ANIMA WG
Internet-Draft
Intended status: Informational
Expires: May 4, 2017

M. Pritikin
P. Kampanakis
Cisco Systems

October 31, 2016

BRSKI over CoAP
draft-pritikin-coap-bootstrap-01

Abstract

This document provides an initial discussion of Bootstrapping of Remote Secure key infrastructures (BRSKI) when the device being bootstrapped speaks CoAP. The HTTPS REST methods leveraged by BRSKI are mapped to CoAP methods. Fragmentation management of large messages during EST certificate enrollment is addressed.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 4, 2017.

Copyright Notice

Table of Contents

1. Introduction

Many IoT and other devices are expected to use CoAP over UDP extensively. Bootstrapping these devices without requiring a full TCP stack is an often raised requirement for [I-D.ietf-anima-bootstrapping-keyinfra]. BRSKI provides REST methods over TLS that can be functional in a UDP setting with the folling necessary additions:

DTLS: Because the CoAP use of DTLS includes support for large handshake messages there is little to describe here. BRSKI and EST [RFC7030] are expanded to include DTLS.

REST: The mapping of BRSKI and EST messages to CoAP REST calls is described.

Fragmentation: Use of block chaining to support fragmentation of large BRSKI and EST messages is described.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Scope of solution

The definition of BRSKI over DTLS and CoAP is not intended to expand the scope of BRSKI to highly constrained devices. (ref: [RFC7228]). Instead it is intended to ensure that bootstrapping works for less constained devices that choose to limit their communications stack to UDP/CoAP.

The BRSKI document details extensions to EST as well as making section 5.7 requirements on EST flows. This document's references to BRSKI are intended to include all BRSKI extensions and all existing EST messages. This document could replace BRSKI -03 section 5.7.5. [[TODO: making this section 5.8 might make the most sense.]]

Support for Observe CoAP options (https://tools.ietf.org/html/rfc7641) in Blocks with BRSKI is not supported in the current BRSKI/EST message flows and is thus out-of-scope for this discussion. Observe options could be used by the server to notify clients about a change in the cacerts or csr attributes (resources) and might be an area of future work.

4. DTLS

COAP was designed to avoid fragmentation. DTLS is used to secure COAP messages. When using DTLS, even though it can be avoided by using pre-shared keys or ECC ciphersuites, sometimes fragmentation will be needed. During the DTLS handshake, if fragmentation is necessary, "DTLS provides a mechanism for fragmenting a handshake message over a number of records, each of which can be transmitted separately, thus avoiding IP fragmentation" [RFC6347].

Within BRSKI and EST when "TLS" is referred to, it is understood that CoAP security is provided using DTLS instead. No other changes are necessary (all provisional modes etc are the same as for TLS).

In a constrained CoAP environment, endpoints can't afford to

establish a DTLS connection for every EST transaction. Authenticating and negotiating DTLS keys requires resources on low-end endpoints and consumes valuable bandwidth. The DTLS connection SHOULD remain open for persistent EST connections. For example, an EST cacerts request that is followed by an simpleenroll request can use the same authenticated DTLS connection. Given that after a successfull enrollment, it is more likely that a new EST transaction will take place after a significant amount of time, the DTLS connections SHOULD only be kept alive for EST messages that are relatively close to each other.

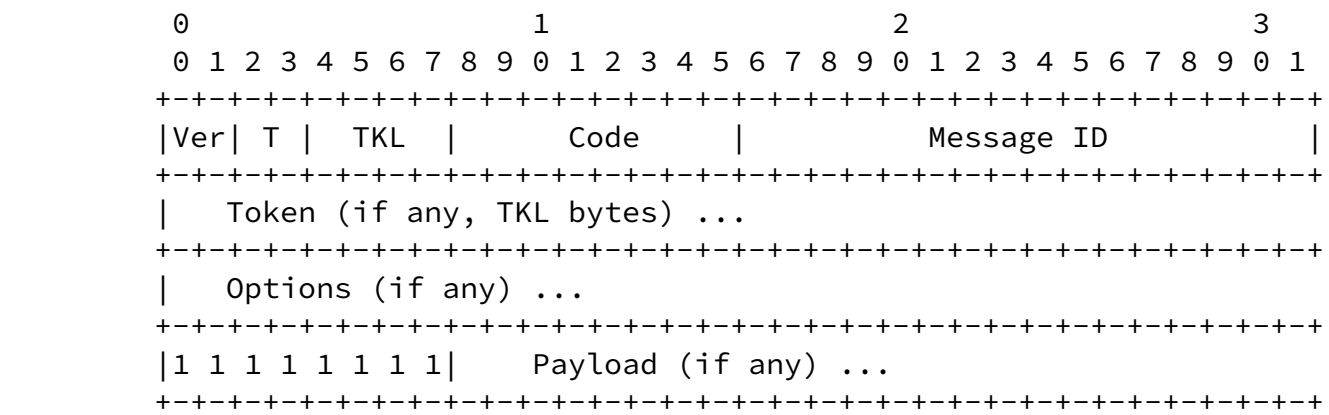
5. Message Bindings

This section describes BRSKI to CoAP message mappings.

CoAP defines confirmed (CON), acknowledgements (ACK), reset (RST) and non-corfirmed (NON) message types. For confirmable messages, the responses are CoAP ACKs or RSTs. All /cacerts, /simpleenroll, /simplereenroll, /csrattrs, /fullcmc and /serverkeygen EST messages expect a response, so they are all COAP CON messages.

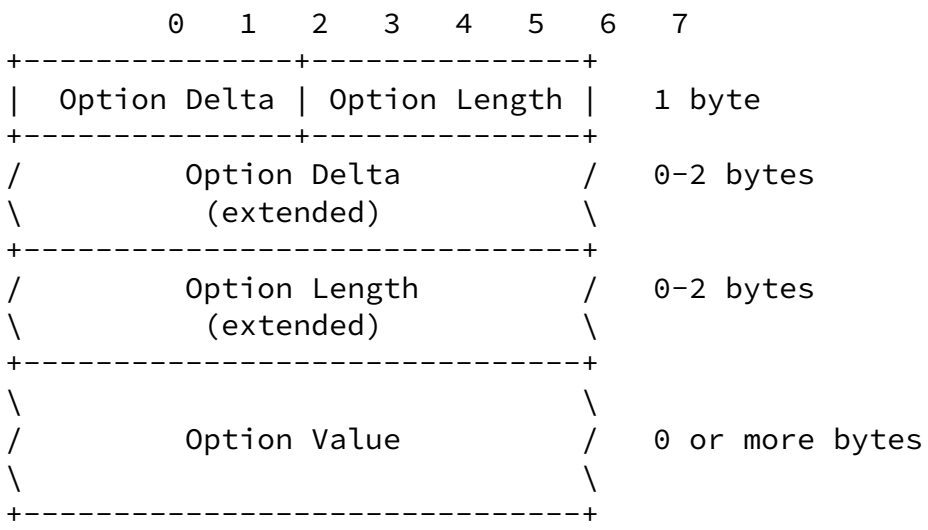
The HTTP responses in BRSKI are translated to COAP Response Codes as explained in [RFC7252] section 5.3.1

A CoAP message has the following fields ([I-D.ietf-core-block]):



Then Ver, TKL, Token, Message ID are not affected in BRSKI. Their use is the same as in CoAP.

The options that can be used in a CoAP header have the following format (from [I-D.ietf-core-block] Figure 8):



Options are used to convey Uri-Host, Uri-Path, Uri-Port, Content-Format and more in COAP. For BRSKI, COAP Options are used to communicate the HTTP fields used in the BRSKI REST messages.

BRSKI URLs are HTTPS based (https://), in CoAP these will be assumed to be transformed to coaps (coaps://)

Some examples of how an BRSKI message would be translated in CoAP follow. [[TODO: This section to be expanded to ensure it covers all BRSKI edge conditions.]]

5.1. cacerts

First let's see how a get cacerts message in EST would be in CoAP. The HTTPS cacerts message can be

```
GET /.well-known/est/cacerts HTTP/1.1
   User-Agent: curl/7.22.0 (i686-pc-linux-gnu) libcurl/7.22.0
               OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23 librtmp/2.3
   Host: 192.0.2.1:8085
   Accept: */*
```

The corresponding CoAP fields would be:

```
  Ver = 1
  T = 0 (CON)
  Code = 0x01 (0.01 is GET)
  Options
   Option1 (Uri-Host)
     Option Delta = 0x3
     Option Length = 0x9
     Option Value = 192.0.2.1
   Option2 (Uri-Port)
     Option Delta = 0xA
     Option Length = 0x4
     Option Value = 8085
   Option3 (Uri-Path)
     Option Delta = 0xD
     Option Length = 0xD
     Extended Option Delta = 0x08
     Extended Option Length = 0x14
     Option Value = /.well-known/est/cacerts HTTP/1.1
  Payload = [Empty]
```

Now let's say we have a 200 OK response with a cert in EST:


```
   HTTP/1.1 200 OK
   Status: 200 OK
   Content-Type: application/pkcs7-mime
   Content-Transfer-Encoding: base64 (TODO: Verify if we need a new
                                      option registry for Encoding?)
   Content-Length: 4246 (TODO: this example overflows and would
                         need fragmentation. Choose a better example.
                         Regardless we might need an CoAP option for
                         the content-length ie the CoAP payload?)

   MIIMOQYJKoZIhvcNAQcCoIIMKjCCDCYCAQExADALBgkqhkiG9w0BBwGgggwMMIIC
   +zCCAeOgAwIBAgIJAJpY3nUZO3qcMA0GCSqGSIb3DQEBBQUAMBsxGTAXBgNVBAMT
   ...
```

The corresponding CoAP fields would be:

```
Ver = 1
T = 2 (ACK)
Code = 0x21 (TODO: Maybe we need to create a 0x200 response code.)
Options
  Option1 (Content-Format)
    Option Delta = 0xC
    Option Length = 0xD
    Extended Option Length = 0x09
    Option Value = <number for application/pkcs7-mime>
              (TODO: We need a new CoAP IANA registered value
              application/pkcs7-mime; smime-type=certs-only,
              application/csrattrs, application/pkcs10,
              application/pkcs8,
              application/pkcs12 )
Payload = MIIMOQYJKoZIhvcNAQcCoIIMKjCCDCYCAQExA \
            DALBgkqhkiG9w0BBwGgggwMMIIC...
```

5.2. enroll / reenroll

[[TODO: username/password authentication can be described here but is not a primary focus for BRSKI. It is important for generic EST exchanges but would an endpoint device with sufficient user interface to allow username/password input from an end user be required to use CoAP instead of a full HTTPS exchange?]]

[[TODO: We might need a new Option for the Retry-After response message. We might need a new Option for the WWW-Authenticate response.]]

5.3. csrattr

5.4. requestauditttoken, requestauditlog

6. Data Fragmentation

Even though ECC certificates are small in size, they can vary greatly based on signature algorithms, key sizes, and OID fields used. Even

with ECC certs, BRSKI CoAP messages carrying such certificates can still exceed sizes in MTU of 1280 for IPv6 or 60-80 bytes for 6LoWPAN [RFC4919]) (section 2 of [I-D.ietf-core-block]). For 256-bit curves, common ECDSA cert sizes are 500-1000 bytes which could fluctuate further based on the algorithms, OIDs, SANs and cert fields. For 384-bit curves, ECDSA certs increase in size and can sometimes reach 1.5KB. Additionally, there are times when the EST cacert response from the server can include multiple certs that amount large payloads.

CoAP RFC section 4.6 describes the possible payload sizes: "if nothing is known about the size of the headers, good upper bounds are 1152 bytes for the message size and 1024 bytes for the payload size". Also "If IPv4 support on unusual networks is a consideration, implementations may want to limit themselves to more conservative IPv4 datagram sizes such as 576 bytes; per [RFC0791], the absolute minimum value of the IP MTU for IPv4 is as low as 68 bytes, which would leave only 40 bytes minus security overhead for a UDP payload". Thus, after the DTLS connection is established, fragmentation will sometimes be needed for the CoAP messages which involve certificate enrollment and management. A fragmentation solution for BRSKI and EST CoAP message is required.

The [I-D.ietf-core-block] document describes how fragmentation can be done by using a pair of Block options added to the CoAP flow. Block1 options are used by the client PUT and POST requests. A Block1 in a client request requires a Block1 option in the responses. A Block2 comes from a server response that will also need Block2 from the client to acknowledge the block and get the rest of blocks from the server. So, Block1 is used when a request (POST for example) is done in BRSKI over CoAP with a payload that needs fragmentation. Then the server responds with Block1 option to acknowledge the fragment-blocks. Block2 is used when a BRSKI server response is big and needs fragmentation. The Block2 acknowledgements are requests with the same options as the initial request and a Block2 option. "To influence the block size used in a response, the requester MAY also use the Block2 Option on the initial request, giving the desired size, a block number of zero and an M bit of zero". As explained in see section 2.8 of [I-D.ietf-core-block]), blockwise transfers SHOULD

be used in Confirmable COAP messages to avoid the exacerbation of

lost blocks.

In a scenario with a big BRSKI POST request we might have Block1 options from client to server and Block2 from server to client. In this case the Block1 blocks get completed and then the Block2 comes the other direction.

The BLOCK draft also defines Size1 and Size2 options. These are used to convey the size of the resources in the requests or responses. The Size1 response MAY be parsed by the client as an size indication of the Block2 resource in the server response or by the server as a request for a size estimate by the client. Similarly, Size2 option defined in BLOCK should be parsed by the server as an indication of the size of the resource carried in Block1 options and by the client as a maximum size expected in the 4.13 (Request Entity Too Large) response to a request.

6.1. Fragmented response example with Block2

An example of a server cacerts response that exceeds the MTU is:

An example of a server cacerts response that exceeds the MTU is

```
HTTP/1.1 200 OK
   Status: 200 OK
   Content-Type: application/pkcs7-mime; smime-type=certs-only
   Content-Transfer-Encoding: base64
   Content-Length: 1122

   MIIDOAYJKoZIhvcNAQcCoIIDKTCCAyUCAQExADALBgkqhkiG9w0BBwGgggMLMIID
   BzCCAe+gAwIBAgIBFTANBgkqhkiG9w0BAQUFADAbMRkwFwYDVQQDExBlc3RFeGFt
   cGxlQ0EgTndOMB4XDTEzMDUwOTIzMTU1M1oXDTE0MDUwOTIzMTU1M1owHzEdMBsG
   A1UEAxMUZGVtb3N0ZXA0IDEzNjgxNDEzNTIwggEiMA0GCSqGSIb3DQEBAQUAA4IB
   DwAwggEKAoIBAQClNp+kdz+Nj8XpEp9kaumWxDZ3eFYJpQKz9ddD5e5OzUeCm103
   ZIXQIxc0eVtMCatnRr3dnZRCAxGjwbqoB3eKt29/XSQffVv+odbyw0WdkQOIbntC
   Qry8YdcBZ+8LjI/N7M2krmjmoSLmLwU2V4aNKf0YMLR5Krmah3Ik31jmYCSvwTnv
   6mx6pr2pTJ82JavhTEIIt/fAYq1RYhkM1CXoBL+yhEoDanN7TzC94skfS3VV+f53
   J9SkUxTYcy1Rw0k3VXfxWwy+cSKEPREl7I6k0YeKtDEVAgBIEYM/L1S69RXTLuji
   rwnqSRjOquzkAkD31BE961KZCxeYGrhxaR4PAgMBAAGjUjBQMA4GA1UdDwEB/wQE
   AwIEsDAdBgNVHQ4EFgQU/qDdB6ii6icQ8wGMXvy1jfE4xtUwHwYDVR0jBBgwFoAU
   scRp5lujBKfYl6OLO7+5arIyQjwwDQYJKoZIhvcNAQEFBQADggEBACmxg1hvL6+7
   a+lFTARoxainBx5gxdZ9omSb0L+qL+4PDvg/+KHzKsDnMCrcU6M4YP5n0EDKmGa6
   4lY8fbET4tt7juJg6ixb95/760Th0vuctwkGr6+D6ETTfqyHnrbhX3lAhnB+0Ja7
   o1gv4CWxh1I8aRaTXdpOHORvN0SMXdcrlCys2vrtOl+LjR2a3kajJO6eQ5leOdzF
   QlZfOPhaLWen0e2BLNJI0vsC2Fa+2LMCnfC38XfGALa5A8e7fNHXWZBjXZLBCza3
   rEs9Mlh2CjA/ocSC/WxmMvd+Eqnt/FpggRy+F8IZSRvBaRUCtGE1lgDmu6AFUxce
   R4POrT2xz8ChADEA
```

Block options in CoAP messages can contain fields, SZX, M and NUM which are not affected by BRSKI.

Let's assume that the cacerts message will need to be broken up to 3 messages. The first Block2 will be:

```
  Ver = 1
  T = 2 (ACK)
  Code = 0x21 (2.01 success message.
         TODO: Do we need to create a 0x200 respond code.)
  Options
    Option1 (Content-Format)
      Option Delta = 0xC
      Option Length = 0xD
      Extended Option Length = 0x09
      Option Value = <number for application/pkcs7-mime;
                      smime-type=certs-only>
                (TODO: We need a new CoAP IANA registered value
                application/pkcs7-mime, application/csrattrs,
                application/pkcs10, application/pkcs8,
                application/pkcs12 )
    Option2 (Block2)
      Option Delta = 0xD
      Option Length = 0x1
      Extended Option Delta = 0x16
      Option Value = 0x0D
  Payload = MIIMOQYJKoZIhvcNAQcCoIIMKjCCDCYC \
            AQExADALBgkqhkiG9w0BBwGgggwMMIIC... (512 bytes)
```

The second Block2:

```
     Ver = 1
     T = 2 (means ACK)
     Code = 0x21
     Options
       Option1 (Content-Format)
         Option Delta = 0xC
         Option Length = 0xD
         Extended Option Length = 0x09
         Option Value = <number for application/pkcs7-mime;
                         smime-type=certs-only>
                    (TODO: We need a new CoAP IANA registered value
                    application/pkcs7-mime, application/csrattrs,
                    application/pkcs10, application/pkcs8,
                    application/pkcs12 )
       Option2 (Block2)
         Option Delta = 0xD
         Option Length = 0x1
         Extended Option Delta = 0x16
         Option Value = 0x1D
     Payload = ... (512 bytes)
```

The third and final Block2:

```
     Ver = 1
     T = 2 (means ACK)
     Code = 0x21
     Options
       Option1 (Content-Format)
         Option Delta = 0xC
         Option Length = 0xD
         Extended Option Length = 0x09
         Option Value = <number for application/pkcs7-mime;
                          smime-type=certs-only>
                    (TODO: We need a new CoAP IANA registered value
                    application/pkcs7-mime, application/csrattrs,
                    application/pkcs10, application/pkcs8,
                    application/pkcs12 )
       Option2 (Block2)
         Option Delta = 0xD
```

```
        Option Length = 0x1
        Extended Option Delta = 0x16
        Option Value = 0x25
    Payload = ...
```


[[TODO: This section to be populated. It will address how proxying can take place by an entity that resides at the edge of the CoAP network, such as the Registrar, and can reach the BRSKI server residing in a traditional "TCP setting".]]

[[TODO: This section to be populated. It will address transmission parameters for BRSKI described in sections 4.7 and 4.8 of the CoAP draft. BRSKI does not impose any unique parameters that affect the CoAP parameters in Table 2 and 3 in the CoAP draft but the ones in CoAP could be affecting BRSKI. For example the processing delay of CAs could be less then 2s, but in this case they should send an CoAP ACK every 2s while processing.]]

[[TODO: This section to be populated. This document describes an existing protocol moved to CoAP and there should not be additional security concerns added beyond the protocol's or CoAP's specifics security considerations.]]

-01:

Added more binding usecases and Block examples subsections to be expanded on later. Made various text improvements and clarifications.

Added two clarifying sentences in the relevant sections to address Brian C.'s comments and explain that message fragmentation can be avoided to a degree in DTLS and that fragments of block transfers should be confirmed to prevent exacerbated fragment loss in constrained networks.

11. Normative References

[I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Behringer, M., and S. Bjarnason, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-03 (work in progress), June 2016.

[I-D.ietf-core-block]
Bormann, D. and Z. Shelby, "Block-wise transfers in CoAP", draft-ietf-core-block-20 (work in progress), April 2016.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <http://www.rfc-editor.org/info/rfc7030>.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <http://www.rfc-editor.org/info/rfc7228>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <http://www.rfc-editor.org/info/rfc7252>.

Authors' Addresses

Max Pritikin
Cisco Systems

Email: pritikin@cisco.com

Panos Kampanakis
Cisco Systems

Email: pkampana@cisco.com