An Origin Attribute for the STUN Protocol
draft-ietf-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

Copyright Notice

Table of Contents

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. This
specification also defines an origin for SIP and XMPP users of STUN
and TURN.

An important use case that the STUN Origin helps solve is the
operation of a multi-tenanted TURN server (i.e. a TURN server that
serves multiple, perhaps tens of thousands of different domains).
The problem associated with this use case is described in 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. With the TURN server configured with a mapping between
a domain (conveyed in the Origin) and the realm string (to be used in
the authentication challenge), a single TURN URI could be used across
all domains, and the resulting JavaScript code would be portable.

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

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

1.1. Requirements Language

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
   document are to be interpreted as described in RFC 2119 [RFC2119].

2. STUN ORIGIN attribute

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

   This specification defines a new Attribute to the STUN protocol
   [RFC5389]. The attribute is called ORIGIN and uses the syntax
   defined in Section 15 of [RFC5389]. The number used for this in the
   type field is 0x802F, chosen in the comprehension optional range.
   The value of ORIGIN is a variable-length value. It MUST contain a
   UTF-8 [RFC3629] encoded sequence of characters. Senders MAY include
   multiple ORIGIN attributes in a request, and receivers MUST support
   parsing and receiving multiple ORIGIN attributes. The size of
   ORIGINs included in a STUN message can have a major impact on the
   size of a STUN packet, and could potentially cause UDP fragmentation.
   HTTP origins are less than 267 characters (the maximum 253 character
   domain name plus 8 characters for the URI scheme plus 5 characters
   for the port number).

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

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

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

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

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

2.1. 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. Including the ORIGIN attribute in the
   Send method is NOT RECOMMENDED. 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]. Multiple origins can occur when the same resource is
fetched by multiple origins at the same time (e.g. multiple tabs,
windows, frames, etc.). In the context of WebRTC, it doesn't make
sense for a STUN binding or TURN allocation to be shared across
origins (e.g. Peer Connections). Based on their definitions,
multiple SIP and XMPP origins also do not apply here.

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.

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

The STUN ORIGIN attribute 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.

The STUN ORIGIN attribute also has privacy implications in that the
origin information is shared with a STUN or TURN server which
otherwise might not know this information. This information could be
used to track usage of real-time communication services. A STUN or
TURN server will always know the public IP address of each user, but
the ORIGIN attribute provides more information about which service or
provider is being used. The particular STUN and TURN servers used
are selected by the real-time communications service provider (i.e.
the web provider for WebRTC or the SIP or XMPP service provider). In
addition, they are usually also run by the same provider, or by a
trusted partner, especially for TURN. However, a service or provider
using a public STUN server needs to recognize that the operator of
the public STUN server will learn the identity of the service or
provider through this extension.

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-12
          (work in progress), October 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-05 (work in progress),
          August 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.

     [RFC5122]   Saint-Andre, P., "Internationalized Resource Identifiers
                    (IRIs) and Uniform Resource Identifiers (URIs) for the
                    Extensible Messaging and Presence Protocol (XMPP)",
                    RFC 5122, February 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.

     [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