An Origin Attribute for the STUN Protocol
draft-johnston-tram-stun-origin-02

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 October 2, 2014.

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

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

Table of Contents

1. Introduction ....................................................... 3
   1.1. Requirements Language ........................................ 5
2. STUN ORIGIN attribute ............................................. 5
   2.1. STUN Usage .................................................. 6
   2.2. TURN Usage .................................................. 6
   2.3. ICE Usage .................................................... 7
3. IANA Considerations ............................................... 7
4. Security Considerations ........................................... 7
5. Acknowledgements .................................................. 8
6. Normative References .............................................. 8
Authors’ Addresses ................................................... 9
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.

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 for logging and other purposes throughout the session even after authentication.

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 the 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
caracter domain name plus 8 characters for the URI scheme plus 5
characters for the port number.

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.

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.

2.3. 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.

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.

If the STUN MESSAGE-INTEGRITY attribute is present, the contents of the ORIGIN attribute are integrity protected. 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. Acknowledgements

Thanks to John Selbie, Tirumaleswar Reddy, and Simon Perreault for
their feedback and reviews.

6. Normative References

[I-D.ietf-rtcweb-overview]
Alvestrand, H., "Overview: Real Time Protocols for Browser-
based Applications", draft-ietf-rtcweb-overview-09 (work
in progress), February 2014.

[I-D.ietf-tram-auth-problems]
Reddy, T., R., Perumal, M., and A. Yegin, "Problems
with STUN Authentication for TURN",
draft-ietf-tram-auth-problems-00 (work in progress),
March 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-01
(work in progress), March 2014.

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

[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., HuiVema, 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


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