Attacks on CoAP

draft-ietf-core-attacks-on-coap

John Preuß Mattsson, John Fornehed, Göran Selander, Francesca Palombini,
Christian Amsüss

IETF117 San Francisco, CoRE, 2023-07-25

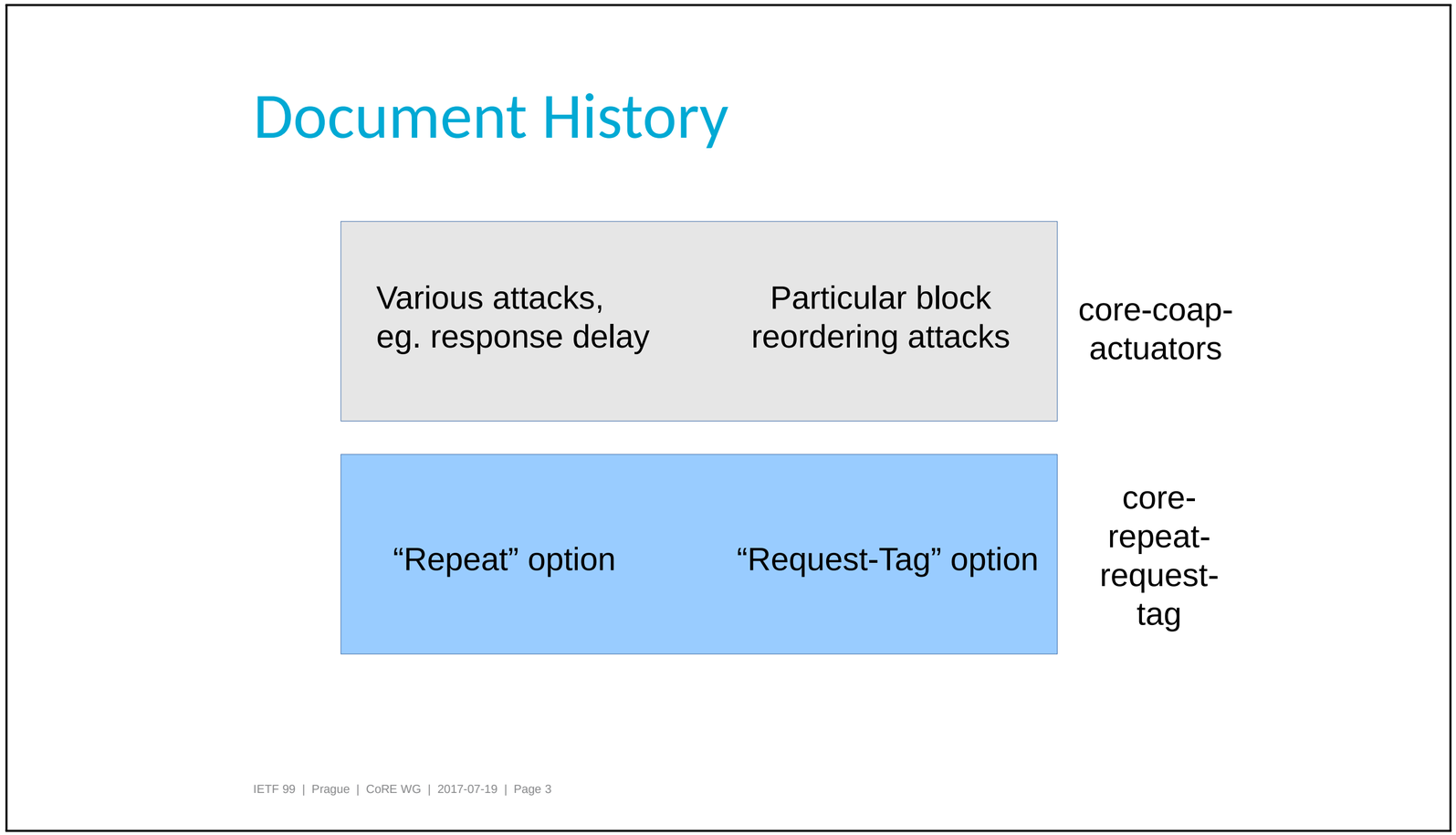
History: IETF99

Document History

Various attacks, eg. response delay	Particular block reordering attacks
“Repeat” option	“Request-Tag” option
core-coap-actuators	core-request-tag

IETF 99 | Prague | CoRE WG | 2017-07-19 | Page 2

History: IETF99



`repeat-request-tag` became `echo-request-tag`, and eventually RFC 9175

`coap-actuators` became `attacks-on-coap`.

Content

"Here is what can go wrong in the presence of an on-path attacker that may even guess plain text but does not break the cryptographic primitives, and there (eg. RFC 9175) is how to avoid it."

Except, we're may not be done :'-(

Return of the Request Fragment Rearrangement

Paraphrasing dialogue between Jon Shallow and Christian Amsüss

J That can also happen when there is no Block1 phase.

C No, because then it's concurrent requests, and those don't work without Request-Tag anyway. In the Block2 phase, the request payload is repeated.

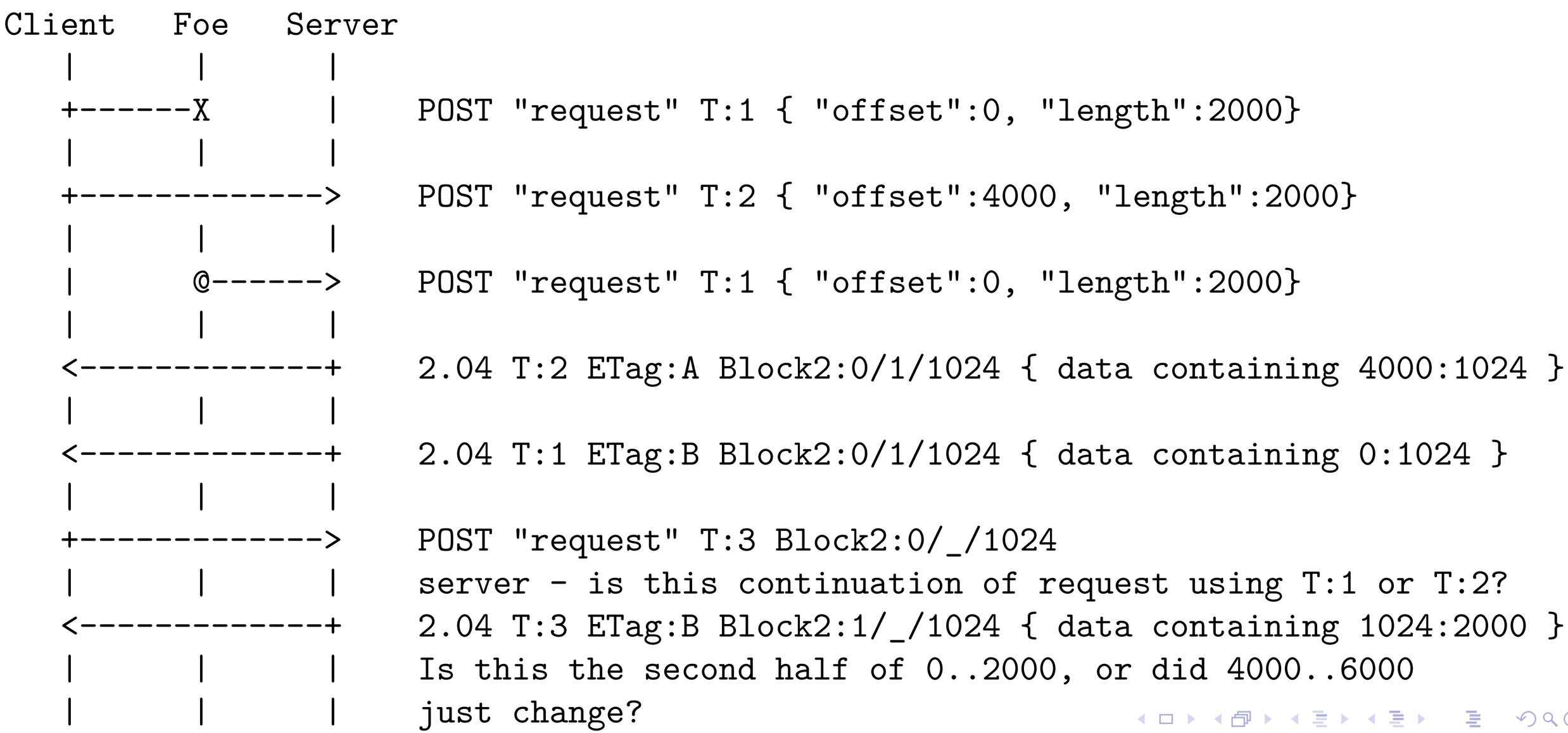
J The problem is still there.

C *reads RFC 7959 and blushes.*

Turns out, the request payload is very much not repeated.

Return of the Request Fragment Rearrangement

Example



Possible solution A: Use Request-Tag more often

Old: “When a client fragments a request body into multiple message payloads [the body integrity is not protected]. To gain that protection, use the Request-Tag mechanism as follows:”

New: “When a client fragments a request body into multiple message payloads, or on requests that are matchable to requests the client may have produced and is prepared to accept fragmented responses for (even if the request body is small or empty), [the body integrity is not protected]. To gain that protection, use the Request-Tag mechanism a follows:”

Could be short document, updates RFC 9175 (maybe also RFC 9177).

Practical Impact of using Request-Tag more often

When using OSCORE: Negligible.
Constrained OSCORE implementations can always use the absent Request-Tag option as long as they spot some conditions after which they bump their sender sequence number.

When using DTLS: Almost all requests that could have requests that could have resquest and response bodies will need a Request-Tag, typically a counter.
...But they already tolerate having tokens that practically need to be counting up. How large is the impact really?

Possible solution B: Allow server to declare Request-Tag for client to use

Suggestion from Jon

Would update RFC 9175 to allow use in requests.
Shifts logic and state tracking to the server.

Possible solution C: Tighten ETag

Add some small requirements around ETags to ensure they prevent the attack.

"...are necessary even if the represention does not change over time but is a function of the request payload..."
"...ETags must be unique to the body even across request payloads ..." (example: When request payload slices, a hash ETag needs to be fed the slice parameters.)

Could be short document, updates RFC 7252 or RFC 7959 (maybe also RFC 9175).

Roadmap

1. Tune example, decide whether single example covers it well.
2. Write text for proposed solutions.
3. Pick best solution.
4. Ask WG to adopt solution.
5. Continue with this document when we can point to solution.

Thanks

Comments?

Questions?