IETF86-KARP

Key Management and Adjacency Management for KARP-based Routing Systems

J. William Atwood
Revathi Bangalore Somanatha
Concordia University, Montreal

Overall Structure in akam-rp

- Layer 1: GCKS <-> each router
  - Step 1: mutual authentication between an individual router and the GCKS
  - Step 2: push key management configuration information to each individual router
- Layer 2: router to neighbors
  - Step 3: mutual authentication between a router and its neighbors, possibly using information supplied by the GCKS
  - Step 4: push or negotiate keys, etc.

--

- The local routers retain key management information across re-boots, to avoid any possible issues with (apparent) DoS attacks on the GCKS when recovering from a general power failure
- (We put forth this architecture in Vancouver, based on a new protocol derived from IKE/gdoi)

Scope of keys

- akam-rp
  - More than one possible scope for keys:
    - Entire administrative area
    - Routers on a network segment
    - This router plus its immediate neighbor routers
    - This router plus its neighbor routers on an interface
    - This router and a single peer router
- karp-ops-model
  - More than one possible scope for keys
  - Required scope may be fixed by the protocol spec
  - Any AKM **must** enforce this

Relationship of akam-rp to other karp drafts

- RKMP
  - RKMP works out the details of key management for steps 3 and 4, for the case where the key scope is “single peer”
- MaRK (MRKMP)
  - MaRK works out the details for steps 3 and 4, for the case where the key scope is “neighbor routers on an interface”

Configuration management

- karp-ops-model
  - There are many other things that may need to be considered/configured/controlled:
    - Key table consistency
    - Key update rules
    - Key derivation rules
    - Naming of peer groups
    - Fault handling
    - Upgrade rules
  - Routing security may be considered to be “just one more set of configuration parameters”

Framework relevance

- The ***framework*** proposed in akam-rp may be right, but …
  - The amount of information required to manage and configure keys is actually quite large
  - Defining a “new protocol” (i.e., an extension of IKE/ gdoi) to transfer the key management information may not have been the best idea, in the sense that it would be good to have something that is itself extensible

Structural questions

- What are we trying to achieve here?
  - Movement of key-management information that is specific to the needs of “secured routing”
- How can we achieve this information movement?
  - Modify/extend IKEv2 messages (or some similar security protocol)
    - Done by rkmp, mrkmp, and akam-rp drafts
  - Create a new “information exchange” protocol and transport it using a known, secure existing protocol
    - (Which is my understanding of how SIDR works: move various messages on top of mutually-authenticated TCP-AO connections)

Observations

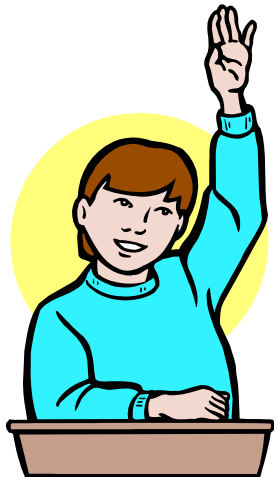
- The need to mutually authenticate would argue for using something like TCP-AO for all the configuration-exchange steps
- The need to do general configuration would argue for something like NETCONF as a vehicle

Questions

- Is it worth exploring such a general framework, i.e., one that is “beyond” the key management proposals?

- Does anyone favor
  - TCP-AO?
  - NETCONF?
- Why?

Thank You!



Questions?