

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: December 30, 2014

A. Johnston
Avaya
J. Uberti
Google
J. Yoakum
K. Singh
Avaya
June 28, 2014

An Origin Attribute for the STUN Protocol
draft-johnston-tram-stun-origin-03

Abstract

STUN, or Session Traversal Utilities for NAT, is a protocol used to assist other protocols traverse Network Address Translators or NATs. STUN, and STUN extensions such as TURN, or Traversal Using Relays around NAT, and ICE, Interactive Communications Establishment, have been around for many years but with WebRTC, Web Real-Time Communications, STUN and related extensions are about to see major deployments and implementation due to these protocols being implemented in browsers. 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 and SIP 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 December 30, 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
1.1. Requirements Language	5
2. STUN ORIGIN attribute	5
2.1. STUN Usage	6
2.2. TURN Usage	7
2.3. NAT Behavior Discovery Usage	7
2.4. ICE Usage	7
2.5. Media Keep-Alive Usage	7
2.6. SIP Keep-Alive Usage	7
2.7. Multiple Origins	7
3. IANA Considerations	8
4. Security Considerations	8
5. Implementation Status	9
6. Acknowledgements	10
7. Normative References	11
Authors' Addresses	12

1. Introduction

STUN, or Session Traversal Utilities for NAT, 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 [RFC5389] 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. In addition, the resource requirements of operating a STUN server 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 other cases, 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.

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 Section 4.5 of [I-D.ietf-tram-auth-problems]. 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. To implement this, a TURN server needs to know which realm to include in authentication challenge to TURN clients. One way to do this would be to create a unique TURN URI for each realm. This would require either a separate IP address or port for each realm, and this unique URI would need to be correctly provisioned by each domain (i.e. included in JavaScript, which then could not be copied between domains). Clearly, this doesn't scale for hundreds or thousands of domains. Origin information solves this problem since TURN requests will contain the domain in the Origin attribute. The TURN server just needs to be configured with a mapping between a domain (conveyed in the Origin) and the realm string (to be used in the authentication challenge). Thus, a single TURN URI could be used across all domains, and the resulting JavaScript code would be portable. There is no need for thousands of IP addresses or ports to

be allocated and managed.

It has been suggested that this origin insight is not needed if the server_name TLS extension in [RFC6066] is supported. This extension allows a TLS client to provide to the TLS server the name of the server they are contacting. In this case, the STUN or TURN client using TLS transport would provide the domain from the TURN server URI during the TLS client hello, allowing the TURN server to respond with the appropriate server certificate. For the TURN server domain to be used by the TURN server to choose the appropriate realm, this would require a unique TURN URI to be provisioned per realm. This is not scalable for supporting thousands of realms. Also, this URI would need to be provisioned in the JavaScript, making the resulting code non-portable. Finally, [RFC6066] provides no help for UDP or TCP transport, which are the most commonly used transports today for STUN and TURN.

Another approach that could be pursued is for the client to be explicitly provisioned with a realm value, to which its username and password are scoped. When using the long-term authentication method to authenticate to a TURN server, the client would include this REALM value in the initial, unauthorized requests, allowing the TURN server to know which REALM to use in its authorization challenge. This approach avoids many of the issues with the RFC 6066 approach, but it still requires the realm value to be explicitly provisioned in Javascript. In addition, it does not work in unauthenticated usages, i.e. STUN binding requests sent to a STUN server.

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", "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 less than 268 bytes. The value of 268 is chosen to be larger than the maximum 253 character domain name plus 8 characters for the URI scheme plus 5 characters for the port number. Senders MAY include multiple ORIGIN attributes in a request, and receivers MUST support parsing and receiving multiple ORIGIN attributes.

Editor's Note: At the appropriate time, the authors will work with the chairs of TRAM to follow [RFC4020] procedures to ensure that no attribute collisions occur while running code is being developed and tested.

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. To ensure backwards compatibility with [RFC3489], the ORIGIN attribute is padded to ensure its length is a multiple of 4 octets.

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).

For a Jabber client [RFC6120] using STUN and TURN, the ORIGIN attribute is the Jabber ID (JID) [RFC6122] of the Jabber Server that the client is using.

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. 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 SHOULD include the ORIGIN attribute. 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.

2.2. TURN Usage

For STUN requests sent using the long-term authentication method, such as TURN [RFC5766] allocate requests, the STUN client SHOULD include the ORIGIN attribute. 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. See the section below on "Multiple Origins."

2.3. NAT Behavior Discovery Usage

For the NAT Behavior Discovery Usage in [RFC5780], the ORIGIN attribute SHOULD be included in requests sent to a STUN server. This usage is most similar to the STUN Usage described earlier.

2.4. ICE Usage

For STUN requests sent using the short-term authentication method, such as ICE connectivity checks [RFC5245], the use of the ORIGIN attribute is NOT RECOMMENDED. No valid use cases for the ORIGIN attribute have been identified to date.

2.5. Media Keep-Alive Usage

For media keep-alive STUN requests described in Section 20 of [RFC5245], the use of the ORIGIN attribute is NOT RECOMMENDED. No valid use cases for the ORIGIN attribute have been identified to date.

2.6. SIP Keep-Alive Usage

For SIP keep-alive STUN requests described in [RFC5626], the use of the ORIGIN attribute is NOT RECOMMENDED. No valid use cases for the ORIGIN attribute have been identified to date.

2.7. Multiple Origins

Multiple Origins for HTTP Requests are described in Section 7.2 of [RFC6454]. As a result, a browser MAY generate a STUN request with multiple ORIGIN attributes present. A browser MUST NOT include multiple origins in a single ORIGIN attribute and instead use one Origin per attribute. If the ORIGIN attribute is being used by a TURN server to select the realm for authentication, the TURN server MAY use any or all of the Origins to select the realm. Any future

new usage scenarios of the ORIGIN attribute need to describe how to handle multiple Origins.

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 Origin header field via JavaScript, browsers should not allow a web application through JavaScript to set the STUN ORIGIN attribute.

The STUN MESSAGE-INTEGRITY attribute can provide integrity protection for all attributes present in a STUN request. However, MESSAGE-INTEGRITY is not present in the initial STUN message sent, so the initial ORIGIN attribute will not have integrity protection and hence could be modified or removed from a STUN request without the server knowing. Subsequent STUN requests containing the MESSAGE-INTEGRITY attribute will provide integrity protection for the contents of the ORIGIN attribute. The strength of this protection is a function of the secret used to generate the MESSAGE-INTEGRITY value.

The STUN ORIGIN attribute does have privacy implications. The recipient of the STUN request learns the web origin of the user. In addition, an on-path attacker could determine this information by inspecting STUN messages between the STUN client and STUN server, depending on the transport used. 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. STUN has a defined TLS transport; however, TLS transport is generally unsuitable for the real-time media flows that follow STUN requests and must use the same transport. The DTLS transport for STUN [I-D.ietf-tram-stun-dtls] provides a very good privacy solution to

this problem. In cases where privacy is paramount, the ORIGIN attribute SHOULD NOT be included or only included if DTLS or TLS transport is used.

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 Moskalkenko 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://code.google.com/p/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. Normative References

- [I-D.ietf-rtcweb-overview]
Alvestrand, H., "Overview: Real Time Protocols for Browser-based Applications", draft-ietf-rtcweb-overview-10 (work in progress), June 2014.
- [I-D.ietf-tram-auth-problems]
Reddy, T., R, R., Perumal, M., and A. Yegin, "Problems with STUN long-term Authentication for TURN", draft-ietf-tram-auth-problems-01 (work in progress), May 2014.
- [I-D.ietf-tram-stun-dtls]
Petit-Huguenin, M. and G. Salgueiro, "Datagram Transport Layer Security (DTLS) as Transport for Session Traversal Utilities for NAT (STUN)", draft-ietf-tram-stun-dtls-05 (work in progress), June 2014.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC3489] Rosenberg, J., Weinberger, J., Huitema, C., and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", RFC 3489, March 2003.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.
- [RFC4020] Kompella, K. and A. Zinin, "Early IANA Allocation of Standards Track Code Points", RFC 4020, February 2005.
- [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.
- [RFC5626] Jennings, C., Mahy, R., and F. Audet, "Managing Client-

Initiated Connections in the Session Initiation Protocol (SIP)", RFC 5626, October 2009.

- [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.
- [RFC5780] MacDonald, D. and B. Lowekamp, "NAT Behavior Discovery Using Session Traversal Utilities for NAT (STUN)", RFC 5780, May 2010.
- [RFC6066] Eastlake, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, January 2011.
- [RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, March 2011.
- [RFC6122] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Address Format", RFC 6122, March 2011.
- [RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, December 2011.
- [RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, July 2013.
- [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

Phone:
Email: alan.b.johnston@gmail.com

Justin Uberti
Google
Kirkland, WA
USA

Phone:
Email: justin@uberti.name

John Yoakum
Avaya
Cary, NC
USA

Phone:
Email: yoakum@avaya.com

Kundan Singh
Avaya
San Francisco, CA
USA

Phone:
Email: kundan10@gmail.com

TRAM
Internet-Draft
Intended status: Standards Track
Expires: August 16, 2015

P. Martinsen
H. Wildfeuer
Cisco
February 12, 2015

Differentiated prIorities and Status Code-points Using Stun Signalling
(DISCUSS)
draft-martinsen-tram-discuss-02

Abstract

This draft describes a mechanism for information exchange between an application and the network. The information provided from the application to the network MAY be used by a network element in the path to modify its behavior to improve application quality of experience (QoE). Likewise, the information provided by the network to the application MAY be used by the application to modify its behavior to optimize for QoE.

The information provided from the application to the network can also be useful for middleboxes that are responsible for security at edges of network (e.g. firewalls) or other middleboxes in determining how to treat the packets delivered from this application.

Requirements Language

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 RFC 2119 [RFC2119].

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 August 16, 2015.

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. General Usage	3
3. Network Processing	6
3.1. Packet Processing by Network Device	6
3.2. Interaction with DSCP	7
4. Interaction with ICE	7
5. Multiplexed Streams	8
6. New STUN attributes	9
6.1. STREAM-TYPE	9
6.2. BANDWIDTH-USAGE	10
6.3. STREAM-PRIORITY	10
6.4. NETWORK-STATUS	11
6.5. SUB-STREAM-TYPE, SUB-STREAM-PRIORITY	12
7. IANA Considerations	13
8. Implementation Status	13
8.1. NATtools	13
9. Security Considerations	14
10. Acknowledgements	14
11. References	14
11.1. Normative References	14
11.2. Informational References	15
Authors' Addresses	16

1. Introduction

In the context of Content, Mobile, Fixed Service, Service Providers, Enterprise and Private networks have a need to prioritize packet flows end-to-end. These flows are often dynamic, time-bound, encrypted, peer-to-peer, possibly asymmetric, and might have different priorities depending on network conditions, direction, time of the day, dynamic user preferences and other factors. These

factors may be time variant, and thus need to be signalled. Moreover, in many cases of peer-to-peer communication, flow information is known only to the endpoint. These considerations, coupled with the trend to use encryption for browser-to-browser communication [I-D.ietf-rtcweb-security-arch], imply that access lists, deep packet inspection and other static prioritization methods cannot be employed successfully to prioritize packet flows.

The lack of congestion control in UDP may lead to problems as described in [I-D.eggert-tsvwg-rfc5405bis]. The mechanism described in this document can be used to introduce fairness and congestion control for UDP streams.

There is a need for a solution that is easy for the application developer to use. That means consistent behavior on all supported platforms and preferably without need of administrative privileges to set and read values. The solution also needs to be able to cross administrative domains without the risk of being rewritten. [[Q1: This draft will only offer tamper detection of some of the values. Further discussion regarding the incentive to lie is needed. --palmarti]]

This draft describes how these problems can be solved by defining a few strictly defined STUN [RFC5389] attributes which can be added to any STUN message the client wants to send. STUN messages are typically sent during the ICE [RFC5245] connectivity check phase when the media session is established, or when keep-alive STUN messages are sent after the session is established. The application is not limited to those two scenarios, if some communication between application and network is needed it can choose to do so at any time.

Devices on the media path can use the information in the STUN attributes to prioritize the flow, perform traffic engineering, provide network analytics or as a gateway to existing methods for prioritizing flows (DSCP [RFC2474]). Applications can use information in network status attribute to influence rate stating points or rate adaption mechanisms.

2. General Usage

This draft defines several attributes that can be added to a STUN message; STREAM-TYPE, BANDWIDTH-USAGE, STREAM-PRIORITY and NETWORK-STATUS. See Section 6 for the formal description. It is RECOMMENDED to add them to a STUN request response pair, especially if the NETWORK-STATUS attribute is in use. This allows the information gathered to be sent back to the requesting agent in the the STUN response.

The STREAM-TYPE, BANDWIDTH-USAGE, STREAM-PRIORITY attributes MUST be added before any INTEGRITY attribute. It is RECOMMENDED to only add these attributes to STUN messages containing a INTEGRITY attribute as this prevents tampering with the content of the attribute.

If the client wants to receive feedback from the network it must add a null NETWORK-STATUS attribute. A null NETWORK_STATUS attribute is created by filling in the all the fields in the attribute with 0x0 values. This attribute MUST be added after the INTEGRITY attribute, as on-path devices may write information into this attribute. Having a readily available attribute to write into will save the the on-path device from growing buffers to add their own attribute. On path devices SHOULD not add their own NETWORK-STATUS attribute (or any other STUN attribute).

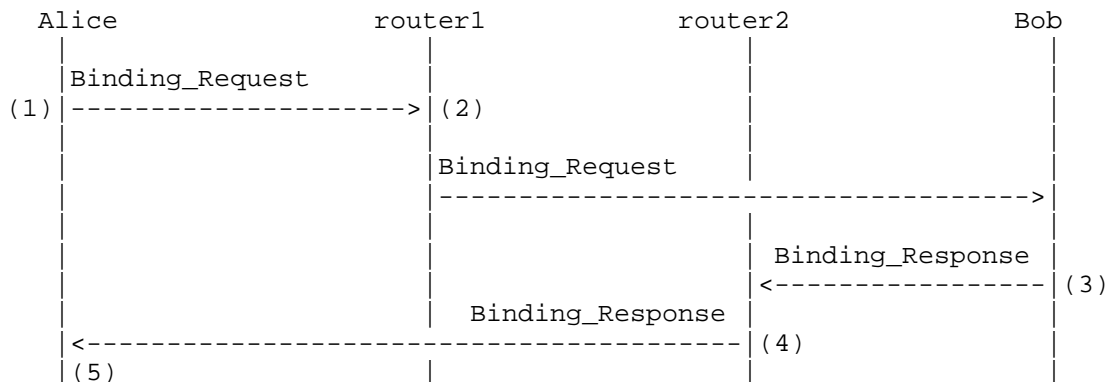
If an agent receives a STUN request with a NETWORK-STATUS attribute after the INTEGRITY attribute, it should copy the content into a new NETWORK-STATUS attribute and add it before the INTEGRITY attribute when sending the STUN response. A new null NETWORK-STATUS attribute can be added after the INTEGRITY attribute. New STUN attributes described in this draft can also be added describing the stream in this direction.

If an agent receives a STUN response with a NETWORK-STATUS attribute before the INTEGRITY attribute, this describes the stream in the upstream direction. A NETWORK-STATUS attribute after the INTEGRITY attribute describes the stream in the downstream direction.

It might make sense to distinguish DISCUSS packets from normal STUN packets. This would prevent unnecessary processing of normal STUN packets on the network nodes.

[[Q2: A few alternatives (Needs discussion): ---1: Alter the STUN magic cookie. (But than i would not be a STUN packet anymore, and that raises a new set of problems) ---2: Add a special this is DISCUSS attribute. This must be the first attribute in the message. This allows for network node to look for DISCUSS packets at a fixed offset without needing to parse the entire packet. ---3: Alter the transaction id. This might be problematic if using it in conjunction with ICE connectivity checks. But probably fine in other scenarios. ---4: Define a new STUN Method. Also brakes ICE, makes it harder to tag onto attributes to already in use messages. --palmarti]]

[[Q3: Do we want to restrict this to req/resp or do we want to allow for the attributes to be added in this fashion for indications as well? --palmarti]]



DISCUSS example flow

1. Alice creates a Binding Request, adds STREAM-TYPE, BANDWIDTH-USAGE, STREAM-PRIORITY attributes before the INTEGRITY attribute and a single null NETWORK-STATUS attribute after the INTEGRITY attribute.
2. Router1 inspects the STUN Request message and reads any STREAM-TYPE, BANDWIDTH-USAGE, or STREAM-PRIORITY attributes and the information they contain. It then updates the NETWORK-STATUS attribute with any information the router have. It then forwards the request.
3. Bob processes the STUN Request. When Bob builds the response, it copies the NETWORK-STATUS attribute into the STUN Response before the INTEGRITY check and adds new null NETWORK-STATUS attribute after the INTEGRITY attribute. Bob then transmits the message.
4. Router2 (first DISCUSS network element for the downstream direction) inspects the Response message, reads the STREAM-TYPE, BANDWIDTH-USAGE, or STREAM-PRIORITY attributes and MAY alter the NETWORK-STATUS attribute located after the INTEGRITY attribute. It then transmits the message.
5. When Alice receives the STUN Response, she can extract the STREAM-TYPE, BANDWIDTH-USAGE, or STREAM-PRIORITY attributes and the two NETWORK-STATUS attributes to get a complete picture of what the remote agent is sending and how the status of both the upstream and downstream path.

3. Network Processing

This section describes the processing of DISCUSS packets by network devices.

3.1. Packet Processing by Network Device

Network devices are said to support DISCUSS if they perform inspection of packets being forward or switched in order to identify an DISCUSS STUN packet. These devices will also be able to read/write STUN attributes to/from this packet. It is not required that every network device in the path support DISCUSS. It is expected that DISCUSS will have the most value being implemented at certain points in the network (PIN's) such as WAN edge devices, wireless access devices, and Internet gateways.

Network devices that support DISCUSS MAY utilize the information provided by the application in the STUN attributes to modify their behavior. These include the attributes defined in this document with the exception of the NETWORK-STATUS attribute. The NETWORK-STATUS attribute SHOULD NOT be used by the DISCUSS capable network device to modify its behavior. The intent of the NETWORK-STATUS attribute is for the application to modify its behavior.

If the NETWORK-STATUS attributes exists in a DISCUSS packet after an INTEGRITY attribute, the DISCUSS capable network device MUST process it as described in this section. NETWORK-STATUS attributes that exist before the INTEGRITY attribute MUST NOT be modified by the network device. The modifications to the NETWORK-STATUS attribute are:

- o Update the Node Cnt field in the attribute. The device SHALL increment this field by one unless it is at its maximum (saturated) value. If the field is at its maximum value, the device SHALL NOT modify this field.
- o Overwrite the attribute CS bit if the value at this device is "worst" than the current value. In other words, only write to this bit if the device is experiencing congestion on relevant queues/interfaces for this flow AND the current value of this field is 0 (Off).

The determination of congestion at a device is out of the scope of this document. Setting of CS bit to On by the device is meant to provide direct feedback to the application of potential or current loss of packets in its flow (s). The application can then react to this indication by altering its encoding of information in the packet to deal with congestion/packet loss, e.g. reduce its encoding rate or

switch to embedded encoding. Devices SHOULD ensure that the DISCUSS capable applications that do react to congestion notification by reducing their transmission rate be treated properly to ensure fairness with non reacting applications (i.e. ensure fairness for well behaving applications).

The DISCUSS STUN packet SHOULD experience minimal extra processing delay through the DISCUSS capable network device relative to non-DISCUSS packets in the flow. The DISCUSS STUN packet MAY be placed out of order in the packet flow, but SHOULD NOT be delayed more than a few packet interval times.

3.2. Interaction with DSCP

One of the attributes that may be added to the STUN packet by the application is the STREAM-PRIORITY attribute. This attribute indicates the relative priority of streams inside of an application session. This attribute is compatible with the use of DSCP (or other priority markings) at the networking layer as described in this section.

Since transport layer markings may be modified by middle boxes or devices in the path or at the interface of the application itself due to the lack of support in the OS network stack, the STREAM-PRIORITY attribute can be used as a mechanism for ensuring proper QoS treatment through multiple domains. DISCUSS capable device may use the STREAM-PRIORITY attribute to remark the DSCP value to the appropriate value. DSCP re-marking based on STREAM-PRIORITY attribute may make sense at certain PIN's, e.g. gateway between network domains (e.g. managed network to/from Internet), access switches in managed network, etc. The translation from the Priority number in the STREAM-PRIORITY attribute to the correct transport layer marking (e.g. DSCP) is implementation specific and out of the scope of this document.

[I-D.dhesikan-tsvwg-rtcweb-qos] provides the recommended DSCP values for webrtc enabled browsers to use for various classes of traffic.

4. Interaction with ICE

An ICE connectivity check is performed by sending a STUN Binding indication. Prior to sending the agent can add one STREAM-TYPE attribute. If added, only one MUST be added. This is to avoid unnecessary large STUN packets during the connectivity checks. If the connectivity check is sent on a 5-tuple that multiplexes different types of media and more detailed information wants to be signalled it should be done after the connectivity check phase is finished.

This limits the information the STUN messages are able to convey during the connectivity checks, but also avoids adding network confusion with BANDWIDTH-USAGE attributes describing different paths that never going to be utilized.

[[Q4: Problem with consent freshness if not based on STUN.
--palmarti]]

5. Multiplexed Streams

In some scenarios a 5-tuple can be used to transport several media streams. BUNDLE [I-D.ietf-mmusic-sdp-bundle-negotiation] describes such a mechanism.

At times, the different "streams" carried in this bundle require very different treatment from the network, including the ability to prioritize some of these "streams" over others. For example, the application may bundle video and audio in the same 5-tuple flow, but would like the network to prioritize the delivery of audio over that of video in the case of congestion. Another example is the use of embedded (or scalable) coding for video. Per RFC 6190 [RFC6190], using Multi-session Transmission (MST) the layers are transported in separate sub-flows (RTP sessions) within the bundled flow. Using the STREAM-TYPE attribute with the extension to identify the sub-flow and its priority would allow network elements, if capable, to provide differentiated services even in the case of bundling.

For RTP/SRTP based flows, the existence and attributes for sub-flows in the flow MAY be indicated by the application via the SUB-STREAM-xxx attributes. These attributes MUST only be included if the equivalent STREAM-xxxx attributes are included. It is expected that only a sub-set of network elements representing bottleneck Points in Network (PIN) will be able to inspect the higher layer protocols to differentiate sub-flows, so it is important to describe the aggregate flow, and then the sub-flows. The SUB-STREAM-xxxx attributes are similar to the corresponding STREAM-xxxx attributes with the addition of the application layer identifier field. For the case of RTP/SRTP, this field is the SSRC assigned to the flow. Note that this will only work for non-header encrypted SRTP.

When describing the aggregate stream with a STREAM-TYPE attribute there are two possibilities to describe the streams that are multiplexed. Adding one attribute for each type (Audio, Video,++), or to save a few bits on the wire it is also possible to construct the STREAM-TYPE so a one type value describes several types. For example audio have the value of 1 and application data have the value of 4. If the STREAM_TYPE value is set to 5 the only combination that gives that is audio and application data. As previously discussed,

in the case of bundling, the aggregate stream attribute MUST be included before the optional sub-stream attributes

The other attributes BANDWIDTH-USAGE, STREAM-PRIORITY and NETWORK-STATUS SHOULD only be added once as they describe the behavior of the 5-tuple and not individual streams.

6. New STUN attributes

This STUN extension defines the following new attributes:

```

0xXXX0: STREAM-TYPE
0xXXX1: BANDWIDTH-USAGE
0xXXX2: STREAM-PRIORITY
0xXXX8: SUB-STREAM-TYPE
0xXXX9: SUB-BANDWIDTH-USAGE
0XXXXA: SUB-STREAM-PRIORITY
0xYYYY: NETWORK-STATUS

```

6.1. STREAM-TYPE

This attribute have a length that are multiples of 4 (32) so no padding is necessary.

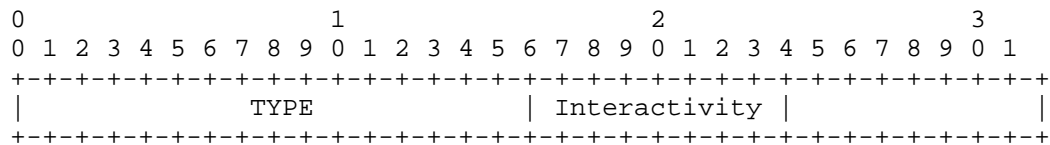


Figure 1: STREAM TYPE Attribute

TYPE: STREAM TYPE is a 16 bit value defined in Figure 2 below describing the flow.

```

0x0001 Audio
0x0002 Video
0x0004 Application Data
0x0008 Other

```

Figure 2: STREAM Types

Interactivity: Is a 8 bit value defined in Figure 3 below describing the flow.

```

0x00 Undef
0x01 Stream (Broadcast? Oneway?)
0x02 Interactive

```

Figure 3: Interactivity Types

It is possible to combine the stream types if a stream contains more than one type.

If a 5-tuple is used to send both a audio and video stream, the stream type can be set to 0x0006. This can be useful if the application wants to hint that the 5-tuple contains several streams, This is useful if the attribute is added to STUN binding requests during ICE connectivity checks. If more information regarding multiplexed streams is needed it is possible to add more than one attribute to a STUN message (See section ??). This can be done to STUN messages that are being sent after the connectivity check phase is finished (Keep-alive, consent freshness). During this phase the added size of the STUN messages pose no security threat.

6.2. BANDWIDTH-USAGE

This attribute have a length that are multiples of 4 (32) so no padding is necessary.

```

      0               1               2               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     AVERAGE (kbps)                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     MAX (kbps)                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure 4: BANDWIDTH USAGE Attribute

AVERAGE: Expected sustained bandwidth usage for this stream. Note that for elastic types of streams like video, sudden large movements in the picture may lead to this value being inaccurate.

MAX: The maximum bandwidth usage for this stream. If the sustained and max value differ greatly it might be safe to assume that an elastic encoder is in use. (Would it be useful to say something about expected BURST lengths?)

6.3. STREAM-PRIORITY

This attribute have a length that are multiples of 4 (32) so no padding is necessary.

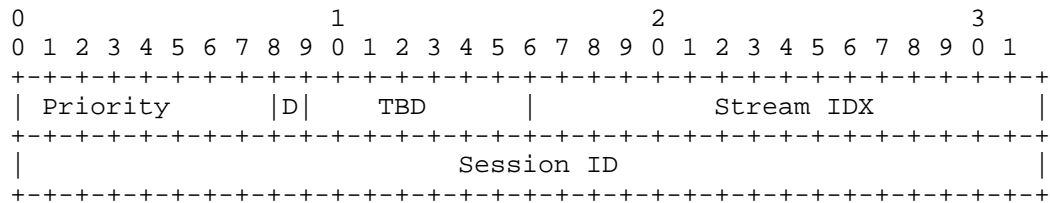


Figure 5: STREAM PRIORITY Attribute

Priority: Describes this streams priority among other streams coming from this endpoint/application (With same session ID). Values range from 255 (0xFF) to 0.

D: Delay sensitive. The application can set this bit as a hint to the network element that the stream is delay sensitive. (Unsure if this is useful)

Stream IDX: Application can choose set this to ease debugging in the network. A reasonable value can for example be the index have in the SDP.

Session ID: Identification to distinguish what session this stream is part of. This MUST have the same value for all the media streams the application wants to give differentiated services. (Note that this ID may overlap with other streams that originates from a different IP address. The network element MUST only prioritize among streams with the same Session Id originating from the same IP)

6.4. NETWORK-STATUS

This attribute have a length that are multiples of 4 (32) so no padding is necessary. The values are kept in the same attribute to make it easier for the network element to process it. Only one attribute, with static placement of the fields. [[Q5: Does this matter? Could we have several attributes with possible different ordering without any problem for the network element? --palmarti]]

This attribute MUST be added after any INTEGRITY attribute in the STUN message. Values in this attribute can be updated along the network path by nodes that are not able to regenerate a correct INTEGRITY attribute.

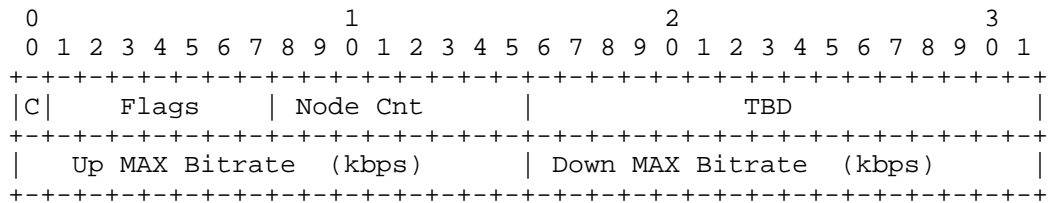


Figure 6: NETWORK-STATUS Attribute

C: Congestion Status. This bit is set to indicate that there is congestion at the network device's relevant queues/interfaces for this flow. The network element should set this bit to 1 (On) if it is experiencing congestion. This bit is set to 0 (off) when the attribute is created by the application. The application that sees this bit set might act on it by doing some rate adaption or similar action.

Flags: 7 more bits available for flags.

Node Cnt: Numbers of nodes that supports DISCUSS in the network path. Any router on the path that understands DISCUSS should increase this number. This field is set to 0 when the attribute is created by the application.

TBD: 16 more bits available for useful info.

Up MAX Bitrate: Available MAX bit-rate the router is able to handle for the 5-tuple in the UP direction. (Same direction as the packet is moving)

Down MAX Bitrate: Available MAX bit-rate the router is able to handle for the 5-tuple in the DOWN direction. (Opposite direction as the packet is moving)

6.5. SUB-STREAM-TYPE, SUB-STREAM-PRIORITY

These attributes are identical format to their aggregate stream version (STREAM-TYPE, STREAM-PRIORITY) with the addition of a transport layer identifier. The transport layer identifier is a 64 bit field which contains the unique identifier of the sub-stream for which the attribute applies.

Currently, only RTP transport is supported with the identifier being the SSRC currently used by the sub-stream.

7. IANA Considerations

IANA is requested to add the following attributes to the STUN attribute registry [iana-stun],

- o 0xxx0: STREAM-TYPE (0xxx0, in the comprehension-optional range)
- o 0xxx1: BANDWIDTH-USAGE (0xxx1, in the comprehension-optional range)
- o 0xxx2: STREAM-PRIORITY (0xxx2, in the comprehension-optional range)
- o 0yyyy: NETWORK-STATUS (0yyyy, in the comprehension-optional range)

8. Implementation Status

[Note to RFC Editor: Please remove this section and reference to [RFC6982] prior to publication.]

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".

8.1. NATtools

Organization: Cisco

Description: Open-Source ICE, TURN and STUN implementation.

Implementation: <https://github.com/cisco/NATTools>

Level of maturity: Code is stable. Tests being run to learn more on how to leverage the information shared between client and network.

Coverage: Implements the DISCUSS attributes

Licensing: BSD

Implementation experience: Draft was implemented with internal video test clients. Wireless access point implemented STUN detection in the media path and acted on the information in the DISCUSS attributes. After running tests in different congestion scenarios it is clear that sharing information between endpoint and network can help with congestion and end-user experience. This approach required little effort to implement on the client side.

Contact: Paal-Erik Martinsen <palmarti@cisco.com>.

9. Security Considerations

Due to the security implications described in [I-D.thomson-mmusic-ice-webrtc] where large STUN packet are used to amplify an attack, keeping the added STUN attributes small is a important design consideration.

To avoid unwanted information leakage the new defined STUN attributes defined in this draft are strictly defined. No more information should be leaked that an on-path device could learn by observing the stream over time or do some deep packet analysis. This draft would benefit from more discussions on this topic.

It is also worth noticing that the STUN attributes defined should be treated as hints, and more work is needed regarding how to deal with misbehaving clients or network devices.

10. Acknowledgements

Authors would like to thank Dan Wing, Anca Zamfir, Jon Snyder and Cullen Jennings for their comments and review.

11. References

11.1. Normative References

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

- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.
- [RFC6190] Wenger, S., Wang, Y., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, May 2011.
- [RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, July 2013.

11.2. Informational References

- [I-D.ietf-rtcweb-security-arch]
Rescorla, E., "WebRTC Security Architecture", draft-ietf-rtcweb-security-arch-09 (work in progress), February 2014.
- [I-D.thomson-mmusic-ice-webrtc]
Thomson, M., "Using Interactive Connectivity Establishment (ICE) in Web Real-Time Communications (WebRTC)", draft-thomson-mmusic-ice-webrtc-01 (work in progress), October 2013.
- [I-D.dhesikan-tsvwg-rtcweb-qos]
Dhesikan, S., Druta, D., Jones, P., and J. Polk, "DSCP and other packet markings for RTCWeb QoS", draft-dhesikan-tsvwg-rtcweb-qos-06 (work in progress), March 2014.
- [I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C., Alvestrand, H., and C. Jennings, "Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers", draft-ietf-mmusic-sdp-bundle-negotiation-05 (work in progress), October 2013.

[I-D.eggert-tsvwg-rfc5405bis]

Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", draft-eggert-tsvwg-rfc5405bis-01 (work in progress), June 2014.

[iana-stun]

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

Authors' Addresses

Paal-Erik Martinsen
Cisco Systems, Inc.
Philip Pedersens vei 20
Lysaker, Akershus 1366
Norway

Email: palmarti@cisco.com

Herb Wildfeuer
Cisco Systems, Inc.
821 Alder Drive
Milpitas, California 95035
United States

Email: hwildfeu@cisco.com

TRAM
Internet-Draft
Intended status: Standards Track
Expires: November 3, 2014

P. Patil
T. Reddy
D. Wing
Cisco
May 2, 2014

TURN Server Auto Discovery
draft-patil-tram-turn-serv-disc-01

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 or the ISP, 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 two 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 November 3, 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	2
2. Terminology	3
3. Discovery Procedure	3
4. Discovery using Service Resolution	4
4.1. Retrieving Domain Name	4
4.1.1. DHCP	4
4.1.2. IP Address	5
4.1.3. From own Identity	5
4.2. Resolution	5
4.2.1. SOA	6
5. Discovery using Anycast	7
6. Deployment Considerations	7
6.1. Mobility and Changing IP addresses	7
7. IANA Considerations	8
7.1. Anycast	8
8. Security Considerations	8
8.1. Service Resolution	8
8.2. Anycast	8
9. Acknowledgements	9
10. References	9
10.1. Normative References	9
10.2. Informative References	10
Appendix A. Change History	10
A.1. Change from draft-patil-tram-serv-disc-00 to -01	10
Authors' Addresses	11

1. Introduction

TURN [RFC5766] is a protocol that is often used to improve the connectivity of P2P applications. By providing a relay service, TURN ensures that a connection can be established even when one or both sides is 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 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. WebRTC usages and

related extensions, which are mostly based on web applications, need this immediately.

This document describes two discovery mechanisms. The reason for providing two mechanisms is to maximize the opportunity for discovery, based on the network in the which the TURN client sees itself.

- 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 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 MUST proceed with discovery in the following order:

1. Local Configuration : Local or manual configuration should be tried first, as it may be an explicit preferred choice of a user. 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 every address family.
2. Service Resolution : The TURN client attempts to perform TURN service resolution using the DNS domain name that the host belongs to OR the hosts' global IP address. The TURN client will attempt to do this for each combination of interface and address family. The retrieved DNS domain names OR IP addresses are then used for NAPTR lookups.
3. Anycast : Send TURN allocate request to the assigned TURN anycast request for each combination of interface and address family.

While it is expected that Step-3 be performed if Step-2 fails, an implementation may choose to perform steps 2 and 3 in parallel.

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).

On hosts with more than one interface or address family (IPv4/v6), the TURN server discovery procedure has to be run for each combination of interface and address family.

4.1. Retrieving Domain Name

The domain, in which the client is located, can be determined using one of the techniques provided below. An implementation can choose to use any or all techniques.

Implementations may allow the user to specify a default name that is used if no specific name has been configured. Other means of retrieving domain names may be used, which are outside the scope of this document e.g. local configuration.

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. [RFC5986] defines 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 still may be useful if the option defined in [RFC5986] is 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. IP Address

Typically, but not necessarily, the DNS domain name is the domain name in which the client is located, i.e., a PTR lookup on the client's IP address (according to [RFC1035], Section 3.5 for IPv4 or [RFC3596], Section 2.5 for IPv6) would yield a similar name. However, due to the widespread use of Network Address Translation (NAT), the client MAY need to determine its public IP address using mechanisms described in [I-D.ietf-geopriv-res-gw-lis-discovery].

4.1.3. From own Identity

A TURN client could also 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.).

4.2.1. SOA

If no TURN-specific S-NAPTR records can be retrieved using the previous step, additional steps described in this section have to be followed. First, an SOA record for the "reverse zone" i.e., the zone in the in-addr.arpa. or ip6.arpa. domain that contains the IP address(s) in question, has to be retrieved. IP addresses can be determined, if not done already, as described in Section 4.1.2.

A sample SOA record could be:

```

100.51.198.in-addr.arpa
IN  SOA dns1.isp.example.net. hostmaster.isp.example.net. (
                                1           ; Serial
                                604800      ; Refresh
                                86400       ; Retry
                                2419200     ; Expire
                                604800 )    ; Negative Cache TTL

```

If this lookup fails, the discovery procedure is aborted without a result.

Once the SOA record is available, the discovery procedure extracts the MNAME field, i.e., the responsible master name server from the SOA record. An example MNAME could be: dns1.isp.example.net. Then, an S-NAPTR lookup as specified in the previous step Section 4.2 is performed on this MNAME to discover the TURN service. 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).

5. Discovery using Anycast

IP anycast is an elegant solution 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.

6. Deployment Considerations

6.1. 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-mmusic-ice-mobility] can be used for ongoing streams.

7. IANA Considerations

7.1. 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/>>

8. Security Considerations

In general, it is recommended that a TURN client authenticate with the TURN server to identify a rouge server.

[I-D.petithuguenin-tram-turn-dtls] can be potentially used by a client to validate a previously unknown server.

8.1. 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.

8.2. 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.

9. Acknowledgements

Discovery using Service Resolution described in Section 4 of this document was derived from similar techniques described in ALTO Server Discovery [I-D.ietf-alto-server-discovery] and [I-D.kist-alto-3pdisc].

10. References

10.1. Normative References

- [I-D.ietf-geopriv-res-gw-lis-discovery]
Thomson, M. and R. Bellis, "Location Information Server (LIS) Discovery using IP address and Reverse DNS", draft-ietf-geopriv-res-gw-lis-discovery-08 (work in progress), December 2013.
- [I-D.petithuguenin-tram-turn-dtls]
Petit-Huguenin, M. and G. Salgueiro, "Datagram Transport Layer Security (DTLS) as Transport for Traversal Using Relays around NAT (TURN)", draft-petithuguenin-tram-turn-dtls-00 (work in progress), January 2014.
- [I-D.wing-mmusic-ice-mobility]
Wing, D., Reddy, T., Patil, P., and P. Martinsen, "Mobility with ICE (MICE)", draft-wing-mmusic-ice-mobility-06 (work in progress), February 2014.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2132] Alexander, S. and R. Droms, "DHCP Options and BOOTP Vendor Extensions", RFC 2132, March 1997.

- [RFC3188] Hakala, J., "Using National Bibliography Numbers as Uniform Resource Names", RFC 3188, October 2001.
- [RFC3596] Thomson, S., Huitema, C., Ksinant, V., and M. Souissi, "DNS Extensions to Support IP Version 6", RFC 3596, October 2003.
- [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.
- [RFC5928] Petit-Huguenin, M., "Traversal Using Relays around NAT (TURN) Resolution Mechanism", RFC 5928, August 2010.
- [RFC5986] Thomson, M. and J. Winterbottom, "Discovering the Local Location Information Server (LIS)", RFC 5986, September 2010.

10.2. Informative References

- [I-D.ietf-alto-server-discovery]
Kiesel, S., Stiemerling, M., Schwan, N., Scharf, M., and S. Yongchao, "ALTO Server Discovery", draft-ietf-alto-server-discovery-10 (work in progress), September 2013.
- [I-D.kist-alto-3pdisc]
Kiesel, S., Krause, K., and M. Stiemerling, "Third-Party ALTO Server Discovery (3pdisc)", draft-kist-alto-3pdisc-05 (work in progress), January 2014.

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

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: August 15, 2014

M. Petit-Huguenin
Jive Communications
G. Salgueiro
Cisco Systems
February 11, 2014

Datagram Transport Layer Security (DTLS) as Transport for Session
Traversal Utilities for NAT (STUN)
draft-petithuguenin-tram-stun-dtls-00

Abstract

This document specifies the usage of Datagram Transport Layer Security (DTLS) as a transport protocol for Session Traversal Utilities for NAT (STUN). It provides guidances on when and how to use DTLS with the currently standardized STUN Usages. It also specifies modifications to the STUN URIs and TURN URIs and to the TURN resolution mechanism to facilitate the resolution of STUN URIs and TURN URIs into the IP address and port of STUN and TURN servers supporting DTLS as a transport protocol.

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 August 15, 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	2
2. Terminology	3
3. DTLS as Transport for STUN	3
4. STUN Usages	4
4.1. NAT Discovery Usage	4
4.1.1. DTLS Support in STUN URIs	4
4.2. Connectivity Check Usage	4
4.3. Media Keep-Alive Usage	5
4.4. SIP Keep-Alive Usage	5
4.5. NAT Behavior Discovery Usage	5
4.6. TURN Usage	5
4.6.1. DTLS Support in TURN URIs	6
4.6.2. Resolution Mechanism for TURN over DTLS	6
5. Implementation Status	7
5.1. turnuri	8
5.2. rfc5766-turn-server	8
6. Security Considerations	9
7. IANA Considerations	9
7.1. S-NAPTR application protocol tag	9
7.2. Service Name and Transport Protocol Port Number	9
7.2.1. The stuns Service Name	10
7.2.2. The turns Service Name	10
8. Acknowledgements	10
9. References	10
9.1. Normative References	10
9.2. Informative References	12
Appendix A. Examples	12
Appendix B. Release notes	13
B.1. Modifications between petithuguenin-tram-turn-dtls-00 and petithuguenin-tram-stun-dtls-00	13
Authors' Addresses	14

1. Introduction

STUN [RFC5389] defines Transport Layer Security (TLS) over TCP (simply referred to as TLS [RFC5246]) as the transport for STUN due to additional security advantages it offers over plain UDP or TCP transport. But TLS-over-TCP is not an optimal transport when STUN is used for its originally intended purpose, which is to support multimedia sessions. This sub-optimality primarily stems from the

added latency incurred by the TCP-based head-of-line (HOL) blocking problem coupled with additional TLS buffering (for integrity checks). This is a well documented and understood transport limitation for secure real-time communications.

TLS-over-UDP (referred to as DTLS [RFC6347]) offers the same security advantages as TLS-over-TCP, but without the undesirable latency concerns.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] when they appear in ALL CAPS. When these words are not in ALL CAPS (such as "must" or "Must"), they have their usual English meanings, and are not to be interpreted as RFC 2119 key words.

3. DTLS as Transport for STUN

STUN [RFC5389] defines three transports: UDP, TCP, and TLS. This document adds DTLS as a valid transport for STUN.

STUN over DTLS MUST use the same retransmission rules as STUN over UDP (as described in Section 7.2.1 of [RFC5389]). It MUST also use the same rules that are described in Section 7.2.2 of [RFC5389] to verify the server identity. STUN over DTLS MUST, at a minimum, support `TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256` [[TODO: What is the recommendation these days?]]. The same rules established in Section 7.2.2 of [RFC5389] for keeping open and closing TCP/TLS connections MUST be used as well for DTLS associations.

In addition to the path MTU rules described in Section 7.1 of [RFC5389], if the path MTU is unknown, the actual STUN message needs to be adjusted to take into account the size of the (13-byte) DTLS Record header, the MAC size, the padding size and the eventual compression applied to the payload.

By default, STUN over DTLS MUST use port 5349, the same port as STUN over TLS. However, the SRV procedures can be implemented to use a different port (as described in Section 9 of [RFC5389]). When using SRV records, the service name MUST be set to "stuns" and the application name to "udp".

Classic STUN [RFC3489] defines only UDP as a transport and DTLS MUST NOT be used. Any STUN request or indication without the magic cookie over DTLS MUST always result in an error. [[TODO: Note that it is a departure from RFC 5389, which does not explicitly state what to do in that case. Are we OK with this?]]

4. STUN Usages

[RFC5389] Section 7.2 states that STUN usages must specify which transport protocol is used. The following sections discuss if and how the existing STUN usages are used with DTLS as the transport. Future STUN usages MUST take into account DTLS as a transport and discuss its applicability. [[TODO: Note that Section 14 of RFC 5389 omitted to say that transport applicability MUST be discussed. Is this a reasonable addition?]].

4.1. NAT Discovery Usage

As stated by Section 13 of [RFC5389], "...TLS provides minimal security benefits..." for this particular STUN usage. DTLS will also similarly offer only limited benefit. This is because the only mandatory attribute that is TLS/DTLS protected is the XOR-MAPPED-ADDRESS, which is already known by an on-path attacker, since it is the same as the source address and port of the STUN request. On the other hand, using TLS/DTLS will prevent an active attacker to inject XOR-MAPPED-ADDRESS in responses. The TLS/DTLS transport will also protect the SOFTWARE attribute, which can be used to find vulnerabilities in STUN implementations.

Regardless, this usage is rarely used by itself, since TURN [RFC5766] is generally mandatory to use with ICE [RFC5245], and TURN provides the same NAT Discovery feature as part of an Allocation creation. In fact, with ICE, the NAT Discovery usage is only used when there is no longer any resource available for new Allocations in the TURN server.

4.1.1. DTLS Support in STUN URIs

This document does not make any changes to the syntax of a STUN URI [RFC7064]. As indicated in Section 3.2 of [RFC7064], secure transports like STUN over TLS, and now STUN over DTLS, MUST use the "stuns" URI scheme.

The <host> value MUST be used when using the rules in Section 7.2.2 of [RFC5389] to verify the server identity. [[TODO: What happens if an IP address is used in the URI? Should we forbid that?]]

4.2. Connectivity Check Usage

Using DTLS would hide the USERNAME, PRIORITY, USE-CANDIDATE, ICE-CONTROLLED and ICE-CONTROLLING attributes. But because MESSAGE-INTEGRITY protects the entire STUN response using a password that is known only by looking at the SDP exchanged, it is not possible for an attacker to inject an incorrect XOR-MAPPED-ADDRESS, which would subsequently be used as a peer reflexive candidate.

Adding DTLS on top of the connectivity check would delay, and consequently impair, the ICE process. There is, in fact, a proposal ([I-D.thomson-rtcweb-ice-dtls]) to use the DTLS handshake used by the WebRTC SRTP streams as a replacement for the connectivity checks, proving that adding additional round-trips to ICE is undesirable.

This usage MUST NOT be used with a STUN URI.

4.3. Media Keep-Alive Usage

The media keep-alive (described in Section 20 of [RFC5245]) runs inside an RTP or RTCP session, so it is already protected if the RTP or RTCP session is also protected (i.e., SRTP/SRTCP). Adding DTLS inside the SRTP/SRTCP session would add overhead, with minimal security benefit.

This usage MUST NOT be used with a STUN URI.

4.4. SIP Keep-Alive Usage

The SIP keep-alive (described in [RFC5626]) runs inside a SIP flow. This flow would be protected if a SIP over DTLS transport mechanism is implemented (such as described in [I-D.jennings-sip-dtls]).

This usage MUST NOT be used with a STUN URI.

4.5. NAT Behavior Discovery Usage

The NAT Behavior Discovery usage is Experimental and to date has never been effectively deployed. Despite this, using DTLS would add the same security properties as for the NAT Discovery Usage (Section 4.1).

The STUN URI can be used to access the NAT Discovery feature of a NAT Behavior Discovery server, but accessing the full features would require definition of a "stun-behaviors:" URI, which is out of scope for this document.

4.6. TURN Usage

TURN [RFC5766] defines three combinations of transports/allocations: UDP/UDP, TCP/UDP and TLS/UDP. This document adds DTLS/UDP as a valid combination. A TURN server using DTLS MUST implement the denial-of-service counter-measure described in Section 4.2.1 of [RFC6347].

[RFC6062] states that TCP allocations cannot be obtained using a UDP association between client and server. The fact that DTLS uses UDP implies that TCP allocations MUST NOT be obtained using a DTLS association between client and server.

By default, TURN over DTLS uses port 5349, the same port as TURN over TLS. However, the SRV procedures can be implemented to use a different port (as described in Section 6 of [RFC5766]). When using SRV records, the service name MUST be set to "turns" and the application name to "udp".

4.6.1. DTLS Support in TURN URIs

This document does not make any changes to the syntax of a TURN URI [RFC7065]. As indicated in Section 3 of [RFC7065], secure transports like TURN over TLS, and now TURN over DTLS, MUST use the "turns" URI scheme. When using the "turns" URI scheme to designate TURN over DTLS, the transport value of the TURN URI, if set, MUST be "udp".

4.6.2. Resolution Mechanism for TURN over DTLS

This document defines a new Straightforward Naming Authority Pointer (S-NAPTR) application protocol tag: "turn.dtls".

The <transport> component, as provisioned or resulting from the parsing of a TURN URI, is passed without modification to the TURN resolution mechanism defined in Section 3 of [RFC5928], but with the following alterations to that algorithm:

- o The acceptable values for transport name are extended with the addition of "dtls".
- o The acceptable values in the ordered list of supported TURN transports is extended with the addition of "Datagram Transport Layer Security (DTLS)".
- o The resolution algorithm check rules list is extended with the addition of the following step:

If <secure> is true and <transport> is defined as "udp" but the list of TURN transports supported by the application does not contain DTLS, then the resolution MUST stop with an error.

- o The 5th rule of the resolution algorithm check rules list is modified to read like this:

If <secure> is true and <transport> is not defined but the list of TURN transports supported by the application does not

contain TLS or DTLS, then the resolution MUST stop with an error.

- o Table 1 is modified to add the following line:

+-----+	+-----+	+-----+
<secure>	<transport>	TURN Transport
+-----+	+-----+	+-----+
true	"udp"	DTLS
+-----+	+-----+	+-----+

- o In step 1 of the resolution algorithm the default port for DTLS is 5349.
- o In step 4 of the resolution algorithm the following is added to the list of conversions between the filtered list of TURN transports supported by the application and application protocol tags:

"turn.dtls" is used if the TURN transport is DTLS.

Note that using the [RFC5928] resolution mechanism does not imply that additional round trips to the DNS server will be needed (e.g., the TURN client will start immediately if the TURN URI contains an IP address).

5. Implementation Status

[[Note to RFC Editor: Please remove this section and the reference to [RFC6982] before publication.]]

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".

5.1. turnuri

Organization: Impedance Mismatch

Name: turnuri 0.5.0 <http://debian.implementers.org/stable/source/turnuri.tar.gz>

Description: A reference implementation of the URI and resolution mechanism defined in this document, RFC 7065 [RFC7065] and RFC 5928 [RFC5928].

Level of maturity: Beta.

Coverage: Fully implements the URIs and resolution mechanism defined in this specification, in RFC 7065 and in RFC 5928.

Licensing: AGPL3

Implementation experience: TBD

Contact: Marc Petit-Huguenin <marc@petit-huguenin.org>.

5.2. rfc5766-turn-server

Organization: This is a public project, the full list of authors and contributors here: <http://turnserver.open-sys.org/downloads/AUTHORS>.

Name: <http://code.google.com/p/rfc5766-turn-server/>

Description: A mature open-source TURN server specs implementation (RFC 5766, RFC 6062, RFC 6156, etc) designed for high-performance applications, especially geared for WebRTC.

Level of maturity: Production level.

Coverage: Fully implements DTLS with TURN protocol.

Licensing: BSD: <http://turnserver.open-sys.org/downloads/LICENSE>

Implementation experience: DTLS is recommended for secure media applications. It has benefits of both UDP and TLS.

Contact: Oleg Moskalenko <mom040267@gmail.com>

6. Security Considerations

STUN over DTLS as a STUN transport does not introduce any specific security considerations beyond those for STUN over TLS detailed in [RFC5389].

The usage of "udp" as a transport parameter with the "stuns" URI scheme does not introduce any specific security issues beyond those discussed in [RFC7064].

TURN over DTLS as a TURN transport does not introduce any specific security considerations beyond those for TURN over TLS detailed in [RFC5766].

The usage of "udp" as a transport parameter with the "turns" URI scheme does not introduce any specific security issues beyond those discussed in [RFC7065].

The new S-NAPTR application protocol tag defined in this document as well as the modifications this document makes to the TURN resolution mechanism described in [RFC5928] do not introduce any additional security considerations beyond those outlined in [RFC5928].

7. IANA Considerations

7.1. S-NAPTR application protocol tag

This specification contains the registration information for one S-NAPTR application protocol tag (in accordance with [RFC3958]).

Application Protocol Tag: turn.dtls

Intended Usage: See Section 4.6.2

Interoperability considerations: N/A

Security considerations: See Section 6

Relevant publications: This document

Contact information: Marc Petit-Huguenin

Author/Change controller: The IESG

7.2. Service Name and Transport Protocol Port Number

This specification contains the registration information for two Service Name and Transport Protocol Port Numbers (in accordance with [RFC6335]).

7.2.1. The stuns Service Name

Service Name: stuns
Transport Protocol(s): UDP
Assignee: IESG
Contact: Marc Petit-Huguenin
Description: STUN over DTLS
Reference: This document
Port Number: 5349

7.2.2. The turns Service Name

Service Name: turns
Transport Protocol(s): UDP
Assignee: IESG
Contact: Marc Petit-Huguenin
Description: TURN over DTLS
Reference: This document
Port Number: 5349

8. Acknowledgements

Thanks to Alan Johnston, Oleg Moskalenko, and Simon Perreault for the comments, suggestions, and questions that helped improve this document.

9. References

9.1. Normative References

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

- [RFC3489] Rosenberg, J., Weinberger, J., Huitema, C., and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", RFC 3489, March 2003.
- [RFC3958] Daigle, L. and A. Newton, "Domain-Based Application Service Location Using SRV RRs and the Dynamic Delegation Discovery Service (DDDS)", RFC 3958, January 2005.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.
- [RFC5626] Jennings, C., Mahy, R., and F. Audet, "Managing Client-Initiated Connections in the Session Initiation Protocol (SIP)", RFC 5626, October 2009.
- [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.
- [RFC5928] Petit-Huguenin, M., "Traversal Using Relays around NAT (TURN) Resolution Mechanism", RFC 5928, August 2010.
- [RFC6062] Perreault, S. and J. Rosenberg, "Traversal Using Relays around NAT (TURN) Extensions for TCP Allocations", RFC 6062, November 2010.
- [RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, August 2011.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.
- [RFC7064] Nandakumar, S., Salgueiro, G., Jones, P., and M. Petit-Huguenin, "URI Scheme for the Session Traversal Utilities for NAT (STUN) Protocol", RFC 7064, November 2013.

- [RFC7065] Petit-Huguenin, M., Nandakumar, S., Salgueiro, G., and P. Jones, "Traversal Using Relays around NAT (TURN) Uniform Resource Identifiers", RFC 7065, November 2013.

9.2. Informative References

- [RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, July 2013.

[I-D.thomson-rtcweb-ice-dtls]

Thomson, M., "Using Datagram Transport Layer Security (DTLS) For Interactivity Connectivity Establishment (ICE) Connectivity Checking: ICE-DTLS", draft-thomson-rtcweb-ice-dtls-00 (work in progress), March 2012.

[I-D.jennings-sip-dtls]

Jennings, C. and N. Modadugu, "Using Interactive Connectivity Establishment (ICE) in Web Real-Time Communications (WebRTC)", draft-jennings-sip-dtls-05 (work in progress), October 2007.

Appendix A. Examples

Table 1 shows how the <secure>, <port> and <transport> components are populated for a TURN URI that uses DTLS as its transport. For all these examples, the <host> component is populated with "example.net".

URI	<secure>	<port>	<transport>
turns:example.net?transport=udp	true		DTLS

Table 1

With the DNS RRs in Figure 1 and an ordered TURN transport list of {DTLS, TLS, TCP, UDP}, the resolution algorithm will convert the TURN URI "turns:example.net" to the ordered list of IP address, port, and protocol tuples in Table 2.

```

example.net.
IN NAPTR 100 10 "" RELAY:turn.udp:turn.dtls "" datagram.example.net.
IN NAPTR 200 10 "" RELAY:turn.tcp:turn.tls "" stream.example.net.

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

stream.example.net.
IN NAPTR 100 10 S RELAY:turn.tcp "" _turn._tcp.example.net.
IN NAPTR 200 10 A RELAY:turn.tls "" a.example.net.

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

_turn._tcp.example.net.
IN SRV 0 0 5000 a.example.net.

_turns._udp.example.net.
IN SRV 0 0 5349 a.example.net.

a.example.net.
IN A 192.0.2.1

```

Figure 1

Order	Protocol	IP address	Port
1	DTLS	192.0.2.1	5349
2	TLS	192.0.2.1	5349

Table 2

Appendix B. Release notes

This section must be removed before publication as an RFC.

B.1. Modifications between petithuguenin-tram-turn-dtls-00 and petithuguenin-tram-stun-dtls-00

- o Add RFC 6982 information for rfc5766-turn-server project.
- o Rename the draft as TURN is now just one of the usages.
- o Remove the references in the abstract to make idnits happy.

- o No longer updates other standard drafts.
- o Rewrite from a STUN over DTLS point of view. The previous text becomes section 4.6.
- o Add IANA request for stuns port.
- o Add acknowledgement section.

Authors' Addresses

Marc Petit-Huguenin
Jive Communications
1275 West 1600 North, Suite 100
Orem, UT 84057
USA

Email: marcph@getjive.com

Gonzalo Salgueiro
Cisco Systems
7200-12 Kit Creek Road
Research Triangle Park, NC 27709
US

Email: gsalguei@cisco.com

BEHAVE
Internet-Draft
Intended status: Standards Track
Expires: April 03, 2014

T. Reddy
Ram. Ravindranath
Muthu. Perumal
Cisco
A. Yegin
Samsung
September 30, 2013

Problems with STUN Authentication for TURN
draft-reddy-behave-turn-auth-04

Abstract

This document discusses some of the issues with STUN authentication for TURN messages.

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 03, 2014.

Copyright Notice

Copyright (c) 2013 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. Notational Conventions	3
3. Scope	3
4. Problems with usage of STUN Authentication	3
5. Security Considerations	4
6. IANA Considerations	5
7. Acknowledgments	5
8. References	5
8.1. Normative References	5
8.2. Informative References	5
Authors' Addresses	6

1. Introduction

The TURN server is a building block to support interactive, real-time communication using audio, video, collaboration, games, etc., between two peer web browsers using the Web Real-Time communication (WebRTC) [I-D.ietf-rtcweb-overview] framework. The use-case explained in "Simple Video Communication Service, enterprise aspects" (Section 3.2.5 of [I-D.ietf-rtcweb-use-cases-and-requirements]) refers to deploying a TURN[RFC5766] server in the DMZ to audit all media sessions from inside an Enterprise premises to any external peer. TURN server could also be deployed for RTP Mobility [I-D.wing-mmusic-ice-mobility] etc.

TURN server is also used in the following scenarios:

- o Users of RTCWEB based web application may use TURN server to hide host candidate addresses from the remote peer for privacy.
- o Enterprise networks deploy firewalls which typically block UDP traffic. When SIP user agents or WebRTC endpoints are deployed behind such firewalls, media cannot be sent over UDP across the firewall, but must be sent using TCP (which causes a different user experience). In such cases a TURN server deployed in the DMZ MAY be used to traverse Firewalls.
- o TURN Server may be used for IPv4-to-IPv6, IPv6-to-IPv6, and IPv6-to-IPv4 relaying [RFC6156].
- o ICE connectivity checks using server-reflexive candidates could fail when the endpoint is behind NAT that performs Address-dependent mapping. In such cases relayed candidate allocated from the TURN server is used for media.

STUN [RFC5389] specifies an authentication mechanism called the long-term credential mechanism. TURN [RFC5766] in section 4 specifies that TURN servers and clients MUST implement this mechanism and the TURN server MUST demand that all requests from the client be authenticated using this mechanism, or that a equally strong or stronger mechanism for client authentication be used.

In the above scenarios RTCWEB based web applications would use Interactive Connectivity Establishment (ICE) protocol [RFC5245] for gathering candidates. ICE agent can use TURN to learn server-reflexive and relayed candidates. If the TURN server requires the TURN request to be authenticated then ICE agent will use the long-term credential mechanism explained in section 10 of [RFC5389] for authentication and message integrity. TURN specification [RFC5766] in section 10 explains the importance of long-term credential mechanism to mitigate various attacks. With proposals like[I-D.thomson-mmusic-rtcweb-bw-consent] that defines a STUN BANDWIDTH attribute for requesting bandwidth allocation at a TURN server, STUN authentication becomes further important to prevent unauthorized users from accessing the TURN server and misuse of credentials could impose significant cost on the victim TURN server.

This note focuses on listing the problems with current STUN authentication for TURN so that it can serve as the basis for stronger authentication mechanisms.

Compared to a Binding request the Allocate request is more likely to be identified by a server administrator as needing client authentication and integrity protection of messages exchanged. Hence, the issues discussed here in STUN authentication are applicable mainly in the context of TURN messages.

2. Notational Conventions

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 note uses terminology defined in [RFC5389], [RFC5766].

3. Scope

This document can be used as an input to design solution(s) to address the problems with the current STUN authentication for TURN messages.

4. Problems with usage of STUN Authentication

1. The long-term credential mechanism in [RFC5389] could use traditional "log-in" username and password given to users which does not change for extended periods of time and uses the key derived from user credentials to generate message integrity for every TURN request/response. An attacker that is capable of eavesdropping on a message exchange between a client and server can determine the password by trying a number of candidate passwords and checking if one of them is correct by calculating the message-integrity of the message using these candidate passwords and comparing with the message integrity value in the MESSAGE-INTEGRITY attribute.
2. When TURN server is deployed in DMZ and requires requests to be authenticated using the long-term credential mechanism in [RFC5389], TURN server needs to be aware of the username and password to validate the message integrity of the requests and to provide message integrity for responses. This results in management overhead on the TURN server.
3. The long-term credential mechanism in [RFC5389] requires that the TURN client must include username value in the USERNAME STUN attribute. An adversary snooping the TURN messages between the TURN client and server can identify the users involved in the call resulting in privacy leakage. In certain scenarios TURN usernames need not be linked to any real usernames given to users as they are just provisioned on a per company basis.
4. An Attacker posing as a TURN server challenges the client to authenticate, learns the USERNAME of the client and later snoops the traffic from the client identifying the user activity resulting in privacy leakage.
5. Hosting multiple realms on a single IP address is challenging with TURN. When a TURN server needs to send the REALM attribute in response to an unauthenticated request, it has no useful information for determining which realm it should send, except the source transport address of the TURN request. Note this is a problem with multi-tenant scenarios only. This may not be a problem when TURN server is located in enterprise premises.
6. In WebRTC the Javascript needs to know the username and password to use in W3C RTCPeerConnection API to access the TURN server. This exposes the user credentials to the Javascript which could be malicious.

5. Security Considerations

This document lists problems with current STUN authentication for TURN so that it can serve as the basis for stronger authentication mechanisms.

6. IANA Considerations

This document does not require any action from IANA.

7. Acknowledgments

Authors would like to thank Dan Wing, Harald Alvestrand, Sandeep Rao, Prashanth Patil, Pal Martinsen and Simon Perreault for their comments and review.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.
- [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.
- [RFC6156] Camarillo, G., Novo, O., and S. Perreault, "Traversal Using Relays around NAT (TURN) Extension for IPv6", RFC 6156, April 2011.

8.2. Informative References

- [I-D.ietf-rtcweb-overview] Alvestrand, H., "Overview: Real Time Protocols for Brower-based Applications", draft-ietf-rtcweb-overview-08 (work in progress), September 2013.
- [I-D.ietf-rtcweb-use-cases-and-requirements] Holmberg, C., Hakansson, S., and G. Eriksson, "Web Real-Time Communication Use-cases and Requirements", draft-ietf-rtcweb-use-cases-and-requirements-11 (work in progress), June 2013.
- [I-D.thomson-mmusic-rtcweb-bw-consent]

Thomson, M. and B. Aboba, "Bandwidth Constraints for Session Traversal Utilities for NAT (STUN)", draft-thomson-mmusic-rtcweb-bw-consent-00 (work in progress), October 2012.

[I-D.wing-mmusic-ice-mobility]

Wing, D., Reddy, T., Patil, P., and P. Martinsen, "Mobility with ICE (MICE)", draft-wing-mmusic-ice-mobility-05 (work in progress), September 2013.

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

[RFC6544] Rosenberg, J., Keranen, A., Lowekamp, B., and A. Roach, "TCP Candidates with Interactive Connectivity Establishment (ICE)", RFC 6544, March 2012.

Authors' Addresses

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

Email: tiredddy@cisco.com

Ram Mohan Ravindranath
Cisco Systems, Inc.
Cessna Business Park, Varthur Hobli
Sarjapur Marathalli Outer Ring Road
Bangalore, Karnataka 560103
India

Email: rmohanr@cisco.com

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

Email: mperumal@cisco.com

Alper Yegin
Samsung
Istanbul
Turkey

Email: alper.yegin@yegin.org

TRAM
Internet-Draft
Intended status: Standards Track
Expires: December 21, 2014

T. Reddy
P. Patil
R. Ravindranath
Cisco
J. Uberti
Google
June 19, 2014

TURN Extension for Third Party Authorization
draft-reddy-tram-turn-third-party-authz-03

Abstract

This document proposes the use of OAuth to obtain and validate ephemeral tokens that can be used for TURN authentication. The usage of ephemeral tokens ensure that access to a TURN server can be controlled even if the tokens are compromised, as is the case in WebRTC where TURN credentials must be specified in Javascript.

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 21, 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	2
2. Terminology	3
3. Solution Overview	3
4. Obtaining a Token Using OAuth	5
4.1. Key Establishment	7
4.1.1. DSKPP	8
4.1.2. HTTP interactions	8
4.1.3. Manual provisioning	9
5. Forming a Request	10
6. STUN Attributes	10
6.1. THIRD-PARTY-AUTHORIZATION	10
6.2. ACCESS-TOKEN	10
7. Receiving a request with ACCESS-TOKEN attribute	12
8. Changes to TURN Client	13
9. Security Considerations	13
10. IANA Considerations	13
11. Acknowledgements	14
12. References	14
12.1. Normative References	14
12.2. Informative References	14
Authors' Addresses	15

1. Introduction

Traversal Using Relay NAT (TURN) TURN [RFC5766] is a protocol that is often used to improve the connectivity of P2P applications. By providing a cloud-based relay service, TURN ensures that a connection can be established even when one or both sides is incapable of a direct P2P connection. However, as a relay service, it imposes a nontrivial cost on the service provider. Therefore, access to a TURN service is almost always access-controlled.

TURN provides a mechanism to control access via "long-term" username/password credentials that are provided as part of the TURN protocol. It is expected that these credentials will be kept secret; if the credentials are discovered, the TURN server could be used by unauthorized users or applications. However, in web applications, ensuring this secrecy is typically impossible. To address this problem and the ones described in [I-D.ietf-tram-auth-problems], this document proposes the use of third party authorization using OAuth for TURN.

To achieve third party authorization, a resource owner e.g. WebRTC server, authorizes a TURN client to access resources on the TURN server.

Using OAuth, a client obtains an ephemeral token from an authorization server e.g. WebRTC server, and the token is presented to the TURN server instead of the traditional mechanism of presenting username/password credentials. The TURN server validates the authenticity of the token and provides required services.

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].

- o WebRTC Server: A web server that supports WebRTC [I-D.ietf-rtcweb-overview].
- o Access Token: OAuth 2.0 access token.
- o mac_key: The session key generated by the authorization server. Note that the lifetime of the session key is equal to the lifetime of the access token.
- o kid: An ephemeral and unique key identifier. The kid also allows the resource server to select the appropriate keying material for decryption.

3. Solution Overview

This specification uses the token type 'Assertion' (aka self-contained token) described in [RFC6819] where all the information necessary to authenticate the validity of the token is contained within the token itself. This approach has the benefit of avoiding a protocol between the TURN server and the authorization server for token validation, thus reducing latency. The exact mechanism used by a client to obtain a token from the OAuth authorization server is outside the scope of this document. For example, a client could make an HTTP request to an authorization server to obtain a token that can be used to avail TURN services. The TURN token is returned in JSON, along with other OAuth Parameters like token type, mac_key, kid, token lifetime etc. The client is oblivious to the content of the token. The token is embedded within a TURN request sent to the TURN server. Once the TURN server has determined the token is valid, TURN services are offered for a determined period of time.

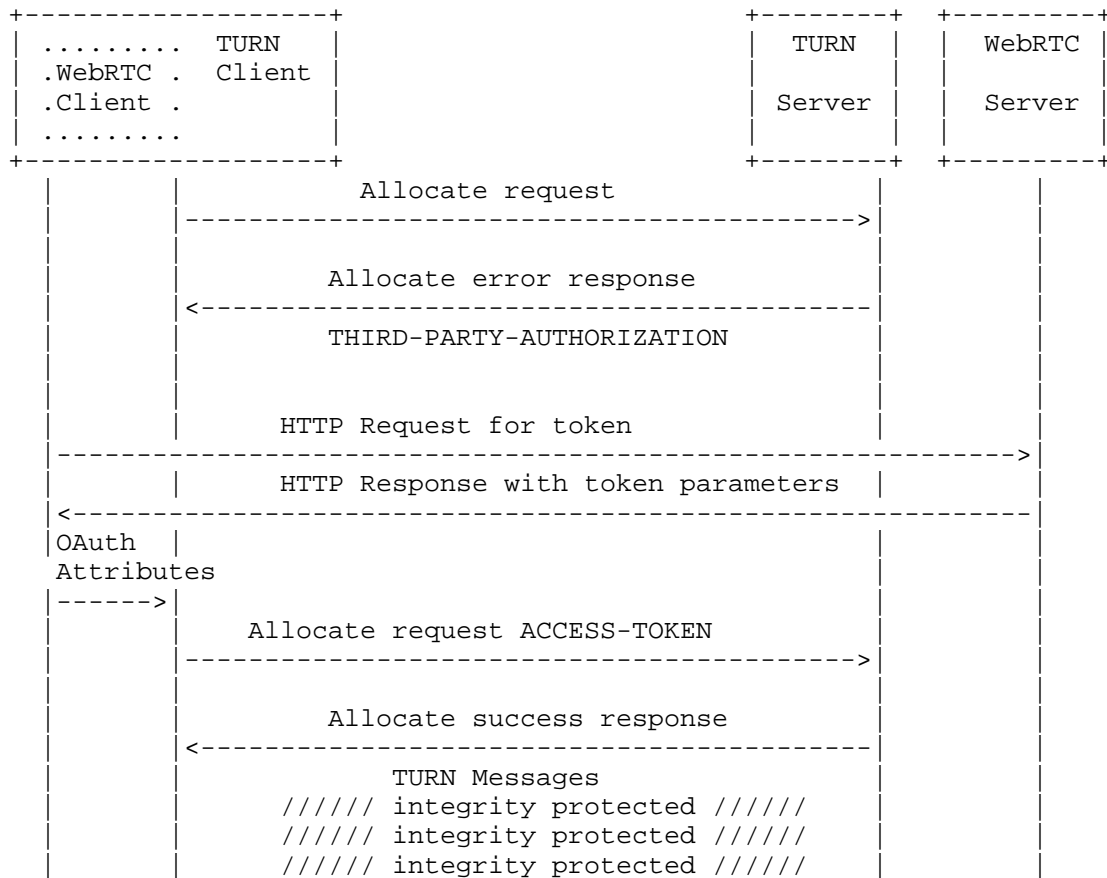


Figure 1: TURN Third Party Authorization

Note : An implementation may choose to contact the WebRTC server to obtain a token even before it makes an allocate request, if it knows the server details before hand. For example, once a client has learnt that a TURN server supports Third Party authorization from a WebRTC server, the client can obtain the token before making subsequent allocate requests.

For example, the client learns the TURN server name "turn1@example.com" from THIRD-PARTY-AUTHORIZATION attribute value and makes the following HTTP request for the access token using transport-layer security (with extra line breaks for display purposes only):

```
POST /o/oauth2/token HTTP/1.1
Audience: turn1@example.com
Content-Type: application/x-www-form-urlencoded
timestamp=1361471629
grant_type=implicit
```

Figure 2: Request

If the client is authorized then the authorization server issues an access token. An example of successful response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "access_token":
  "U2FsdGVkXl8qJK/kkWmRcnfHglrVTJSpsS6yU32kmHmOrfGyI3m1gQj1jRPsr0uBb
  HctuycAgsfRX7nJW2BdukGyKMXSiNGNnBzigkAofP6+Z3vkJlQ5pWbfSRroOkWBn",
  "token_type": "mac",
  "expires_in": 1800,
  "kid": "22BIjxU93h/IgwEb",
  "mac_key": "v51N62OM65kyMvfTI080"
}
```

Figure 3: Response

Access token and other attributes issued by the authorization server are explained in Section 6.2.

4. Obtaining a Token Using OAuth

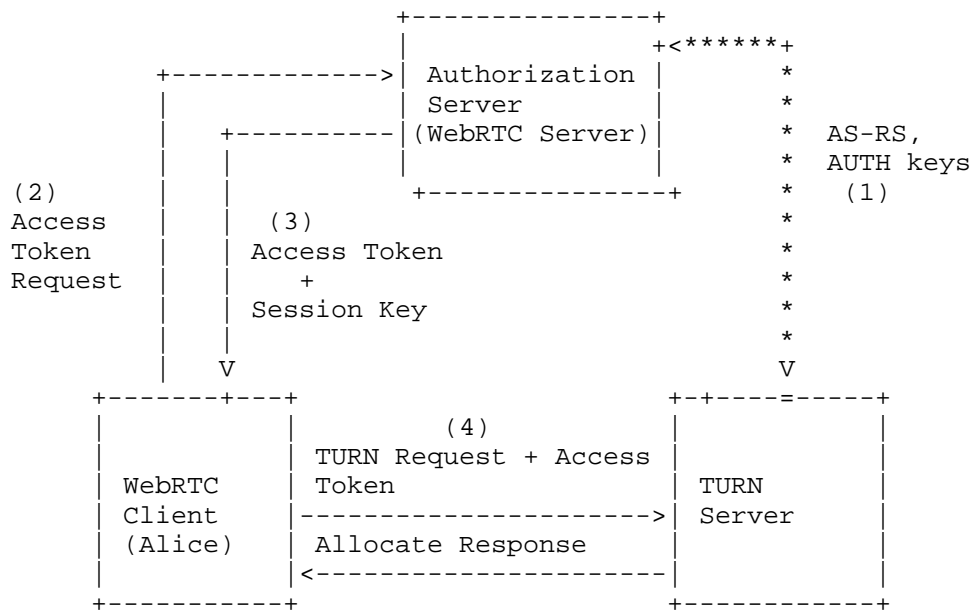
A TURN client should know the authentication capability of the TURN server before deciding to use third party authorization with it. A TURN client initially makes a request without any authorization. If the TURN server supports or mandates third party authorization, it will return an error message indicating support for third party authorization. The TURN server includes an ERROR-CODE attribute with a value of 401 (Unauthorized), a nonce value in a NONCE attribute and a SOFTWARE attribute that gives information about the TURN server's software. The TURN servers also includes additional STUN attribute THIRD-PARTY-AUTHORIZATION signaling the TURN client that the TURN server supports third party authorization.

The following mapping of OAuth concepts to WebRTC is used :

OAuth	WebRTC
Client	WebRTC client
Resource owner	WebRTC server
Authorization server	Authorization server
Resource server	TURN Server

Figure 4: OAuth terminology mapped to WebRTC terminology

Using the OAuth 2.0 authorization framework, a WebRTC client (third-party application) obtains limited access to a TURN (resource server) on behalf of the WebRTC server (resource owner or authorization server). The WebRTC client requests access to resources controlled by the resource owner (WebRTC server) and hosted by the resource server (TURN server). The WebRTC client obtains access token, lifetime, session key (in the mac_key parameter) and key id (kid). The TURN client conveys the access token and other OAuth parameters learnt from the authorization server to the resource server (TURN server). The TURN server obtains the session key from the access token. The TURN server validates the token, computes the message integrity of the request and takes appropriate action i.e permits the TURN client to create allocations. This is shown in an abstract way in Figure 5.



User : Alice

****: Out-of-Band Long-Term Key Establishment

Figure 5: Interactions

OAuth in [RFC6749] defines four grant types. This specification uses the OAuth grant type "Implicit" explained in section 1.3.2 of [RFC6749] where the WebRTC client is issued an access token directly. The scope of the access token explained in section 3.3 of [RFC6749] MUST be TURN.

4.1. Key Establishment

The TURN and authorization servers MUST establish a symmetric key (K), using an out of band mechanism. Symmetric key MUST be chosen to ensure that the size of encrypted token is not large because usage of asymmetric keys will result in large encrypted tokens which may not fit into a single STUN message. The AS-RS, AUTH keys will be derived from K. AS-RS key is used for encrypting the self-contained token and message integrity of the encrypted token is calculated using the AUTH key. The TURN and authorization servers MUST establish the symmetric key over an authenticated secure channel. The establishment of symmetric key is outside the scope of this specification. For example, implementations could use one of the following mechanisms in to establish a symmetric key.

4.1.1.1. DSKPP

The two servers could choose to use Dynamic Symmetric Key Provisioning Protocol (DSKPP) [RFC6063] to establish a symmetric key (K). The encryption and MAC algorithms will be negotiated using the KeyProvClientHello, KeyProvServerHello messages. A unique key identifier (referred to as KeyID) for the symmetric key is generated by the DSKPP server (i.e. Authorization server) and signalled to the DSKPP client (i.e TURN server) which is equivalent to the kid defined in this specification. The AS-RS, AUTH keys would be derived from the symmetric key using (HMAC)-based key derivation function (HKDF) [RFC5869] and the default hash function is SHA-256. For example if the input symmetric key (K) is 32 octets length, encryption algorithm is AES_128_CBC and HMAC algorithm is HMAC-SHA-256-128 then the secondary keys AS-RS, AUTH are generated from the input key K as follows

1. HKDF-Extract(zero, K) -> PRK
2. HKDF-Expand(PRK, zero, 16) -> AS-RS key
3. HKDF-Expand(PRK, zero, 32) -> AUTH key

4.1.1.2. HTTP interactions

The two servers could choose to use REST API to establish a symmetric key. To retrieve a new symmetric key, the TURN server makes an HTTP GET request to the authorization server, specifying TURN as the service to allocate the symmetric keys for, and specifying the name of the TURN server. The response is returned with content-type "application/json", and consists of a JSON object containing the symmetric key.

Request

service - specifies the desired service (turn)
name - TURN server name be associated with the key

example: GET /?service=turn&name=turn1@example.com

Response

key - Long-term key (K)
ttl - the duration for which the key is valid, in seconds.

example:

```
{
  "key" :
  "ESIZRFVmd4iZABEiM0RVZgKn6WjLaTC1FXAghRMVTzkBGNaan496523WIISKerLi",
  "ttl" : 86400,
  "kid" : "22BIjxU93h/IgwEb"
}
```

The AS-RS, AUTH keys are derived from K using HKDF as discussed in Section 4.1.1. Authorization server must also signal a unique key identifier (kid) to the TURN server which will be used to select the appropriate keying material for decryption. The default encryption algorithm to encrypt the self-contained token could be Advanced Encryption Standard (AES) in Cipher Block Chaining (CBC) mode (AES_128_CBC). The default HMAC algorithm to calculate the integrity of the token could be HMAC-SHA-256-128. In this case AS-RS key length must be 128-bit, AUTH key length must be 256-bit (section 2.6 of [RFC4868]).

4.1.3. Manual provisioning

TURN and authorization servers could be manually configured with a symmetric key (K) and kid. The default encryption and HMAC algorithms could be AES_256_CBC, HMAC-SHA-256-128.

Note : The mechanisms specified in Section 4.1.2 Section 4.1.3 are easy to implement and deploy compared to DSKPP but lack encryption and HMAC algorithm agility.

5. Forming a Request

When a TURN server responds that third party authorization is required, a TURN client re-attempts the request, this time including access token and kid values in ACCESS-TOKEN and USERNAME STUN attributes. The TURN client includes a MESSAGE-INTEGRITY attribute as the last attribute in the message over the contents of the TURN message. The HMAC for the MESSAGE-INTEGRITY attribute is computed as described in section 15.4 of [RFC5389] where the mac_key is used as the input key for the HMAC computation. The TURN client and server will use the mac_key to compute the message integrity and doesn't have to perform MD5 hash on the credentials.

6. STUN Attributes

The following new STUN attributes are introduced by this specification to accomplish third party authorization.

6.1. THIRD-PARTY-AUTHORIZATION

This attribute is used by the TURN server to inform the client that it supports third party authorization. This attribute value contains the TURN server name. The TURN server may have tie-up with multiple authorization servers and vice versa, so the client MUST provide the TURN server name to the authorization server so that it can select the appropriate keying material to generate the self-contained token. The THIRD-PARTY-AUTHORIZATION attribute is a comprehension-optional attribute (see Section 15 from [RFC5389]).

6.2. ACCESS-TOKEN

The access token is issued by the authorization server. OAuth does not impose any limitation on the length of the access token but if path MTU is unknown then STUN messages over IPv4 would need to be less than 548 bytes (Section 7.1 of [RFC5389]), access token length needs to be restricted to fit within the maximum STUN message size. Note that the self-contained token is opaque to the client and it MUST NOT examine the ticket. The ACCESS-TOKEN attribute is a comprehension-optional attribute (see Section 15 from [RFC5389]).

The token is structured as follows:


```
struct {  
    opaque {  
        ushort key_length;  
        opaque mac_key[key_length];  
        opaque timestamp[8];  
        long    lifetime;  
    } encrypted_block;  
    opaque mac[mac_length];  
} token;
```

Figure 6: Self-contained token format

The fields are described below:

key_length: Length of the session key. Key length of 160-bits MUST be supported (i.e only 160-bit key is used by HMAC-SHA-1 for message integrity of STUN message). The key length facilitates the hash agility plan discussed in section 16.3 of [RFC5389].

mac_key: The session key generated by the authorization server.

Timestamp: 64-bit unsigned integer field containing a timestamp. The value indicates the time since January 1, 1970, 00:00 UTC, by using a fixed point format. In this format, the integer number of seconds is contained in the first 48 bits of the field, and the remaining 16 bits indicate the number of 1/64K fractions of a second (Native format - Unix).

Lifetime: The lifetime of the access token, in seconds. For example, the value 3600 indicates one hour. The Lifetime value SHOULD be equal to the "expires_in" parameter defined in section 4.2.2 of [RFC6749].

mac: The Hashed Message Authentication Code (HMAC) is calculated with AUTH key over the encrypted portion of the token and the TURN server name (N) conveyed in the THIRD-PARTY-AUTHORIZATION response. Encryption is applied before authentication on the sender side and conversely on the receiver side. The length of the mac field is known to the TURN and authorization server based on the negotiated MAC algorithm.

For example the encryption process can be illustrated as follows. Here C, N denote the ciphertext and TURN server name.

- o C = AES_128_CBC(AS-RS, encrypted_block)
- o mac = HMAC-SHA-256-128(AUTH, C || N)

The token MUST be encoded as defined in Section 4 of [RFC4648] and then encrypted using the symmetric long-term key established between the resource server and the authorization server, as shown in Figure 5 as AS-RS key. HMAC is computed using the encrypted portion of the token and TURN server name to ensure that the client does not use the same token to gain illegal access to other TURN servers provided by the same administrative domain. This attack is possible when multiple TURN servers in a single administrative domain share the same symmetric key with the authorization server. Since the access token is valid for a specific period of time the resource server MUST cache it so that it need not to be provided in every request within an existing allocation. The access token can be re-used for multiple Allocate requests to the same TURN server.

The TURN client MUST include the ACCESS-TOKEN attribute only in Allocate and Refresh requests.

7. Receiving a request with ACCESS-TOKEN attribute

The TURN server, on receiving a request with ACCESS-TOKEN attribute, performs checks listed in section 10.2.2 of [RFC5389] in addition to the following steps to verify that the access token is valid:

- o TURN server selects the keying material based on kid signalled in the USERNAME attribute.
- o It performs the verification of the token message integrity by calculating HMAC over the encrypted portion in the self-contained token and TURN server name using AUTH key and if the resulting value does not match the mac field in the self-contained token then it rejects the request with an error response 401 (Unauthorized).
- o TURN server obtains the mac_key by retrieving the content of the access token (which requires decryption of the self-contained token using the AS-RS key).
- o The TURN server verifies that no replay took place by performing the following check:
 - * The access token is accepted if the timestamp field (TS) in the self-contained token is recent enough to the reception time of the TURN request (RDnew) using the following formula: $\text{Lifetime} + \text{Delta} > \text{abs}(\text{RDnew} - \text{TS})$. The RECOMMENDED value for the allowed Delta is 5 seconds. If the timestamp is NOT within the boundaries then the TURN server discards the request with error response 401 (Unauthorized).

- o The TURN server uses the `mac_key` to compute the message integrity over the request and if the resulting value does not match the contents of the MESSAGE-INTEGRITY attribute then it rejects the request with an error response 401 (Unauthorized).
- o If all the checks pass, the TURN server continues to process the request. Any response generated by the server MUST include the MESSAGE-INTEGRITY attribute, computed using the `mac_key`.

The lifetime provided by the TURN server in the Allocate and Refresh responses MUST be less than or equal to the lifetime of the token.

8. Changes to TURN Client

- o A TURN response is discarded by the client if the value computed for message integrity using `mac_key` does not match the contents of the MESSAGE-INTEGRITY attribute.
- o If the access token expires then the client MUST obtain a new token from the authorization server and use it for new allocations. The client MUST also use the new token to refresh existing allocations. This way client has to maintain only one token per TURN server.

9. Security Considerations

When OAuth is used the interaction between the client and the authorization server requires Transport Layer Security (TLS) with a ciphersuite offering confidentiality protection. The session key MUST NOT be transmitted in clear since this would completely destroy the security benefits of the proposed scheme. If an attacker tries to replay message with ACCESS-TOKEN attribute then the server can detect that the transaction ID as used for an old request and thus prevent the replay attack.

Security considerations discussed in [I-D.ietf-oauth-v2-http-mac] and [RFC5766] are to be taken into account.

10. IANA Considerations

IANA is requested to add the following attributes to the STUN attribute registry [iana-stun],

- o THIRD-PARTY-AUTHORIZATION
- o ACCESS-TOKEN

11. Acknowledgements

Authors would like to thank Dan Wing, Pal Martinsen, Oleg Moskalenko and Charles Eckel for comments and review. The authors would like to give special thanks to Brandon Williams for his help.

12. References

12.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.
- [RFC4868] Kelly, S. and S. Frankel, "Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec", RFC 4868, May 2007.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.
- [RFC6749] Hardt, D., "The OAuth 2.0 Authorization Framework", RFC 6749, October 2012.
- [iana-stun] IANA, , "IANA: STUN Attributes", April 2011, <<http://www.iana.org/assignments/stun-parameters/stun-parameters.xml>>.

12.2. Informative References

- [I-D.ietf-oauth-v2-http-mac] Richer, J., Mills, W., Tschofenig, H., and P. Hunt, "OAuth 2.0 Message Authentication Code (MAC) Tokens", draft-ietf-oauth-v2-http-mac-05 (work in progress), January 2014.
- [I-D.ietf-rtcweb-overview] Alvestrand, H., "Overview: Real Time Protocols for Browser-based Applications", draft-ietf-rtcweb-overview-10 (work in progress), June 2014.
- [I-D.ietf-tram-auth-problems] Reddy, T., R, R., Perumal, M., and A. Yegin, "Problems with STUN long-term Authentication for TURN", draft-ietf-tram-auth-problems-01 (work in progress), May 2014.

- [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.
- [RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, May 2010.
- [RFC6063] Doherty, A., Pei, M., Machani, S., and M. Nystrom, "Dynamic Symmetric Key Provisioning Protocol (DSKPP)", RFC 6063, December 2010.
- [RFC6819] Lodderstedt, T., McGloin, M., and P. Hunt, "OAuth 2.0 Threat Model and Security Considerations", RFC 6819, January 2013.

Authors' Addresses

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

Email: tiredddy@cisco.com

Prashanth Patil
Cisco Systems, Inc.
Bangalore
India

Email: praspati@cisco.com

Ram Mohan Ravindranath
Cisco Systems, Inc.
Cessna Business Park,
Kadabeesanahalli Village, Varthur Hobli,
Sarjapur-Marathahalli Outer Ring Road
Bangalore, Karnataka 560103
India

Email: rmohanr@cisco.com

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

Email: justin@uberti.name

TRAM
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2015

M. Thomson
Mozilla
B. Aboba
Microsoft
A. Johnston
Avaya
O. Moskalkenko
public project
rfc5766-turn-server
July 4, 2014

A Bandwidth Attribute for TURN
draft-thomson-tram-turn-bandwidth-01

Abstract

An attribute is defined for Session Traversal Utilities for NAT (STUN) that allows for declarations of bandwidth limits on the negotiated flow. The application of this attribute is the negotiation of bandwidth between a Traversal Using Relays around NAT (TURN) client and a TURN server.

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 January 5, 2015.

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. The BANDWIDTH Attribute	4
4. Applications	4
4.1. STUN Usage	4
4.2. TURN Usage	5
4.3. ICE Usage	5
5. Bandwidth Measurement Considerations	5
5.1. Rate Enforcement	6
6. Security Considerations	6
7. IANA Considerations	6
8. Implementation Status	6
9. References	7
9.1. Normative References	7
9.2. Informative References	8
Authors' Addresses	8

1. Introduction

This document defines a BANDWIDTH attribute that can be used to request and allocate bandwidth at a Traversal Using Relays around NAT (TURN) relay [RFC5766].

The operator of a TURN server will likely wish to provide fairness between relayed sessions. A TURN server might also wish to limit the use of service to audio-only sessions, or low bandwidth video and audio sessions. In addition, the server may apply rate-limiting policy depending on the credential used for authentication, or the origin of the client. Without the BANDWIDTH attribute, there is no way for a client to indicate the expected bandwidth utilization, or for the server to indicate the maximum bandwidth utilization allowed before rate limiting could be applied.

This attribute is used for indicating a bandwidth limit that is set in policy. The sender is not advised or required to utilize bandwidth up to this limit; limits are usually set well in excess of application needs. Senders also limit their use of bandwidth in reaction to path congestion and "circuit breakers".

Note that the BANDWIDTH attribute was originally in the TURN draft up to version draft-ietf-behave-turn-07 where it was removed as "the requirements for this feature were not clear and it was felt the feature could be easily added later." This draft proposes adding this attribute back into TURN. A related error code 507 "Insufficient Bandwidth Capacity" was also defined in the TURN Internet-Draft, but is not proposed in this draft. This attribute has also been proposed to be used by ICE to provide communication consent [I-D.thomson-mmusic-rtcweb-bw-consent]. No use cases have been identified where bandwidth information is useful for a STUN server which is responding to STUN binding requests.

There have been discussions about what other media-related information could be usefully exchanged between a TURN client and a TURN server. One proposal was for the actual media type (voice, video, data) to be exchanged. Other proposals include more granularity over the bandwidth, including max, min, average, etc. While these could be added, the authors do not feel the use cases for these data have been sufficiently developed yet. Also, this information is known in signaling through the SDP attributes and parameters. In a particular implementation, it could be possible for a signaling-aware entity to share this information with a TURN server in order to apply policy for the media relay.

2. Terminology

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

The terms client, server, and peer are those used for TURN, as defined in [RFC5766].

3. The BANDWIDTH Attribute

The BANDWIDTH attribute (identifier TBD) identifies the rate of packet transmission in kilobits per second that is permitted for a given transport flow. The BANDWIDTH attribute is a comprehension-optional attribute (see Section 15 from [RFC5389]). Figure 1 shows the format of this attribute.

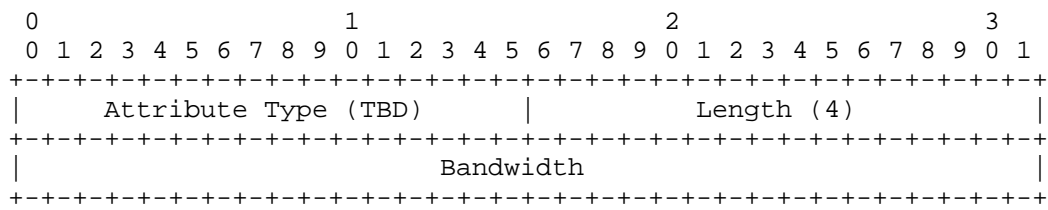


Figure 1: Bandwidth Attribute Format

The value of this attribute is an unsigned integer that represents the maximum bandwidth for the flow in kilobits per second (1 kilobit = 1024 bits). This is the original format of the Bandwidth attribute. This format could include a maximum and average bandwidth, as the BANDWIDTH-USAGE attribute proposed in [I-D.martinsen-tram-discuss].

4. Applications

This section discusses the application of the BANDWIDTH attribute for STUN, TURN, and ICE.

4.1. STUN Usage

Since the bandwidth of a communications session has no bearing on a STUN server that simply responds to binding requests, this attribute MUST NOT be used for client-server STUN requests or responses.

4.2. TURN Usage

This attribute can be useful for communication between a TURN client and a TURN server.

The BANDWIDTH attribute indicates a limit to available bandwidth for TURN [RFC5766] allocation. The bandwidth limit is symmetric; the value covers the bandwidth of data sent from a peer toward the TURN server and the bandwidth of data sent from client to the TURN server.

A BANDWIDTH attribute MAY be present in an Allocate request. This attribute indicates that the given bandwidth is requested. A BANDWIDTH attribute MAY be present in an Allocate response. This attribute in a response indicates the limit that will be applied by the TURN server. The value a TURN server provides could be influenced by the value that a TURN client requests at the discretion of server policy. A client could use this bandwidth limitation of the TURN server in choosing media types or in choosing codecs for a media session.

4.3. ICE Usage

While [I-D.thomson-mmusic-rtcweb-bw-consent] proposed the use of the BANDWIDTH attribute to provide bandwidth consent for ICE, this draft does not do so. This attribute MUST NOT be used with ICE.

5. Bandwidth Measurement Considerations

Allocation messages (Binding and Allocate) sent to and from the TURN server are exempt from any bandwidth measurement accounting.

In calculating bandwidth, the entire IP packet - including the header - is measured. This is identical to the measurement performed by the Real-time Transport Protocol (RTP) [RFC3550]. At a TURN server, bandwidth measurement is performed on the packets arriving at or leaving from the TURN server, prior to the encapsulation that occurs between TURN server and TURN client.

Determining the rate requires that the bits be allocated to specific intervals of time. How bits are allocated MAY vary between implementations.

Measurement of bandwidth is imperfect and inconsistent. Packet jitter can result in fluctuations in received packet rate so that a receiver might see an instantaneous bandwidth that is different to what the sender might have transmitted. Jitter can cause the observed bandwidth of incoming packets to temporarily increase above

the permitted rate. At a minimum, implementations SHOULD allow for short periods of excessive bandwidth to allow for these temporary increases.

5.1. Rate Enforcement

Enforcement of limits by the TURN server SHOULD provide an allowance for application usages that temporarily exceed the limit. For example, assessing observed bandwidth usage as an average over 10 seconds ensures that real-time video does not clip unnecessarily; shorter durations could result in the enforcement affecting valuable intra-frames.

6. Security Considerations

For STUN requests or responses that are not sent using TLS or DTLS transport, the bandwidth information contained in the BANDWIDTH attribute will be available to an eavesdropper who could use it to learn about the nature of a session to be established. For example, they might be able to deduce from the bandwidth requested that the session is likely to be audio only, or audio and video. However, an on-path attacker can likely learn this same information from either the signaling channel or by inspecting the RTP packet headers, which are in the clear for SRTP, or simply by measuring the media bandwidth used.

If a STUN request or response is transported using TCP or UDP, the BANDWIDTH attribute will have integrity protection from the MESSAGE-INTEGRITY attribute if the request is authenticated using the STUN short-term or long-term authentication method. Unauthenticated TCP or UDP requests will not have integrity protection and could be modified by a MitM attacker. The use of DTLS transport [I-D.ietf-tram-stun-dtls] provides integrity protection for the BANDWIDTH attribute regardless of the STUN authentication method used.

7. IANA Considerations

The STUN BANDWIDTH attribute uses the TBD value in the comprehension-optional range. This attribute is registered in the "STUN Attribute" Registry following the procedures of Section 18.2 of [RFC5389].

8. 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".

A multiple realms capable advanced open source 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 STUN BANDWIDTH attribute support, among other TRAM Working Group STUN and TURN extensions.

'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://code.google.com/p/coturn/>

9. References

9.1. Normative References

[I-D.ietf-tram-stun-dtls]

Petit-Huguenin, M. and G. Salgueiro, "Datagram Transport Layer Security (DTLS) as Transport for Session Traversal Utilities for NAT (STUN)", draft-ietf-tram-stun-dtls-05 (work in progress), June 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate

Requirement Levels", BCP 14, RFC 2119, March 1997.

- [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.

9.2. Informative References

- [I-D.martinsen-tram-discuss] Martinsen, P. and H. Wildfeuer, "Differentiated priorities and Status Code-points Using Stun Signalling (DISCUSS)", draft-martinsen-tram-discuss-00 (work in progress), February 2014.
- [I-D.thomson-mmusic-rtcweb-bw-consent] Thomson, M. and B. Aboba, "Bandwidth Constraints for Session Traversal Utilities for NAT (STUN)", draft-thomson-mmusic-rtcweb-bw-consent-00 (work in progress), October 2012.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, July 2013.

Authors' Addresses

Martin Thomson
Mozilla
331 E Evelyn Street
Mountain View, CA 94041
USA

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

Bernard Aboba
Microsoft
One Microsoft Way
Redmond, WA 98052
USA

Email: bernard_aboba@outlook.com

Alan Johnston
Avaya
St. Louis, MO
USA

Email: alan.b.johnston@gmail.com

Oleg Moskalenko
public project rfc5766-turn-server
Walnut Creek, CA
USA

Email: mom040267@gmail.com
URI: <https://code.google.com/p/rfc5766-turn-server/>

