IKEv2 Optional SA&TS Payloads in Child Exchange


Sandeep Kampati  Huawei Technologies
Meduri S S Bharath  Huawei Technologies
Wei Pan  Huawei Technologies

IETF 106, Singapore
November 2019
Recap

**Purpose:** To optimize unnecessary payloads at rekeying SAs

- Omit SA payloads at rekeying IKE SAs
- Omit SA & TS payloads at rekeying Child SAs

**Rationale**

- Configurations (e.g., cryptographic suites) don’t change frequently.
- SA & TS payloads at rekeying SAs are the same as the ones at creating SAs.
- Just use the previous SA & TS payloads instead of sending them again at rekeying.

**Motivations**

- Repeatedly sending SA & TS payloads is a redundant operation and unnecessarily consumes resources such as bandwidth and CPU.
- IKE SAs and Child SAs (IPSec SAs) rekeying happen periodically. This means periodic redundancies and burdens, especially for the constrained devices.
- When setting IPSec SA lifetime to be based on the transported traffic, rekeying happens more frequently (may even less than 20 minutes).
- Situations become much severer in 5G network as there will be more than 100,000 IKE/IPSec tunnels established.
Updates

- Make IKE SAs rekeying optimization and Child SAs rekeying optimization optional.
  - It’s up to implementer to optimize IKE SAs rekeying or Child SAs rekeying or both.
  - If you don’t think IKE SAs rekeying optimization is essential, you can choose not to optimize it.

- Simplify the rekeying optimization processes
  - The Initiator optimizes the rekeying message and the Responder accepts this optimization.
    Initiator                              Responder
    -------------------------------------
    Optimized rekeying request -->       <--- Optimized rekeying response

  - The Initiator optimizes the rekeying message and the Responder rejects this optimization.
    Initiator                              Responder
    -------------------------------------
    Optimized rekeying request -->       <--- NO_PROPOSAL_CHOSEN

  - (Discarded) The Initiator doesn’t optimize and the Responder optimizes the rekeying message.
    Initiator                              Responder
    -------------------------------------
    Original rekeying request -->        <--- Optimized rekeying response
Solution Overview

- **Negotiate the support of this optimization**
  - Send the `MINIMAL_REKEY_SUPPORTED` notification at the `IKE_AUTH` message exchange.

- **Optimize the IKE SAs rekeying (Optional implementation)**
  - The Initiator sends the optimized rekeying request and the Responder accepts this optimization.
    
    ![Diagram]

    Initiator: `HDR, SK {N(SA_UNCHANGED), Ni, KEi} -->`  
    Responder: `<-- HDR, SK {N(SA_UNCHANGED), Nr, KEr}`

  - The Initiator sends the optimized rekeying request and the Responder rejects this optimization.
    
    ![Diagram]

    Initiator: `HDR, SK {N(SA_UNCHANGED), Ni, KEi} -->`  
    Responder: `<-- HDR, SK {N(NO_PROPOSAL_CHOSEN), Nr, KEr}`

- **Optimize the Child SAs rekeying (Optional implementation)**
  - The Initiator sends the optimized rekeying request and the Responder accepts this optimization.
    
    ![Diagram]

    Initiator: `HDR, SK {N(REKEY_SA), N(SA_TS_UNCHANGED), Ni, [KEi]} -->`  
    Responder: `<-- HDR, SK {N(SA_TS_UNCHANGED), Nr, [KEr]}`

  - The Initiator sends the optimized rekeying request and the Responder rejects this optimization.
    
    ![Diagram]

    Initiator: `HDR, SK {N(REKEY_SA), N(SA_TS_UNCHANGED), Ni, [KEi]} -->`  
    Responder: `<-- HDR, SK {N(NO_PROPOSAL_CHOSEN), Nr, KEr}`
Future Plan

- More feedbacks, comments and reviews
- Looking for WG Adoption