

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: January 21, 2015

A. Johnston
Avaya
J. Uberti
Google
J. Yoakum
K. Singh
Avaya
July 20, 2014

An Origin Attribute for the STUN Protocol
draft-ietf-tram-stun-origin-00

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 January 21, 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 |
| 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 Moskalenko and is freely licensed under the New BSD license. This reference implementation and proof-of-concept provides a clone (a spin-off) of the rfc5766-turn-server project adding Origin-based multiple realms support.

'Coturn' is backward-compatible with rfc5766-turn-server project but the code is more complex and it uses a different (also more complex) database structure. It is the intent to add all IETF TRAM TURN server related capabilities to this project as they mature. 'Coturn' is publicly available and can be found at:
<https://code.google.com/p/coturn/>

6. Acknowledgements

Thanks to John Selbie, Tirumaleswar Reddy, Simon Perreault, Marc Petit-Huguenin, Andy Hutton, and Oleg Moskalenko for their feedback and reviews. Special thanks to Graham Yoakum and Ryan Yoakum of Skobalt LLC and Oleg Moskalenko 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-02 (work in progress), July 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: January 22, 2015

T. Reddy
P. Patil
R. Ravindranath
Cisco
J. Uberti
Google
July 21, 2014

TURN Extension for Third Party Authorization
draft-ietf-tram-turn-third-party-authz-00

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 January 22, 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 | 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.

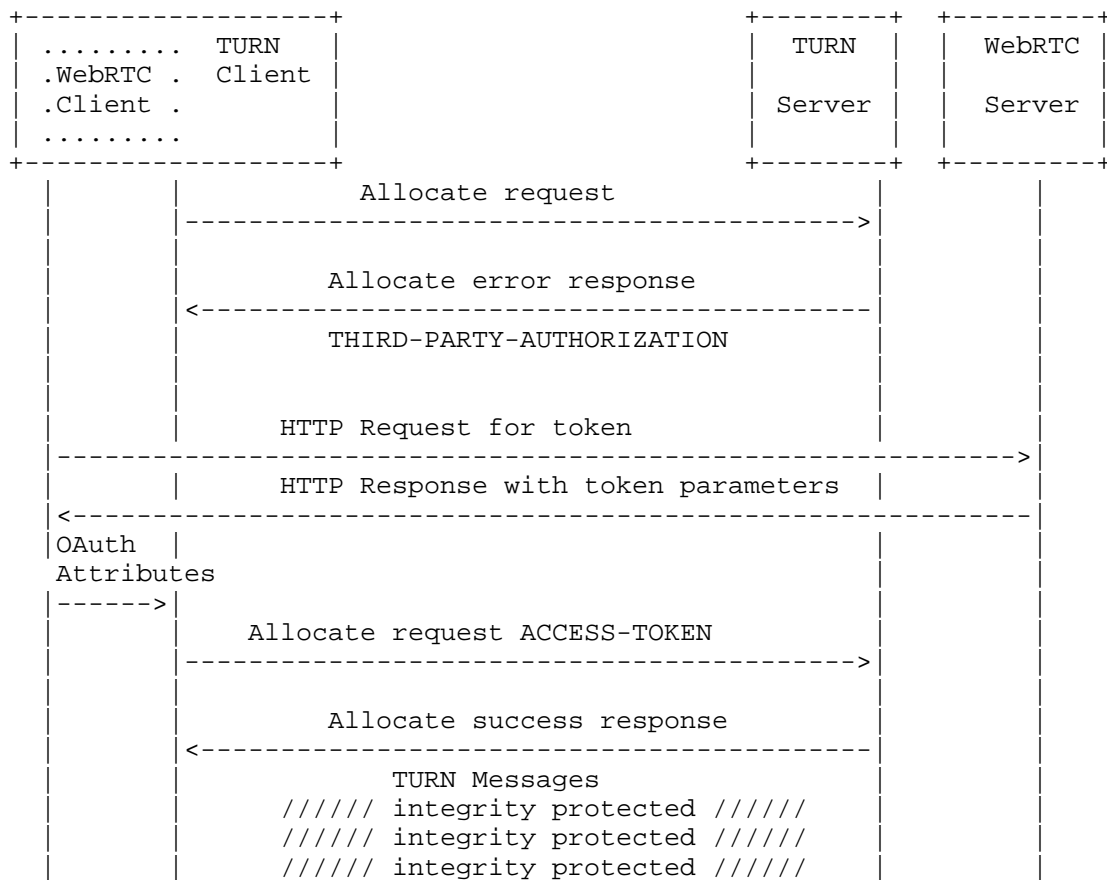


Figure 1: TURN Third Party Authorization

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

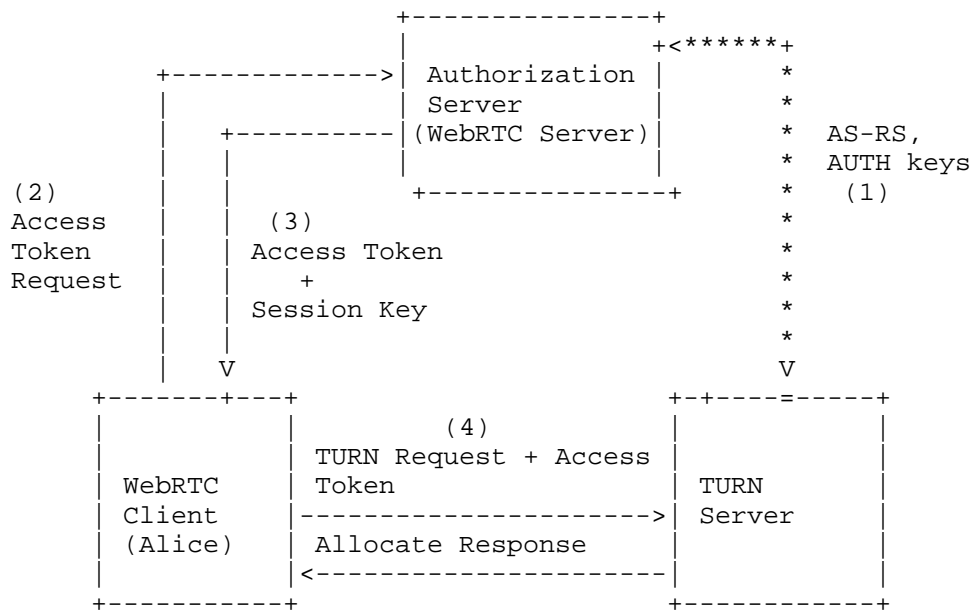
A TURN client should know the authentication capability of the TURN server before deciding to use third party authorization with it. A TURN client initially makes a request without any authorization. If the TURN server supports or mandates third party authorization, it will return an error message indicating 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" :
  "ESIZRFVmd4iZABEiM0RVZgKn6WjLaTC1FXAghRMVTzkBGNaan496523WIIKerLi",
  "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-02 (work in progress), July 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: tireddy@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: September 18, 2014

P. Martinsen
Cisco
J. Uberti
Google
O. Moskalkenko
public project rfc5766-turn-server
March 17, 2014

Single SOcket Dual Allocation with TURN
draft-martinsen-tram-ssoda-00

Abstract

This draft describes a simple method for allocating one IPv4 and one IPv6 relay address from a single ALLOCATE request to the TURN server. This saves local ports on the client, reduces the number of candidates gathered by the client, and reduces the number of messages sent between the client and the TURN server.

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 September 18, 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. Creating an Allocation | 3 |
| 2.1. Sending an Allocate Request | 3 |
| 2.2. Receiving an Allocate Request | 3 |
| 2.3. Receiving an Allocate Success Response | 4 |
| 2.4. Receiving an Allocate Error Response | 5 |
| 3. Refreshing an Allocation | 5 |
| 3.1. Sending a Refresh Request | 5 |
| 3.2. Receiving a Refresh Request | 6 |
| 3.3. CreatePermission | 6 |
| 3.3.1. Sending a CreatePermission Request | 6 |
| 3.3.2. Receiving a CreatePermission Request | 6 |
| 4. Channels | 7 |
| 5. Acknowledgements | 7 |
| 6. Normative References | 7 |
| Authors' Addresses | 7 |

1. Introduction

The main motivation for this draft is to reduce the number of local ports on the client, reduce the number of candidates gathered during the discovery process, and reduce the number of messages that need to be exchanged to allocate the relay addresses needed for ICE.

Reducing the number of local ports is important as it saves resources at three places in the network. First, the number of open ports on the client is reduced, leading to fewer host candidates. Secondly, with fewer local host ports there will be fewer NAT bindings for the NAT to keep track of, and fewer server reflexive candidates. Lastly, with a single 5-tuple in use, it reduces the number of open ports the TURN server needs to open on the interface towards the client (Private side). As ports are a scarce resource (16-bit number) preserving them on the NAT and at the TURN server can make large scale deployments easier.

2. Creating an Allocation

The behavior specified here affects the processing defined in Section 6 of [RFC5766] and Section 4 of [RFC6156].

2.1. Sending an Allocate Request

A client that wishes to obtain one IPv6 and one IPv4 by sending one Allocate request MUST include two REQUESTED-ADDRESS-FAMILY attributes, one for each address family, in the Allocate request that it sends to the TURN server. The order of the REQUESTED-ADDRESS-FAMILY is arbitrary, because the server either understands SSODA (then the order does not matter) or the server does not understand SSODA (then the server behavior is undefined - it may return a 400 error, or it may take the first attribute, or it may take the last attribute). Multiple candidates of the same family are not supported; the client MUST NOT include more than one REQUESTED-ADDRESS-FAMILY attribute for a given address family. The mechanism to formulate an Allocate request is described in Section 6.1 of [RFC5766].

Clients MUST NOT include a REQUESTED-ADDRESS-FAMILY attribute in an Allocate request that contains a RESERVATION-TOKEN or an EVEN-PORT attributes. The SSODA mechanism is not available when using the odd/even port allocation scheme.

2.2. Receiving an Allocate Request

Once a server has verified that the request is authenticated and has not been tampered with, the TURN server processes the Allocate request following the rules in [RFC5766] and [RFC6156].. Only one REQUESTED-ADDRESS-FAMILY attribute with the same family value is allowed in the request. If two attributes with the same family value exist the server MUST return 400 Bad Request error.

If no REQUESTED-ADDRESS-FAMILY attributes are present, the server MUST treat this as if the request contained a single REQUESTED-ADDRESS-FAMILY specifying the IPv4 address family.

If the server can successfully process the request, it allocates a relay address for each of the REQUESTED-ADDRESS-FAMILY attributes present in the Allocate request. The allocated relay addresses are returned in separate XOR-RELAYED-ADDRESS attributes in the Allocate response message. The ordering of the XOR-RELAYED-ADDRESS attributes in the response is arbitrary.

If the server cannot satisfy the request at all, because none of the specified address families are supported, the server MUST return a 440 error code, as indicated in [RFC6156].

If the server cannot satisfy the request at all, because the server could not allocate any of the specified addresses, the server MUST return a 508 (Insufficient Capacity) error code as indicated in [RFC5766].

If some of the requested address could be allocated, but some could not, either because the requested address family is not supported, or the server currently lacks capacity, the server MUST indicate this partial success by returning an Allocate Success Response that contains XOR-RELAYED-ADDRESS attributes for the addresses that were successfully allocated, as well as XOR-RELAYED-ADDRESS with ANY addresses (that is, IPv4 address 0.0.0.0:0 or IPv6 address [::0]:0) corresponding to the address families that could not be allocated. This will notify the client that the desired REQUESTED-ADDRESS-FAMILY was understood, but could not be allocated. A success response with ANY addresses MUST NOT be returned if all allocation requests cannot be satisfied; instead, an error response should be returned, as indicated above.

This somewhat unusual pattern of partial success is used to avoid the need for an additional roundtrip when the client just wants whatever address families the TURN server supports.

Note that while allocating multiple address families at the same time is supported, doing this sequentially is not. The server MUST reject any attempt to "add" an address family to an existing allocation with a 437 (Allocation Mismatch) error code.

[OPEN ISSUE 1: do we need to include REQUESTED-ADDRESS-FAMILY attribute(s) with failed address family (or families) to help the client to recognize whether this is an "old" non-SSODA server or a "new" SSODA-supporting server ?]

[OPEN ISSUE 2: do we have to consider a particular ordering of REQUESTED-ADDRESS-FAMILY and REQUESTED-ADDRESS-FAMILY attributes in the ALLOCATE request and response ? Can attribute ordering provide some benefits in this case ?]

2.3. Receiving an Allocate Success Response

This section describes how the client must react on receiving a response to the dual allocation request. If the client is not using dual allocation, then the behavior is the same as the rules in [RFC5766] and in [RFC6156].

If the client receives an Allocate Success Response containing a non-ANY (ANY as defined above) XOR-RELAYED-ADDRESS attribute for each of the REQUESTED-ADDRESS-FAMILY attributes in the Allocate request sent by the client, the client knows that the TURN server supports multiple address family allocation over a single socket. All relay addresses can now be used by the client.

If the Allocate response contains both usable XOR-RELAYED-ADDRESS attributes as well as ANY XOR-RELAYED-ADDRESS attributes, then the client knows that the TURN server "understands" dual allocation SSODA request, but the server either does not support one of the requested address families or cannot currently allocate an address of that family. The allocated non-ANY address can be used, but the client SHOULD NOT try to allocate any of the unsupported families on a different 5-tuple.

If the Allocate Response contains only one XOR-RELAYED-ADDRESS attribute, then the client knows that the TURN server does not support SSODA. The client can retry the missing address family allocations on new 5-tuples, if desired. Subsequent Allocate requests towards the same TURN server SHOULD NOT include multiple REQUESTED-ADDRESS-FAMILY attributes.

2.4. Receiving an Allocate Error Response

When doing dual allocation, if the client receives an Allocate error response with the 440 (Unsupported Address Family) error code, then the client knows that the TURN server does not support any of the desired address families, or might be a non-SSODA server that misinterpreted the included REQUESTED-ADDRESS-FAMILY attributes in the Allocate request. The client SHOULD retry its IPv4 request on the same 5-tuple, with no REQUESTED-ADDRESS-FAMILY attribute, and MAY retry other address families on different local ports, by sending an Allocate request with only one REQUESTED-ADDRESS-FAMILY attribute.

3. Refreshing an Allocation

The behavior specified here affects the processing defined in Section 7 of [RFC5766] and Section 5 of [RFC6156]. This section MUST only be used if the client has verified that the TURN server supports SSODA during the allocation creation described in Section 2.1. Otherwise, revert back to RFC 5766 or RFC 6156 behaviour.

3.1. Sending a Refresh Request

To perform an allocation refresh, the client generates a Refresh Request as described in Section 7.1 of [RFC5766]. When refreshing a dual allocation, the client SHOULD include one or more REQUESTED-

ADDRESS-FAMILY attributes describing the the family types that should be refreshed; the client MUST only include family types that it previously allocated and has not yet deleted. When refreshing a (single) allocation on a server that does not support SSODA, REQUESTED-ADDRESS-FAMILY should be omitted, for backwards compatibility.

This process can also be used to delete an allocation of a specific address type, by setting the lifetime of that refresh request to 0. It is possible to delete one or more allocations depending on how many REQUESTED-ADDRESS-FAMILY attributes are included. Deleting a single allocation destroys any permissions or channels associated with that particular allocation; it MUST NOT affect any permissions or channels associated with allocations for other address families.

3.2. Receiving a Refresh Request

The server refreshes the allocated address families that match the supplied REQUESTED-ADDRESS-FAMILY values. If any of the values in the request do not match a currently allocated address, the server MUST respond with a 437 (Allocation Mismatch) error. [OPEN ISSUE: discuss whether this is the right error code for the situation] If no REQUESTED-ADDRESS-FAMILY is present, the request should be treated as applying to all current allocations, for backward compatibility.

The server MUST then refresh or delete the specified allocations, and return a Refresh Success Response.

3.3. CreatePermission

The behavior specified here affects the processing defined in Section 9 of [RFC5766] and Section 6 of [RFC6156]

3.3.1. Sending a CreatePermission Request

The client MUST only include XOR-PEER-ADDRESS attributes with addresses that match an address family of one of the currently allocated addresses.

3.3.2. Receiving a CreatePermission Request

If an XOR-PEER-ADDRESS attribute contains an address of an address family different than that any of the relayed transport addresses allocated, the server MUST generate an error response with the 443 (Peer Address Family Mismatch) response code, which is defined in Section 6.2.1 of [RFC6156].

4. Channels

The session channels setup process follows the same rules as in [RFC5766] and in [RFC6156]; the client is allowed to set up multiple channels within the same 5-tuple session. However, when using SSODA and dual allocation, the peer addresses of those channels may be of different families. Thus, a single 5-tuple session may create several IPv4 channels and several IPv6 channels.

5. Acknowledgements

Authors would like to thank Simon Perreault for providing ideas direction and insight.

6. Normative References

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

Authors' Addresses

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

Email: palmarti@cisco.com

Justin Uberti
Google
Kirkland, WA
USA

Email: justin@uberti.name

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

Email: mom040267@gmail.com

URI: <https://code.google.com/p/rfc5766-turn-server/>

TRAM
Internet-Draft
Intended status: Standards Track
Expires: October 24, 2014

P. Patil
T. Reddy
G. Salgueiro
Cisco
M. Petit-Huguenin
Jive Communications
April 22, 2014

Application Layer Protocol Negotiation (ALPN) for Session Traversal
Utilities for NAT (STUN)
draft-patil-tram-alpn-00

Abstract

An Application Layer Protocol Negotiation (ALPN) label for the Session Traversal Utilities for NAT (STUN) protocol is defined in this document to allow the application layer to negotiate STUN within the Transport Layer Security (TLS) connection. The STUN ALPN protocol identifier applies to both TLS and Datagram Transport Layer Security (DTLS).

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 October 24, 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. IANA Considerations | 3 |
| 4. Security Considerations | 4 |
| 5. Acknowledgements | 4 |
| 6. References | 4 |
| 6.1. Normative References | 4 |
| 6.2. Informative References | 5 |
| Authors' Addresses | 5 |

1. Introduction

STUN can be securely transported using TLS-over-TCP (referred to as TLS [RFC5246]), as specified in [RFC5389], or TLS-over-UDP (referred to as DTLS [RFC6347]), as specified in [I-D.petithuguenin-tram-turn-dtls].

ALPN [I-D.ietf-tls-applayerprotoneg] enables an endpoint to positively identify STUN protocol uses in TLS/DTLS and distinguish them from other TLS/DTLS protocols. With ALPN, the client sends the list of supported application protocols as part of the TLS/DTLS ClientHello message. The server chooses a protocol and sends the selected protocol as part of the TLS/DTLS ServerHello message. The application protocol negotiation can thus be accomplished within the TLS/DTLS handshake, without adding network round-trips, and allows the server to associate a different certificate with each application protocol, if desired.

For example, a firewall could block all outgoing traffic except for TCP traffic to specific ports (e.g., 443 for HTTPS). A TURN server listening on its default ports (3478 for TCP/UDP, 5349 for TLS) would not be reachable in this case. However, despite the restrictions imposed by the firewall, the TURN server can still be reached on the allowed HTTPS port if an ALPN STUN protocol identifier is used to establish the STUN application layer protocol as part of the TLS handshake. In this case, the STUN ALPN identifier sent by the client will be used by the server to identify that the client intends to make a TURN request and it must act as a TURN server to relay the traffic to and from the remote peer. Similarly, with Quick UDP Internet Connections (QUIC) [QUIC], a UDP-based transport protocol

that operates under SPDY [I-D.mbelshe-httpbis-spy], a TURN server could be operated on the same ports as that of a SPDY server.

This document defines an entry ("stun") in the "Application Layer Protocol Negotiation (ALPN) Protocol IDs" registry established by [I-D.ietf-tls-applayerprotoneg] to identify the STUN protocol.

[[TODO: In various offline discussions some have expressed a desire to add an additional ALPN protocol identifier for TURN (see IANA Considerations below for example registration). ALPN can be used more granularly to externally identify more of the protocol variants and their different properties (i.e., STUN and TURN over TLS/DTLS). The advantage in dividing it this way is that these different forms can be externally identified (obviously, there isn't any inherent value in the different identifiers from within the TLS handshake). There are two main disadvantages. the first is that this two application protocol approach may make implementations more complicated/confusing. The second is that there may be difficulty in differentiating the two with ALPN when TURN was specifically designed to be able to run on the same port as STUN usage (in section 13 of RFC 5389). Section 4.1.1.2 of RFC 5245 explicitly says that "If the Allocate request is rejected because the server lacks resources to fulfill it, the agent SHOULD instead send a Binding request to obtain a server reflexive candidate." Does that prove there is no need to differentiate TURN and STUN request on UDP/TCP or TLS and now DTLS? Are there sufficiently meaningful differences between the usages to warrant separate STUN and TURN ALPN identifiers?]]

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. IANA Considerations

The following entry is to be added to the "Application Layer Protocol Negotiation (ALPN) Protocol IDs" registry established by [I-D.ietf-tls-applayerprotoneg].

The "stun" label identifies STUN over TLS/DTLS:

Protocol: Session Traversal Utilities for NAT (STUN)

Identification Sequence: 0x73 0x74 0x75 0x6E ("stun")

Specification: This document (RFCXXXX)

[[TODO: Shown only as an example. Remove the below registry entry if open issue above dictates a single STUN ALPN identifier is sufficient.]]

The "turn" label identifies TURN over TLS/DTLS:

Protocol: Traversal Using Relays around NAT (TURN)

Identification Sequence: 0x74 0x75 0x72 0x6E ("turn")

Specification: This document (RFCXXXX)

4. Security Considerations

The ALPN STUN protocol identifier does not introduce any specific security considerations beyond those detailed in the TLS ALPN Extension specification [I-D.ietf-tls-applayerprotoneg]. It also does not impact the security of TLS/DTLS session establishment nor the application data exchange.

5. Acknowledgements

This work benefited from the discussions and invaluable input by the various members of the TRAM working group. These include Simon Perrault, Paul Kyzivat, and Andrew Hutton. Special thanks to Martin Thomson and Oleg Moskalenko for their constructive comments, suggestions, and early reviews that were critical to the formulation and refinement of this document.

6. References

6.1. Normative References

[I-D.ietf-tls-applayerprotoneg]
Friedl, S., Popov, A., Langley, A., and S. Emile,
"Transport Layer Security (TLS) Application Layer Protocol
Negotiation Extension", draft-ietf-tls-applayerprotoneg-05
(work in progress), March 2014.

[I-D.mbelshe-httpbis-spdy]
Belshe, M. and R. Peon, "SPDY Protocol", draft-mbelshe-
httpbis-spdy-00 (work in progress), February 2012.

[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.

- [QUIC] <http://www.ietf.org/proceedings/88/slides/slides-88-tsvarea-10.pdf>, "QUIC Slide Deck at IETF88", .
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [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.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

6.2. Informative References

- [RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.

Authors' Addresses

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

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

Email: gsalguei@cisco.com

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

Email: marcph@getjive.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

AVTCORE
Internet-Draft
Updates: 5764 (if approved)
Intended status: Standards Track
Expires: January 3, 2015

M. Petit-Huguenin
Jive Communications
G. Salgueiro
Cisco Systems
July 2, 2014

Multiplexing Scheme Updates for Secure Real-time Transport Protocol
(SRTP) Extension for Datagram Transport Layer Security (DTLS)
draft-petithuguenin-avtcore-rfc5764-mux-fixes-00

Abstract

This document defines how Datagram Transport Layer Security (DTLS), Real-time Transport Protocol (RTP), Real-time Transport Control Protocol (RTCP), Session Traversal Utilities for NAT (STUN), and Traversal Using Relays around NAT (TURN) packets are multiplexed on a single receiving socket. It overrides the guidance from SRTP Extension for DTLS [RFC5764], which suffered from three issues described and fixed in this document.

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 3, 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 | 2 |
| 1.1. Implicit Allocation of Codepoints for New STUN Methods . | 3 |
| 1.2. Implicit Allocation of New Codepoints for TLS ContentTypes | 4 |
| 1.3. Multiplexing of TURN Channels | 4 |
| 2. Terminology | 5 |
| 3. RFC 5764 Updates | 5 |
| 4. Implementation Status | 6 |
| 5. Security Considerations | 7 |
| 6. IANA Considerations | 7 |
| 6.1. STUN Methods | 7 |
| 6.2. TLS ContentType | 8 |
| 6.3. TURN Channel Numbers | 8 |
| 7. Acknowledgements | 8 |
| 8. References | 9 |
| 8.1. Normative References | 9 |
| 8.2. Informative References | 9 |
| Authors' Addresses | 10 |

1. Introduction

Section 5.1.2 of Secure Real-time Transport Protocol (SRTP) Extension for DTLS [RFC5764] defines a scheme for a Real-time Transport Protocol (RTP) [RFC3550] receiver to demultiplex Datagram Transport Layer Security (DTLS) [RFC6347], Session Traversal Utilities for NAT (STUN) [RFC5389] and Secure Real-time Transport Protocol (SRTP)/Secure Real-time Transport Control Protocol (SRTCP) [RFC3711] packets that are arriving on the RTP port. Unfortunately, this demultiplexing scheme has created three problematic issues:

1. It implicitly allocated codepoints for new STUN methods without an IANA registry reflecting these new allocations.
2. It implicitly allocated codepoints for new Transport Layer Security (TLS) ContentTypes without an IANA registry reflecting these new allocations.
3. It did not take into account the fact that the Traversal Using Relays around NAT (TURN) usage of STUN can create TURN channels that also need to be demultiplexed with the other packet types explicitly mentioned in Section 5.1.2 of RFC 5764.

These flaws in the demultiplexing scheme were unavoidably inherited by other documents, such as [I-D.ietf-mmusic-udptl-dtls] and [I-D.ietf-mmusic-sdp-bundle-negotiation]. These will need to be corrected with the updates this document provides when it become normative.

1.1. Implicit Allocation of Codepoints for New STUN Methods

The demultiplexing scheme in [RFC5764] states that the receiver can identify the packet type by looking at the first byte. If the value of this first byte is 0 or 1, the packet is identified to be STUN. The problem that arises as a result of this implicit allocation is that this restricts the codepoints for STUN methods (as described in Section 18.1 of [RFC5389]) to values between 0x000 and 0x07F, which in turn reduces the number of possible STUN method codepoints assigned by IETF Review (i.e., the range from (0x000 - 0x7FF) from 2048 to only 128 and entirely obliterating those STUN method codepoints assigned by Designated Expert (i.e., the range 0x800 - 0xFFFF). In fact, RFC 5764 implicitly (and needlessly) allocated a very large range of STUN methods, but at a minimum the IANA STUN Methods registry should properly reflect this.

There are only a few STUN method codepoints currently allocated, but this is largely attributed to the fact that STUN did not see much deployment until the development of WebRTC. For this reason, simply marking the implicit allocations made by RFC 5764 in the STUN Method registry may create a shortage of codepoints at a time when interest in STUN and STUN Usages (especially TURN) is growing rapidly. Consequently, this document also changes the RFC 5764 packet identification algorithm to expand the range assigned to the STUN protocol from 0 - 1 to 0 - 19, as the values 2-19 are unused.

In addition to explicitly allocating STUN methods codepoints from 0x500 to 0xFFFF as Reserved values, this document also updates the IANA registry such that the STUN method codepoints assigned via IETF Review are in the 0x000-0x27F range and those assigned via Designated Expert are in the 0x280-0x4FF range. The proposed changes to the STUN Method Registry is:

OLD:

| | |
|--------------|-------------------|
| 0x000-0x7FF | IETF Review |
| 0x800-0xFFFF | Designated Expert |

NEW:

| | |
|--------------|-------------------|
| 0x000-0x27F | IETF Review |
| 0x280-0x4FF | Designated Expert |
| 0x500-0xFFFF | Reserved |

1.2. Implicit Allocation of New Codepoints for TLS ContentTypes

The demultiplexing scheme in [RFC5764] dictates that if the value of the first byte is between 20 and 63 (inclusive), then the packet is identified to be DTLS. The problem that arises is that this restricts the TLS ContentType codepoints (as defined in Section 12 of [RFC5246]) to this range, and by extension implicitly allocates ContentType codepoints 0 to 19 and 64 to 255. Unlike STUN, TLS is a mature protocol that is already well established and widely implemented and thus we expect only relatively few new codepoints to be assigned in the future. With respect to TLS packet identification, this document simply explicitly reserves the codepoints from 0 to 19 and from 64 to 255 so they are not inadvertently assigned in the future.

1.3. Multiplexing of TURN Channels

When used with ICE [RFC5245], an RFC 5764 implementation can receive packets on the same socket from three different paths, as shown in Figure 1:

1. Directly from the source
2. Through a NAT
3. Relayed by a TURN server

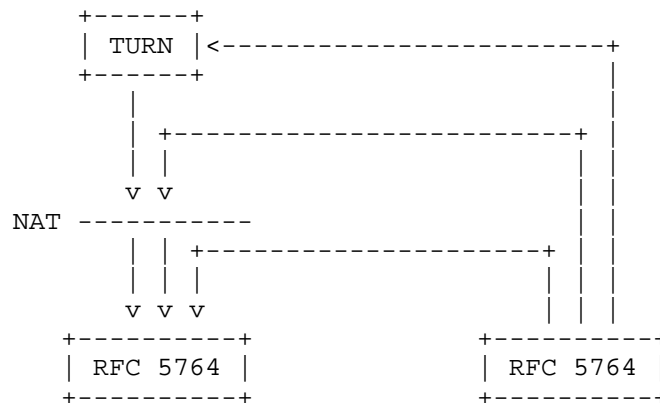


Figure 1: Packet Reception by an RFC 5764 Implementation

Even if the ICE algorithm succeeded in selecting a non-relayed path, it is still possible to receive data from the TURN server. For instance, when ICE is used with aggressive nomination the media path can quickly change until it stabilizes. Also, freeing ICE candidates is optional, so the TURN server can restart forwarding STUN connectivity checks during an ICE restart.

TURN channels are an optimization where data packets are exchanged with a 4-byte prefix, instead of the standard 36-byte STUN overhead (see Section 2.5 of [RFC5766]). The problem is that the RFC 5764 demultiplexing scheme does not define what to do with packets received over a TURN channel since these packets will start with a first byte whose value will be between 64 and 127 (inclusive). If the TURN server was instructed to send data over a TURN channel, then the current RFC 5764 demultiplexing scheme will reject these packets. Current implementations violate RFC 5764 for values 64 to 127 (inclusive) and they instead parse packets with such values as TURN. In order to prevent future documents from assigning values from the unused range to a new protocol, this document modifies the RFC 5764 demultiplexing algorithm to properly account for TURN channels.

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. RFC 5764 Updates

This document updates the text in Section 5.1.2 of [RFC5764] as follows:

OLD TEXT

The process for demultiplexing a packet is as follows. The receiver looks at the first byte of the packet. If the value of this byte is 0 or 1, then the packet is STUN. If the value is in between 128 and 191 (inclusive), then the packet is RTP (or RTCP, if both RTCP and RTP are being multiplexed over the same destination port). If the value is between 20 and 63 (inclusive), the packet is DTLS. This process is summarized in Figure 3.

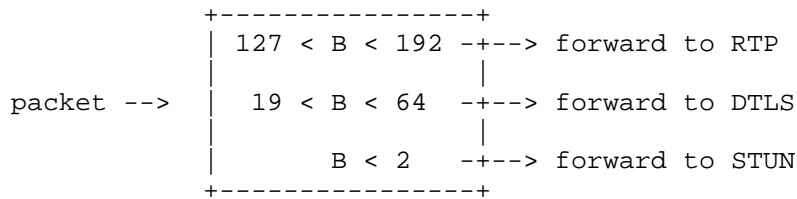


Figure 3: The DTLS-SRTP receiver's packet demultiplexing algorithm.
Here the field B denotes the leading byte of the packet.

END OLD TEXT

NEW TEXT

The process for demultiplexing a packet is as follows. The receiver looks at the first byte of the packet. If the value of this byte is in between 0 and 19 (inclusive), then the packet is STUN. If the value is in between 128 and 191 (inclusive), then the packet is RTP (or RTCP, if both RTCP and RTP are being multiplexed over the same destination port). If the value is between 20 and 63 (inclusive), the packet is DTLS. If the value is between 64 and 127 (inclusive), the packet is TURN Channel. This process is summarized in Figure 3.

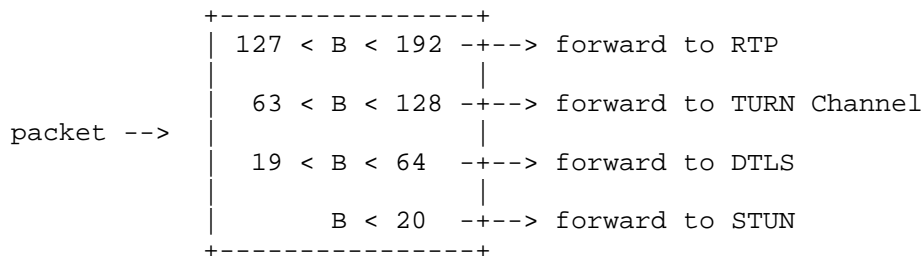


Figure 3: The DTLS-SRTP receiver's packet demultiplexing algorithm.
Here the field B denotes the leading byte of the packet.

END NEW TEXT

[[Note: we may want to use "<=" instead of "<" to make it easier on implementers.]]

4. 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".

Note that there is currently no implementation declared in this section, but the intent is to add RFC 6982 templates here from implementers that support the modifications in this document.

5. Security Considerations

This document simply updates existing IANA registries and does not introduce any specific security considerations beyond those detailed in [RFC5764].

6. IANA Considerations

6.1. STUN Methods

This specification contains the registration information for 2816 STUN Methods codepoints, as explained in Section 1.1 and in accordance with the procedures defined in Section 18.1 of [RFC5389].

Value: 0x500-0xFFF

Name: Reserved

Reference: RFC5764, RFCXXXX

This specification also reassigns the ranges in the STUN Methods Registry as follow:

Range: 0x000-0x27F

Registration Procedures: IETF Review

Range: 0x280-0x4FF

Registration Procedures: Designated Expert

6.2. TLS ContentType

This specification contains the registration information for 212 TLS ContentType codepoints, as explained in Section 1.2 and in accordance with the procedures defined in Section 12 of [RFC5246].

Value: 0-19

Description: Reserved

DTLS-OK: N/A

Reference: RFC5764, RFCXXXX

Value: 64-255

Description: Reserved

DTLS-OK: N/A

Reference: RFC5764, RFCXXXX

6.3. TURN Channel Numbers

This specification contains the registration information for 32768 TURN Channel Numbers codepoints, as explained in Section 1.3 and in accordance with the procedures defined in Section 18 of [RFC5766].

Value: 0x8000-0xFFFF

Name: Reserved

Reference: RFCXXXX

[RFC EDITOR NOTE: Please replace RFCXXXX with the RFC number of this document.]

7. Acknowledgements

The implicit STUN Method codepoint allocations problem was first reported by Martin Thomson in the RTCWEB mailing-list and discussed further with Magnus Westerlund.

Thanks to Simon Perreault and Colton Shields for the comments, suggestions, and questions that helped improve this document.

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.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.
- [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.
- [RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, May 2010.
- [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.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

8.2. Informative References

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

[I-D.ietf-mmusic-udptl-dtls]

Holmberg, C., Sedlacek, I., and G. Salgueiro, "UDP Transport Layer (UDPTL) over Datagram Transport Layer Security (DTLS)", draft-ietf-mmusic-udptl-dtls-10 (work in progress), June 2014.

[I-D.ietf-mmusic-sdp-bundle-negotiation]

Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", draft-ietf-mmusic-sdp-bundle-negotiation-07 (work in progress), April 2014.

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

TRAM
Internet-Draft
Intended status: Standards Track
Expires: November 20, 2014

M. Petit-Huguenin
Jive Communications
O. Johansson
Edvina AB
G. Salgueiro
Cisco Systems
May 19, 2014

Using DNS-based Authentication of Named Entities (DANE) to validate TLS
certificates for the Session Traversal Utilities for NAT (STUN) protocol
draft-petithuguenin-tram-stun-dane-00

Abstract

This document defines how DNS-based Authentication of Named Entities (DANE) can be used to validate TLS certificates with the Session Traversal Utilities for NAT (STUN) 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 November 20, 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 | 2 |
| 3. Operations | 3 |
| 4. Security Considerations | 3 |
| 5. IANA Considerations | 3 |
| 6. References | 3 |
| 6.1. Normative References | 3 |
| 6.2. Informative References | 4 |
| Appendix A. Examples | 4 |
| Authors' Addresses | 6 |

1. Introduction

This document defines how DNS-based Authentication of Named Entities [RFC6698] (DANE) can be used to validate TLS certificates with the Session Traversal Utilities for NAT [RFC5389] (STUN) protocol.

STUN [RFC5389] uses Transport Layer Security [RFC5246] (TLS) as a secure transport and [I-D.ietf-tram-stun-dtls] subsequently added Datagram Transport Layer Security [RFC6347] as a secure transport more suited for the originally intended purpose of STUN, which is to support multimedia sessions. Both transports require to have the certificate presented by the server validated following the rules established by [RFC2818]. Additionally [RFC5389] provides rules on how to use DNS SRV Resource Records [RFC2782] to resolve a domain name to a list of host name for the purpose of load balancing and increased reliability. These rules were subsequently enhanced to support S-NAPTR Resource Records [RFC5928] to add the possibility of selecting the preferred transport and redirect between domains.

DANE [RFC6698] improves the mechanism established by [RFC2818] by enabling the administrators of domain names to specify the keys used the secure servers in their domains. The benefits of this approach encompass increasing flexibility, getting less reliance on trust anchors, enabling Perfect Forward Secrecy (PFS) and much more.

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.

"Source Domain" and "Host Name" are defined in [I-D.ietf-dane-srv].

3. Operations

STUN clients that conform with this specification, and that are using one or more DNS lookups to find the server, and that have established that all DNS Resource Records from the Source Domain to the Host Name are secure according to DNSsec [RFC4033] (i.e. that the AD bit is set in all the DNS answers), and that have selected a secure protocol (e.g. TLS or DTLS) MUST lookup for a TLSA Resource Record for the protocol, port and Host Name selected. If the TLSA Resource Record is secure then the STUN client MUST use it to validate the certificate presented by the STUN server. If there is no TLSA Resource Record or if the Resource Record is not secure, then the client MUST fallback to the validation process defined in [RFC5389] and [I-D.ietf-tram-stun-dtls].

Note that only STUN Usages where the connection is the result of a DNS lookup are to be used with DANE which, for the list of STUN Usages listed in [I-D.ietf-tram-stun-dtls], means these:

NAT Discovery Usage

NAT Behavior Discovery Usage

TURN Usage

4. Security Considerations

Using DANE as (D)TLS certificate validation mechanism does not introduce any specific security considerations beyond those for STUN over TLS detailed in [RFC5389] and those for STUN over DTLS detailed in [I-D.ietf-tram-stun-dtls].

5. IANA Considerations

This document has no IANA actions.

6. References

6.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.
- [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.
- [RFC5928] Petit-Huguenin, M., "Traversal Using Relays around NAT (TURN) Resolution Mechanism", RFC 5928, August 2010.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.
- [RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, August 2012.
- [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-02 (work in progress), May 2014.
- [I-D.ietf-dane-srv]
 Finch, T., Miller, M., and P. Saint-Andre, "Using DNS-Based Authentication of Named Entities (DANE) TLSA records with SRV and MX records.", draft-ietf-dane-srv-05 (work in progress), February 2014.

6.2. Informative References

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

Appendix A. Examples

With the DNS RRs in Figure 1 and an ordered TURN transport list of {DTLS, TLS, TCP, UDP}, a TURN client conform to this specification and using the TURN URI [RFC7065] "turns:example.com" will try first to connect to the TURN server at address 192.0.2.1:5389 using DTLS and using DANE to verify the certificate subsequently presented by the server.

If this connection does not succeed, the client will then try to connect to the TURN server at 192.0.2.1:5000 but will not use DANE for the certificate verification even as a TLSA RR is available, because the DNSsec validation chain is broken in this case.

Using a TURN URI of "turns:example.com;transport=udp" bypasses the NAPTR lookup, but at the expense of preventing the TLS fallback.

```
example.com.
IN NAPTR 100 10 "" RELAY:turn.tls:turn.dtls "" example.net.
IN RRSIG NAPTR ...

_turns._tcp.example.com.
IN SRV 0 0 5000 a.example.net.

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

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

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.
IN RRSIG NAPTR ...

stream.example.net.
IN NAPTR 200 10 S RELAY:turn.tcp "" _turn._tcp.example.net.
IN NAPTR 200 10 S RELAY:turn.tls "" _turns._tcp.example.net.
IN RRSIG NAPTR ...

_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.  
IN RRSIG SRV ...  
  
_turns._tcp.example.net.  
IN SRV      0 0 5000 a.example.net.  
  
a.example.net.  
IN A        192.0.2.1  
IN RRSIG A  ...  
  
_5389._udp.a.example.net.  
IN TLSA ...  
IN RRSIG TLSA ...  
  
_5000._tcp.a.example.net.  
IN TLSA ...  
IN RRSIG TLSA ...
```

Figure 1

Authors' Addresses

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

Email: marcph@getjive.com

Olle E. Johansson
Edvina AB
Runbovaegen 10
Sollentuna SE-192 48
SE

Email: oej@edvina.net

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

Email: gsalguei@cisco.com

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.

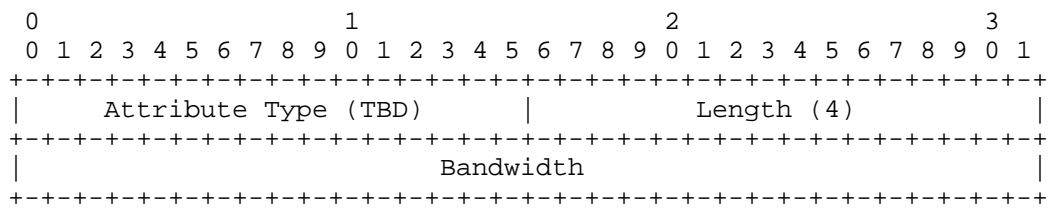


Figure 1: Bandwidth Attribute Format

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 Moskaleiko 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/>

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

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

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

Abstract

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

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 12, 2014.

Copyright Notice

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

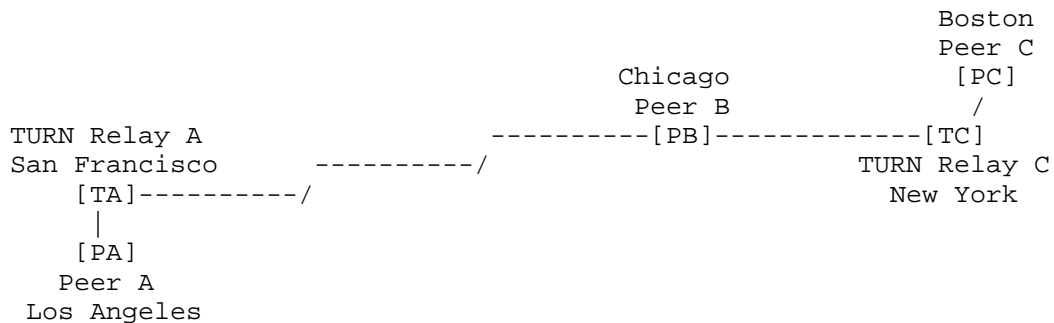
This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|----|
| 1. Introduction | 3 |
| 2. Terminology | 4 |
| 3. Peer-specific Server Redirect Mechanism | 4 |
| 3.1. Attribute Usage | 4 |
| 3.2. Sending a CreatePermission or ChannelBind Request | 6 |
| 3.2.1. The CHECK-ALTERNATE Attribute | 6 |
| 3.2.2. The XOR-OTHER-ADDRESS attribute | 7 |
| 3.3. Receiving a CreatePermission or ChannelBind Request | 7 |
| 3.4. Receiving a CreatePermission or ChannelBind Error Response | 8 |
| 3.5. Receiving a CreatePermission or ChannelBind Success Response | 8 |
| 4. Security Considerations | 9 |
| 4.1. CHECK-ALTERNATE Flood | 9 |
| 4.2. Unsolicited or Invalid ALTERNATE-SERVER | 10 |
| 5. IANA Considerations | 11 |
| 6. References | 11 |
| 6.1. Normative References | 11 |
| 6.2. Informative References | 11 |
| Authors' Addresses | 12 |

1. Introduction

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



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

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

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

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

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

2. Terminology

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

3. Peer-specific Server Redirect Mechanism

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

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

3.1. Attribute Usage

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

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

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

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

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

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

3.2. Sending a CreatePermission or ChannelBind Request

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

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

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

3.2.1. The CHECK-ALTERNATE Attribute

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

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

```

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

```

The Value contains a single 1-bit flag:

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

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

3.2.2. The XOR-OTHER-ADDRESS attribute

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

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

3.3. Receiving a CreatePermission or ChannelBind Request

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

The server checks the following:

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

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

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

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

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

3.4. Receiving a CreatePermission or ChannelBind Error Response

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

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

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

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

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

3.5. Receiving a CreatePermission or ChannelBind Success Response

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

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

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

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

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

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

4. Security Considerations

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

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

4.1. CHECK-ALTERNATE Flood

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

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

4.2. Unsolicited or Invalid ALTERNATE-SERVER

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

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

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

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

5. IANA Considerations

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

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

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

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

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

6. References

6.1. Normative References

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

6.2. Informative References

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

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

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

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

Authors' Addresses

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

Email: brandon.williams@akamai.com

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

Email: tiredy@cisco.com

TRAM
Internet-Draft
Intended status: Standards Track
Expires: December 16, 2014

D. Wing
P. Patil
T. Reddy
P. Martinsen
Cisco
June 14, 2014

Mobility with TURN
draft-wing-tram-turn-mobility-00

Abstract

It is desirable to minimize traffic disruption caused by changing IP address during a mobility event. One mechanism to minimize disruption is to expose a shorter network path to the mobility event so only the local network elements are aware of the changed IP address but the remote peer is unaware of the changed IP address.

This draft provides such an IP address mobility solution using TURN. This is achieved by allowing a client to retain an allocation on the TURN server when the IP address of the client changes.

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 16, 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. Notational Conventions | 3 |
| 3. Mobility using TURN | 3 |
| 3.1. Creating an Allocation | 4 |
| 3.1.1. Sending an Allocate Request | 4 |
| 3.1.2. Receiving an Allocate Request | 4 |
| 3.1.3. Receiving an Allocate Success Response | 5 |
| 3.1.4. Receiving an Allocate Error Response | 5 |
| 3.2. Refreshing an Allocation | 5 |
| 3.2.1. Sending a Refresh Request | 5 |
| 3.2.2. Receiving a Refresh Request | 6 |
| 3.2.3. Receiving a Refresh Response | 6 |
| 3.3. New STUN Attribute MOBILITY-TICKET | 6 |
| 3.4. New STUN Error Response Code | 7 |
| 4. IANA Considerations | 7 |
| 5. Implementation Status | 7 |
| 5.1. open-sys | 7 |
| 6. Security Considerations | 8 |
| 7. Acknowledgements | 8 |
| 8. References | 8 |
| 8.1. Normative References | 8 |
| 8.2. Informative References | 9 |
| Authors' Addresses | 9 |

1. Introduction

When moving between networks, the endpoint's IP address can change or (due to NAT) the endpoint's public IP address can change. Such a change of IP address breaks upper layer protocols such as TCP and RTP. Various techniques exist to prevent this breakage, all tied to making the endpoint's IP address static (e.g., Mobile IP, Proxy Mobile IP, LISP). Other techniques exist, which make the change in IP address agnostic to the upper layer protocol (e.g., SCTP). The mechanism described in this document are in that last category.

A TURN [RFC5766] server relays media packets and is used for a variety of purposes, including overcoming NAT and firewall traversal issues. The existing TURN specification does not permit a TURN client to reuse an allocation across client IP address changes. Due

to this, when the IP address of the client changes, the TURN client has to request for a new allocation, create permissions for the remote peer, create channels etc. In addition to notifying the remote peer of the address change, and punching new pinholes through any NAT/FW that might be on the path.

This specification describes a mechanism to seamlessly reuse allocations across client IP address changes without any of the hassles described above. A critical benefit of this technique is that the remote peer does not have to support mobility, or deal with any of the address changes. The client, that is subject to IP address changes, does all the work. The mobility technique works across and between network types (e.g., between 3G and wired Internet access), so long as the client can still access the TURN server. The technique should also work seamlessly when (D)TLS is used as a transport protocol for STUN. When there is a change in IP address, the client uses (D)TLS Session Resumption without Server-Side State as described in [RFC5077] to resume secure communication with the TURN server, using the changed client IP address.

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].

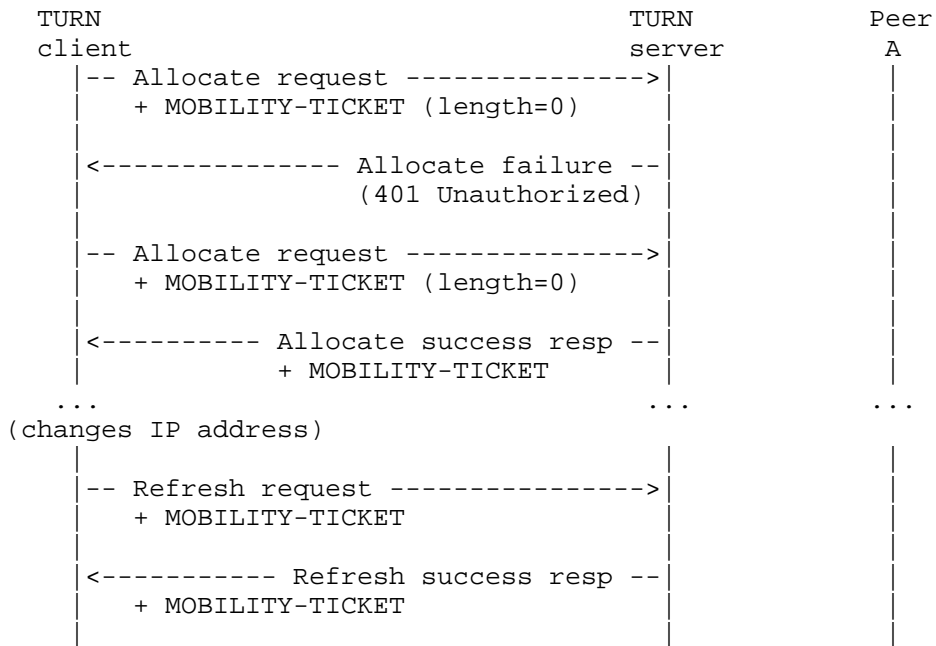
This note uses terminology defined in [RFC5245], and the following additional terminology:

3. Mobility using TURN

To achieve mobility, a TURN client should be able to retain an allocation on the TURN server across changes in the client IP address as a consequence of movement to other networks.

When the client sends the initial Allocate request to the TURN server, it will include a new STUN attribute MOBILITY-TICKET (with zero length value), which indicates that the client is capable of mobility and desires a ticket. The TURN server provisions a ticket that is sent inside the new STUN attribute MOBILITY-TICKET in the Allocate Success response to the client. The ticket will be used by the client when it wants to refresh the allocation but with a new client IP address and port. This ensures that an allocation can only be refreshed by the same client that allocated relayed transport address. When a client's IP address changes due to mobility, it presents the previously obtained ticket in a Refresh Request to the TURN server. If the ticket is found to be valid, the TURN server will retain the same relayed address/port for the new IP address/port

allowing the client to continue using previous channel bindings -- thus, the TURN client does not need to obtain new channel bindings. Any data from external peer will be delivered by the TURN server to this new IP address/port of the client. The TURN client will continue to send application data to its peers using the previously allocated channelBind Requests.



3.1. Creating an Allocation

3.1.1. Sending an Allocate Request

In addition to the process described in Section 6.1 of [RFC5766], the client includes the MOBILITY-TICKET attribute with length 0. This indicates the client is a mobile node and wants a ticket.

3.1.2. Receiving an Allocate Request

In addition to the process described in Section 6.2 of [RFC5766], the server does the following:

If the MOBILITY-TICKET attribute is included, and has length zero, and the TURN session mobility is forbidden by local policy, the server MUST reject the request with the new Mobility Forbidden error code. If the MOBILITY-TICKET attribute is included and has non-zero length then the server MUST generate an error response with an error

code of 400 (Bad Request). Following the rules specified in [RFC5389], if the server does not understand the MOBILITY-TICKET attribute, it ignores the attribute.

If the server can successfully process the request create an allocation, the server replies with a success response that includes a STUN MOBILITY-TICKET attribute. TURN server can store system internal data into the ticket that is encrypted by a key known only to the TURN server and sends the ticket in the STUN MOBILITY-TICKET attribute as part of Allocate success response.

The ticket is opaque to the client, so the structure is not subject to interoperability concerns, and implementations may diverge from this format. TURN Allocation state information is encrypted using 128-bit key for Advance Encryption Standard (AES) and 256-bit key for HMAC-SHA-256 for integrity protection.

3.1.3. Receiving an Allocate Success Response

In addition to the process described in Section 6.3 of [RFC5766], the client will store the MOBILITY-TICKET attribute, if present, from the response. This attribute will be presented by the client to the server during a subsequent Refresh request to aid mobility.

3.1.4. Receiving an Allocate Error Response

If the client receives an Allocate error response with error code TBD (Mobility Forbidden), the error is processed as follows:

- o TBD (Mobility Forbidden): The request is valid, but the server is refusing to perform it, likely due to administrative restrictions. The client considers the current transaction as having failed. The client MAY notify the user or operator and SHOULD NOT retry the same request with this server until it believes the problem has been fixed.

All other error responses must be handled as described in [RFC5766].

3.2. Refreshing an Allocation

3.2.1. Sending a Refresh Request

If a client wants to refresh an existing allocation and update its time-to-expiry or delete an existing allocation, it will send a Refresh Request as described in Section 7.1 of [RFC5766]. If the client wants to retain the existing allocation in case of IP change, it will include the MOBILITY-TICKET attribute received in the Allocate Success response. If a Refresh transaction was previously

made, the MOBILITY-TICKET attribute received in the Refresh Success response of the transaction must be used.

3.2.2. Receiving a Refresh Request

In addition to the process described in Section 7.2 of [RFC5766], the client does the following:

If the STUN MOBILITY-TICKET attribute is included in the Refresh Request then the server will not retrieve the 5-tuple from the packet to identify an associated allocation. Instead TURN server will decrypt the received ticket, verify the ticket's validity and retrieve the 5-tuple allocation using the ticket. If this 5-tuple obtained does not identify an existing allocation then the server MUST reject the request with an error.

If the source IP address and port of the Refresh Request is different from the stored 5-tuple allocation, the TURN server proceeds with MESSAGE-INTEGRITY validation to identify the that it is the same user which had previously created the TURN allocation. If the above checks are not successful then server MUST reject the request with a 441 (Wrong Credentials) error.

If all of the above checks pass, the TURN server understands that the client has moved to a new network and acquired a new IP address. The source IP address of the request could either be the host transport address or server-reflexive transport address. The server then updates it's 5-tuple with the new client IP address and port. TURN server calculates the ticket with the new 5-tuple and sends the new ticket in the STUN MOBILITY-TICKET attribute as part of Refresh Success response.

3.2.3. Receiving a Refresh Response

In addition to the process described in Section 7.3 of [RFC5766], the client will store the MOBILITY-TICKET attribute, if present, from the response. This attribute will be presented by the client to the server during a subsequent Refresh Request to aid mobility.

3.3. New STUN Attribute MOBILITY-TICKET

This attribute is used to retain an Allocation on the TURN server. It is exchanged between the client and server to aid mobility. The value of MOBILITY-TICKET is encrypted and is of variable-length.

3.4. New STUN Error Response Code

This document defines the following new error response code:

Mobility Forbidden: Mobility request was valid but cannot be performed due to administrative or similar restrictions.

4. IANA Considerations

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

- o MOBILITY-TICKET (0x802E, in the comprehension-optional range)

and to add a new STUN error code "Mobility Forbidden" with the value 405 to the STUN Error Codes registry [iana-stun].

5. 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".

5.1. open-sys

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

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.

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

Level of maturity: The Mobile ICE feature implementation can be qualified as "production" - it is well tested and fully implemented, but not widely used, yet..

Coverage: Fully implements MICE with TURN protocol.

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

Implementation experience: MICE implementation is somewhat challenging for a multi-threaded performance-oriented application (because the mobile ticket information must be shared between the threads) but it is doable.

Contact: Oleg Moskalenko <mom040267@gmail.com>.

6. Security Considerations

TURN server MUST use strong encryption and integrity protection for the ticket to prevent an attacker from using a brute force mechanism to obtain the ticket's contents or refreshing allocations.

Security considerations described in [RFC5766] are also applicable to this mechanism.

7. Acknowledgements

Thanks to Alfred Heggstad, Lishitao, Sujing Zhou, Martin Thomson, Emil Ivov and Oleg Moskalenko for review and comments.

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.
- [RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, January 2008.

- [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.

8.2. Informative References

- [RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, July 2013.
- [iana-stun] IANA, , "IANA: STUN Attributes", April 2011, <<http://www.iana.org/assignments/stun-parameters/stun-parameters.xml>>.

Authors' Addresses

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

Email: dwing@cisco.com

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: tireddy@cisco.com

Paal-Erik Martinsen
Cisco Systems, Inc.
Philip Pedersens vei 22
Lysaker, Akershus 1325
Norway

Email: palmarti@cisco.com