Signature Forms Ambiguity in IKEv2

Valery Smyslov

svan@elvis.ru

IETF 97

Problem Overview

- In IKEv2 there is no negotiation of auth methods, so each side may use what she thinks is appropriate
- RFC7427 adds a mechanism that allows peers to announce their support for hash functions that can be used in digital signatures
  - each peer sends `SIGNATURE_HASH_ALGORITHMS` notification containing a list of supported hash functions
- However, currently there is no way for peers to indicate supported signature forms
  - if some signature algorithm has several forms that can equally be used with the same key, then peers cannot tell each other what forms are supported

Real Life Interoperability Issue

- RSA signature currently has two forms:
  1. RSASSA-PKCS1 v1.5 (legacy)
  2. RSASSA-PSS (newer, more secure)
- An implementation may support both forms or only one of them and still be compliant with RFC7427
  - draft-ietf-ipsecme-rfc4307bis specifies that RSASSA-PSS MUST be supported and RSASSA-PKCS1 v1.5 MAY be supported
- If an implementation supports only one of the above forms, then IKE SA may fail even if the other side supports both.
  - if Responder supports both forms it can use the same form as Initiator used
  - however if Initiator supports both forms it has no clue what form to use:
    - she can use some heuristics based on information from IKE_SA_INIT (unreliable)
    - she can use some pre-configuration (doesn’t scale)
    - she can try RSASSA-PSS first and revert to RSASSA-PKCS1 if it fails (complicates code and slows down IKE SA setup)
  - since currently RSASSA-PSS is not widely used, the simplest solution for Initiator is to always use RSASSA-PKCS1, that will further slow down PSS adoption

Possible Future Issues

Similar issues may arise in future if several signature forms can be used with one key type:

- ECDSA vs EdDSA with Edwards curve keys?
- Prehashed vs non-prehashed forms of EdDSA?
  - draft-nir-ipsecme-eddsa specifies that pre-hashed form SHOULD NOT be used
- Different `AlgorithmIdentifier` OIDs for the same signature form?
- New forms of ECC signatures using existing curves?
- Hash based signatures? (e.g. XMSS vs XMSS^MT)

What to Do: Do Nothing

Consider the RSASSA-PSS issue as temporary and insignificant, that will gone once draft-ietf-ipsecme-rfc4307bis is adopted (until that happens work around RSASSA-PSS issue as suggested before). Envision that no such issues will occur in the future.

Pros:

- no changes to the protocol

Cons:

- complicates code to work around current RSASSA-PSS issue
- slow down RSASSA-PSS adoption
- if similar issues occur in the future then we'll face the same problem and it's unclear now whether reasonable workarounds will be found

What to Do: Make a Quick Fix

Add a fake hash algorithm `RSASSA_PSS_SUPPORTED` in `SIGNATURE_HASH_ALGORITHMS` notification.

Pros:

- fixes current problem

Cons:

- clear protocol hack
- needs some time to be adopted, so the problem may have already gone once draft-ietf-ipsecme-rfc4307bis is adopted
- slightly increases IKE_SA_INIT message size
- fixes only current problem, so if similar issues occur in the future then we'll face the same problem and it's unclear now whether reasonable workarounds will be found

What to Do: Solve Generic Problem

Define a new notification that will contain a list of supported signature forms (as `AlgorithmIdentifier` OIDs or as code points from new IKEv2 registry).

Pros:

- fixes the problem completely

Cons:

- increases IKE_SA_INIT message size
- may partially overlap with `SIGNATURE_HASH_ALGORITHMS` functionality
- reveals some information about peers capabilities to passive eavesdroppers (also true for `SIGNATURE_HASH_ALGORITHMS`)

Any thoughts?

Thank you