A Usage of Resource Location and Discovery (RELOAD) for Public Switched Telephone Network (PSTN) Verification
draft-rosenberg-dispatch-vipr-reload-usage-01

Abstract

Verification Involving PSTN Reachability (ViPR) is a technique for inter-domain SIP federation. ViPR makes use of the RELOAD protocol to store unverified mappings from phone numbers to RELOAD nodes, with whom a validation process can be run. This document defines the usage of RELOAD for this purpose.

Legal

This document and the information contained therein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION THEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.
Table of Contents

1. Introduction ................................................. 4
2. ViPR Usage ................................................. 4
3. PeerID Shim ................................................. 6
4. Security Considerations ................................. 7
5. IANA Considerations ........................................ 7
6. References ................................................. 7
   6.1. Normative References ............................... 7
   6.2. Informative References ............................. 7
Authors’ Addresses ............................................. 7
1. Introduction

This document relies heavily on the concepts and terminology defined in [I-D.rosenberg-dispatch-vipr-overview] and will not make sense if you have not read that document first. As it defines a usage for RELOAD [I-D.ietf-p2psip-base], it assumes the reader is also familiar with that specification. The same DHT can also be used for a RELOAD SIP usage [I-D.ietf-p2psip-sip].

2. ViPR Usage

The ViPR usage defines details for how the DHT is used for ViPR operations.

The ViPR usage defines kind-id 0x00000001. This kind-id is a dictionary entry. Its resource ID is defined through a transformation which takes an E.164 based number, and computes a resourceID as the least significant 128 bits of the SHA1 hash of the following string: Cat(CHOICE(null, "COPY", "COPY2"), number) That is, the resource ID is the hash of a string which is the concatenation of the number, prefixed with nothing, or the words "COPY1" or "COPY2".

For example, for number +17327662496:

ResourceID = least128(SHA1("+17327662496"))

or

ResourceID = least128(SHA1("COPY1+7327662496"))

or

ResourceID = least128(SHA1("COPY2+7327662496"))

The object stored at this resource ID is a dictionary entry, which uses data model 0x??. Object = {key,value} Here, the key is formed by taking the peerID of the storing node in hex format, without the "0x", appending a "+", followed by the VserviceID in hex format, without the "0x". For example, if a peer with peerID 0x8h60f5gab753037g64ab6c53947fd532 receives a Publish with a Vservice of 0x7ggb6a7036478351
The resulting key is:

8h60f5gab753037g64ab6c53947fd532+7ggb6a7036478351

Both parts of this key are important. Using the peerID of the node performing the store basically segments the keyspace of the dictionary so that no two peers every store using the same key. Indeed, the responsible node will verify the signature over the stored data and check the peerID against the value of the key, to make sure that a conflict does not take place. The usage of the Vservice allows for a single ViPR server to service multiple clusters, and to ensure that numbers published by one cluster (using one Vservice ID) do not clobber or step on numbers published by another cluster (using a different Vservice ID). The responsible node does not verify or check the VserviceID.

When a node receives a Store operation for this usage, the data itself has a signature. The node responsible for storing the data must verify this signature; the certificate will always be included in the data and indicate which peerID is used. The responsible node must check that this peerID is included in the cert. If the signature verifies, the responsible node checks that the data model is a dictionary entry. The key must meet the format above. The responsible node must check that it is a 32 character sequence of numbers and letters a-f, followed by a +, followed by a 16 character sequence of numbers and letters a-f. If this checks, the key is split in half along the plus. The first 32 characters are considered a hex value and compared with the peerID used for the signature. If they match, it is good. Otherwise the Store is rejected. If they did match, next the responsible node checks the value. It must be a TLV as defined below, and must contain a peerID. The peerID must match that used for the signature. If they don’t match, the Store is rejected. If they do match, the next step is a quota check.

For each peer that the responsible node is storing data for, it must maintain a count of the number of unique dictionary entries being stored for that peerID. For each resourceID, each key constitutes a unique dictionary entry. So if a peer is storing 5 resource IDs, and at each of those 5, there are two keys whose first 32 bits correspond to a particular peerID, it means this node is currently storing 10 unique dictionary entries for that peerID.

It takes the StorageQuota configuration parameter for this DHT, which measures the amount of numbers a particular node can store. That value is multiplied by nine (a 3x factor to account for the application-layer copies (COPY1 and COPY2), and another 3x factor for replicas). Then, an addition 3x factor is added for rounding to make sure that the probability is low that a rejection occurs due to
imperfect distribution of resourceIDs across the ring. (Open Issue: need to adjust this multiplier - basically birthday problem!) and then divided by the fraction of the hashspace owned by this ViPR server. If the result is less than one, it is rounded up to two. This is the max number of unique entries that can be stored for this storing peer ID. If the ViPR server is not yet storing this many entries for that peer ID, the store is allowed.

The method for merging data after a partition follows the normal RELOAD rules around temporal ordering.

3. PeerID Shim

Because the ViPR implementation of RELOAD protocol makes use of the concept of multiple peerID on the same physical box, utilizing a single cert, the TLS handshakes alone are not sufficient to determine the entity on both sides of the TLS connection. As such, we will have a small "shim" type of protocol, which runs after TLS, but is not formally part of RELOAD.

When a node initiates a TLS connection towards another node, after the TLS completes, it sends this message. The message contains the peerID associated with this connection. The recipient gets this, and sends back a similar message, containing its peerID. Both sides will verify that, the peerID sent by the other side, are amongst the peerIDs listed in the certificate. The connections are then stored in the connection tables, indexed by this peerID.

Furthermore, if, after this exchange, a node determines that it already has a connection in its connection table with that peerID on the far side, the older connection is closed. This is actually a critical security function! Without this, a user could clone ViPR servers utilizing the same certs, and each one can join the network.

Finally, once the exchange has taken place, the node compares the peerID from its peers with the current set of blacklisted peerID from the ACL that is distributed through the DHT. If the remote peerID appears on the list, the node closes the TCP/TLS connection immediately.

The reason we are using a non-reload message for this, is that we need to be 100% sure that this never propagates. It is strictly over a single connection and should never be routed. Indeed, had we not had this idea of multiple peerID in a single cert, this would have effectively been accomplished through TLS. Alternatively, there is a TLS command for telling the other side who I expect them to be; however this is not implemented in older versions of OpenSSL, and so
our shim forms an alternative to that which can be run on top of OpenSSL.

4. Security Considerations

TBD

5. IANA Considerations

TBD. Need to register items in IANA registries created by RELOAD.

6. References

6.1. Normative References

[I-D.ietf-p2psip-base]

[I-D.rosenberg-dispatch-vipr-overview]

6.2. Informative References

[I-D.ietf-p2psip-sip]

Authors' Addresses

Jonathan Rosenberg
jdrosen.net
Monmouth, NJ
US

Email: jdrosen@jdrosen.net
URI: http://www.jdrosen.net
Cullen Jennings
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Phone: +1 408 421-9990
Email: fluffy@cisco.com