draft-ietf-radext-radiusdtls-bis
Making RADIUS/(D)TLS a Proposed Standard

radext interim | 08.10.2024

Janfred Rieckers | DFN-Verein

What happened since IETF120

- A lot of Reviews and Pull requests \o/ (Thanks to all for their comments)
- Secdir early review from Russ Housley

Summary of open PRs on github

- Connection handling
  - Idle Timeout (#5, #11)
  - Parallel Connections, exponential back-off (#12)
  - NAT considerations, downgrade attacks (#16), IP-Range clients (#17)
  - Implementation considerations for sockets (#14)
- Proxying/Load-balancing
  - How to deal with EAP or other multi-packet sessions (#13)

Summary of open PRs on github

- TLS-related
  - Server Name Indication and Server Identity (#1)
  - RFC9525 (Service Identity) guidance for cert validation (#4)
  - Trust list considerations (#15)
- Editorial
  - TLS response cache invalidation to common section (#6)
  - Reference to Crypto-Agility (RFC6421) (#18)

TODOs in the document not covered by PRs

- Section 4.1 (D)TLS requirements: Which recommendations from RFC9325 (TLS BCP) should be followed?
- Section 4.2.1 (TLS-X.509-PKIX trust model): Open discussion about re-assessing the peer's continued authorization after trust base change (removal of CA, new CRL, ...) [also commented on by Russ in secdir review]
- Section 4.4 (RADIUS Datagrams): Send ProtocolError packet instead of Accounting-Response with Error-Cause Attribute when received an unwanted Accounting-Request
- Section 7.1 (RADIUS Proxies): Add reference how to handle dynamic discovery

Connection Handling – Parallel Connections (#12) and Load Balancing (#13)

- Currently no specification on handling parallel connections, distributing packets, reconnect mechanisms
- Suggested in #12: Add one paragraph in Section 4.1 ((D)TLS requirements)
  - Comment on the PR from Alan: move into a new section, add some considerations around distributing packets as well. (May fall in the Loadbalancing issue as well)
- For Multi-Packet Sessions load-balancing is tricky (see #13)
  - Decision to make: Mandate a specific load balancing behavior/algorithm or not?
  - What RADIUS attributes could/should be used for load-balancing decisions?
    - CallingStationID (may not always be present) – User-Name (may be too similar, if anonymous) – something else?
- My suggestion: Add the contents of #12 to the result of #13, drop #12 and adjust and then merge #13
  - For load-balancing: Mandate that no load-balancing must be done unless explicitly configured (failsafe default, allow switching to new server only once the other connection is marked DOWN)
    - Open question/ref to #1: Is a new session to the same configured endpoint the same server?
    - Recommend load-balancing algorithm on basis of CallingStationID or User-Name

TLS – SNI and Server Identity (#1) – Part 1

- Problem statement 1: How does the client know that it connected to the same server? (Or previous problem: Does the client even care?) Possible scenarios:
  - Server was discovered via dynamic discovery, but maybe already a statically configured connection exists, that could be re-used for this?
  - A client wants to open a second connection to the server to increase the packet-in-flight limit, but does not want to deal with load-balancing. Important to keep track for disconnects (a disconnect may be tolerated, if the client has multiple connections to the same server, but trigger a reconnect otherwise)
- Origin of the problem: When should a client mark a specific server as „DOWN“? RFC6613 (RADIUS/TCP) says: “RADIUS clients using RADIUS/TCP MUST NOT decide that a RADIUS server is unresponsive until all TCP connections to it have been marked DOWN.“
- Suggestion on ML (back in the day): Is it enough to say „with the same config item“?
  - Would treat dynamic connections separately from static ones
  - Might be problematic if configured with a hostname, but the hostname resolves to different IP addresses?

TLS – SNI and Server Identity (#1) – Part 2

- Problem statement 2: RFC6614/RFC7360 did not specify how to deal with SNI.
  - SNI also means that Server IP Addr and port are now definitely not enough to reliably check if it is the same server with at least some certainty.

- I have no suggestion here. Maybe the „same server" discussion is fixed with the „multi-packet session" discussion?

- SNI should be included anyway, it is a good way to signal things, i.e. if the server has two different certificates configured for different use cases (Example: eduroam/openroaming). Would probably require updates to RFC7585 (dynamic discovery) to use the hostname discovered in the NAPTR lookup as name in SNI.

TLS – RFC 9525 guidance for validation (#4)

- Suggestion in #4: Re-phrase the whole certificate validation part
- Changes from current spec:
  - Certificate CN MUST NOT be used (possibly breaks config backwards compatibility)
  - Add Text for wildcard certificates
  - Unify text for dynamic discovery with „normal“configuration
- Open questions: How to deal with loopback-attack?
  - Basic scenario: An attacker mirrors the TLS Client Hello back, causing a loop
    - Requires specific configuration that would accept the client certificate on the server side and the server certificate on the client side
    - Example: Dynamic configuration, authorization by Policy OID only, ignoring host names.
- My suggestion: Merge #4, maybe add bullet point to „changes from 6613/6614/7360“, add text in security considerations section for loopback-attack
  - Do we need specification to prevent loopback-attacks? (i.e. „don’t accept your own certificate“)
    Take a closer look, because this is also a (maybe even bigger) problem with TLS-PSK

TLS – Trust List considerations (#15)

- Suggestion in #15: RADIUS/(D)TLS peers should not come with a pre-configured set of trust-anchors.
  - Rationale: RADIUS/(D)TLS is a highly specific use case, we don’t need to communicate with unknown entities, so we don’t need the Root-CA list for trust checks.
- My suggestion: Merge #15, maybe coordinate with #4
  - I.e. the „just trust the cert, we don’t have any other information“ behavior is forbidden unless explicitly configured with a specific trust anchor, and not the default one

Connection Handling – Timeouts (#5, #11)

- Currently there is no consistent specification on timeouts
  - For TLS in Sec 5.3 (TCP appl are not UDP appl)
  - For DTLS in Sec 6.4.1.1 (for servers) and Sec 6.4.2 (for clients)
- For DTLS: SHOULD close if no appl traffic for more than 3 watchdog timeouts
- Suggested in #5: New section inside Section 4 (Connection handling), remove text from other parts of the document where it is mentioned
  - Unified text around connection timeout
  - Drop „SHOULD" for closing sessions (may not what you want, i.e. for Reverse-COA), instead make it configurable
- Suggested in #11: (just for DTLS) Idle timeout configurable SHOULD be 1 min < x < 10 min
- My suggestion: Accept #5, drop #11, look through document to make sure everything aligns with this suggestion

Connection Handling – Downgrade/Duplicate IP Addresses (#16)/Clients w/o IP restriction (#17)

- Problem statement in #16: A RADIUS server could have conflicting view of client if a client behind a NAT would connect to both RADIUS/UDP and RADIUS/DTLS over the same source port, due to NAT translating the traffic to the same source port
- Suggestion in #17: Add Section in Security Considerations regarding clients without specific IP restriction (summary: It is safe to use, but you have to uniquely identify the client by Certificate/PSK-ID/...
- My suggestion: Merge contents of #16 and #17 to a single section.
  - Identifying the client is important, we don’t really need the IP address any more (like in RADIUS/UDP), beware of issues with NAT where it could happen that the same IP addr and source port connect to both RADIUS/UDP and RADIUS/DTLS. (or RADIUS/TCP and RADIUS/TLS, though more unlikely and probably less problematic due to being connection-based anyway)

Connection Handling – Connected Sockets (#14)

- Suggested in #14: Add new section „Implementation Guidelines“
  - Suggestion to always use connected sockets on the client side and to not use connected sockets on the server side
    - Objection to not using connected sockets on the server side from Fabian
- Duplicate text from Section 7.6 (Security Considerations, Client Subsystems) around having a local proxy for multiple RADIUS clients on the same server
- My suggestion: Re-evaluate need for whole Implementation Guidelines section, maybe move text for connected sockets in “DTLS specific specifications“ section, double-check that section 7.6 and suggested addition are in fact similar in content.

Other open PRs / TODOs to dicuss

- Add Reference to RFC6421 (RADIUS Crypto Agility) (#18)
  - Needs eyes, maybe a little bit more text.
- Move session invalidation to common section (#6)
  - Seems straight-forward, would appreciate two more sets of eyes to verify, then merge.
- Unwanted AccountingRequest: Break backwards compatibility and use ProtocolError?
- Any other open questions?

Discussion/Questions?

- Contact
  - Jan-Frederik Rieckers

Mail: rieckers@dfn.de
Phone: 0049 30 884299-339
Fax: 0049 30 884299-370

Address:
DFN-Verein, Geschäftsstelle
Alexanderplatz1
10178 Berlin