Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 09, 2014

J. Uberti
Google

July 08, 2013

A REST API For Access To TURN Services
draft-uberti-rtcweb-turn-rest-00

Abstract

This document describes a proposed standard REST API for obtaining access to TURN services via ephemeral (i.e. time-limited) credentials. These credentials are vended by a web service over HTTP, and then supplied to and checked by a TURN server using the standard TURN protocol. The usage of ephemeral credentials ensures that access to the TURN server can be controlled even if the credentials can be discovered by the user, 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 09, 2014.

Table of Contents

1. Introduction

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, this document proposes an API that can be used to retrieve ephemeral TURN credentials from a web service. For simplicity, the design has been kept intentionally stateless; to use this mechanism, the only interaction needed between the web service and the TURN service is to share a secret key.

2. HTTP Interactions

To retrieve a new set of credentials, the client makes a HTTP POST request, specifying TURN as the service to allocate credentials for, and optionally specifying a user id parameter. The purpose of the user id parameter is to simplify debugging on the TURN server, as well as provide the ability to control the number of credentials handed out for a specific user, if desired. The TURN credentials and their lifetime are returned as JSON, along with URIs that indicate how to connect to the server using the TURN protocol. To avoid the need for state passing between the web service and TURN server, the returned credentials consist of a TURN username that encodes all the necessary state (expiry time and application user id), and a TURN password that is a digest of this state, signed with the shared secret key. Since the returned credentials are ephemeral, they will eventually expire. This does not affect existing TURN allocations, as 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 candidates, including TURN candidates. 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, login cookie, API key, etc.

2.1. Request

The request includes the following parameters, specified in the URL:

- service: specifies the desired service (turn)
- username: an optional user id to be associated with the credentials
- key: if an API key is used for authentication, the API key

Example:

```
POST /?service=turn&username=fred
```

2.2. Response

The response is returned with content-type "application/json", and consists of a JSON object with the following parameters:

- username: the TURN username to use, which is a colon-delimited combination of the expiration timestamp and the username parameter from the request (if specified). The timestamp is intended to be opaque to the web application, so its format is arbitrary, but for simplicity, use of UNIX timestamps is recommended.

- 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 algorithm that can be used, but any algorithm that incorporates a shared secret is acceptable, as long as both the web server and TURN server use the same algorithm and secret.

- ttl: the duration for which the username and password are valid, in seconds. A value of one day (86400 seconds) is recommended.

- uris: an array of TURN URIs, in the form specified in [I-D.petithuguenin-behave-turn-uris]. This is used to indicate the different addresses and/or protocols that can be used to reach the TURN server.

Example:

```
 {
  "username" : "12334939:fred",
  "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"
  ]
}
```

3. WebRTC Interactions

The returned JSON is parsed and supplied when creating a WebRTC RTCPeerConnection, to tell it how to access the TURN server.

Example:

```
var iceServer = {
  "username": response.username,
  "credential": response.password,
  "uris": response.uris
};
var config = {"iceServers": [iceServer]};
var pc = new RTCPeerConnection(config);
```

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

[TODO: make sure this behavior is specified in the W3C API spec]

4. TURN Interactions

4.1. Client

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

4.2. Server

When processing ALLOCATE requests, the TURN server will split the USERNAME attribute into its timestamp and user id components, and verify that the timestamp, which indicates when the credentials expire, has not yet been reached. If this verification fails, it SHOULD reject the request with a 401 (Unauthorized) error.

If desired, the TURN server can optionally verify that the parsed user id value corresponds to a currently valid user of an external service (e.g. is currently logged in to the web app that is making use of TURN). This requires proprietary communication between the TURN server and external service on each ALLOCATE request, so this usage is not recommended for typical applications. If this external verification fails, it SHOULD reject the request with a 401 (Unauthorized) error.

For non-ALLOCATE 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 password associated with the supplied USERNAME. For the usage outlined in this document, the password will always be constructed using the supplied username and the shared secret as indicated in the "HTTP Interactions" section above.

5. Implementation Notes

5.1. Revocation

In the system as described here, revoking specific 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, TURN server blacklists of timestamp+username values can be supplied by an administrator to stop abuse of specific credential sets.

5.2. Key Rotation

As indicated in [RFC2104], periodic rotation of the shared secret to protect against key compromise is RECOMMENDED. 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.

6. Security Considerations

Because the USERNAME values in a TURN ALLOCATE request are typically visible to eavesdroppers, inclusion of an externally identifying user id, such as a login name, may allow a passive attacker to determine the identities of the parties in a conversation. To prevent this problem, use of opaque user id values is recommended.

7. IANA Considerations

None.

8. Acknowledgements

Harald Alvestrand, Alfred Godoy, and Philipp Hancke provided key input on the initial design. Dave Cridland, Cullen Jennings, Oleg Moskalenko, and Matthew Robertson pointed out several errors and omissions.

9. References

9.1. Normative References

[I-D.petithuguenin-behave-turn-uris]
Petit-Huguenin, M., Nandakumar, S., Salgueiro, G., and P. Jones, "Traversal Using Relays around NAT (TURN) Uniform Resource Identifiers", draft-petithuguenin-behave-turn-uris-03 (work in progress), January 2013.

[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

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

Author's Address

Justin Uberti
Google
747 6th St S
Kirkland, WA 98033
USA

Email: justin@uberti.name