Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: August 8, 2014

A. Johnston
Avaya
J. Uberti
Google
J. Yoakum
K. Singh
Avaya

February 4, 2014

An Origin Attribute for the STUN Protocol
draft-johnston-tram-stun-origin-01

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.

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.

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.reddy-behave-turn-auth]. 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 URL for each realm. This would require either a separate IP address or port for each realm, and this unique URL 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 URL 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 URL 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 URL to be provisioned per realm. This is not scalable for supporting thousands of realms. Also, this URL 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]. A STUN Attribute type is a hex number in the range 0x0000 - 0xFFFF. The ORIGIN attribute value is 0x802F, chosen in the comprehension optional range.

Editor's Note: At the appropriate time, the authors will work with the chairs of TRAM to follow RFC 4020 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.

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).

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 DNS and 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. A DTLS transport for STUN would provide 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. Normative References

[I-D.ietf-rtcweb-overview]
Alvestrand, H., "Overview: Real Time Protocols for Brower-

based Applications", draft-ietf-rtcweb-overview-08 (work in progress), September 2013.

[I-D.reddy-behave-turn-auth]
Reddy, T., R, R., Perumal, M., and A. Yegin, "Problems with STUN Authentication for TURN", draft-reddy-behave-turn-auth-04 (work in progress), September 2013.

[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.

[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.

[RFC6066] Eastlake, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, January 2011.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, December 2011.

[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