TURN Server REST API

Justin Uberti, Google
Version 0.92 (DRAFT)
April 9, 2013

This document describes a proposed standard API for allocating TURN services via HTTP, via the use of ephemeral credentials. These credentials are vended by a web service, and enforced by the TURN server. This usage of ephemeral credentials ensures that access to the TURN server is controlled even if the credentials cannot be kept secret, as is the case in WebRTC where the TURN credentials need to be specified in JS.

To use this mechanism, the only interaction needed between the web service and the TURN service is to share a secret key.

HTTP Interactions

A typical GET request is made with a username parameter, and optional TTL to control the lifetime of the TURN credentials. The TURN server addresses and credentials are returned as JSON, as well as the actual TTL to use.

Since the returned credentials are time-limited, they will eventually expire. This does not affect existing TURN allocations, since they are tied to a specific 5-tuple, but requests to allocate new TURN ports will fail after the expiry time. This is significant in the case of an ICE restart, where the client will need to allocate a new set of ports. To get a new set of ephemeral credentials, the client can simply re-issue the original HTTP request with the same parameters, which will return the new credentials in its JSON response.

To prevent unauthorized use, the HTTP requests can be ACLed by various means, e.g. IP address (if coming from a server), Origin header, User-Agent header, cookie, API key, etc.

Request

The request includes the following parameters:

- **service**: specifies the desired service (turn)
- **username**: an optional user identifier for the request
- **ttl**: an optional TTL request for the lifetime of the credentials, in seconds

Example:

```
GET /?service=turn&username=<optional_username>&ttl=<optional_requested_ttl>
```

Response

The response includes the following parameters:

- **username**: the TURN username to use, which is a combination of the username parameter from the request, with a timestamp in time_t format, colon-separated (note that in order to support Chrome 27 and earlier, another delimited (e.g. '+') should be used). If username was not specified in the request, just the timestamp is present in the response.
- **password**: the TURN password to use; this value is computed from the a secret key shared with the TURN server and the returned username value, by performing base64(hmac(secret key, returned username)). HMAC-SHA1 is one HMAC that can be used, but any algorithm is acceptable, as long as both the web server and TURN server use the same one.
- **ttl**: the duration for which the username and password are valid, in seconds. This number will be less than or equal to the requested TTL.
- **uris**: an array of TURN URIs, in the form of http://tools.ietf.org/html/draft-petithuguenin-behave-turn-uris-03. This is used to indicate the different addresses and/or protocols that can be used to reach the TURN server.

Example:

```
 {
  "username" : "foo:12334939",
  "password" : "adfsaflsjflds",
  "ttl" : 86400,
  "uris" : [
    "turn:1.2.3.4:9991?transport=udp",
    "turn:1.2.3.4:9992?transport=tcp",
    "turns:1.2.3.4:443?transport=tcp"
  ]
}
```

WebRTC Interactions

The returned JSON is parsed and supplied when creating a PeerConnection, to tell it how to access the TURN server. Note that for Chrome 27 and earlier, the username must be included into the URI (using URL encoding), e.g. "turn:foo@1.2.3.4:9991?transport=udp".

```
var config = { "iceServers": [] };
for (var i = 0; i < response.uris.length; ++i) {
  var uri = response.uris[i];
  var iceServer = {
    "username":response.username,
    "credential":response.password,
    "uri":uri
```

```
  };
  config.iceServers.push(iceServer);
}
var pc = new PeerConnection(config);
```

When the credentials are updated (i.e. because they are about to expire), they can be supplied to the existing PeerConnection via the updateIce method. This update must not affect existing TURN allocations, because TURN requires that the username stay constant for an allocation, but can be used for any new allocations.

TURN Interactions

Client

WebRTC's TURN request uses the "username" value for its USERNAME attribute, and the "password" value for the MESSAGE-INTEGRITY hash.

Server

When processing ALLOCATE requests, the TURN server will split the USERNAME attribute into its username and timestamp components, verify that the timestamp is within the configured TTL, and optionally verify that the username corresponds to a currently active user of the service (e.g. is currently logged in). If either verification fails, it SHOULD reject the request with a 401 (Unauthorized) error.
For all other requests, the TURN server merely verifies that the USERNAME matches the USERNAME that was used in the ALLOCATE (since it must remain constant).

As in RFC 5766, the TURN server MUST verify the MESSAGE-INTEGRITY using the shared secret key to validate that it matches the supplied USERNAME.

Discussion

Revocation

In the system as described here, revoking credentials is not possible. The assumption is that TURN services are of low enough value that waiting for the timeout to expire is a valid approach for dealing with possibly-compromised credentials.
In extreme abuse cases, specific TURN server blacklists of username:timestamp values can be used to stop abuse of a specific credential.

Shared secret compromise

If the shared secret between the TURN server and the Web server is compromised, a new

shared secret can be installed. To facilitate the rollover, the TURN server SHOULD be able to validate incoming MESSAGE-INTEGRITY tokens based on at least 2 shared secrets at any time.