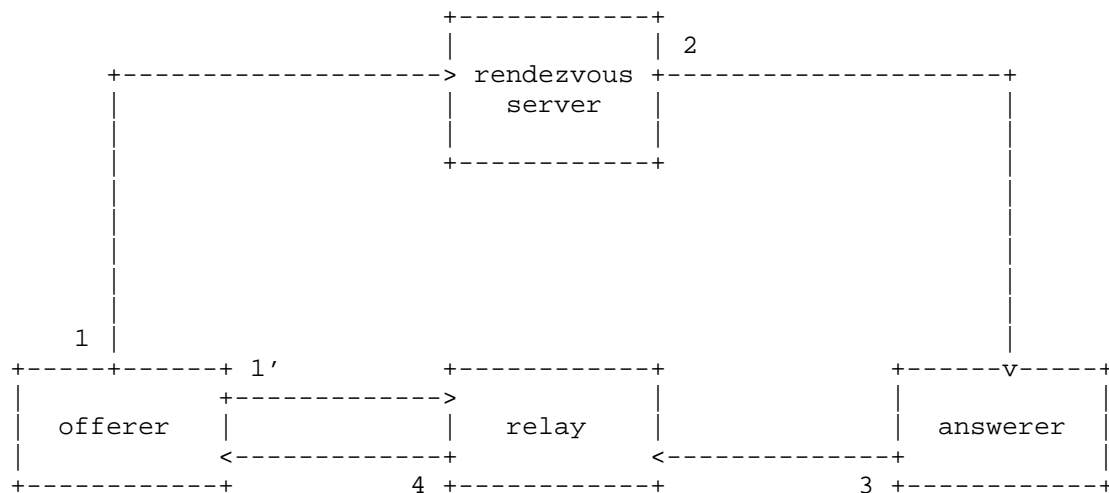
2. Terminology

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. Ufrag Permission Usage

To allow successful connectivity checks from the answerer, the offerer registers a new type of permission, known as a ufrag permission, with the relay server. Instead of using an XOR-PEER-ADDRESS attribute to identify the remote peer, a ufrag permission specifies the offerer's ufrag as the value for a LOCAL-UFRAG attribute. A ufrag permission allows any ICE connectivity check from a remote peer to be accepted by the relay if the RFRAG in that message matches the ufrag specified for the permission. Note that the LOCAL-UFRAG attribute is only allowed for TURN permission requests. ChannelBind requests cannot make use of this type of permission.

Message flow for successful connectivity checks using ufrag permissions looks fairly similar to the direct connectivity case where timing of the first successful check is concerned.



The offerer first establishes a TURN allocation with the relay (not pictured) to learn its relay candidate(s). At the point when the offerer sends the offer to the rendezvous server (1), it also sends a ufrag permission request to the relay (1'). The rendezvous server forwards the offer to the answerer (2), at which point the answerer can immediately send ICE connectivity checks (3) that can be accepted by the relay and forwarded to the offerer (4). Provided that the relay is fairly close to the offerer or at least inline between the offerer and the answerer, the primary difference in latency between direct and relay connectivity checks is the time required for candidate gathering (i.e. the allocation request).

3.1. Forming a CreatePermission Request

A ufrag permission request is formed in the same general way as a permission associated with an IP address, with the only exception being that it contains a LOCAL-UFRAG attribute to provide the ufrag value. As with any other CreatePermission request, multiple permissions may be established using a single CreatePermission request, meaning that a combination of one or more XOR-PEER-ADDRESS attributes and one or more LOCAL-UFRAG attributes may be present in a single request, with each resulting in a separate permission.

The LOCAL-UFRAG attribute is an understanding required attribute with the type TBD, and it contains a single value, which is the sender's ufrag value. This is allowed to be from 4 to 256 bytes in length.

NOTE: The authors considered two alternatives: providing the ufrag in either an XOR-PEER-ADDRESS or a USERNAME attribute. In both cases, it this change would modify the semantics for the attribute enough that it seemed better to defined a new attribute type.

3.2. Processing a CreatePermission Request

When the server receives a CreatePermission request, it processes it as per [RFC5766]. The rest of this section describes processing for cases where the request contains a LOCAL-UFRAG attribute.

If the server understands but does not support ufrag addresses, it rejects the request with a 403 (Forbidden) error.

If the request is valid, then the server installs or refreshes a permission for the ufrag contained in the LOCAL-UFRAG attribute. The server then responds with a CreatePermission success response.

NOTE: Careful consideration of the ufrag permission's lifetime is required. It needs to be long enough to be useful for its intended purpose but short enough to limit security exposure. A future revision of the draft will cover this in more detail.

3.3. Server Backward Compatibility

A server that does not recognize the LOCAL-UFRAG attribute will reject the request and send a 420 (Unknown Attribute) error response and otherwise ignore the request.

If the request sent by the client contained IP address XOR-PEER-ADDRESS attributes in addition to a LOCAL-UFRAG attribute, the client MAY resend the request without the LOCAL-UFRAG attribute in order to retry registration of the IP address permissions.

3.4. Processing a ChannelBind Request

A ChannelBind request received on the server MUST be considered invalid if it contains a LOCAL-UFRAG attribute. The server MUST reject such a request with a 403 (Forbidden) error.

3.5. Processing a Message on the Relay Transport Address

When a message is received on the relay transport address, the server first checks whether the allocation has a matching IP/IPv6 permission. If it does not have a matching IP/IPv6 permission but it does have one or more ufrag permissions, the server examines the message to determine whether it is an ICE connectivity check message, meaning: it is a STUN Binding request that contains all of the required attributes: FINGERPRINT, PRIORITY, ICE-CONTROLLED or ICE-CONTROLLING, USERNAME, and MESSAGE-INTEGRITY. If the message is not a structurally valid ICE connectivity check, the server MUST discard the message if there is no IP/IPv6 permission that applies.

If the message is an ICE connectivity check with no matching IP/IPv6 permission, the server then parses the value of the USERNAME attribute to extract the RFRAG value, which is the second colon-separated field. If a ufrag permission exists for the RFRAG value, the relay server generates a Data indication for the message. The Data indication is then sent to the TURN client.

NOTE: TURN-TCP [RFC6062] should be discussed in this document if/when it moves forward.

3.6. Processing a Data Indication

When the client receives a structurally valid Data indication, the client first checks whether the XOR-PEER-ADDRESS attribute value contains an IP address with which the client believes there is an active permission. If there is no such permission and the message is not a structurally valid ICE connectivity check, the client SHOULD discard the message. If the message is a structurally valid ICE connectivity check, the client parses, validates, and responds to the message as per standard ICE behavior.

4. ICE Interactions

The following subjects have been identified that should be discussed in greater detail:

- o Interaction with ice-lite.
- o Interactions with vanilla ice.

- o Interactions with trickle ice.

In particular, this section should discuss setting IP address permissions as a result of receiving a valid ICE connectivity check and/or learning the true candidates via the answer.

5. Security Considerations

The following subjects have been identified that must be discussed in greater detail:

- o An open port could be used to provide an unauthorized service. At this time, this is the primary security concern identified by the authors and some suitable mitigations should be discussed in this document.
- o A valid ICE connectivity check could be replayed by an attacker. This risk is shared by existing address-based permissions and so is not a significant concern for this draft.
- o An invalid ICE connectivity check using a snooped ufrag value could be forwarded to the client. This risk is also shared by existing address-based permissions and so is not a significant concern for this draft.
- o Others?

6. IANA Considerations

[Paragraphs below in braces should be removed by the RFC Editor upon publication]

[The LOCAL-UFRAG attribute requires that IANA allocate a value in the "STUN attributes Registry" from the comprehension-required range (0x0000-0x7FFF), to be replaced for TBD throughout this document]

This document defines the LOCAL-UFRAG attribute, described in Section 3.1. IANA has allocated the comprehension-required codepoint TBD for this attribute.

7. Acknowledgements

Thanks to T. Reddy for early review of this draft.

8. 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>>.
- [RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, DOI 10.17487/RFC5766, April 2010,
<<http://www.rfc-editor.org/info/rfc5766>>.
- [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>>.

Authors' Addresses

Brandon Williams
Akamai, Inc.
150 Broadway
Cambridge, MA 02142
USA

Email: brandon.williams@akamai.com

Justin Uberti
Google
747 6th Ave S
Kirkland, WA 98033
USA

Email: justin@uberti.name