IETF 87 – RADIUS Extensions WG Meeting
Berlin, 30 July 2013

RESTENA

draft-ietf-radext-dynamic-discovery -07

Document status

- -07 submitted some time ahead of cutoff
- Integrated comments from Jim Schaad
- Remaining issues : few
  - TRAC #168 (remaining comments from Jim)
  - NAPTR → SRV → A/AAAA traversal thoroughness and loop detection
  - Alan DeKok's recent ML comments
    - text about risk of open TLS ports in Sec Con
    - text about keeping record of negative connection attempts to save computational power of excessive retries
    - Alan is happy:-)

MTI mech for server authz

- Cert property needed to express « authoritative server for a NAI realm »
  - SubjectAltName:dNSName does not do the job (properly)
  - A new property for NAI realms is needed
    - subjectAltName:nAIRealm
    - UTF8Name (assuming realms are always UTF-8) ?
    - Wildcard match in intermediate portions of realm ?
    - requested at pkix, led to discussion, but no results yet

Privacy implications ?

- See Kim Schaad's comments on ML and TRAC #168
- I'm inclined to think that no interesting knowledge can be won by observing execution of the dynamic discovery algorithm
- I.e. : no text update needed.
- Comments ?

NAPTR → SRV → A/AAAA

- S-NAPTR RFC allows for partial execution of discovery
  - As soon as the highest-priority server is resolved, break out of algorithm
  - Try that server
  - If connection doesn't come up satisfactorily, get back to discovery and continue to next-best option
  - lather, rinse, repeat
- Makes « forward-to-self » detection harder/impossible
- But also makes the whole discovery process faster
- Breakout/return code is more complex, but that's an implementation problem :-)

Loop Detection

- Forward-to-self detection in NAPTR discovery can only capture loops introduced due to discovery
- Other loops may exist but will be undetected
- Decision needed :
  - make NAPTR discovery as thorough as possible, minimising (but not zeroing) risk
  - or don't insist on full discovery of all targets in the interest of speed
- NB : we would not need a decision if loop detection were solved in the general case. If only we had that ! ;-)

Loop Detection I-D ?

- We've had arguments about a loop-detection attribute previously
  - Processed-By : <someid>
  - Proxy-State : <blob>
- Both inflate the packet until it reaches 4K boundary and dies
- Packet-TTL : int, decrement → would not inflate
- Previous arguments were : the packet size boundary will take care of this, so no need to bother
- Enter : Sam Hartman's proposal of lifting the boundary
  - Many more loop rounds until packet dies, performance hit !