Large Payloads in IKEv2

draft-tjhai-ikev2-beyond-64k-limit-01

CJ Tjhai (Post-Quantum)
Tobias Heider (genua GmbH)
Valery Smyslov (ELVIS-PLUS)

IETF 111

Motivation

- draft-ietf-ipsecme-ikev2-multiple-ke addresses issues of using large keys for Key Exchange methods (common in PQC) in IKEv2
- This draft still limits the size of any single public key to 64K – the maximum size of IKEv2 payload
  - most NIST Third Round Candidate Algorithms fit into this restriction
  - notable exception - Classic McElice PQKE which smallest public key is 255 KB
- However, some national regulators (e.g. BSI) **recommend** using Classic McElice PQKE
- It is also anticipated that PQ Digital Signatures will be used in IKEv2
  - Some NIST Third Round Candidate Digital Signature Algorithms have either public key size (Rainbow) or signature size (Picnic) greater than 64 KB

Goals

- The goal of the document is to define a way for using some specific data blobs in IKEv2 if they grow beyond 64K
  - public keys for key exchange methods (KE)
  - signatures (AUTH)
  - certificates (CERT)
- The defined mechanism must be backward compatible
- Reliability of transferring large data in IKEv2 should be addressed
- Performance of IPsec traffic should not degrade
- The defined mechanism must be simple and must introduce minimal changes to IKEv2

Not Goal

- There is no goal to define a generic mechanism for IKEv2 which would allow **any** payload be greater than 64K

Proposed Approach

- If amount of data doesn’t fit into a single payload then split data into chunks less than 64K and put them into a sequence of payloads of the same type; receiving end will concatenate data from a sequence of payloads having the same type
  - this approach works well if only one payload of this type may appear in the message according to IKEv2 (true for KE and AUTH, not true for CERT, but can be worked around)
  - if such sequence of payloads appears inside Encrypted payload, then the Length field of the Encrypted payload would be overflowed, but this doesn’t matter, since the length of Encrypted payload can always be deduced from the length of IKE message, so we can use value 0 for it

Example

Initiator		Responder
IKE_SA_INIT HDR, SAi1, KE1i, Ni	→ ←	**IKE_SA_INIT** HDR, SAr1, KE1r, Nr, [CERTREQ,]
IKE_INTERMEDIATE HDR, SK{**KE2i, KE2i, KE2i**}	→ ←	**IKE_INTERMEDIATE** HDR, SK{**KE2r, KE2r**}
IKE_AUTH HDR, SK{IDi,[**CERT,CERT,CERT**,][CERTREQ,] [IDr,] **AUTHi, AUTHi**, SAi2, TSi, TSr}	→ ←	**IKE_AUTH** HDR, SK{IDr,[**CERT, CERT**,] **AUTHr, AUTHr**, SAr2, TSi, TSr}

Changes from -00 version

- IKE Fragmentation is now mandatory for both UDP and TCP transport
  - with TCP the size of a single IKE message is still limited to 64 KB, so we need IKE Fragmentation to transmit larger messages (with TCP they may be fragmented to 64 KB fragments)
- Mixed Transport Mode is introduced
  - with this mode IKE starts from UDP port 4500, then switches to TCP on the first INTERMEDIATE exchange and continues to use TCP for all subsequent exchanges; however, Child SAs created with this IKE SA use either direct transport or UDP encapsulation
  - this mode is negotiated by exchange of new notification IKE_OVER_TCP
  - Mixed Transport Mode allows IKE to reliably transfer large blobs of data still avoiding performance implications of using TCP for ESP
- Added clarifications for tweaking Length field of Encrypted Payload

Future Discussion

- DoS attacks are an important concern for this extension; we are going to discuss how to defend against them in next version

Thanks

- Comments? Questions?
- WG adoption?