networkRADIUS

IETF 99

256 Is Not Enough

THE PROBLEM

- after 256 outstanding packets, a connection is “full”
- With 64-bit CPUs and gigabit home networks, an 8-bit protocol is embarrassing
- Clients can open a new connection to get more free IDs.... but...

PROBLEMS WITH THE SOLUTION

- Low-load systems are fine, and don’t need it.
- High load systems may open thousands of network connections
- Each connection operates independently of all others
- Each connection to a server independently discovers server availability
- UDP hits Ethernet packet rate limits before the network is “full”
- TCP doesn’t help, because we can’t “fill” a TCP connection
- It is generally better to have a few “full” connections than many “empty’ ones

REQUIREMENTS FOR A BETTER SOLUTION

- No changes to RADIUS packet format
- No changes to RADIUS security
- No changes to RADIUS data types
- No changes to RADIUS attribute format
- Use standard data types
- Works with all existing transports
- Compatible with existing RADIUS
- Does not affect proxying

COMPATIBILITY

- Negotiate via Status-Server (the de-facto solution)
- Clients can fall back to normal RADIUS with *no negotiation* if the server starts using normal RADIUS
- Clients use normal RADIUS until the new capability has been negotiated
- low-load systems do not need this specification
- As always…
  - requires code changes on clients and servers to implement

BENEFITS

- Low-load systems don’t require changes
- High load systems open one connection
- Different connection still operate independently of all others
- One connection per server to discover server availability, *once*
- UDP still hits Ethernet packet rate limits before the network is “full”
- TCP connections get “filled”
- Fewer connections, but “full” ones.
- Implementations track vectors, not file descriptor

NEGOTIATION

- Via Status-Server
- client -> server
  - Can we do this?
- server -> client
  - ACK, NAK, or radio silence (== NAK)
- Clients can still send old-style requests before negotiation has completed!
- Servers can immediately send new-style replies to old-style requests
  - because servers ALWAYS get old-style requests!

THE DETAILS

- Servers *may* use Request Authenticator as a unique ID
  - All packets from clients are *completely unchanged*
- Servers echo the Request Authenticator in reply packets
  - via the Original-Request-Authenticator attribute
  - just like Original-Packet-Code from RFC 7930, Protocol-Error
  - No other change to the protocol
- both sides need to track packets via the tuple:
  - (src / dst IP / port, code, ID, *Request Authenticator*)

WHY THIS WORKS

- Request Authenticator is either:
  - 16 random octets (Access-Request)
  - 16 octet MD5 signature (other packets) … i.e. mostly random octets
- The MD5 signature is unique, and “good enough” for an Identifier
  - essentially impossible for an attacker to forge
- We expect collisions every 2^64 packets or so
  - i.e. never, even at giga-packet rates

WHY MD5 IS OK

- Any change in packet contents will change the MD5 signature
  - Event-Timestamp, packet counters, etc.
- But MD5 collisions can be created by an attacker!
  - Only if they know the shared secret.
  - If you don’t trust the trusted people, all bets are off
- So if the packets are different, the MD5 hashes are different
- If the packets are identical, the MD5 hashes are identical
  - Duplicate detection for free, without taking additional steps!

COMPARISON TO OTHER PROPOSALS

- Multiple source ports
  - complex to manage, OS / application overhead
- Diameter
  - too complicated for a minor upgrade
  - very little outside of 3G supports Diameter
- Multiple RADIUS packets in one UDP packet
  - Bad. Doesn’t solve the ID exhaustion or TCP problem

COMPARISON TO OTHER PROPOSALS (2)

- Changing the RADIUS packet header
  - runs away screaming…
  - no, no, just… no. Did I mention “no”
- Extended ID?
  - Already used in some form by vendor(s)

COMPARISON TO EXTENDED ID

- ✓ Pretty similar to this proposal
- ✓ Tracking a new 32 or 64-bit Identifier is not hard
  - could just be an incrementing counter
- ✦ If a client misbehaves, the “Extended ID” attribute could be sent to a server which doesn’t support it… and get proxied upstream
- ✦ Doesn’t get duplicate detection for free

- ▸ Not a huge difference between the two proposals

IMPLEMENTATION

- Ongoing in FreeRADIUS v4
  - has to wait for some other architectural changes first
  - Probably September
- Could be implemented in v3
  - Extended-ID is ~300 LoC including full negotiation
  - This will likely be similar

DRAFT

- The draft has a detailed explanation of everything
  - pros and cons
  - what led me to this proposal
  - comparisons to other proposals
- Describes impact and inter-operability with existing systems
- Implementation guidelines and suggestions