Choosing Sensible Signer Parameters

January 2013
What Parameters?

- We’re talking here about technical parameters
  - key sizes, algorithms, rollover schedules
  - authenticated denial of existence
  - signature validity periods
- We’re not talking about key management policy
Key Algorithm

• What algorithm?
  • RSASHA1 is mandatory to implement in the validator
  • RSASHA256 is used in the root zone
  • hard to recommend alternatives, unless there are layer-9 reasons not to use RSA
  • both RSASHA1 and RSASHA256 are reasonable choices
Key Sizes

- Common wisdom suggests that 1024 bit RSA ZSK keys are safe to use for the next 5 years
- KSKs may use longer keys, but it isn’t strictly proven to be necessary
- ZSKs are easier to roll than KSKs
- KSKs are exercised less frequently than ZSKs
- KSKs are harder to roll, since they involve parents
  - using a 2048 bit KSK does not seem bad
ZSK Rollover

- ZSK rollover can be automated, so it’s not painful to do
- Replacing keys provides an opportunity to reset your documentation trail and key management processes
- Silly to roll the ZSK too frequently, but doing it a few times a year seems prudent
KSK Rollover

• Rolling a KSK will invariably involve manual interaction with a parent zone

• You might consider that given the cost of the operation you might only plan to roll your KSK when you need to (i.e. when there is a suspicion of compromise)

• Remember however that operational procedures are difficult if you don’t practice them
Non-Existence

• NSEC is simpler to understand than NSEC3, and hence easier to troubleshoot
  • but more overhead than NSEC3 with opt-out

• NSEC facilitates zone-walking

• NSEC3 with opt-out may be the best option for large zones and zones with privacy concerns
Signing Parameters

• Every RRSIG is a ticking time bomb!
• choose your parameters wisely
• The parameters described in the slides which follow are based on elements in the OpenDNSSEC KASP
Resign Interval

• The interval between successive runs of the OpenDNSSEC signer engine

• it is generally not harmful to run the signer engine fairly frequently

• signatures can be re-used in many cases
Signature Refresh

• Signature Refresh is the minimum remaining validity of any signature in your zone

• How much time do you estimate would be needed to recover from a catastrophic failure in your signer infrastructure?
  • double it, add some more

• Think: risk management & analysis
Signature Validity

• Signatures remain valid for use by a cache until they expire
• a stolen signature can be used as an attack vector
• the maximum validity period is hence the window of opportunity for such an attack
• longer than the refresh interval
Jitter

• You can spread your signing load more evenly by applying jitter to signature inception and expiration times

  • times will be shifted by a random amount within +/- the value specified

  • ensure that the worst-case jitter does not present operational risks
Inception Offset

- It’s prudent to specify a signature inception time which is in the past
- even if your clocks are accurate, validators run by others might not be
- the window of opportunity to exploit a signature in the past is, well, past
Signature Lifetime

Inception time  Signing time  Expiration time

Inception offset  Validity period  +/- random (jitter)
Re-Use of Signatures
TTL Parameters

• The DNSKEY RRSet TTL determines how long your keys might be cached elsewhere, and is relevant for rollover scheduling.

• The SOA RRSet TTL determines (with the MINIMUM field) the negative cache TTL, and since OpenDNSSEC generates the SOA, we need to specify both in the KASP.
Further Reading