LISP Working Group
Internet-Draft
Intended status: Experimental
Expires: September 24, 2015

V. Ermagan
Cisco Systems
A. Rodriguez-Natal
F. Coras
A. Cabellos-Aparicio
Technical University of Catalonia
F. Maino
Cisco Systems

March 23, 2015

YANG model for LISP
draft-ermagan-lisp-yang-00

Abstract

This document describes a YANG data model to use with the Locator/ID Separation Protocol (LISP).

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 September 24, 2015.

Copyright Notice

Table of Contents

1. Introduction

The Locator/ID Separation Protocol (LISP) defines several network elements subject to be configured. This document presents a YANG data model to define the basic configuration of all major LISP elements.

2. Tree view

The tree view of the model is depicted below. The following notation is used to describe elements within the tree. For readability purposes, the tree depth is limited to 9 levels.

Each node is printed as:

<status> <flags> <name> <opts> <type> <if-features>

<status> is one of:
+ for current
x for deprecated
o for obsolete

<flags> is one of:
rw for configuration data
ro for non-configuration data
-x for rpcs
-n for notifications

<name> is the name of the node
(<name>) means that the node is a choice node
:(<name>) means that the node is a case node

If the node is augmented into the tree from another module, its name is printed as <prefix>:<name>.

<opts> is one of:
? for an optional leaf or choice
! for a presence container
* for a leaf-list or list
[<keys>] for a list's keys

<type> is the name of the type for leafs and leaf-lists

<if-features> is the list of features this node depends on, printed within curly brackets and a question mark "{...}?"

```
module: lisp
   +--rw itr-cfg! {itr}?
   |  +--rw rloc-probing!
   |  |  +--rw interval?           uint16
   |  |  +--rw retries?            uint8
   |  |  +--rw retries-interval?   uint16
   |  +--rw itr-rlocs
   |  |  +--rw itr-rloc* [id]
   |  |     +--rw id         string
   |  |     +--rw address
   |  |        +--rw afi?           enumeration
   |  |        +--rw instance-id?   instance-id-type
   |  |        +--rw (address)?
   |  |           +--:(ipv4)
```

```
|  |           |  +--rw ipv4?          inet:ipv4-address
|  |           +--:(ipv6)
|  |           |  +--rw ipv6?          inet:ipv6-address
|  |           +--:(mac-address)
|  |           |  +--rw mac-address?   yang:mac-address
|  |           +--:(lcaf)
|  |              +--rw lcaf
|  |                 +--rw lcaf-type?              enumeration
|  |                 +--rw (address)?
|  |                    +--:(as-number)
|  |                    |     ...
|  |                    +--:(sourc-dest-key)
|  |                    |     ...
|  |                    +--:(explicit-locator-path)
|  |                          ...
|  +--rw local-eids
|  |  +--rw local-eid* [id]
|  |     +--rw id             eid-id
|  |     +--rw eid-address
|  |        +--rw afi?           enumeration
|  |        +--rw instance-id?   instance-id-type
|  |        +--rw (address)?
|  |           +--:(ipv4)
|  |           |  +--rw ipv4?          inet:ipv4-address
|  |           +--:(ipv6)
|  |           |  +--rw ipv6?          inet:ipv6-address
|  |           +--:(mac-address)
|  |           |  +--rw mac-address?   yang:mac-address
|  |           +--:(lcaf)
|  |              +--rw lcaf
|  |                 +--rw lcaf-type?              enumeration
|  |                 +--rw (address)?
|  |                    +--:(as-number)
|  |                    |     ...
|  |                    +--:(sourc-dest-key)
|  |                    |     ...
|  |                    +--:(explicit-locator-path)
|  |                          ...
|  +--rw map-resolvers
|  |  +--rw map-resolver* [id]
|  |     +--rw id                      eid-id
|  |     +--rw eid-address
|  |     |  +--rw afi?           enumeration
|  |     |  +--rw instance-id?   instance-id-type
|  |     |  +--rw (address)?
|  |     |     +--:(ipv4)
|  |     |     |  +--rw ipv4?          inet:ipv4-address
|  |     |     +--:(ipv6)
```

```
|  |     |     |  +--rw ipv6?          inet:ipv6-address
|  |     |     +--:(mac-address)
|  |     |     |  +--rw mac-address?   yang:mac-address
|  |     |     +--:(lcaf)
|  |     |        +--rw lcaf
|  |     |           +--rw lcaf-type?              enumeration
|  |     |           +--rw (address)?
|  |     |              +--:(as-number)
|  |     |              |     ...
|  |     |              +--:(sourc-dest-key)
|  |     |              |     ...
|  |     |              +--:(explicit-locator-path)
|  |     |                    ...
|  |     +--rw map-resolver-address*   inet:ip-address
|  +--rw proxy-etrs
|  |  +--rw proxy-etr* [id]
|  |     +--rw id                    eid-id
|  |     +--rw eid-address
|  |     |  +--rw afi?           enumeration
|  |     |  +--rw instance-id?   instance-id-type
|  |     |  +--rw (address)?
|  |     |     +--:(ipv4)
|  |     |     |  +--rw ipv4?          inet:ipv4-address
|  |     |     +--:(ipv6)
|  |     |     |  +--rw ipv6?          inet:ipv6-address
|  |     |     +--:(mac-address)
|  |     |     |  +--rw mac-address?   yang:mac-address
|  |     |     +--:(lcaf)
|  |     |        +--rw lcaf
|  |     |           +--rw lcaf-type?              enumeration
|  |     |           +--rw (address)?
|  |     |              +--:(as-number)
|  |     |              |     ...
|  |     |              +--:(sourc-dest-key)
|  |     |              |     ...
|  |     |              +--:(explicit-locator-path)
|  |     |                    ...
|  |     +--rw proxy-etr-address*   inet:ip-address
|  +--rw map-cache
|     +--rw mapping* [id]
|        +--rw id                   eid-id
|        +--rw eid
|        |  +--rw afi?           enumeration
|        |  +--rw instance-id?   instance-id-type
|        |  +--rw (address)?
|        |     +--:(ipv4)
|        |     |  +--rw ipv4?          inet:ipv4-address
|        |     +--:(ipv6)
```

```
|        |     |  +--rw ipv6?          inet:ipv6-address
|        |     +--:(mac-address)
|        |     |  +--rw mac-address?   yang:mac-address
|        |     +--:(lcaf)
|        |        +--rw lcaf
|        |           +--rw lcaf-type?              enumeration
|        |           +--rw (address)?
|        |              +--:(as-number)
|        |              |     ...
|        |              +--:(sourc-dest-key)
|        |              |     ...
|        |              +--:(explicit-locator-path)
|        |                    ...
|        +--rw ttl?               uint32
|        +--rw (locator-list)?
|        |  +--:(negative-mapping)
|        |  |  +--rw map-reply-action?   map-reply-action
|        |  +--:(positive-mapping)
|        |     +--rw rlocs
|        |        +--rw rloc* [name]
|        |           +--rw name                  string
|        |           +--rw (address-type)?
|        |           |  +--:(interface-address)
|        |           |  |     ...
|        |           |  +--:(lisp-address)
|        |           |        ...
|        |           +--rw priority?             uint8
|        |           +--rw weight?               uint8
|        |           +--rw multicast-priority?   uint8
|        |           +--rw multicast-weight?     uint8
|        +--rw static?             boolean
+--rw etr-cfg! {etr}?
|  +--rw local-eids
|     +--rw local-eid* [id]
|        +--rw id                        eid-id
|        +--rw eid-address
|        |  +--rw afi?           enumeration
|        |  +--rw instance-id?   instance-id-type
|        |  +--rw (address)?
|        |     +--:(ipv4)
|        |     |  +--rw ipv4?          inet:ipv4-address
|        |     +--:(ipv6)
|        |     |  +--rw ipv6?          inet:ipv6-address
|        |     +--:(mac-address)
|        |     |  +--rw mac-address?   yang:mac-address
|        |     +--:(lcaf)
|        |        +--rw lcaf
|        |           +--rw lcaf-type?              enumeration
```

```
|        |           +--rw (address)?
|        |              +--:(as-number)
|        |              |     ...
|        |              +--:(sourc-dest-key)
|        |              |     ...
|        |              +--:(explicit-locator-path)
|        |                    ...
|        +--rw map-servers
|        |  +--rw map-server* [address]
|        |     +--rw address          inet:ip-address
|        |     +--rw auth-key?        string
|        |     +--rw auth-key-type?   auth-key-type
|        +--rw rlocs
|        |  +--rw rloc* [name]
|        |     +--rw name                  string
|        |     +--rw (address-type)?
|        |     |  +--:(interface-address)
|        |     |  |  +--rw interface?           interface-name
|        |     |  +--:(lisp-address)
|        |     |     +--rw locator-address
|        |     |        +--rw afi?           enumeration
|        |     |        +--rw instance-id?   instance-id-type
|        |     |        +--rw (address)?
|        |     |              ...
|        |     +--rw priority?             uint8
|        |     +--rw weight?               uint8
|        |     +--rw multicast-priority?   uint8
|        |     +--rw multicast-weight?     uint8
|        +--rw record-ttl?              uint32
|        +--rw want-map-notify?         boolean
|        +--rw proxy-reply?             boolean
|        +--rw registration-interval?   uint16
+--rw map-server-cfg! {map-server}?
|  +--rw sites
|  |  +--rw site* [site-id]
|  |     +--rw site-id    uint64
|  |     +--rw devices
|  |        +--rw device* [device-id]
|  |           +--rw device-id    uint64
|  |           +--rw auth-key
|  |           |  +--rw auth-key-value?   string
|  |           |  +--rw auth-key-type?    auth-key-type
|  |           +--rw eids
|  |              +--rw eid* [id]
|  |                 +--rw id                            eid-id
|  |                 +--rw eid-address
|  |                 |  +--rw afi?           enumeration
|  |                 |  +--rw instance-id?   instance-id-type
```

```
|  |                |  +--rw (address)?
|  |                |        ...
|  |                +--rw more-specifics-accepted?      boolean
|  |                +--rw mapping-expiration-timeout?   int16
|  +--rw registered-mappings
|  |  +--rw mapping* [id]
|  |     +--rw id                  eid-id
|  |     +--rw eid
|  |     |  +--rw afi?           enumeration
|  |     |  +--rw instance-id?   instance-id-type
|  |     |  +--rw (address)?
|  |     |     +--:(ipv4)
|  |     |     |  +--rw ipv4?          inet:ipv4-address
|  |     |     +--:(ipv6)
|  |     |     |  +--rw ipv6?          inet:ipv6-address
|  |     |     +--:(mac-address)
|  |     |     |  +--rw mac-address?   yang:mac-address
|  |     |     +--:(lcaf)
|  |     |        +--rw lcaf
|  |     |           +--rw lcaf-type?              enumeration
|  |     |           +--rw (address)?
|  |     |              +--:(as-number)
|  |     |              |     ...
|  |     |              +--:(sourc-dest-key)
|  |     |              |     ...
|  |     |              +--:(explicit-locator-path)
|  |     |                    ...
|  |     +--rw ttl?               uint32
|  |     +--rw (locator-list)?
|  |        +--:(negative-mapping)
|  |        |  +--rw map-reply-action?   map-reply-action
|  |        +--:(positive-mapping)
|  |           +--rw rlocs
|  |              +--rw rloc* [name]
|  |                 +--rw name                  string
|  |                 +--rw (address-type)?
|  |                 |  +--:(interface-address)
|  |                 |  |     ...
|  |                 |  +--:(lisp-address)
|  |                 |        ...
|  |                 +--rw priority?             uint8
|  |                 +--rw weight?               uint8
|  |                 +--rw multicast-priority?   uint8
|  |                 +--rw multicast-weight?     uint8
|  +--rw (mapping-system)
|     +--:(ddt-mapping-system)
|     |  +--rw ddt-mapping-system!
|     |     +--rw ddt-root*             inet:ip-address
```

```
|     |     +--rw authoritative-eids
|     |        +--rw authoritative-eid* [id]
|     |           +--rw id             eid-id
|     |           +--rw eid-address
|     |              +--rw afi?           enumeration
|     |              +--rw instance-id?   instance-id-type
|     |              +--rw (address)?
|     |                 +--:(ipv4)
|     |                 |     ...
|     |                 +--:(ipv6)
|     |                 |     ...
|     |                 +--:(mac-address)
|     |                 |     ...
|     |                 +--:(lcaf)
|     |                       ...
|     +--:(alt-mapping-system)
|        +--rw alt-mapping-system!
+--rw map-resolver-cfg! {map-resolver}?
|  +--rw (mapping-system)
|     +--:(ddt-mapping-system)
|     |  +--rw ddt-mapping-system!
|     |     +--rw ddt-root*   inet:ip-address
|     +--:(alt-mapping-system)
|        +--rw alt-mapping-system!
+--rw proxy-itr-cfg! {proxy-itr}?
|  +--rw servicing-eids
|  |  +--rw eid* [id]
|  |     +--rw id             eid-id
|  |     +--rw eid-address
|  |        +--rw afi?           enumeration
|  |        +--rw instance-id?   instance-id-type
|  |        +--rw (address)?
|  |           +--:(ipv4)
|  |           |  +--rw ipv4?          inet:ipv4-address
|  |           +--:(ipv6)
|  |           |  +--rw ipv6?          inet:ipv6-address
|  |           +--:(mac-address)
|  |           |  +--rw mac-address?   yang:mac-address
|  |           +--:(lcaf)
|  |              +--rw lcaf
|  |                 +--rw lcaf-type?              enumeration
|  |                 +--rw (address)?
|  |                    +--:(as-number)
|  |                    |     ...
|  |                    +--:(sourc-dest-key)
|  |                    |     ...
|  |                    +--:(explicit-locator-path)
|  |                          ...
```

```
|  +--rw map-resolvers
|  |  +--rw map-resolver* [id]
|  |     +--rw id              eid-id
|  |     +--rw eid-address
|  |     |  +--rw afi?           enumeration
|  |     |  +--rw instance-id?   instance-id-type
|  |     |  +--rw (address)?
|  |     |     +--:(ipv4)
|  |     |     |  +--rw ipv4?          inet:ipv4-address
|  |     |     +--:(ipv6)
|  |     |     |  +--rw ipv6?          inet:ipv6-address
|  |     |     +--:(mac-address)
|  |     |     |  +--rw mac-address?   yang:mac-address
|  |     |     +--:(lcaf)
|  |     |        +--rw lcaf
|  |     |           +--rw lcaf-type?              enumeration
|  |     |           +--rw (address)?
|  |     |              +--:(as-number)
|  |     |              |     ...
|  |     |              +--:(sourc-dest-key)
|  |     |              |     ...
|  |     |              +--:(explicit-locator-path)
|  |     |                    ...
|  |     +--rw map-resolver*   inet:ip-address
|  +--rw map-cache
|     +--rw mapping* [id]
|        +--rw id                 eid-id
|        +--rw eid
|        |  +--rw afi?           enumeration
|        |  +--rw instance-id?   instance-id-type
|        |  +--rw (address)?
|        |     +--:(ipv4)
|        |     |  +--rw ipv4?          inet:ipv4-address
|        |     +--:(ipv6)
|        |     |  +--rw ipv6?          inet:ipv6-address
|        |     +--:(mac-address)
|        |     |  +--rw mac-address?   yang:mac-address
|        |     +--:(lcaf)
|        |        +--rw lcaf
|        |           +--rw lcaf-type?              enumeration
|        |           +--rw (address)?
|        |              +--:(as-number)
|        |              |     ...
|        |              +--:(sourc-dest-key)
|        |              |     ...
|        |              +--:(explicit-locator-path)
|        |                    ...
|        +--rw ttl?               uint32
```

```
|        +--rw (locator-list)?
|           +--:(negative-mapping)
|           |  +--rw map-reply-action?   map-reply-action
|           +--:(positive-mapping)
|              +--rw rlocs
|                 +--rw rloc* [name]
|                    +--rw name                  string
|                    +--rw (address-type)?
|                    |  +--:(interface-address)
|                    |  |     ...
|                    |  +--:(lisp-address)
|                    |        ...
|                    +--rw priority?             uint8
|                    +--rw weight?               uint8
|                    +--rw multicast-priority?   uint8
|                    +--rw multicast-weight?     uint8
|     +--rw proxy-etr-cfg! {proxy-etr}?
|        +--rw servicing-eids
|           +--rw eid* [id]
|              +--rw id             eid-id
|              +--rw eid-address
|                 +--rw afi?           enumeration
|                 +--rw instance-id?   instance-id-type
|                 +--rw (address)?
|                    +--:(ipv4)
|                    |  +--rw ipv4?          inet:ipv4-address
|                    +--:(ipv6)
|                    |  +--rw ipv6?          inet:ipv6-address
|                    +--:(mac-address)
|                    |  +--rw mac-address?   yang:mac-address
|                    +--:(lcaf)
|                       +--rw lcaf
|                          +--rw lcaf-type?               enumeration
|                          +--rw (address)?
|                             +--:(as-number)
|                             |     ...
|                             +--:(sourc-dest-key)
|                             |     ...
|                             +--:(explicit-locator-path)
|                                   ...
```

3. YANG model

This section contains the YANG model for lisp configuration and the companion lisp-address-types module.

<CODE BEGINS> file "lisp-address-types@2015-03-23.yang"

```
module lisp-address-types {
  namespace "urn:ietf:params:xml:ns:yang:lisp-address-types";
  prefix lisp;

  import ietf-inet-types {
    prefix inet;
  }
  import ietf-yang-types {
    prefix yang;
  }

  organization "IETF LISP (Locator/ID Separation Protocol) Working Group";
  contact
    "lisp@ietf.org";
  description
    "This YANG module defines the LISP Canonical Address Formats (LCAF)
     for LISP. The module can be extended by vendors
     to define vendor-specific parameters.

     Copyright (c) 2015 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject
     to the license terms contained in, the Simplified BSD License
     set forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (http://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC 6338; see
     the RFC itself for full legal notices.

    ";

  revision 2015-03-23 {
    description
      "Initial revision.";
  }

  typedef instance-id-type {
    type uint32 {
      range "0..16777214";
    }
  }

  typedef simple-address {
    type union {
      type inet:ip-address;
```

```
      type yang:mac-address;
    }
  }

  grouping lcaf-address {
    leaf lcaf-type {
      type enumeration {
        enum "null";
        enum "afi-list";
        enum "instance-id";
        enum "as-number";
        enum "application-data";
        enum "geo-coordinates";
        enum "opaque-key";
        enum "nat-ttraversal";
        enum "nonce-locator";
        enum "multicast-info";
        enum "explicit-locator-path";
        enum "security-key";
        enum "source-dest-key";
        enum "replication-list";
        enum "json-data-model";
        enum "key-value-address";
        enum "encapsulation-format";
      }
    }
    choice address {
      container as-number {
        when "lcaf-type = as-number";
        leaf as {
          type inet:as-number;
        }
        leaf address {
          type simple-address;
        }
      }
      container sourc-dest-key {
        when "lcaf-type = source-dest-key";
        leaf source {
          type inet:ip-prefix;
        }
        leaf dest {
          type inet:ip-prefix;
        }
      }
      container explicit-locator-path {
        when "lcaf-type = explicit-locator-path";
        list hop {
```

```
          key "address";
          leaf address {
            type simple-address;
          }
          leaf lrs-bits {
            type bits {
              bit lookup;
              bit rloc-probe;
              bit strict;
            }
          }
        }
      }
    }
  }

  grouping lisp-address {
    leaf afi {
      type enumeration {
        enum "ipv4" {
          value 1;
        }
        enum "ipv6" {
          value 2;
        }
        enum "mac-address" {
          value 6;
        }
        enum "lcaf" {
          value 16387;
        }
      }
    }
    leaf instance-id {
      type instance-id-type;
    }
    choice address {
      case ipv4 {
        when "afi = ipv4";
        leaf ipv4 {
          type inet:ipv4-address;
        }
      }
      case ipv6 {
        when "afi = ipv6";
        leaf ipv6 {
          type inet:ipv6-address;
        }
```

```
      }
      case mac-address {
        when "afi = mac-address";
        leaf mac-address {
          type yang:mac-address;
        }
      }
      case lcaf {
        when "afi = lcaf";
        container lcaf {
          uses lcaf-address;
        }
      }
    }
  }
}

<CODE ENDS>

<CODE BEGINS> file "lisp@2015-03-23.yang"

module lisp {
  namespace "urn:ietf:params:xml:ns:yang:lisp";
  prefix lisp;

  import ietf-inet-types {
    prefix inet;
  }
  import lisp-address-types {
    prefix lcaf;
  }

  organization "IETF LISP (Locator/ID Separation Protocol) Working Group";
  contact
    "lisp@ietf.org";
  description
    "This YANG module defines the generic configuration
     data for LISP. The module can be extended by vendors
     to define vendor-specific LISP configuration
     parameters and policies.

     Copyright (c) 2015 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject
     to the license terms contained in, the Simplified BSD License
     set forth in Section 4.c of the IETF Trust's Legal Provisions
```

```
     Relating to IETF Documents
     (http://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC 6338; see
     the RFC itself for full legal notices.

    ";

  revision 2015-03-23 {
    description
      "Initial revision.";
  }

  typedef interface-name {
    description
      "Name of a device interface";
    type string;
  }

  typedef map-reply-action {
    description
      "Defines the lisp map-cache ACT type";
    type enumeration {
      enum "no-action" {
        value 0;
      }
      enum "natively-forward" {
        value 1;
      }
      enum "send-map-request" {
        value 2;
      }
      enum "drop" {
        value 3;
      }
    }
  }

  typedef eid-id {
    type string;
  }

  typedef instance-id-type {
    type uint32 {
      range "0..16777214";
    }
  }
```

```
  typedef auth-key-type {
    type enumeration {
      enum "none" {
        value 0;
      }
      enum "hmac-sha-1-96" {
        value 1;
      }
      enum "hmac-sha-256-128" {
        value 2;
      }
    }
  }

  feature itr {
    description
      "ITR operation supported";
  }

  feature etr {
    description
      "ETR operation supported";
  }

  feature proxy-itr {
    description
      "PITR operation supported";
  }

  feature proxy-etr {
    description
      "PETR operation supported";
  }

  feature map-server {
    description
      "MS operation supported";
  }

  feature map-resolver {
    description
      "MR operation supported";
  }

  grouping locators {
    list rloc {
      key "name";
      leaf name {
```

```
        type string;
      }
      choice address-type {
        case interface-address {
          leaf interface {
            type interface-name;
          }
        }
        case lisp-address {
          container locator-address {
            uses lcaf:lisp-address;
          }
        }
      }
      leaf priority {
        type uint8;
      }
      leaf weight {
        type uint8;
      }
      leaf multicast-priority {
        type uint8;
      }
      leaf multicast-weight {
        type uint8;
      }
    }
  }

  grouping mappings {
    list mapping {
      key "id";
      leaf id {
        type eid-id;
      }
      container eid {
        uses lcaf:lisp-address;
      }
      leaf ttl {
        type uint32;
      }
      choice locator-list {
        case negative-mapping {
          leaf map-reply-action {
            type map-reply-action;
          }
        }
        case positive-mapping {
```

```
          container rlocs {
            uses locators;
          }
        }
        default "positive-mapping";
      }
    }
  }

  container itr-cfg {
    if-feature itr;
    presence "LISP ITR operation enabled";
    config true;
    container rloc-probing {
      presence "RLOC probing active";
      leaf interval {
        type uint16;
        units "seconds";
        description
          "Interval in seconds";
      }
      leaf retries {
        type uint8;
        description
          "Number of retries";
      }
      leaf retries-interval {
        type uint16;
        units "seconds";
        description
          "Interval in seconds between retries";
      }
    }
    container itr-rlocs {
      description
        "List of RLOCs of the ITR included in Map-Requests";
      list itr-rloc {
        key "id";
        leaf id {
          type string;
        }
        container address {
          uses lcaf:lisp-address;
        }
      }
    }
    container local-eids {
      list local-eid {
```

```
        min-elements 1;
        key "id";
        leaf id {
          type eid-id;
        }
        container eid-address {
          uses lcaf:lisp-address;
        }
      }
    }
    container map-resolvers {
      list map-resolver {
        min-elements 1;
        key "id";
        leaf id {
          type eid-id;
        }
        container eid-address {
          uses lcaf:lisp-address;
        }
        leaf-list map-resolver-address {
          type inet:ip-address;
        }
      }
    }
    container proxy-etrs {
      list proxy-etr {
        key "id";
        leaf id {
          type eid-id;
        }
        container eid-address {
          uses lcaf:lisp-address;
        }
        leaf-list proxy-etr-address {
          type inet:ip-address;
        }
      }
    }
    container map-cache {
      uses mappings {
        augment "mapping" {
          leaf static {
            description
              "A configured mapping is a static mapping. If the mapping
               is learned, it is operational data and static is false.";
            type boolean;
            default "true";
```

```
          }
        }
      }
    }
  }
  container etr-cfg {
    if-feature etr;
    presence "LISP ETR operation enabled";
    config true;
    container local-eids {
      list local-eid {
        min-elements 1;
        key "id";
        leaf id {
          type eid-id;
        }
        container eid-address {
          uses lcaf:lisp-address;
        }
        container map-servers {
          list map-server {
            key "address";
            leaf address {
              type inet:ip-address;
            }
            leaf auth-key {
              type string;
            }
            leaf auth-key-type {
              type auth-key-type;
            }
          }
        }
        container rlocs {
          uses locators;
        }
        leaf record-ttl {
          type uint32;
        }
        leaf want-map-notify {
          type boolean;
        }
        leaf proxy-reply {
          type boolean;
        }
        leaf registration-interval {
          units "seconds";
          type uint16;
```

```
          default "60";
        }
      }
    }
  }
  container map-server-cfg {
    if-feature map-server;
    presence "LISP Map Server operation enabled";
    config true;
    container sites {
      list site {
        key "site-id";
        leaf site-id {
          type uint64;
        }
        container devices {
          list device {
            key "device-id";
            leaf device-id {
              type uint64;
            }
            container auth-key {
              leaf auth-key-value {
                description
                  "clear text authentication key";
                type string;
              }
              leaf auth-key-type {
                type auth-key-type;
              }
            }
            container eids {
              list eid {
                key "id";
                leaf id {
                  type eid-id;
                }
                container eid-address {
                  uses lcaf:lisp-address;
                }
                leaf more-specifics-accepted {
                  type boolean;
                }
                leaf mapping-expiration-timeout {
                  type int16;
                  units "seconds";
                  default "180";
                }
```

```
              }
            }
          }
        }
      }
      container registered-mappings {
        uses mappings;
      }
      choice mapping-system {
        mandatory true;
        container ddt-mapping-system {
          presence "DDT Mapping System in use";
          leaf-list ddt-root {
            type inet:ip-address;
          }
          container authoritative-eids {
            list authoritative-eid {
              key "id";
              leaf id {
                type eid-id;
              }
              container eid-address {
                uses lcaf:lisp-address;
              }
            }
          }
        }
        container alt-mapping-system {
          presence "ALT Mapping System in use";
        }
      }
    }
    container map-resolver-cfg {
      if-feature map-resolver;
      presence "LISP Map-Resolver operation enabled";
      config true;
      choice mapping-system {
        mandatory true;
        container ddt-mapping-system {
          presence "DDT Mapping System in use";
          leaf-list ddt-root {
            type inet:ip-address;
          }
        }
        container alt-mapping-system {
          presence "ALT Mapping System in use";
        }
```

```
    }
  }
  container proxy-itr-cfg {
    if-feature proxy-itr;
    presence "LISP PITR operation enabled";
    config true;
    container servicing-eids {
      list eid {
        key "id";
        leaf id {
          type eid-id;
        }
        container eid-address {
          uses lcaf:lisp-address;
        }
      }
    }
    container map-resolvers {
      list map-resolver {
        key "id";
        leaf id {
          type eid-id;
        }
        container eid-address {
          uses lcaf:lisp-address;
        }
        leaf-list map-resolver {
          min-elements 1;
          type inet:ip-address;
        }
      }
    }
    container map-cache {
      uses mappings;
    }
  }
  container proxy-etr-cfg {
    if-feature proxy-etr;
    presence "LISP PETR operation enabled";
    config true;
    container servicing-eids {
      list eid {
        key "id";
        leaf id {
          type eid-id;
        }
        container eid-address {
          uses lcaf:lisp-address;
```

```
        }
      }
    }
  }
}
```

<CODE ENDS>

4. Acknowledgments

The tree view and the YANG model shown in this document have been formated with the ’pyang’ tool.

5. IANA Considerations

This memo includes no request to IANA.

6. Security Considerations

Security Considerations TBD

7. Normative References

[I-D.ietf-lisp-ddt]
Fuller, V., Lewis, D., Ermagan, V., and A. Jain, "LISP Delegated Database Tree", draft-ietf-lisp-ddt-02 (work in progress), October 2014.

[I-D.ietf-lisp-lcaf]
Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", draft-ietf-lisp-lcaf-04 (work in progress), January 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, January 2013.

[RFC6833] Fuller, V. and D. Farinacci, "Locator/ID Separation Protocol (LISP) Map-Server Interface", RFC 6833, January 2013.

[RFC6836] Fuller, V., Farinacci, D., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol Alternative Logical Topology (LISP+ALT)", RFC 6836, January 2013.

Authors' Addresses

Vina Ermagan
Cisco Systems
170 Tasman Drive
San Jose, CA
USA

Email: vermagan@cisco.com

Alberto Rodriguez-Natal
Technical University of Catalonia
Barcelona
Spain

Email: arnatal@ac.upc.edu

Florin Coras
Technical University of Catalonia
Barcelona
Spain

Email: fcoras@ac.upc.edu

Albert Cabellos-Aparicio
Technical University of Catalonia
Barcelona
Spain

Email: acabello@ac.upc.edu

Fabio Maino
Cisco Systems
170 Tasman Drive
San Jose, CA
USA

Email: fmaino@cisco.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: June 17, 2015

V. Moreno
Cisco Systems
D. Farinacci
lispers.net

December 14, 2014

Signal-Free LISP Multicast
draft-farinacci-lisp-signal-free-multicast-02

Abstract

When multicast sources and receivers are active at LISP sites, the core network is required to use native multicast so packets can be delivered from sources to group members. When multicast is not available to connect the multicast sites together, a signal-free mechanism can be used to allow traffic to flow between sites. The mechanism within here uses unicast replication and encapsulation over the core network for the data-plane and uses the LISP mapping database system so encapsulators at the source LISP multicast site can find de-capsulators at the receiver LISP multicast sites.

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 [RFC2119].

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 June 17, 2015.

Table of Contents

1. Introduction

When multicast sources and receivers are active at LISP sites, and the core network between the sites does not provide multicast support, a signal-free mechanism can be used to create an overlay that will allow multicast traffic to flow between sites and connect the multicast trees at the different sites.

The signal-free mechanism here proposed does not extend PIM over the overlay as proposed in [RFC6831], nor does the mechanism utilize direct signaling between the Receiver-ETRs and Sender-ITRs as described in [I-D.farinacci-lisp-mr-signaling]. The signal-free mechanism proposed reduces the amount of signaling required between sites to a minimum and is centered around the registration of Receiver-sites for a particular multicast-group or multicast-channel with the LISP Mapping System.

Registrations from the different receiver-sites will be merged at the Mapping System to assemble a multicast-replication-list inclusive of all RLOCs that lead to receivers for a particular multicast-group or multicast-channel. The replication-list for each specific multicast-entry is maintained as a LISP database mapping entry in the Mapping Database.

When the ITR at the source-site receives multicast traffic from sources at its site, the ITR can query the mapping system by issuing Map-Request messages for the (S,G) source and destination addresses in the packets received. The Mapping System will return the RLOC replication-list to the ITR, which the ITR will cache as per standard LISP procedure. Since the core is assumed to not support multicast, the ITR will replicate the multicast traffic for each RLOC on the replication-list and will unicast encapsulate the traffic to each RLOC. The combined function or replicating and encapsulating the traffic to the RLOCs in the replication-list is referred to as "rep-encapsulation" in this document.

The document describes the General Procedures and information encoding that are required at the Receiver-sites and Source-sites to achieve signal-free multicast interconnectivity. The General Procedures for Mapping System Notifications to different sites are also described. A section dedicated to the specific case of SSM trees discusses the implications to the General Procedures for SSM multicast trees over different topological scenarios. At this stage ASM trees are not supported with LISP Signal-Free multicast.

2. Definition of Terms

LISP related terms, notably Map-Request, Map-Reply, Ingress Tunnel Router (ITR), Egress Tunnel Router (ETR), Map-Server (MS) and Map-Resolver (MR) are defined in the LISP specification [RFC6830].

Extensions to the definitions in [RFC6830] for their application to multicast routing are documented in [RFC6831].

Terms defining interactions with the LISP Mapping System are defined in [RFC6833].

The following terms are consistent with the definitions in [RFC6830] and [RFC6831]. The terms are specific cases of the general terms and are here defined to facilitate the descriptions and discussions within this particular document.

Source: Multicast source end-point. Host originating multicast packets.

Receiver: Multicast group member end-point. Host joins multicast group as a receiver of multicast packets sent to the group.

Receiver-site: LISP site where multicast receivers are located.

Source-site: LISP site where multicast sources are located.

RP-site: LISP site where an ASM PIM Rendezvous Point is located. The RP-site and the Source-site may be the same in some situations.

Receiver-ETR: LISP xTR at the Receiver-site. This is a multicast ETR.

Source-ITR: LISP xTR at the Source-site. This is a multicast ITR.

RP-xTR: LISP xTR at the RP-site. This is typically a multicast ITR.

Replication-list: Mapping-entry containing the list of RLOCs that have registered Receivers for a particular multicast-entry.

Multicast-entry: A tuple identifying a multicast tree. Multicast-entries are in the form of (S-prefix, G-prefix).

Rep-encapsulation: The process of replicating and then encapsulating traffic to multiple RLOCs.

3. Reference Model

The reference model that will be used for the discussion of the Signal-Free multicast tree interconnection is illustrated in Figure 1.

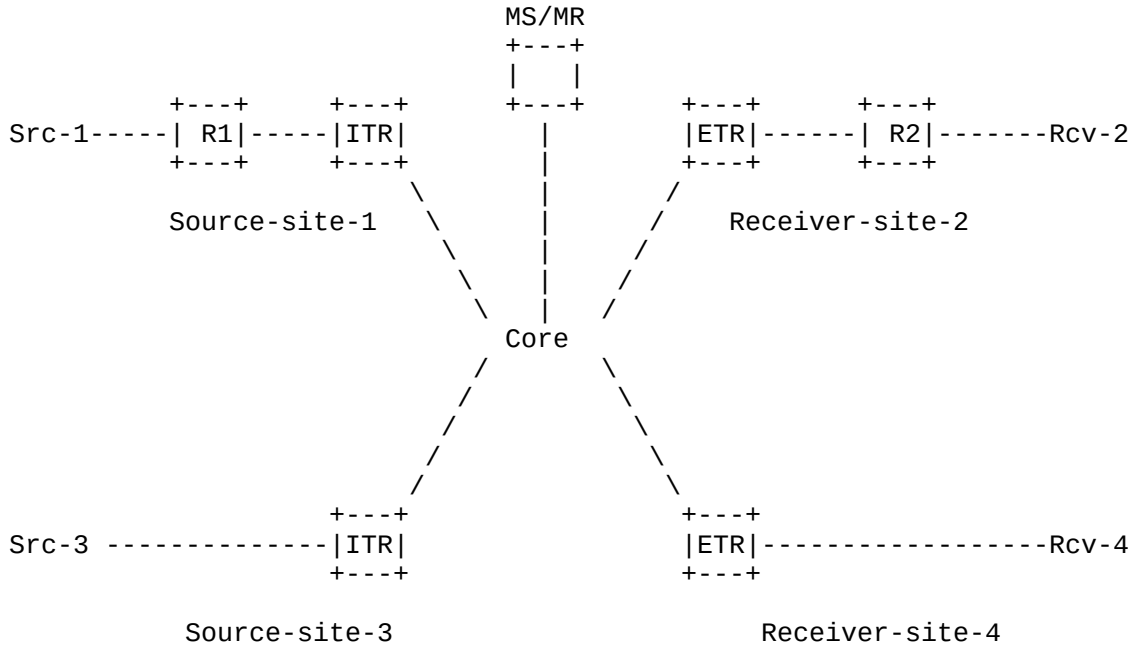


Figure 1: LISP Multicast Generic Reference Model

Sites 1 and 3 are Source-sites.

Source-site-3 presents a Source (Src-3) that is directly connected to the Source-ITR

Source-site-1 presents a Source (Src-1) that is one hop or more away from the Source-ITR

Receiver-site-2 and 4 are receiver sites with not-directly connected and directly connected Receiver end-points respectively

R1 is a router in Source-site-1.

R2 is a PIM router at the Receiver-site.

The Map-Servers and Resolvers are reachable in the RLOC space in the Core, only one is shown for illustration purposes, but these can be many or even part of a DDT tree.

The procedures for interconnecting multicast Trees over an overlay can be broken down into three functional areas:

- Receiver-site procedures
- Source-site procedures
- LISP notification procedures

The receiver site procedures will be common for most tree types and topologies.

The procedures at the source site can vary depending on the type of trees being interconnected as well as based on the topological relation between sources and source-site xTRs. For ASM trees, a special case of the Source-site is the RP-site for which a variation of the Source-site procedures may be necessary if ASM trees are to be supported in future specifications of LISP Signal-Free multicast.

The LISP notification procedures between sites are normalized for the different possible scenarios. Certain scenarios may benefit from a simplified notification mechanism or no notification requirement at all.

4. General Procedures

The interconnection of multicast trees across different LISP sites involves the following procedures to build the necessary multicast distribution trees across sites.

1. The presence of multicast Receiver end-points is detected by the Receiver-ETRs at the Receiver-sites.
2. Receiver-ETRs register their RLOCs as part of the replication-list for the multicast-entry the detected Receivers subscribe to.
3. The Mapping-system merges all receiver-ETR or delivery-group RLOCs to build a comprehensive replication-list inclusive of all Receiver-sites for each multicast-entry.
4. LISP Map-Notify messages should be sent to the Source-ITR informing of any changes in the replication-list.

5. Multicast-tree building at the Source-site is initiated when the Source-ITR receives the LISP Notification.

Once the multicast distribution trees are built, the following forwarding procedures may take place:

1. The Source sends multicast packets to the multicast group destination address.

2. Multicast traffic follows the multicast tree built at the Source-site and makes its way to the Source-ITRs.

3. The Source-ITR will issue a map-request to resolve the replication-list for the multicast-entry.

4. The Mapping System responds to the Source-ITR with a map-reply containing the replication-list for the multicast group requested.

5. The Source-ITR caches the replication-list received in the map-reply for the multicast-entry.

6. Multicast traffic is rep-encapsulated. That is, the packet is replicated for each RLOC in the replication-list and then encapsulated to each one.

4.1. General Receiver-site Procedures

4.1.1. Multicast receiver detection

When the Receiver-ETRs are directly connected to the Receivers (e.g. Receiver-site-4 in Figure 1), the Receiver-ETRs will receive IGMP Reports from the Receivers indicating which group the Receivers wish to subscribe to. Based on these IGMP Reports, the receiver-ETR is made aware of the presence of Receivers as well as which group they are interested in.

When the Receiver-ETRs are several hops away from the Receivers (e.g. Receiver-site-2 in Figure 1), the Receiver-ETRs will receive PIM join messages which will allow the Receiver-ETR to know that there are multicast Receivers at the site and also learn which multicast group the Receivers are for.

4.1.2. Receiver-site Registration

Once the Receiver-ETRs detect the presence of Receivers at the Receiver-site, the Receiver-ETRs will issue Map-Register messages to

include the Receiver-ETR RLOCs in the replication-list for the multicast-entry the Receivers joined.

The Map-Register message will use the multicast-entry (Source, Group) tuple as its EID record type with the Receiver-ETR RLOCs conforming the locator set.

The EID in the Map-Register message must be encoded using the Multicast Information LCAF type defined in [I-D.ietf-lisp-lcaf]. The R, L and J bits in the Multicast-info LCAF frame are not used and should be set to zero.

The RLOC in the Map-Register message must be encoded using the Replication List Entry (RLE) LCAF type defined in [I-D.ietf-lisp-lcaf] with the Level Value fields for all entries set to 128 (decimal).

The encoding described above must be used consistently for Map-Register messages, entries in the Mapping Database, Map-reply messages as well as the map-cache at the Source-ITRs.

The Map-Register messages [RFC6830] sent by the receiver-ETRs should have the following bits set as here specified:

1. merge-request-bit set to 1. The Map-Register messages must be sent with "Merge Semantics". The Map-Server will receive registrations from a multitude of Receiver-ETRs. The Map-Server will merge the registrations for common EIDs and maintain a consolidated replication-list for each multicast-entry.

2. want-map-notify-bit (M) set to 0. This tells the Mapping System that the receiver-ETR does not expect to receive Map-Notify messages as it does not need to be notified of all changes to the replication-list.

3. proxy-reply-bit (P) set to 1. The merged replication-list is kept in the Map-Servers. By setting the proxy-reply bit, the receiver-ETRs instruct the Mapping-system to proxy reply to map-requests issued for the multicast entries.

Map-Register messages for a particular multicast-entry should be sent for every receiver detected, even if previous receivers have been detected for the particular multicast-entry. This allows the replication-list to remain up to date.

4.1.3. Consolidation of the replication-list

The Map-Server will receive registrations from a multitude of Receiver-ETRs. The Map-Server will merge the registrations for common EIDs and consolidate a replication-list for each multicast-entry.

4.2. General Source-site Procedures

Source-ITRs must register the unicast EIDs of any Sources or Rendezvous Points that may be present on the Source-site. In other words, it is assumed that the Sources and RPs are LISP EIDs.

The registration of the unicast EIDs for the Sources or Rendezvous Points allows the map-server to know where to send Map-Notify messages to. Therefore, the Source-ITR must register the unicast S-prefix EID with the want-map-notify-bit set in order to receive Map-Notify messages whenever there is a change in the replication-list.

4.2.1. Multicast Tree Building at the Source-site

When the source site receives the Map-Notify messages from the mapping system as described in Section 4.3, it will initiate the process of building a multicast distribution tree that will allow the multicast packets from the Source to reach the Source-ITR.

The Source-ITR will issue a PIM join for the multicast-entry for which it received the Map-Notify message. The join will be issued in the direction of the source or in the direction of the RP for the SSM and ASM cases respectively.

4.2.2. Multicast Destination Resolution

On reception of multicast packets, the source-ITR must obtain the replication-list for the (S,G) addresses in the packets.

In order to obtain the replication-list, the Source-ITR must issue a Map-Request message in which the EID is the (S,G) multicast tuple which is encoded using the Multicast Info LCAF type defined in [I-D.ietf-lisp-lcaf].

The Mapping System (most likely the Map-Server) will Map-reply with the merged replication-list maintained in the Mapping System. The Map-reply message must follow the format defined in [RFC6830], its EID must be encoded using the Multicast Info LCAF type and the corresponding RLOC-records must be encoded using the RLE LCAF type. Both LCAF types defined in [I-D.ietf-lisp-lcaf].

4.3. General LISP Notification Procedures

The Map-Server will issue LISP Map-Notify messages to inform the Source-site of the presence of receivers for a particular multicast group over the overlay.

Updated Map-Notify messages should be issued every time a new registration is received from a Receiver-site. This guarantees that the source-sites are aware of any potential changes in the multicast-distribution-list membership.

The Map-Notify messages carry (S,G) multicast EIDs encoded using the Multicast Info LCAF type defined in [I-D.ietf-lisp-lcaf].

Map-Notify messages will be sent by the Map-Server to the RLOCs with which the unicast S-prefix EID was registered.

When both the Receiver-sites and the Source-sites register to the same Map-Server, the Map-Server has all the necessary information to send the Map-Notify messages to the Source-site.

When the Map-Servers are distributed in a DDT, the Receiver-sites may register to one Map-Server while the Source-site registers to a different Map-Server. In this scenario, the Map-Server for the receiver sites must resolve the unicast S-prefix EID in the DDT per standard LISP lookup procedures and obtain the necessary information to send the Map-Notify messages to the Source-site. The Map-Notify messages must be sent with an authentication length of 0 as they would not be authenticated.

When the Map-Servers are distributed in a DDT, different Receiver-sites may register to different Map-Servers. This is an unsupported scenario with the currently defined mechanisms.

5. Source Specific Multicast Trees

The interconnection of Source Specific Multicast (SSM) Trees across sites will follow the General Receiver-site Procedures described in Section 4.1 on the Receiver-sites.

The Source-site Procedures will vary depending on the topological location of the Source within the Source-site as described in Section 5.1 and Section 5.2 .

5.1. Source directly connected to Source-ITRs

When the Source is directly connected to the source-ITR, it is not necessary to trigger signaling to build a local multicast tree at the Source-site. Therefore Map-Notify messages may not be required to initiate building of the multicast tree at the Source-site.

Map-Notify messages are still required to ensure that any changes to the replication-list are communicated to the Source-site so that the map-cache at the Source-ITRs is kept updated.

5.2. Source not directly connected to Source-ITRs

The General LISP Notification Procedures described in Section 4.3 must be followed when the Source is not directly connected to the source-ITR. On reception of Map-Notify messages, local multicast signaling must be initiated at the Source-site per the General Source Site Procedures for Multicast Tree building described in Section 4.2.1.

In the SSM case, the IP address of the Source is known and it is also registered with the LISP mapping system. Thus, the mapping system may resolve the mapping for the Source address in order to send Map-Notify messages to the correct source-ITR.

6. PIM Any Source Multicast Trees

LISP signal-free multicast will not support ASM Trees at this time. A future revision of this specification may include procedures for PIM ASM support.

PIM ASM in shared-tree only mode could be supported in the scenario where the root of the shared tree (the PIM RP) is placed at the source site.

7. Signal-Free Multicast for Replication Engineering

The mechanisms in this draft can be applied to the LISP Replication-Engineering [I-D.coras-lisp-re] design. Rather than having the layered LISP-RE RTR hierarchy use signaling mechanisms, the RTRs can register their availability for multicast tree replication via the mapping database system. As stated in [I-D.coras-lisp-re], the RTR layered hierarchy is used to avoid head-end replication in replicating nodes closest to a multicast source. Rather than have multicast ITRs replicate to each ETR in an RLE entry of a (S,G) mapping database entry, it could replicate to one or more layer-0 RTRs in the LISP-RE hierarchy.

There are two formats an (S,G) mapping database entry could have. One format is a 'complete-format' and the other is a 'filtered-format'. A 'complete-format' entails an (S,G) entry having multiple RLOC records which contain both ETRs that have registered as well as the RTRs at the first level of the LISP-RE hierarchy for the ITR to replicate to. When using 'complete-format', the ITR has the ability to select if it replicates to RTRs or to the registered ETRs at the receiver sites. A 'filtered-format' (S,G) entry is one where the Map-Server returns the RLOC-records that it decides the ITR should use. So replication policy is shifted from the ITRs to the mapping system. The Map-Servers can also decide for a given ITR, if it uses a different set of replication targets per (S,G) entry for which the ITR is replicating for.

The procedure for the LISP-RE RTRs to make themselves available for replication can occur before or after any receivers join an (S,G) entry or any sources send for a particular (S,G) entry. Therefore, newly configured RTR state will be used to create new (S,G) state and inherited into existing (S,G) state. A set of RTRs can register themselves to the mapping system or a third-party can do so on their behalf. When RTR registration occurs, it is done with an (S-prefix, G-prefix) entry so it can advertise its replication services for a wide-range of source/group combinations.

When a Map-Server receives (S,G) registrations from ETRs and (S-prefix, G-prefix) registrations from RTRs, it has the option of merging the RTR RLOC-records for each (S,G) that is more-specific for the (S-prefix, G-prefix) entry or keep them separate. When merging, a Map-Server is ready to return a 'complete-format' Map-Reply. When keeping the entries separate, the Map-Server can decide what to include in a Map-Reply when a Map-Request is received. It can include a combination of RLOC-records from each entry or decide to use one or the other depending on policy configured.

Here is a specific example of (S,G) and (S-prefix, G-prefix) mapping database entries when a source S is behind an ITR and there are receiver sites joined to (S,G) via ETR1, ETR2, and ETR3. And there exists a LISP-RE hierarchy of RTR1 and RTR2 at level-0 and RTR3 and RTR4 at level-1:

```
EID-record: (S,G)
    RLOC-record: RLE: (ETR1, ETR2, ETR3), p1
EID-record: (S-prefix, G-prefix)
    RLOC-record: RLE: (RTR1(L0), RTR2(L0), RTR3(L1), RTR4(L1)), p1
```

The above entries are in the form of how they were registered and stored in a Map-Server. When a Map-Server uses 'complete-format', a Map-Reply it originates has the mapping record encoded as:

```
EID-record: (S,G)
    RLOC-record: RLE: (RTR1(L0), RTR3(L1)), p1
    RLOC-record: RLE: (ETR1, ETR2, ETR3), p1
```

The above Map-Reply allows the ITR to decide if it replicates to the ETRs or if it should replicate only to level-0 RTR1. This decision is left to the ITR since both RLOC-records have priority 1. If the Map-Server wanted to force the ITR to replicate to RTR1, it would set the ETRs RLOC-record to priority greater than 1.

When a Map_server uses "filtered-format', a Map-Reply it originates has the mapping record encoded as:

```
EID-record: (S,G)
    RLOC-record: RLE: (RTR1(L0), RTR3(L1)), p1
```

An (S,G) entry can contain alternate RTRs. So rather than replicating to multiple RTRs, one of a RTR set may be used based on the RTR reachability status. An ITR can test reachability status to any layer-0 RTR using RLOC-probing so it can choose one RTR from a set to replicate to. When this is done the RTRs are encoded in different RLOC-records versus together in one RLE RLOC-record. This moves the replication load off the ITRs at the source site to the RTRs inside the network infrastructure. This mechanism can also be used by level-n RTRs to level-n+1 RTRs.

The following mapping would be encoded in a Map-Reply sent by a Map-Server and stored in the ITR. The ITR would use RTR1 until it went unreachable and then switch to use RTR2:

```
EID-record: (S,G)
    RLOC-record: RTR1, p1
    RLOC-record: RTR2, p2
```

8. Security Considerations

[I-D.ietf-lisp-sec] defines a set of security mechanisms that provide origin authentication, integrity and anti-replay protection to LISP's EID-to-RLOC mapping data conveyed via mapping lookup process. LISP-SEC also enables verification of authorization on EID-prefix claims in Map-Reply messages.

Additional security mechanisms to protect the LISP Map-Register messages are defined in [RFC6833].

The security of the Mapping System Infrastructure depends on the particular mapping database used. The [I-D.ietf-lisp-ddt] specification, as an example, defines a public-key based mechanism

that provides origin authentication and integrity protection to the LISP DDT protocol.

Map-Replies received by the source-ITR can be signed (by the Map-Server) so the ITR knows the replication-list is from a legit source.

Data-plane encryption can be used when doing unicast rep-encapsulation as described in [I-D.farinacci-lisp-crypto]. For further study we will look how to do multicast rep-encapsulation.

9. IANA Considerations

This document has no IANA implications

10. Acknowledgements

The authors want to thank Greg Shepherd, Joel Halpern and Sharon Barkai for their insightful contribution to shaping the ideas in this document. Thanks also goes to Jimmy Kyriannis, Paul Vinciguerra, and Florin Coras for testing an implementation of this draft.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3618] Fenner, B. and D. Meyer, "Multicast Source Discovery Protocol (MSDP)", RFC 3618, October 2003.

[RFC4601] Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", RFC 4601, August 2006.

[RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, August 2006.

11.2. Informative References

[I-D.coras-lisp-re]
Coras, F., Cabellos-Aparicio, A., Domingo-Pascual, J., Maino, F., and D. Farinacci, "LISP Replication Engineering", draft-coras-lisp-re-06 (work in progress), October 2014.

[I-D.farinacci-lisp-crypto]
Farinacci, D., "LISP Data-Plane Confidentiality", draft-farinacci-lisp-crypto-01 (work in progress), July 2014.

[I-D.farinacci-lisp-mr-signaling]
Farinacci, D. and M. Napierala, "LISP Control-Plane Multicast Signaling", draft-farinacci-lisp-mr-signaling-05 (work in progress), August 2014.

[I-D.ietf-lisp-ddt]
Fuller, V., Lewis, D., Ermagan, V., and A. Jain, "LISP Delegated Database Tree", draft-ietf-lisp-ddt-02 (work in progress), October 2014.

[I-D.ietf-lisp-lcaf]
Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", draft-ietf-lisp-lcaf-07 (work in progress), December 2014.

[I-D.ietf-lisp-sec]
Maino, F., Ermagan, V., Cabellos-Aparicio, A., and D. Saucez, "LISP-Security (LISP-SEC)", draft-ietf-lisp-sec-07 (work in progress), October 2014.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, January 2013.

[RFC6831] Farinacci, D., Meyer, D., Zwiebel, J., and S. Venaas, "The Locator/ID Separation Protocol (LISP) for Multicast Environments", RFC 6831, January 2013.

[RFC6833] Fuller, V. and D. Farinacci, "Locator/ID Separation Protocol (LISP) Map-Server Interface", RFC 6833, January 2013.

Appendix A. Document Change Log

A.1. Changes to draft-farinacci-lisp-signal-free-multicast-02

- Posted December 2014.
- Added section about how LISP-RE can use the mechanisms from signal-free-multicast so we can avoid head-end replication and avoid signalling across a layered RE topology.

A.2. Changes to draft-farinacci-lisp-signal-free-multicast-01

- Posted June 2014.
- Changes based on implementation experience of this draft.

A.3. Changes to draft-farinacci-lisp-signal-free-multicast-00

- Posted initial draft February 2014.

Authors' Addresses

Victor Moreno
Cisco Systems
170 Tasman Drive
San Jose, California 95134
USA

Email: vimoreno@cisco.com

Dino Farinacci
lispers.net
San Jose, CA 95120
USA

Email: farinacci@gmail.com

Internet Engineering Task Force
Internet-Draft
Intended status: Experimental
Expires: July 16, 2015

D. Farinacci
lispers.net

January 12, 2015

LISP Data-Plane Confidentiality
draft-ietf-lisp-crypto-00

Abstract

This document describes a mechanism for encrypting LISP encapsulated traffic. The design describes how key exchange is achieved using existing LISP control-plane mechanisms as well as how to secure the LISP data-plane from third-party surveillance attacks.

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 July 16, 2015.

Table of Contents

1. Introduction

The Locator/ID Separation Protocol [RFC6830] defines a set of functions for routers to exchange information used to map from non-routable Endpoint Identifiers (EIDs) to routable Routing Locators (RLOCs). LISP ITRs and PITRs encapsulate packets to ETRs and RTRs. Packets that arrive at the ITR or PITR are typically not modified. Which means no protection or privacy of the data is added. If the source host encrypts the data stream then the encapsulated packets can be encrypted but would be redundant. However, when plaintext packets are sent by hosts, this design can encrypt the user payload to maintain privacy on the path between the encapsulator (the ITR or PITR) to a decapsulator (ETR or RTR).

This draft has the following requirements for the solution space:

- Do not require a separate Public Key Infrastructure (PKI) that is out of scope of the LISP control-plane architecture.

- The budget for key exchange MUST be one round-trip time. That is, only a two packet exchange can occur.

- Use symmetric keying so faster cryptography can be performed in the LISP data plane.

- o Avoid a third-party trust anchor if possible.
- o Provide for rekeying when secret keys are compromised.
- o At this time, encapsulated packet authentication is not a strong requirement.

2. Overview

The approach proposed in this draft is to not rely on the LISP mapping system to store security keys. This will provide for a simpler and more secure mechanism. Secret shared keys will be negotiated between the ITR and the ETR in Map-Request and Map-Reply messages. Therefore, when an ITR needs to obtain the RLOC of an ETR, it will get security material to compute a shared secret with the ETR.

The ITR can compute 3 shared-secrets per ETR the ITR is encapsulating to. And when the ITR encrypts a packet before encapsulation, it will identify the key it used for the crypto calculation so the ETR knows which key to use for decrypting the packet after decapsulation. By using key-ids in the LISP header, we can also get rekeying functionality.

3. Diffie-Hellman Key Exchange

LISP will use a Diffie-Hellman [RFC2631] key exchange sequence and computation for computing a shared secret. The Diffie-Hellman parameters will be passed in Map-Request and Map-Reply messages.

Here is a brief description how Diff-Hellman works:

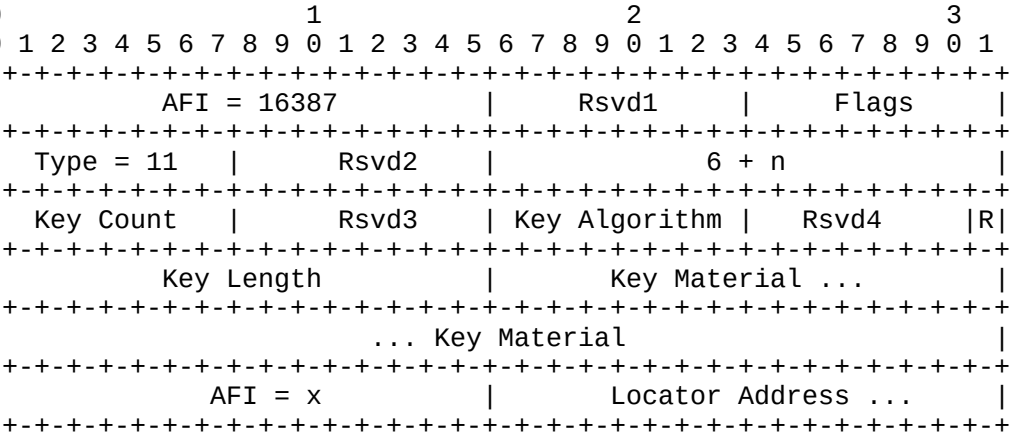
ITR				ETR		
Secret	Public	Calculates	Sends	Calculates	Public	Secret
i	p,g		p,g -->			e
i	p,g,I	g^i mod p=I	I -->		p,g,I	e
i	p,g,I		<-- E	g^e mod p=E	p,g	e
i,s	p,g,I,E	E^i mod p=s		I^e mod p=s	p,g,I,E	e,s

Public-key exchange for computing a shared private key [DH]

Diffie-Hellman parameters 'p' and 'g' must be the same values used by the ITR and ETR. The ITR computes public-key 'I' and transmits 'I' in a Map-Request packet. When the ETR receives the Map-Request, it uses parameters 'p' and 'g' to compute the ETR's public key 'E'. The ETR transmits 'E' in a Map-Reply message. At this point, the ETR has enough information to compute 's', the shared secret, by using 'I' as the base and the ETR's private key 'e' as the exponent. When the ITR receives the Map-Reply, it uses the ETR's public-key 'E' with the ITR's private key 'i' to compute the same 's' shared secret the ETR computed. The value 'p' is used as a modulus to create the width of the shared secret 's'.

4. Encoding and Transmitting Key Material

The Diffie-Hellman key material is transmitted in Map-Request and Map-Reply messages. Diffie-Hellman parameters are encoded in the LISP Security Type LCAF [LCAF].



Diffie-Hellman parameters encoded in Key Material field

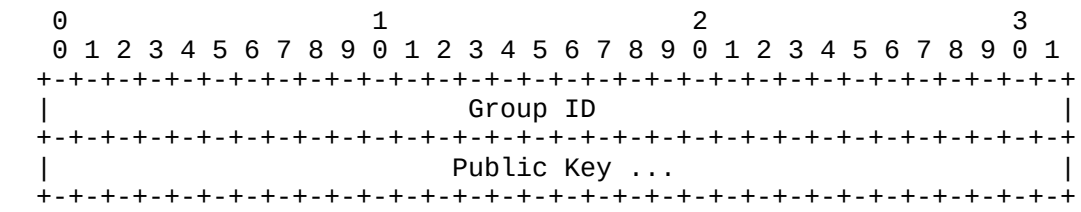
The 'Key Count' field encodes the number of {'Key-Length', 'Key-Material'} fields included in the encoded LCAF. A maximum number of keys that can be encoded are 3 keys, each identified by key-id 1, followed by key-id 2, an finally key-id 3.

The 'R' bit is not used for this use-case of the Security Type LCAF but is reserved for [LISP-DDT] security.

The 'Key Algorithm' encodes the cryptographic algorithm used. The following values are defined:

```
Null:        0
Group-ID:    1
AES:         2
3DES:        3
SHA-256:     4
```

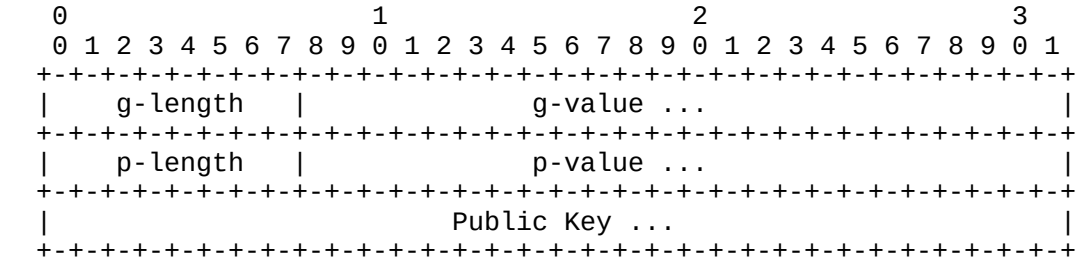
When the 'Key Algorithm' value is 1 (Group-ID), the 'Key Material' field is encoded as:



Points to Key Material values from IANA Registry

The Group-ID values are defined in [RFC2409] and [RFC3526] which describe the Diffie Hellman parameters used for key exchange.

When the 'Key Algorithm' value is not 1 (Group-ID), the 'Key Material' field is encoded as:



Key Length describes the length of the Key Material field

When an ITR or PITR sends a Map-Request, they will encode their own RLOC in Security Type LCAF format within the ITR-RLOCs field. When a ETR or RTR sends a Map-Reply, they will encode their RLOCs in Security Type LCAF format within the RLOC-record field of each EID-record supplied.

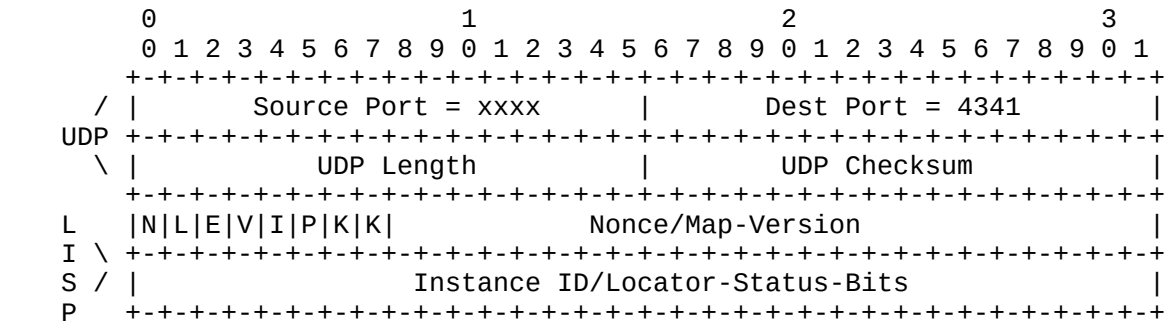
If an ITR or PITR sends a Map-Request with a Security Type LCAF included and the ETR or RTR does not want to have encapsulated traffic encrypted, they will return a Map-Reply with no RLOC records encoded with the Security Type LCAF. This signals to the ITR or PITR

that it should not encrypt traffic (it cannot encrypt traffic anyways since no ETR public-key was returned).

Likewise, if an ITR or PITR wish to include multiple key-ids in the Map-Request but the ETR or RTR wish to use some but not all of the key-ids, they return a Map-Reply only for those key-ids they wish to use.

5. Data-Plane Operation

The LISP encapsulation header [RFC6830] requires changes to encode the key-id for the key being used for encryption.



K-bits indicate when packet is encrypted and which key used

When the KK bits are 00, the encapsulated packet is not encrypted. When the value of the KK bits is 1, 2, or 3, it encodes the key-id of the secret keys computed during the Diffie-Hellman Map-Request/Map-Reply exchange.

When an ITR or PITR receives a packet to be encapsulated, they will first decide what key to use, encode the key-id into the LISP header, and use that key to encrypt all packet data that follows the LISP header. Therefore, the outer header, UDP header, and LISP header travel as plaintext.

There is an open working group item to discuss if the data encapsulation header needs change for encryption or any new applications. This draft proposes changes to the existing header so experimentation can continue without making large changes to the data-plane at this time.

6. Dynamic Rekeying

Since multiple keys can be encoded in both control and data messages, an ITR can encapsulate and encrypt with a specific key while it is negotiating other keys with the same ETR. Soon as an ETR or RTR returns a Map-Reply, it should be prepared to decapsulate and decrypt using the new keys computed with the new Diffie-Hellman parameters received in the Map-Request and returned in the Map-Reply.

RLOC-probing can be used to change keys by the ITR at any time. And when an initial Map-Request is sent to populate the ITR's map-cache, the Map-Requests flows across the mapping system where a single ETR from the Map-Reply RLOC-set will respond. If the ITR decides to use the other RLOCs in the RLOC-set, it MUST send a Map-Request directly to key negotiate with the ETR. This process may be used to test reachability from an ITR to an ETR initially when a map-cache entry is added for the first time, so an ITR can get both reachability status and keys negotiated with one Map-Request/Map-Reply exchange.

A rekeying event is defined to be when an ITR or PITR changes the p, g, or the public-key in a Map-Request. The ETR or RTR compares the p, g, and public-key it last received from the ITR for the key-id, and if any value has changed, it computes a new public-key of its own with the new p and g values from the Map-Request and returns it in the Map-Reply. Now a new shared secret is computed and can be used for the key-id for encryption by the ITR and decryption by the ETR. When the ITR or PITR starts this process of negotiating a new key, it must not use the corresponding key-id in encapsulated packets until it receives a Map-Reply from the ETR with the p and g values it expects (the values it sent in a Map-Request).

Note when RLOC-probing continues to maintain RLOC reachability and rekeying is not desirable, the ITR or RTR can either not include the Security Type LCAF in the Map-Request or supply the same key material as it recieved from the last Map-Reply from the ETR or RTR. This approach signals to the ETR or RTR that no rekeying event is requested.

7. Future Work

By using AES-GCM [RFC5116], or HMAC-CBC [AES-CBC], it has been suggested that encapsulated packet authentication (through encryption [RFC4106]) could be supported. There is current work in progress to investigate these techniques for the LISP data-plane. However, it will require encapsulation header changes to LISP.

For performance considerations, Elliptic-Curve Diffie Hellman (ECDH) can be used as specified in [RFC4492] to reduce CPU cycles required to compute shared secret keys.

8. Security Considerations

8.1. SAAG Support

The LISP working group will seek help from the SAAG working group for security advice. The SAAG will be involved early in the design process so they have early input and review.

8.2. LISP-Crypto Security Threats

Since ITRs and ETRs participate in key exchange over a public non-secure network, a man-in-the-middle (MITM) could circumvent the key exhange and compromise data-plane confidentiality. This can happen when the MITM is acting as a Map-Replier, provides its own public key so the ITR and the MITM generate a shared secret key among each other. If the MITM is in the data path between the ITR and ETR, it can use the shared secret key to decrypt traffic from the ITR.

Since LISP can secure Map-Replies by the authentication process specified in [LISP-SEC], the ITR can detect when a MITM has signed a Map-Reply for an EID-prefix it is not authoritative for. When an ITR determines the signature verification fails, it discards and does not reuse the key exchange parameters, avoids using the ETR for encapsulation, and issues a severe log message to the network adminstrator. Optionally, the ITR can send RLOC-probes to the compromised RLOC to determine if can reach the authoriative ETR. And when the ITR validates the signature of a Map-Reply, it can begin encrypting and encapsulating packets to the RLOC of ETR.

9. IANA Considerations

This draft requires the use of the registry that selects Diffie Hellman parameters. Rather than convey the key exchange parameters directly in LISP control packets, a Group-ID from the registry will be used. The Group-ID values are defined in [RFC2409] and [RFC3526].

10. References

10.1. Normative References

[RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[RFC2631] Rescorla, E., "Diffie-Hellman Key Agreement Method", RFC 2631, June 1999.

[RFC3526] Kivinen, T. and M. Kojo, "More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)", RFC 3526, May 2003.

[RFC4106] Viega, J. and D. McGrew, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)", RFC 4106, June 2005.

[RFC4492] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)", RFC 4492, May 2006.

[RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, January 2008.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, January 2013.

10.2. Informative References

[AES-CBC] McGrew, D., Foley, J., and K. Paterson, "Authenticated Encryption with AES-CBC and HMAC-SHA", draft-mcgrew-aead-aes-cbc-hmac-sha2-03.txt (work in progress), .

[DH] "Diffie-Hellman key exchange", Wikipedia http://en.wikipedia.org/wiki/Diffie-Hellman_key_exchange, .

[LCAF] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format", draft-ietf-lisp-lcaf-04.txt (work in progress), .

[LISP-DDT] Fuller, V., Lewis, D., Ermaagan, V., and A. Jain, "LISP Delegated Database Tree", draft-fuller-lisp-ddt-03 (work in progress), .

[LISP-SEC] Maino, F., Ermagan, V., Cabellos, A., and D. Saucez, "LISP-Secuirty (LISP-SEC)", draft-ietf-lisp-sec-06 (work in progress), .

Appendix A. Acknowledgments

The author would like to thank Dan Harkins, Brian Weis, Joel Halpern, Fabio Maino, Ed Lopez, and Roger Jorgensen for their interest, suggestions, and discussions about LISP data-plane security.

In addition, the support and suggestions from the SAAG working group were helpful and appreciative.

Appendix B. Document Change Log

B.1. Changes to draft-ietf-lisp-crypto-00.txt

- Posted January 2015.
- Changing draft-farinacci-lisp-crypto-01 to draft-ietf-lisp-crypto-00. This draft has become a working group document
- Add text to indicate the working group may work on a new data encapsulation header format for data-plane encryption.

B.2. Changes to draft-farinacci-lisp-crypto-01.txt

- Posted July 2014.
- Add Group-ID to the encoding format of Key Material in a Security Type LCAF and modify the IANA Considerations so this draft can use key exchange parameters from the IANA registry.
- Indicate that the R-bit in the Security Type LCAF is not used by lisp-crypto.
- Add text to indicate that ETRs/RTRs can negotiate less number of keys from which the ITR/PITR sent in a Map-Request.
- Add text explaining how LISP-SEC solves the problem when a man-in-the-middle becomes part of the Map-Request/Map-Reply key exchange process.
- Add text indicating that when RLOC-probing is used for RLOC reachability purposes and rekeying is not desired, that the same key exchange parameters should be used so a reallocation of a pubic key does not happen at the ETR.
- Add text to indicate that ECDH can be used to reduce CPU requirements for computing shared secret-keys.

B.3. Changes to draft-farinacci-lisp-crypto-00.txt

- Initial draft posted February 2014.

Author's Address

Dino Farinacci
lispers.net
San Jose, California
USA

Phone: 408-718-2001
Email: farinacci@gmail.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 7, 2015

D. Saucez
INRIA
L. Iannone
Telecom ParisTech
A. Cabellos
F. Coras
Technical University of Catalonia
March 6, 2015

LISP Impact
draft-ietf-lisp-impact-01.txt

Abstract

The Locator/Identifier Separation Protocol (LISP) aims at improving the Internet scalability properties leveraging on three simple principles: address role separation, encapsulation, and mapping. In this document, based on implementation work, deployment experiences, and theoretical studies, we discuss the impact that the deployment of LISP can have on both the Internet in general and the end-user in particular.

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 September 7, 2015.

Copyright Notice

Table of Contents

1. Introduction

The Locator/Identifier Separation Protocol (LISP) relies on three simple principles to improve the scalability properties of the Internet: address role separation, encapsulation, and mapping. The main goal of LISP is to make the Internet more scalable by reducing the number of prefixes announced in the Default Free Zone (DFZ). As LISP relies on mapping and encapsulation, it turns out that it provides more benefits than just increased scalability. For instance, LISP provides a mean for a LISP site to precisely control its inter-domain outgoing and incoming traffic, with the possibility to apply different policies to different domains exchanging traffic with it. LISP can also be used to ease the transition from IPv4 to IPv6 as it allows to transport IPv4 over IPv6 or IPv6 over IPv4. Furthermore, LISP also provides a solution to perform inter-domain multicast.

This document discusses the impact of LISP's deployment on the Internet and on end-users and shows the consequences of the interworking infrastructure in terms of path-stretch. There still are many, economical rather than technical, open questions related to

the deployment of such infrastructure. Moreover, encapsulation may raise some issues (which have a limited impact in practice) because it reduces the Maximum Transmission Unit (MTU) size. An important impact of LISP on network operations is related to resiliency and troubleshooting. Indeed, as LISP relies on cached mappings and on encapsulation, troubleshooting is harder than in the traditional Internet. Also, encapsulation stresses resiliency as it makes failure detection and recovery slower than with hop-by-hop routing.

2. LISP in a nutshell

The Locator/Identifier Separation Protocol (LISP) relies on three simple principles: address role separation, encapsulation, and mapping.

Addresses are semantically separated in two: the Routing Locators (RLOCs) and the Endpoint Identifiers (EIDs). RLOCs are addresses typically assigned from the Provider Aggregatable (PA) address space. The EIDs are attributed to the nodes in the edge networks, by block of contiguous addresses, which are typically Provider Independent (PI). To limit the scalability problem, only the routes towards the RLOCs are announced in the Internet routing infrastructure, whereas currently EIDs are also propagated.

LISP routers are used at the boundary between the EID and the RLOC spaces. Routers used to exit the EID space are called Ingress Tunnel Router (ITRs) and those used to enter the EID space the Egress Tunnel Routers (ETRs). When a host sends a packet to a remote destination, it sends it as in the current Internet (without LISP). The packet eventually arrives at the border of its site at an ITR. Because EIDs are not routable on the Internet, the packet is encapsulated with the source address set to the ITR RLOC and the destination address set to the ETR RLOC. The encapsulated packet is then forwarded in the Internet until it reaches the selected ETR. The ETR decapsulates the packet and forwards it to its final destination. The acronym xTR for Ingress/Egress tunnel router is used for a router playing these two roles.

The correspondence between EIDs and RLOCs is given by the mappings. When an ITR needs to find ETR RLOCs that serve an EID it queries a mapping system. It is worth noticing that with the LISP Canonical Address Format (LCAF) [I-D.ietf-lisp-lcaf], LISP is not restricted to the Internet Protocol for the EID addresses. With LCAF, any address type can be used as EID (the address is the key for the mapping lookup) and LISP can then transport, for example, Ethernet frames over the Internet.

A more thorough introduction to LISP can be found in [RFC7215]. The complete specifications are given in [RFC6830], [RFC6833], [I-D.ietf-lisp-ddt], [RFC6836], [RFC6832], [RFC6834].

3. LISP for scaling the Internet

The original goal of LISP is to improve the scalability properties of the Internet architecture. LISP achieves such a target thanks to traffic engineering and stub AS prefixes not announced anymore in the DFZ, so that routing tables are smaller and more stable (i.e., they experience less churn). Furthermore, at the edge network, information necessary to forward packets (i.e., the mappings) is obtained on demand using a pull model (whereas the current Internet uses a push model, instantiated by BGP). Therefore, scalability of edge networks is now independent of the Internet's size and is now related its traffic matrix. This scaling improvement is proven by several works.

Quoitin et al. [QIdLB07] show that the separation between locator and identifier roles at the network level improves the routing scalability by reducing the Routing Information Base (RIB) size (up to one order of magnitude) and increases path diversity and thus the traffic engineering capabilities. For instance, at the time of writting, [CAIDA] list 49,757 ASes among which 85% are stub which means that with LISP the number of ASes advertising prefixes could be reduced by 85%.

In addition, Iannone and Bonaventure [IB07] show that the number of mapping entries that must be handled by an ITR of a campus network with 10,000 users is limited to few tens of thousands, and does not represent more than 3 to 4 Megabytes of memory. Furthermore, they show that the signaling traffic (i.e., Map-Request/Map-Reply packets) is in the same order of magnitude like DNS requests/reply traffic and that the encapsulation overhead, while not negligible, is very limited (in the order of few percentage points of the total traffic volume). Similarly, Kim et al. [KIF11] show that the EID-to-RLOC cache size of an ITR responsible of more than 20,000 residential ADSL users of a large ISP is still in the order of few tens of thousands entries and should not exceed 14 Megabytes. These two studies rely on BGP and traffic traces to determine the number of entries to keep in the EID-to-RLOC cache. In both papers, the size of the cache is inferred from the number of entries by considering that every EID is associated with two or three locators. Saucez [S11] confirms these results by looking at the distribution of the number of locators per EID if LISP were deployed in the 2010's Internet. The assumptions in these studies are:

- o contiguous addresses tend to be used similarly and EID prefixes follow the current BGP prefixes decomposition;
- o EIDs are used only at the stub ASes, not in the transit ASes;
- o the RLOCs of an EID prefix are deployed at the edge between the stubs owning the EID prefix and the providers, allocating the RLOCs in a Provider Aggregetable (PA) mode.

While all previous studies consider the case of a timer-based cache eviction policy (i.e., mappings are deleted from the cache upon timeout), Coras et al. [CCD12] have a more general approach for the Least Recently Used (LRU) eviction policy, proposing an analytic model for the EID-to-RLOC cache size when prefix-level traffic has a stationary generating process. The model shows that miss rate can be accurately predicted from the EID-to-RLOC cache size and a small set of easily measurable traffic parameters. The model was validated using four one-day-long packet traces collected at egress points of a campus network and an academic exchange point considering EID-prefixes as being of BGP-prefix granularity. Consequently, operators can provision the EID-to-RLOC cache of their ITRs according to the miss rate they want to achieve for their given traffic.

Results indicate that for a given target miss-ratio, the size of the cache depends only on the parameters of the popularity distribution, being independent of the number of users (the size of the LISP site) and the number of destinations (the size of the EID-prefix space). Assuming that the popularity distribution remains constant, this means that as the number of users and the number of destinations grow, the cache size needed to obtain a given miss rate remains constant O(1).

Under normal user traffic, miss-ratio decreases at an accelerated pace with cache size and finally settles to a power-law decrease. However, Coras et al. [CDLC] extends the previous model to account for scanning attacks, whereby attackers generate a constant flow of packets according to random scans of the destination prefix space and shows that miss-ratios are very high and independent of the cache size. In fact, if the attack is merely 1% of the legitimate traffic, the miss rate does not drop under 1% as long as the cache cannot accommodate the whole prefix space. Locality measurements also suggested that LRU eviction policy should be close to optimal.

TBD: add a paragraph to explain the operational difference while dealing with a pull model instead of a push.

4. Beyond scaling the Internet

Even though it is its main goal, LISP is more than just a scalability solution, it is also a tool to provide both incoming and outgoing traffic engineering ([S11], [I-D.farinacci-lisp-te]) can be used as an IPv6 transition at the routing level, and for inter-domain multicast ([RFC6831], [I-D.coras-lisp-re]). LISP has also proven to be a good protocol for devices' Internet mobility ([I-D.meyer-lisp-mn]) or even virtual machines' mobility in data centers and multi-tenant VPNs. Details of the last two points are not discussed further because out of the scope of the current LISP Working Group charter.

LISP architecture facilitates routing in environments where there is little to no correlation between network endpoints and topological location. In service provider environment this use is evident in a range of consumer use cases which require an inline anchor in-order to deliver a service to a subscribers. Inline anchors provide one of three types of capabilities:

- enable mobility of subscriber end points
- enable chaining of middle-box functions and services
- enable seamless scale-out of functions

Without LISP operators are forced to centralize service anchors in custom built special boxes. This means that end-points can move as long as their traffic ends up on the same mobile gateway, functions can be chained as long as all traffic traverses the same wire or the same DPI box, and capacity can scale out as long as traffic fans out to/from a specific load balancer.

With LISP service providers are able to distribute, virtualize, and instantiate subscriber-service anchors anywhere in the network. Typical use cases that virtualized inline anchors and network functions include: Distributed Mobility and Virtualized Evolved Packet Core (vEPC), where centralization makes way to distributed and virtualized inline anchoring of mobility, Virtualized Customer Premise Equipment or vCPE, where functionality previously anchored at customer premises is now dynamically allocated in-network, Virtualized SGi LAN, where value added mobile services previously anchored inside full-stack boxes or anchored to physical wires with permutation setups aka "Rails", Virtual IMS and Virtual SBC, etc.

Current deployments by ConteXtream, using a pre standards (designed 2006) based architecture, support a total of 100 millions subscribers with such an architecture. A deployment at a tier-1 US Mobile

operator over 50 millions subscribers provides a 39% download rate improvement over LTE.

4.1. Traffic engineering

In the current (non-LISP) Internet, addresses used by stub networks are globally routable and the routing system distributes the routes to reach these stubs. On the contrary, the EID prefixes of a LISP site are not routable in the DFZ, meaning that mappings are needed in order to determine the list of LISP routers to contact to send them packets. The difference is significant for two reasons. First, packets are not sent to a site but to a specific router. Second, a site can control the entry points for its traffic by controlling its mappings.

For traffic engineering purpose, a mapping associates an EID prefix to a list of RLOCs. Each RLOC is annotated with a priority and a weight. When there are several RLOCs, the ITR selects the one with the highest priority and sends the encapsulated packet to this RLOC. If several such RLOCs exist, then the traffic is balanced proportionally to their weight among the RLOCs with the lowest priority value. Traffic engineering in LISP thus allows the mapping owner to have a fine-grained control on the primary and backup path its incoming and outgoing packets use. In addition, it can share the load among its links. An example of the use of such a feature is described by Saucez et al. [SDIB08], showing how to use LISP to direct different types of traffic on different links having different capacity.

Traffic engineering in LISP goes one step further. As every Map-Request contains the Source EID Address of the packet that caused a cache miss and triggered the Map-Request. It is thus possible for a mapping owner to differentiate the answer (Map-Reply) it gives to Map-Requests based on the requester. This functionality is not available today with BGP because a domain cannot control exactly the routes that will be received by domains that are not in the direct neighborhood.

4.2. LISP for IPv6 Co-existence

The LISP encapsulation mechanism is designed to support any combination of locators and identifiers address family. It is then possible to bind IPv6 EIDs with IPv4 RLOCs and vice-versa. This allows transporting IPv6 packets over an IPv4 network (or IPv4 packets over an IPv6 network), making LISP a valuable mechanism to ease the transition to IPv6.

A not so uncommon example is the case of the network infrastructure of a datacenter being IPv4-only while dual-stack front-end load balancers are used. In this scenario, LISP can be used to provide IPv6 access to servers even though the network and the servers only support IPv4. Assuming that the datacenter's ISP offers IPv6 connectivity, the datacenter only needs to deploy one (or more) xTR(s) at its border with the ISP and one (or more) xTR(s) directly connected to the load balancers. The xTR(s) at the ISP's border tunnels IPv6 packets over IPv4 to the xTR(s) directly attached to the load balancer. The load balancer's xTR decapsulates the packets and forward them to the load balancer, which act as proxies, translating each IPv6 packet into an IPv4. IPv4 packets are then sent to the appropriate servers. Similarly, when the server's response arrives at the load balancer, the packet is translated back into an IPv6 packet and forwarded to its xTR(s), which in turn will tunnel it back, over the IPv4-only infrastructure, to an xTR connected to the ISP. The packet is then decapsulated and forwarded to the ISP natively in IPv6.

4.3. Inter-domain multicast

LISP has native support for multicast [RFC6831]. From the data-plane perspective, at a multicast enabled xTR, an EID sourced multicast packet is encapsulated in another multicast packet and subsequently forwarded in a RLOC-level distribution tree. Therefore, xTRs must participate in both EID and RLOC level distribution trees. Control-plane wise, since group addresses have no topological significance they need not to be mapped. It is worth noting that, to properly function, LISP-Multicast requires that inter-domain multicast be available.

LISP Replication Engineering (RE) ([I-D.coras-lisp-re], [CDM12]) leverage LISP messages ([I-D.farinacci-lisp-mr-signaling]) for multicast state distribution to construct xTR based inter-domain multicast distribution trees when inter-domain multicast support is not available. Simulations of three different management strategies for low latency content delivery show that such overlays can support thousands of member xTRs, hundreds of thousands of end-hosts and deliver content at latencies close to unicast ones ([CDM12]). It was also observed that high client churn has a limited impact on performance and management overhead.

Similarly to LISP-RE, Signal-Free LISP Multicast ([I-D.farinacci-lisp-signal-free-multicast]) can be used when the core network does not provide multicast support. But instead of using signaling to build inter-domain multicast trees, signal-free exclusively leverages the map-server for multicast state storage and distribution. As a result, the source ITR generally performs head-

end replication but it might be also used to emulate LISP-RE distribution trees.

5. Impact of LISP on operations and business model

Important implementation efforts ([IOSNXOS], [OpenLISP], [LISPmob], [LISPClick], [LISPcp], and [LISPfritz]) have been made to assess the specifications and interoperability tests ([Was09]) have been a success. World-wide large deployment in the international lisp4.net testbed, which is currently composed of nodes running at least three different implementations, allows to learn operational matters related to LISP.

We have to distinguish the impact of LISP on LISP sites from the impact on non-LISP sites.

5.1. Impact on non-LISP traffic and sites

LISP has no impact on traffic which has neither LISP origin nor LISP destination. However, LISP can have a significant impact on traffic between a LISP site and a non-LISP site. Traffic between a non-LISP site and a LISP site are subject to the same issues than those observed for LISP-to-LISP traffic but also have issues specific to the transition mechanism that allow LISP site to exchange packets with non-LISP site ([RFC6832], [RFC7215]).

Indeed, the transition requires to setup proxy tunnel routers (PxTRs). PxTRs do not cause particular technical issue. However, by definition proxies cause path stretch and make troubleshooting harder. There are still big questions related to PxTRs that have to be answered:

- Where to deploy PxTRs? The placement in the topology has an important impact on the path stretch.
- How many PxTRs? The number of PxTR has a direct impact on the load and the impact of the failure of a PxTR on the traffic.
- What part of the EID space? Will all the PxTRs be proxies for the whole EID space or will it be segmented between different PxTRs?
- Who operates PxTRs? The IETF does not aim at providing business model hints, however, an important question to answer is related to the entities that will deploy PxTRs, how they will manage their CAPEX/OPEX and how the traffic will be carried with respect for the security and privacy.

PxTR also normally have to advertise in BGP the EID prefix they are proxy for. However, if proxies are managed by different entities, they will belong to different ASes. In this case, we have to be sure that it will not cause MOAS (Multi-Origina AS) issues that could negatively influence routing. Moreover, it is important to ensure that the way EID prefixes will be deaggregated by the proxies will remain reasonable to not take part in the BGP scalability issues.

5.2. Impact on LISP traffic and sites

LISP is a protocol based on the map-and-encap paradigm which has the positive effects that we have given in the sections above. However, by design, LISP also has side impact on operations:

MTU issue: as LISP uses encapsulation, the MTU is reduced, this has implication on potentially all the traffic. However, in practice, on the lisp4.net network, no major issue due to the MTU has been observed. This is probably due to the fact that current end-host stacks are well designed to deal with the problem of MTU.

Resiliency issue: the advantage of flexibility and control offered by the Locator/ID separation comes at the cost of increasing the complexity of the reachability detection. Indeed, identifiers are not directly routable and have to be mapped to locators but a locator may be unreachable while others are still reachable. This is an important problem for any tunnel-based solution. In the current Internet, packets are forwarded independently of the border router of the network meaning that in case of the failure of a border router, another one can be used. With LISP, the destination RLOC specifically designate one particular ETR, hence if this ETR fails, the traffic is dropped even though other ETRs are available for the destination site. Another resiliency issue is linked to the fact that mappings are learned on demand. When an ITR fails, all its traffic is redirected to other ITRs that might not have the mappings requested by the redirected traffic. Existing studies ([SKI12], [SD12]) show, based on measurements and traffic traces, that failure of ITRs and RLOC are infrequent but that when such failure happens, an important number of packet can be dropped. Unfortunately, the current techniques for LISP resiliency, based on monitoring or probing are not rapid enough (failure recovery of the order of a few seconds). To tackle this issue [I-D.bonaventure-lisp-preserve] and [I-D.saucez-lisp-itr-graceful] propose techniques based on local failure detection and recovery.

Middle boxes/filters: because of encapsulation, the middle boxes might not understand the traffic which can cause firewall to drop legitimate packets. In addition, LISP allows triangular or even rectangular routing, so it is hard to maintain a correct state even if the middle box perfectly understands LISP. Finally, filtering might also have problems because they might think only one host is generating the traffic (the ITR), as long as it is not decapsulated. To deal with LISP encapsulation, LISP aware firewalls that inspect inner LISP packets are proposed [lispfirewall].

Troubleshooting/debugging: the major issue that years of LISP experimentation have shown is the difficulty of troubleshooting. When there is a problem in the network, it is hard to pin-point the reason as the operator only has a partial view of the network. The operator can see what is in its EID-to-RLOC cache/database, and can try to obtain what is potentially elsewhere by querying the Map Resolvers but the knowledge remains partial. On top of that, ICMP packets only carry the first few tens of bytes of the original packet, which means that when an ICMP arrives at the ITR, it might not contain enough information to make correct troubleshooting. Interestingly, deployment in the beta network has shown that LISP+ALT was not easy to maintain and control, which explains the migration to LISP-DDT [I-D.ietf-lisp-ddt].

Business: the IETF is not aiming at providing business models. However, even though Iannone et al. [IL10] shown that there is economical incentives to migrate to LISP, some questions are on hold. For example, how will the EIDs be allocated to allow aggregation and hence scalability of the mapping system? Who will operate the mapping system infrastructure and for what benefit?

6. IANA Considerations

This document makes no request to the IANA.

7. Security Considerations

Security and threats analysis of the LISP protocol is out of the scope of the present document. A thorough analysis of LISP security threats is detailed in [I-D.ietf-lisp-threats].

8. Acknowledgments

The people that contributed to this document are Sharon Barkai, Vince Fuller, Joel Halpern, Terry Manderson, and Gregg Schudel.

The work of Luigi Iannone has been partially supported by the ANR-13-INFR-0009 LISP-Lab Project (www.lisp-lab.org).

9. References

9.1. Normative References

[I-D.ietf-lisp-ddt] Fuller, V., Lewis, D., Ermagan, V., and A. Jain, "LISP Delegated Database Tree", draft-ietf-lisp-ddt-02 (work in progress), October 2014.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, January 2013.

[RFC6831] Farinacci, D., Meyer, D., Zwiebel, J., and S. Venaas, "The Locator/ID Separation Protocol (LISP) for Multicast Environments", RFC 6831, January 2013.

[RFC6832] Lewis, D., Meyer, D., Farinacci, D., and V. Fuller, "Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites", RFC 6832, January 2013.

[RFC6833] Fuller, V. and D. Farinacci, "Locator/ID Separation Protocol (LISP) Map-Server Interface", RFC 6833, January 2013.

[RFC6834] Iannone, L., Saucez, D., and O. Bonaventure, "Locator/ID Separation Protocol (LISP) Map-Versioning", RFC 6834, January 2013.

[RFC6836] Fuller, V., Farinacci, D., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol Alternative Logical Topology (LISP+ALT)", RFC 6836, January 2013.

[RFC7215] Jakab, L., Cabellos-Aparicio, A., Coras, F., Domingo-Pascual, J., and D. Lewis, "Locator/Identifier Separation Protocol (LISP) Network Element Deployment Considerations", RFC 7215, April 2014.

9.2. Informative References

[CAIDA] "AS Relationships", http://data.caida.org/datasets/as-relationships/, 2015.

[CCD12] Coras, F., Cabellos-Aparicio, A., and J. Domingo-Pascual, "An Analytical Model for the LISP Cache Size", In Proc. IFIP Networking 2012, May 2012.

[CDLC] Coras, F., Domingo, J., Lewis, D., and A. Cabellos, "An Analytical Model for Loc/ID Mappings Caches", IEEE Transactions on Networking, 2014.

[CDM12] Coras, F., Domingo-Pascual, J., Maino, F., Farinacci, D., and A. Cabellos-Aparicio, "Lcast: Software-defined Inter-Domain Multicast", Elsevier Computer Networks, July 2014.

[I-D.bonaventure-lisp-preserve] Bonaventure, O., Francois, P., and D. Saucez, "Preserving the reachability of LISP ETRs in case of failures", draft-bonaventure-lisp-preserve-00 (work in progress), July 2009.

[I-D.coras-lisp-re] Coras, F., Cabellos-Aparicio, A., Domingo-Pascual, J., Maino, F., and D. Farinacci, "LISP Replication Engineering", draft-coras-lisp-re-06 (work in progress), October 2014.

[I-D.farinacci-lisp-mr-signaling] Farinacci, D. and M. Napierala, "LISP Control-Plane Multicast Signaling", draft-farinacci-lisp-mr-signaling-06 (work in progress), February 2015.

[I-D.farinacci-lisp-signal-free-multicast] Moreno, V. and D. Farinacci, "Signal-Free LISP Multicast", draft-farinacci-lisp-signal-free-multicast-02 (work in progress), December 2014.

[I-D.farinacci-lisp-te] Farinacci, D., Kowal, M., and P. Lahiri, "LISP Traffic Engineering Use-Cases", draft-farinacci-lisp-te-07 (work in progress), September 2014.

[I-D.ietf-lisp-lcaf] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", draft-ietf-lisp-lcaf-07 (work in progress), December 2014.

[I-D.ietf-lisp-threats]
Saucez, D., Iannone, L., and O. Bonaventure, "LISP Threats Analysis", draft-ietf-lisp-threats-12 (work in progress), March 2015.

[I-D.meyer-lisp-mn]
Farinacci, D., Lewis, D., Meyer, D., and C. White, "LISP Mobile Node", draft-meyer-lisp-mn-12 (work in progress), January 2015.

[I-D.saucez-lisp-itr-graceful]
Saucez, D., Bonaventure, O., Iannone, L., and C. Filsfils, "LISP ITR Graceful Restart", draft-saucez-lisp-itr-graceful-03 (work in progress), December 2013.

[IB07] Iannone, L. and O. Bonaventure, "On the cost of caching locator/id mappings", In Proc. ACM CoNEXT 2007, December 2007.

[IL10] Iannone, L. and T. Leva, "Modeling the economics of Loc/ID Separation for the Future Internet", Book Chapter, Towards the Future Internet - Emerging Trends from the European Research, IOS Press, May 2010.

[IOSNXOS] Cisco Systems Inc., , "Locator/ID Separation Protocol (LISP)", http://lisp4.cisco.com, 2013.

[KIF11] Kim, J., Iannone, L., and A. Feldmann, "Deep dive into the lisp cache and what isps should know about it", In Proc. IFIP Networking 2011, May 2011.

[LISPClick]
Saucez, D. and V. Nguyen, "LISP-Click: A Click implementation of the Locator/ID Separation Protocol", 1st Symposium on Click Modular Router, 2009, November 2009.

[LISPcp] "The lip6-lisp Project", https://github.com/lip6-lisp/, 2014.

[LISPfritz]
"Unsere FRITZ!Box-Produkte", http://avm.de/produkte/fritzbox/, 2014.

[LISPmob] "LISP Mobile Node for Linux", http://lispmob.org, 2013.

[OpenLISP]
"The OpenLISP Project", http://www.openlisp.org, 2013.

[QIdLB07] Quoitin, B., Iannone, L., de Launois, C., and O. Bonaventure, "Evaluating the benefits of the locator/identifier separation", In Proc. ACM MobiArch 2007, May 2007.

[S11] Saucez, D., "Mechanisms for Interdomain Traffic Engineering with LISP", PhD Thesis, Universite catholique de Louvain, 2011, October 2011.

[SD12] Saucez, D. and B. Donnet, "On the Dynamics of Locators in LISP", In Proc. IFIP Networking 2012, May 2012.

[SDIB08] Saucez, D., Donnet, B., Iannone, L., and O. Bonaventure, "Interdomain Traffic Engineering in a Locator/Identifier Separation Context", In Proc. of Internet Network Management Workshop, 2008, October 2008.

[SKI12] Saucez, D., Kim, J., Iannone, L., Bonaventure, O., and C. Filsfils, "A Local Approach to Fast Failure Recovery of LISP Ingress Tunnel Routers", In Proc. IFIP Networking 2012, May 2012.

[Was09] Wasserman, M., "LISP Interoperability Testing", IETF 76, LISP WG presentation, 2009., November 2009.

[lispfirewall] "LISP and Zone-Based Firewalls Integration and Interoperability", http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/sec_data_zbf/configuration/xe-3s/sec-data-zbf-xe-book/sec-zbf-lisp-inner-pac-insp.html, 2014.

Authors' Addresses

Damien Saucez
INRIA
2004 route des Lucioles BP 93
06902 Sophia Antipolis Cedex
France

Email: damien.saucez@inria.fr

Luigi Iannone
Telecom ParisTech
23, Avenue d'Italie, CS 51327
75214 PARIS Cedex 13
France

Email: luigi.iannone@telecom-paristech.fr

Albert Cabellos
Technical University of Catalonia
C/Jordi Girona, s/n
08034 Barcelona
Spain

Email: fcoras@ac.upc.edu

Florin Coras
Technical University of Catalonia
C/Jordi Girona, s/n
08034 Barcelona
Spain

Email: fcoras@ac.upc.edu

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 6, 2015

D. Saucez
INRIA
L. Iannone
Telecom ParisTech
O. Bonaventure
Universite catholique de Louvain
March 5, 2015

LISP Threats Analysis
draft-ietf-lisp-threats-12.txt

Abstract

This document proposes a threat analysis of the Locator/Identifier Separation Protocol (LISP).

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 September 6, 2015.

Copyright Notice

Table of Contents

1. Introduction

The Locator/ID Separation Protocol (LISP) is specified in [RFC6830]. The present document assess the potential security threats identified

in the LISP specifications if LISP is deployed in the Internet (i.e., a public non-trustable environment).

The document is composed of three main parts: the first defines the general threat model that attackers can follow to mount attacks. The second describes the techniques based on the LISP protocol and architecture that attackers can use to construct attacks. The third discusses mitigation techniques and general solutions to protect the LISP protocol and architecture from attacks.

This document does not consider all the possible uses of LISP as discussed in [RFC6830] and [RFC7215]. The document focuses on LISP unicast, including as well LISP Interworking [RFC6832], LISP-MS [RFC6833], and LISP Map-Versioning [RFC6834]. The reading of these documents is a prerequisite for understanding the present document.

This document assumes a generic IP service and does not discuss the difference, from a security viewpoint, between using IPv4 or IPv6.

2. Threat model

This document assumes that attackers can be located anywhere in the Internet (either in LISP sites or outside LISP sites) and that attacks can be mounted either by a single attacker or by the collusion of several attackers.

An attacker is a malicious entity that performs the action of attacking a target in a network where LISP is (partially) deployed by leveraging the LISP protocol and/or architecture.

An attack is the action of performing an illegitimate action on a target in a network where LISP is (partially) deployed.

The target of an attack is the entity (i.e., a device connected to the network or a network) that is aimed to undergo the consequences of an attack. Other entities can potentially undergo side effects of an attack, even though they are not directly targeted by the attack. The target of an attack can be selected specifically, i.e., a particular entity, or arbitrarily, i.e., any entity. Finally, an attacker can aim at attacking one or several targets with a single attack.

Section 2.1 specifies the different modes of operation that attackers can follow to mount attacks and Section 2.2 specifies the different categories of attacks that attackers can build.

2.1. Attacker's Operation Modes

Attackers can be classified according to the following four modes of operation, i.e., the temporal and spacial diversity of the attacker.

2.1.1. On-path vs. Off-path Attackers

On-path attackers, also known as Men-in-the-Middle, are able to intercept and modify packets between legitimate communicating entities. On-path attackers are located either directly on the normal communication path (either by gaining access to a node on the path or by placing themselves directly on the path) or outside the location path but manage to deviate (or gain a copy of) packets sent between the communication entities. On-path attackers hence mount their attacks by modifying packets initially sent legitimately between communication entities.

An attacker is called off-path attacker if it does not have access to packets exchanged during the communication or if there is no communication. In order for their attacks to succeed, off-path attackers must hence generate packets and inject them in the network.

2.1.2. Internal vs. External Attackers

An internal attacker launches its attack from a node located within a legitimate LISP site. Such an attacker is either a legitimate node of the site or it exploits a vulnerability to gain access to a legitimate node in the site. Because of their location, internal attackers are trusted by the site they are in.

On the contrary, an external attacker launches its attacks from the outside of a legitimate LISP site.

2.1.3. Live vs. Time-shifted attackers

A live attacker mounts attacks for which it must remain connected as long as the attack is mounted. In other words, the attacker must remain active for the whole duration of the attack. Consequently, the attack ends as soon as the attacker (or the used attack vector) is neutralized.

On the contrary, a time-shifted attacker mounts attacks that remain active after it disconnects from the Internet.

2.1.4. Control-plane vs. Data-plane attackers

A control-plane attacker mounts its attack by using control-plane functionalities, typically the mapping system.

A data-plane attacker mounts its attack by using data-plane functionalities.

As there is no complete isolation between the control-plane and the data-plane, an attacker can operate in the control-plane (resp. data-plane) to mount attacks targeting the data-plane (resp. control-plane) or keep the attacked and targeted planes at the same layer (i.e., from control-plane to control-plane or from data-plane to data-plane).

2.1.5. Cross mode attackers

The attacker modes of operation are not mutually exclusive and hence attackers can combine them to mount attacks.

For example, an attacker can launch an attack using the control-plane directly from within a LISP site to which it got temporary access (i.e., internal + control-plane attacker) to create a vulnerability on its target and later on (i.e., time-shifted + external attacker) mount an attack on the data plane (i.e., data-plane attacker) that leverages the vulnerability.

2.2. Threat categories

Attacks can be classified according to the nine following categories.

2.2.1. Replay attack

A replay attack happens when an attacker retransmits at a later time, and without modifying it, a packet (or a sequence of packets) that has already been transmitted.

2.2.2. Packet manipulation

A packet manipulation attack happens when an attacker receives a packet, modifies the packet (i.e., changes some information contained in the packet) and finally transmits the packet to its final destination that can be the initial destination of the packet or a different one.

2.2.3. Packet interception and suppression

In a packet interception and suppression attack, the attacker captures the packet and drops it before it can reach its final destination.

2.2.4. Spoofing

With a spoofing attack, the attacker injects packets in the network pretending being another node. Spoofing attacks are made by forging source addresses in packets.

It should be noted that with LISP, packet spoofing is similar to any other existing tunneling technology currently deployed in the Internet. Generally the term "spoofed packet" indicates a packet containing a source IP address that is not the one of the actual originator of the packet. Hence, since LISP uses encapsulation, the spoofed address could be in the outer header as well as in the inner header, this translates in two types of spoofing.

Inner address spoofing: the attacker uses encapsulation and uses a spoofed source address in the inner packet. In case of data-plane LISP encapsulation, that corresponds to spoof the source EID address of the encapsulated packet.

Outer address spoofing: the attacker does not use encapsulation and spoofs the source address of the packet. In case of data-plane LISP encapsulation, that corresponds to spoof the source RLOC address of the encapsulated packet.

Note that the two types of spoofing are not mutually exclusive, rather all combinations are possible and could be used to perform different kind of attacks. For example, an attacker outside a LISP site can generate a packet with a forged source IP address (i.e., outer address spoofing) and forward it to a LISP destination. The packet is then eventually encapsulated by a PITR so that once encapsulated the attack corresponds to a inner address spoofing. One can also imagine an attacker forging a packet with encapsulation where both inner an outer source addresses are spoofed.

It is important to notice that the combination of inner and outer spoofing makes the identification of the attacker complex as the packet may not contain information that allows to detect the origin of the attack.

2.2.5. Rogue attack

In a rogue attack the attacker manages to appear as a legitimate source of information, without faking its identity (as opposed to a spoofing attacker).

2.2.6. Denial of Service (DoS) attack

A Denial of Service (DoS) attack aims at disrupting a specific targeted service to make it unable to operate properly.

2.2.7. Performance attack

A performance attacks aims at exploiting computational resources (e.g., memory, processor) of a targeted node so to make it unable to operate properly.

2.2.8. Intrusion attack

In an intrusion attack the attacker gains remote access to a resource (e.g., a host, a router, or a network) or information that it normally doesn’t have access to. Intrusion attacks can lead to privacy leakages.

2.2.9. Amplification attack

In an amplification attack, the traffic generated by the target of the attack in response to the attack is larger than the traffic that the attacker must generate.

In some cases, the data-plane can be several order of magnitude faster than the control-plane at processing packets. This difference can be exploited to overload the control-plane via the data-plane without overloading the data-plane.

2.2.10. Multi-category attacks

Attacks categories are not mutually exclusive and any combination can be used to perform specific attacks.

For example, one can mount a rogue attack to perform a performance attack starving the memory of an ITR resulting in a DoS on the ITR.

3. Attack vectors

This section presents techniques that can be used by attackers in order to succeed attacks leveraging the LISP protocol and/or architecture.

3.1. Gleaning

To reduce the time required to obtain a mapping, the optional gleaning mechanism allows an xTR to directly learn a mapping from the LISP data encapsulated packets and the Map-Request packets that it receives. LISP encapsulated data packets contain a source RLOC, destination RLOC, source EID and destination EID. When an xTR receives an encapsulated data packet coming from a source EID for which it does not already know a mapping, it may insert the mapping between the source RLOC and the source EID in its EID-to-RLOC Cache. The same technique can be used when an xTR receives a Map-Request as the Map-Request also contains a source EID address and a source RLOC. Once a gleaned entry has been added to the EID-to-RLOC cache, the xTR sends a Map-Request to retrieve the actual mapping for the gleaned EID from the mapping system.

If a packet injected by an off-path attacker and with a spoofed inner address is gleaned by an xTR then the attacker may divert the traffic meant to be delivered to the spoofed EID as long as the gleaned entry is used by the xTR. This attack can be used as part of replay, packet manipulation, packet interception and suppression, or DoS attacks as the packets are sent to the attacker.

If the packet sent by the attacker contains a spoofed outer address instead of a spoofed inner address then it can achieve a DoS or a performance attack as the traffic normally destined to the attacker will be redirected to the spoofed source RLOC. Such traffic may overload the owner of the spoofed source RLOC, preventing it from operating properly.

If the packet injected uses both inner and outer spoofing, the attacker can achieve a spoofing, a performance, or an amplification attack as traffic normally destined to the spoofed EID address will be sent to the spoofed RLOC address. If the attacked LISP site also generates traffic to the spoofed EID address, such traffic may have a positive amplification factor.

A gleaning attack does not only impact the data-plane but can also have repercussions on the control-plane as a Map-Request is sent after the creation of a gleaned entry. The attacker can then achieve DoS and performance attacks on the control-plane. For example, if an attacker sends a packet for each address of a prefix not yet cached in the EID-to-RLOC cache of an xTR, the xTR will potentially send a Map-Request for each such packet until the mapping is installed which leads to an over-utilisation of the control-plane as each packet generates a control-plane event. In order for this attack to succeed, the attacker may not need to use spoofing. This issue can occur even if gleaning is turned off since whether or not gleaning is

used the ITR may need to send a Map-Request in response to incoming packets whose EID is not currently in the cache.

Gleaning attacks are fundamentally involving a time-shifted mode of operation as the attack may last as long as the gleaned entry is kept by the targeted xTR. RFC 6830 [RFC6830] recommends to store the gleaned entries for only a few seconds which limits the duration of the attack.

Gleaning attacks always involve external data-plane attackers but results in attacks on either the control-plane or data-plane.

It is worth to notice that the outer spoofed address does not need to be the RLOC of a LISP site an may be any address.

3.2. Locator Status Bits

When the L bit in the LISP header is set to 1, it indicates that the second 32-bits longword of the LISP header contains the Locator Status Bits. In this field, each bit position reflects the status of one of the RLOCs mapped to the source EID found in the encapsulated packet. The reaction of a LISP xTR that receives such a packet is left as operational choice in [RFC6830].

When an attacker sends a LISP encapsulated packet with a crafted LSB to an xTR, it can influence the xTR's choice of the locators for the prefix associated to the source EID. In case of an off-path attacker, the attacker must inject a forged packet in the network with a spoofed inner address. An on-path attacker can manipulate the LSB of legitimate packets passing through it and hence does not need to use spoofing. Instead of manipulating the LSB field, an on-path attacker can also obtain the same result of injecting packets with invalid LSB values by replaying packets.

The LSB field can be leveraged to mount a DoS attack by either declaring all RLOCs as unreachable (all LSB set to 0), or by concentrating all the traffic to one RLOC (e.g., all but one LSB set to 0) and hence overloading the RLOC concentrating all the traffic from the xTR, or by forcing packets to be sent to RLOCs that are actually not reachable (e.g., invert LSB values).

The LSB field can also be used to mount a replay, a packet manipulation, or a packet interception and suppression attack. Indeed, if the attacker manages to be on the path between the xTR and one of the RLOCs specified in the mapping, forcing packets to go via that RLOC implies that the attacker will gain access to the packets.

Attacks using the LSB are fundamentally involving a time-shifted mode of operation as the attack may last as long as the reachability information gathered from the LSB is used by the xTR to decide the RLOCs to be used.

3.3. Map-Version

When the Map-Version bit of the LISP header is set to 1, it indicates that the low-order 24 bits of the first 32 bits longword of the LISP header contain a Source and Destination Map-Version. When a LISP xTR receives a LISP encapsulated packet with the Map-Version bit set to 1, the following actions are taken:

- It compares the Destination Map-Version found in the header with the current version of its own configured EID-to-RLOC mapping, for the destination EID found in the encapsulated packet. If the received Destination Map-Version is smaller (i.e., older) than the current version, the ETR should apply the SMR procedure described in [RFC6830] and send a Map-Request with the SMR bit set.

- If a mapping exists in the EID-to-RLOC Cache for the source EID, then it compares the Map-Version of that entry with the Source Map-Version found in the header of the packet. If the stored mapping is older (i.e., the Map-Version is smaller) than the source version of the LISP encapsulated packet, the xTR should send a Map-Request for the source EID.

A cross-mode attacker can use the Map-Version bit to mount a DoS attack, an amplification attack, or a spoofing attack. For instance if the mapping cached at the xTR is outdated, the xTR will send a Map-Request to retrieve the new mapping which can yield to a DoS attack (by excess of signalling traffic) or an amplification attack if the data-plane packet sent by the attacker is smaller, or otherwise uses fewer resources, than the control-plane packets sent in response to the attacker’s packet. With a spoofing attack and if the xTR considers that the spoofed ITR has an outdated mapping, it will send an SMR to the spoofed ITR which can result in performance, amplification, or DoS attack as well.

Map-Version attackers are inherently cross mode as the Map-Version is a method to put control information in the data-plane. Moreover, this vector involves live attackers. Nevertheless, on-path attackers do not take specific advantage over off-path attackers.

3.4. Routing Locator Reachability

The Nonce-Present and Echo-Nonce bits in the LISP header are used to verify the reachability of an xTR. A testing xTR sets the Echo-Nonce and the Nonce-Present bits in LISP data encapsulated packets and include a random nonce in the LISP header of packets. Upon reception of these packets, the tested xTR stores the nonce and echo it whenever it returns a LISP encapsulated data packets to the testing xTR. The reception of the echoed nonce confirms that the tested xTR is reachable.

An attacker can interfere with the reachability test by sending two different types of packets:

1. LISP data encapsulated packets with the Nonce-Present bit set and a random nonce. Such packets are normally used in response to a reachability test.

2. LISP data encapsulated packets with the Nonce-Present and the Echo-Nonce bits both set. These packets will force the receiving ETR to store the received nonce and echo it in the LISP encapsulated packets that it sends. These packets are normally used as trigger for a reachability test.

The first type of packets is used to make xTRs think that an other xTR is reachable while it is not. It is hence a way to mount a DoS attack (i.e., the ITR will send its packet to a non-reachable ETR while it should use another one).

The second type of packets could be exploited to attack the nonce-based reachability test. If the attacker sends a continuous flow of packets that each have a different random nonce, the ETR that receives such packets will continuously change the nonce that it returns to the remote ITR, which can yield to a performance attack. If the remote ITR tries a nonce-reachability test, this test may fail because the ETR may echo an invalid nonce. This hence yields to a DoS attack.

In the case of an on-path attacker, a packet manipulation attack is necessary to mount the attack. To mount such an attack, an off-path attacker must mount an outer address spoofing attack.

If an xTR chooses to periodically check with active probes the liveness of entries in its EID-to-RLOC cache (as described in section 6.3 of [RFC6830]), then this may amplify the attack that caused the insertion of entries being checked.

3.5. Instance ID

LISP allows to carry in its header a 24-bits value called Instance ID and used on the ITR to indicate which local Instance ID has been used for encapsulation, while on the ETR the instance ID decides the forwarding table to use to forward the decapsulated packet in the LISP site.

An attacker (either a control-plane or data-plane attacker) can use the instance ID functionality to mount an intrusion attack.

3.6. Interworking

[RFC6832] defines Proxy-ITR and Proxy-ETR network elements to allow LISP and non-LISP sites to communicate. The Proxy-ITR has functionality similar to the ITR, however, its main purpose is to encapsulate packets arriving from the DFZ in order to reach LISP sites. A Proxy-ETR has functionality similar to the ETR, however, its main purpose is to inject de-encapsulated packets in the DFZ in order to reach non-LISP Sites from LISP sites. As a PITR (resp. PETR) is a particular case of ITR (resp. ETR), it is subject to same attacks than ITRs (resp. ETR).

As any other system relying on proxies, LISP interworking can be used by attackers to hide their exact origin in the network.

3.7. Map-Request messages

A control-plane off-path attacker can exploit Map-Request messages to mount DoS, performance, or amplification attacks. By sending Map-Request messages at high rate, the attacker can overload nodes involved in the mapping system. For instance sending Map-Requests at high rate can considerably increase the state maintained in a Map-Resolver or consume CPU cycles on ETRs that have to process the Map-Request packets they receive in their slow path (i.e., performance or DoS attack). When the Map-Reply packet is larger than the Map-Request sent by the attacker, that yields to an amplification attack. The attacker can combine the attack with a spoofing attack to overload the node to which the spoofed address is actually attached.

It is worth to notice that if the attacker sets the P bit (Probe Bit) in the Map-Request, it is legitimate the send the Map-Request directly to the ETR instead of passing through the mapping system.

The SMR bit can be used to mount a variant of these attacks.

For efficiency reasons, Map-Records can be appended to Map-Request messages. When an xTR receives a Map-Request with appended Map-

Records, it does the same operations as for the other Map-Request messages and is so subject to the same attacks. However, it also installs in its EID-to-RLOC cache the Map-Records contained in the Map-Request. An attacker can then use this vector to force the installation of mappings in its target xTR. Consequently, the EID-to-RLOC cache of the xTR is polluted by potentially forged mappings allowing the attacker to mount any of the attacks categorized in Section 2.2 (see Section 3.8 for more details). It is worth to mention that the attacker does not need to forge the mappings present in the Map-Request to achieve a performance or DoS attack. Indeed, if the attacker owns a large enough EID prefix it can de-aggregate it in many small prefixes, each corresponding to another mapping and it installs them in the xTR cache by mean of the Map-Request.

Moreover, attackers can use Map Resolver and/or Map Server network elements to relay its attacks and hide the origin of the attack. Indeed, on the one hand, a Map Resolver is used to dispatch Map-Request to the mapping system and, on the other hand, a Map Server is used to dispatch Map-Requests coming from the mapping system to ETRs that are authoritative for the EID in the Map-Request.

3.8. Map-Reply messages

Most of the security of the Map-Reply messages depends on the 64 bits nonce that is included in a Map-Request and returned in the Map-Reply. If an ETR does not accept Map-Reply messages with an invalid nonce, the risk of an off-path attack is limited given the size of the nonce (64 bits). Nevertheless, the nonce only confirms that the Map-Reply received was sent in response to a Map-Request sent, it does not validate the contents of that Map-Reply.

If an attacker manages to send a valid (i.e., in response to a Map-Request and with the correct nonce) Map-Reply to an ITR, then it can perform any of the attack categorised in Section 2.2 as it can inject forged mappings directly in the ITR EID-to-RLOC cache. For instance, if the mapping injected to the ITR points to the address of a node controlled by the attacker, it can mount replay, packet manipulation, packet interception and suppression, or DoS attacks as it will receive every packet destined to a destination lying in the EID prefix of the injected mapping. In addition, the attacker can inject plethora of mappings in the ITR to mount a performance attack by filling up the EID-to-RLOC cache of the ITR. If the attacker can also mount an amplification attack as soon as the ITR has to send a lot of packets to the EIDs matching the injected mapping. In this case, the RLOC address associated to the mapping is the address of the real target of the attacker and all the traffic of the ITR will be sent to the target which means that with one single packet the attacker may generate very high traffic towards its final target.

If the attacker is a valid ETR in the system it can mount a rogue attack if it uses prefixes over-claiming. In such a scenario, the attacker ETR replies to a legitimate Map-Request message it received with a Map-Reply message that contains an EID-Prefix that is larger than the prefix owned by the attacker. For instance if the owned prefix is 192.0.2.0/25 but the Map-Reply contains a mapping for 192.0.2.0/24, then the mapping will influence packets destined to other EIDs than the one attacker has authority on. With such technique, the attacker can mount the attacks presented above as it can (partially) control the mappings installed on its target ITR. To force its target ITR to send a Map-Request, nothing prevents the attacker to initiate some communication with the ITR. This method can be used by internal attackers that want to control the mappings installed in their site. To that aim, they simply have to collude with an external attacker ready to over-claim prefixes on behalf of the internal attacker.

It is worth to notice that when the Map-Reply is in response to a Map-Request sent via the mapping system (i.e., not send directly from the ITR to an ETR), the attacker does not need to use a spoofing attack to achieve its attack as by design the source IP address of a Map-Reply is not known in advance by the ITR.

Map-Request and Map-Reply messages are exposed to any type of attackers, on-path or off-path but also external or internal attackers. Also, even though they are control message, they can be leveraged by data-plane attackers. As the decision of removing mappings is based on the TTL indicated in the mapping, time-shifted attackers can take benefit of injecting forged mappings as well.

3.9. Map-Register messages

Map-Register messages are sent by ETRs to Map Servers to indicate to the mapping system the EID prefixes associated to them. The Map-Register message provides an EID prefix and the list of ETRs that are able to provide Map-Replies for the EID covered by the EID prefix.

As Map-Register messages are protected by an authentication mechanism, only a compromised ETR can register itself to its allocated Map Server.

A compromised ETR can over-claim the prefix it owns in order to influence the route followed by Map-Requests for EIDs outside the scope of its legitimate EID prefix (see Section 3.8 for the list of over-claiming attacks).

A compromised ETR can also de-aggregate its EID prefix in order to register more EID prefixes than necessary to its Map Servers (see

Section 3.7 for the impact of de-aggregation of prefixes by an attacker).

Similarly, a compromised Map Server can accept invalid registration or advertise invalid EID prefix to the mapping system.

3.10. Map-Notify messages

Map-Notify messages are sent by a Map Server to an ETR to acknowledge the good reception and processing of a Map-Register message.

Similarly to the pair Map-Request/Map-Reply, the pair Map-Register/ Map-Notify is protected by a nonce making it hard for an attacker to inject a falsified notification to an ETR to make this ETR believe that the registration succeeded while it has not.

4. Note on Privacy

As presented by [RFC6973], universal privacy considerations are impossible to establish as the privacy definition may vary from one to another. As a consequence, this document does not aim at identifying privacy issues related to the LISP protocol but it is necessary to highlight that security threats identified in this document could play a role in privacy threats as defined in section 5 of [RFC6973].

Like public deployments of any other control plane protocols, in an Internet deployment mappings are public and hence provide information about the infrastructure and reachability of LISP sites (i.e., the addresses of the edge routers). Depending upon deployment details, LISP map replies might or might not provide finer grained and more detailed information than is available with currently deployed routing and control protocols.

5. Threats Mitigation

Most of threats can be mitigated with careful deployment and configuration (e.g., filter) and also by applying the general rules in security that consist in activating only features that are necessary in the deployment and verifying the validity of the information obtained from third parties.

The control-plane is the most critical part of LISP from a security viewpoint and it is worth to notice that the specifications already offer authentication mechanism for mappings registration ([RFC6833]) and this mechanism combined with LISP-SEC [I-D.ietf-lisp-sec] strongly mitigates threats in non-trustable environments such as the Internet. Moreover, LISP specifications define an authentication

data field for Map-Request messages and Encapsulated Control messages without specifying how to use it [RFC6830]. The presence of this field in the specifications allows to propose a general authentication mechanisms for the LISP control-plane while staying backward compatible. The exact technique still has to be designed and defined. To maximally mitigate the threats on the mapping system, authentication must be used, whenever possible, for both Map-Request and Map-Reply messages and for messages exchanged internally among elements of the mapping system, such as specified in [I-D.ietf-lisp-sec] and [I-D.ietf-lisp-ddt].

Systematically applying filters and rate-limitation, as proposed in [RFC6830], mitigates most of the threats presented in this document. In order to minimise the risk of overloading the control-plane with actions triggered from data-plane events, such actions should be rate limited.

Finally, all information opportunistically learned (e.g., with LSB or gleaning) should be used with care until they are verified. For instance, a reachability change learned with LSB should not be used directly to decide the destination RLOC, but instead should trigger a rate-limited reachability test. Similarly, a gleaned entry should be used only for the flow that triggered the gleaning procedure until the gleaned entry has been verified [Trilogy].

6. Security Considerations

This entire document is dedicated to threat analysis and mitigation of the Locator/Identifier Separation Protocol, aiming at helping to understand the security risks at stake, and how to mitigate them, while deploying LISP in non-trustable environments.

7. IANA Considerations

This document makes no request to IANA.

8. Acknowledgments

This document builds upon the draft of Marcelo Bagnulo ([I-D.bagnulo-lisp-threat]), where the flooding attack and the reference environment were first described.

The authors would like to thank Ronald Bonica, Albert Cabellos, Ross Callon, Noel Chiappa, Florin Coras, Vina Ermagan, Dino Farinacci, Stephen Farrell, Joel Halpern, Emily Hiltzik, Darrel Lewis, Edward Lopez, Fabio Maino, Terry Manderson, and Jeff Wheeler for their comments.

This work has been partially supported by the INFSO-ICT-216372 TRILOGY Project (www.trilogy-project.org).

The work of Luigi Iannone has been partially supported by the ANR-13-INFR-0009 LISP-Lab Project (www.lisp-lab.org) and the EIT KIC ICT-Labs SOFNETS Project.

9. References

9.1. Normative References

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, January 2013.

[RFC6832] Lewis, D., Meyer, D., Farinacci, D., and V. Fuller, "Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites", RFC 6832, January 2013.

[RFC6833] Fuller, V. and D. Farinacci, "Locator/ID Separation Protocol (LISP) Map-Server Interface", RFC 6833, January 2013.

[RFC6834] Iannone, L., Saucez, D., and O. Bonaventure, "Locator/ID Separation Protocol (LISP) Map-Versioning", RFC 6834, January 2013.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, July 2013.

9.2. Informative References

[I-D.bagnulo-lisp-threat] Bagnulo, M., "Preliminary LISP Threat Analysis", draft-bagnulo-lisp-threat-01 (work in progress), July 2007.

[I-D.ietf-lisp-ddt] Fuller, V., Lewis, D., Ermagan, V., and A. Jain, "LISP Delegated Database Tree", draft-ietf-lisp-ddt-02 (work in progress), October 2014.

[I-D.ietf-lisp-sec] Maino, F., Ermagan, V., Cabellos-Aparicio, A., and D. Saucez, "LISP-Security (LISP-SEC)", draft-ietf-lisp-sec-07 (work in progress), October 2014.

[RFC7215] Jakab, L., Cabellos-Aparicio, A., Coras, F., Domingo-Pascual, J., and D. Lewis, "Locator/Identifier Separation Protocol (LISP) Network Element Deployment Considerations", RFC 7215, April 2014.

[Trilogy] Saucez, D. and L. Iannone, "How to mitigate the effect of scans on mapping systems", Trilogy Future Internet Summer School., 2009.

Appendix A. Document Change Log

- Version 12 Posted March 2015.
  - Addressed comments by Ross Callon on the mailing list (http://www.ietf.org/mail-archive/web/lisp/current/msg05829.html).
  - Addition of a section discussing mitigation techniques for deployments in non-trustable environments.
- Version 11 Posted December 2014.
  - Editorial polishing. Clarifications added in few points.
- Version 10 Posted July 2014.
  - Document completely remodeled according to the discussions on the mailing list in the thread http://www.ietf.org/mail-archive/web/lisp/current/msg05206.html and to address comments from Ronald Bonica and Ross Callon.
- Version 09 Posted March 2014.
  - Updated document according to the review of A. Cabellos.
- Version 08 Posted October 2013.
  - Addition of a privacy consideration note.
  - Editorial changes
- Version 07 Posted October 2013.
  - This version is updated according to the thorough review made during October 2013 LISP WG interim meeting.
  - Brief recommendations put in the security consideration section.

    * Editorial changes

- Version 06 Posted October 2013.

    * Complete restructuration, temporary version to be used at October 2013 interim meeting.

- Version 05 Posted August 2013.

    * Removal of severity levels to become a short recommendation to reduce the risk of the discussed threat.

- Version 04 Posted February 2013.

    * Clear statement that the document compares threats of public LISP deployments with threats in the current Internet architecture.

    * Addition of a severity level discussion at the end of each section.

    * Addressed comments from V. Ermagan and D. Lewis' reviews.

    * Updated References.

    * Further editorial polishing.

- Version 03 Posted October 2012.

    * Dropped Reference to RFC 2119 notation because it is not actually used in the document.

    * Deleted future plans section.

    * Updated References

    * Deleted/Modified sentences referring to the early status of the LISP WG and documents at the time of writing early versions of the document.

    * Further editorial polishing.

    * Fixed all ID nits.

- Version 02 Posted September 2012.

    * Added a new attack that combines over-claiming and de-aggregation (see Section 3.8).

* Editorial polishing.

- Version 01 Posted February 2012.
  * Added discussion on LISP-DDT.
- Version 00 Posted July 2011.
  * Added discussion on LISP-MS>.
  * Added discussion on Instance ID.
  * Editorial polishing of the whole document.
  * Added "Change Log" appendix to keep track of main changes.
  * Renamed "draft-saucez-lisp-security-03.txt.

Authors' Addresses

Damien Saucez
INRIA
2004 route des Lucioles BP 93
06902 Sophia Antipolis Cedex
France

Email: damien.saucez@inria.fr

Luigi Iannone
Telecom ParisTech
23, Avenue d'Italie, CS 51327
75214 PARIS Cedex 13
France

Email: luigi.iannone@telecom-paristech.fr

Olivier Bonaventure
Universite catholique de Louvain
Place St. Barbe 2
Louvain la Neuve
Belgium

Email: olivier.bonaventure@uclouvain.be