draft-ietf-core-attacks-on-coap
Interim July 5 2023

Jon Shallow
Christian Amsüss

Agenda

- Block2 Responses
- Token Manipulation
- Clarifications

Block2 Responses (no RTag / ETag)

```
Client   Foe   Server
   |      |      |
   +------X      |    POST "request" T:1 { "offset":0, "length":2000}
   |      |      |
   +------------->    POST "request" T:2 { "offset":4000, "length":2000}
   |      |      |
   |      @------>    POST "request" T:1 { "offset":0, "length":2000}
   |      |      |
   <-------------+    2.04 T:2 Block2:0/1/1024 { data containing 4000:1024 }
   |      |      |
   <-------------+    2.04 T:1 Block2:0/1/1024 { data containing 0:1024 }
   |      |      |
   +------------->    POST "request" T:3 Block2:1/0/1024
                      server - is this continuation of request using T:1 or T:2 ?
   |      |      |
   <-------------+    2.04 T:3 Block2:1/0/1024 { data containing 1024:2000 }
                      Using T:1 - was this the client's expected data ?
   |      |      |
```

No use of Request-Tag or ETag

Block2 Responses (ETag Only)

```
Client    Foe    Server
   |       |       |
   +-------X       |    POST "request" T:1 { "offset":0, "length":2000}
   |       |       |
   +-------------->    POST "request" T:2 { "offset":4000, "length":2000}
   |       |       |
   |       @------>    POST "request" T:1 { "offset":0, "length":2000}
   |       |       |
   <--------------+    2.04 T:2 Etag:11 Block2:0/1/1024 { data containing 4000:1024 }
   |       |       |
   <--------------+    2.04 T:1 Etag:12 Block2:0/1/1024 { data containing 0:1024 }
   |       |       |
   +-------------->    POST "request" T:3 Block2:1/0/1024 (client asking for T:1)
                       server - is this continuation of request using T:1 or T:2 ?
   |       |       |
   <--------------+    2.04 T:3 Etag:11 Block2:1/0/1024 { data containing 5024:2000 }
                       Using T:2 – what does client do with ETag mismatch?
   |       |       |
```

Use of ETag only makes sure that the client associates the response with the correct request, but this may be a response to a request for the next block which has not yet been issued by client.

Block2 Responses (Request-Tag Only)

```
Client    Foe    Server
   |       |       |
   +-------------->    POST "request" T:1 RT:21 { "offset":0, "length":2000}
   |       |       |
   +-------------->    POST "request" T:2 RT:22 { "offset":4000, "length":2000}
   |       |       |
   <--------------+    2.04 T:1 Block2:0/1/1024 { data containing 0:1024 }
   |       |       |
   <--------------+    2.04 T:2 Block2:0/1/1024 { data containing 4000:1024 }
   |       |       |
   +-------X       |    POST "request" T:3 RT:21 Block2:1/0/1024 (client asking for T:1)
   +-------------->    POST "request" T:4 RT:22 Block2:1/0/1024 (client asking for T:2)
   |       @------->    POST "request" T:3 RT:21 Block2:1/0/1024 (client asking for T:1)
   |       |       |
   <--------------+    2.04 T:4 Block2:1/0/1024 { data containing 5024:2000 }
   <--------------+    2.04 T:3 Block2:1/0/1024 { data containing 1024:2000 }
   |       |       |
```

Use of Request-Tag means server sends correct next block response, but client should correctly associate responses based on Tokens with appropriate requests even if data arrives in wrong order. [Not using ETag means changing data on server not detected]

Issues

- Client has no knowledge of whether a response is going to need to use Block2 or not
- Send Request-Tag with every request? [Request-Tag is supported for requests without Block1/Block2]
  - Unnecessary overhead
  - Size (DTLS requires larger for unpredictability)
  - It is required by RFC9177 Q-Block
- Without Request-Tag, if multiple requests are active, server can select wrong response (FIFO or LIFO request lookup)

Mitigation

- Send Request-Tag with every request
- Prevent client doing concurrent different requests
  - NSTART =1 not enough as all blocks to be returned before next new request sent
- Specify new Signal from server in response indicating (server generated) Request-Tag to use for the next block – how?
  - Request-Tag not allowed in response – change?
  - New Block2 option with embedded Request-Tag to use

Attacks

- Without use of both Request-Tag and ETag, data is subject to corruption even when not under attack
- NSTART = 1 serialization of CON requests can be broken by “foe” ACKing request and converting responses from ACK to CON on the way back to client. Easy then for “foe” to re-order requests

Token Manipulation (NON)

```
Client    Foe    Server
     |      |      |
     +------X      |  NON POST "request1" T:1
     |      |      |
     +------X      |  NON POST "request2" T:2
     |      |      |
     |      @------>  NON POST "request2" T:1 (token replaced)
     |      |      |
     |      @------>  NON POST "request1" T:2 (token replaced)
     |      |      |
     <-------------+  NON 2.04 T:1 { data containing response2}
     |      |      |
     <-------------+  NON 2.04 T:2 { data containing response1}
     |      |      |
```

Token Manipulation (CON)

```
Client    Foe    Server
     |       |       |
     +-------X       |  CON POST "request1" T:1
     <-------@       |  ACK
     +-------X       |  CON POST "request2" T:2
     <-------@       |  ACK
     |       @------->  CON POST "request2" T:1 (token replaced)
     |       @------->  CON POST "request1" T:2 (token replaced)
     |       |       |
     |       X-------+  ACK 2.04 T:1 { data containing response2}
     <-------@       +  CON 2.04 T:1 { data containing response2}
     +-------X       |  ACK
     |       X-------+  ACK 2.04 T:2 { data containing response1}
     <-------@       +  CON 2.04 T:2 { data containing response1}
     +-------X       |  ACK
```

Attacks

- Works with NON (or CON NSTART > 1)
- Works with CON if “foe” ACKs request and updates response from ACK to CON
- OSCORE does protect (Request/Response matching with PIV/AAD)
- (D)TLS does not protect if “foe” is a rogue proxy or “foe” is successful man-in-the-middle

Mitigation

- Use OSCORE
- Do not use No-Sec

Clarifications

- Request using Block1 triggers Block2 response
- Request for next block
  - RFC7959 2.7: “To continue this Block2 transfer, the client continues to send requests similar to the requests in the Block1 phase, but leaves out the Block1 Options and includes a Block2 request option with non-zero NUM”
- Observe Request using Block1
- Observe deregister cancellation
  - Includes original data (with all the Block1s) (only Observe Option changed, ETags ignored) RFC7641 3.6.
  - Cancellation response may include Block2

Thank you