6tisch Working Group
Internet-Draft
Intended status: Informational
Expires: March 10, 2019

M. Richardson
Sandelman Software Works
J. Latour
CIRA Labs
F. Khan
Twelve Dot Systems
September 06, 2018

BRSKI enrollment for Smart Pledges
draft-richardson-anima-smartpledge-00

Abstract

This document details the mechanism used for initial enrollment by a smartphone into a BRSKI based enrollment system.

There are two key differences in assumption from [I-D.ietf-anima-bootstrapping-keyinfra]: that the intended registrar has Internet, and that the Pledge has no user-interface.

This variation on BRSKI is intended to be used in the situation where the registrar device is new out of the box and is the intended gateway to the Internet (such as a home gateway), but has not yet been configured. This work is also intended as a transition to the Wi-Fi Alliance work on the Device Provisioning Protocol (DPP).

Status of This Memo

Table of Contents

1. Introduction

The problem of bootstrapping a new device is described at length in [I-D.ietf-anima-bootstrapping-keyinfra] (aka BRSKI). The problem that BRSKI solves is the case of a smart, properly configured network with a minimum of network connectivity (or previously pre-previoned with nonceless vouchers), and a relatively stupid new device (the Pledge), which lacks a network interface.

The BRSKI problem is one of trust: how does the new device trust that it has found the correct network to join, and how does the new network become convinced that the new device is a device that is intended to join. BRSKI solves the problem well for the case where the network is well connected and easily talk to the device's Manufacturer Authorized Signing Authority (MASA), while providing appropriate proxy mechanisms to enable the new pledge to communicate it's proximity assertion to the MASA as well.

This document is about a variation of the problem: when the new device being introduce has no network connectivity, but a new device is intended to serve as the Registrar for the network. This new device is likely a home (or small office) gateway, and until it is properly configured there will be no direct network connectivity.

There are a number of protocols that permit an ISP to consider a new router brought into a home to be a new pledge to the ISPs' network, and for that new device to integrated into the ISP's (autonomic) network. BRSKI can be used itself, and there are ways to use the Broadband Form's TR-069 to bootstrap the device in this way. This document is not about the situation where the router device is intended to belong to the ISP, but about the situation where the home user intends to own and control the device.

1.1. Additional Motivation

The Wi-Fi Alliance has released the Device Provisioning Protocol [dpp]. The specification is not public. The specification relies on being able to send and receive 802.11 Public Action frames, as well as Generic Advertisement Service (GAS) Public Action frames. Access to send new layer-2 frames is generally restricted in most smartphone operating systems (iOS, Android). At present there are no known public APIs that a generic application writer could use, and therefore the smart-phone side of the DPP can only be implemented at present by the vendors of those operating systems.

As both dominant vendors have competing proprietary mechanisms, it is unclear if generic applications will be produced soon. It is there impossible for a vendor an a smart-appliance to independantly produce an application that can do proper DPP in 2018.

In addition to the above concern, DPP is primary concerned about provisioning WiFi credentials to devices. It assumes that the WiFi Access Point is already provisioned and functioning correctly.

The smartpledge enrollment described in this document is about securely initializing the administrative connection with a device that is the WiFi Access Point.

2. Terminology

The following terminology is copied from [I-D.ietf-anima-bootstrapping-keyinfra]

enrollment: The process where a device presents key material to a network and acquires a network specific identity. For example when a certificate signing request is presented to a certification authority and a certificate is obtained in response.

pledge: The prospective device, which has an identity installed at the factory.

IDevID: a manufacturer signed keypair (different from the QRkey) which is generated at the factory. This is the 802.1AR artifact which is mandated by [I-D.ietf-anima-bootstrapping-keyinfra].

The following new terminology has been added

smartpledge: The prospective administrator device, usually a smartphone equipped with a QR capable camera, wifi and 3G connectivity.

adolescent router (AR): a home router or device containing a registrar. The device does not yet have network connectivity, and has no administrator. It is considered not a "baby" device in the same way that the pledge is, but it is not yet an adult. A better term would be welcome.

SelfDevID: a public/private key pair generated by the smartpledge, formed into a self-signed PKIX certificate. The private key part remains always on the smartpledge, but like other secondary device keys, should be encrypted for backup purposes. {EDNOTE: any references to Apple or Android APIs/specifications here?}

QRkey: a unique, raw ECDSA or EdDSA key pair generated in (or for) the adolescent router at the factory, and stored in the configuration portion of the firmware. The public portion is printed in a QRcode. This key is not formed into a certificate of any kind.

3. Requirements Language

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [RFC2119] and indicate requirement levels for compliant STuPiD implementations.

4. Assumptions and Required Setup

The first assumption is that intended device owner is active and is present. The device owner has a smart-phone that is capable of using Wi-Fi or being wired into the adolescent router (AR).

The smartpledge application generates a self-signed certificate with public/private keypair that it knows. It may generate a unique certificate for each manufacturer. This certificate is called the SelfDevID.

The second assumption is that the device has a QR code printed on the outside of the unit, and/or provided with the packaging/ documentation. The QR code is as specified in section 5.3 of [dpp], with the additions specified in Section 6.1

The third assumption is that the AR, at manufacturing time, has the anchor for it's MASA (same assumption as for BRSKI pledge's). In addition, like the BRSKI pledge, the AR has an IDevID certificate (and associated private key) signed by the manufacturer.

The fourth assumption is that the key in the "K:" attribute Section 6.1 is a different public key pair. It MUST be different from the key used in the IDevID. This key is called the DPP-Keypair.

5. Protocol Overview

This is the overview of the process. {EDNOTE: there are many details here that belong in the next section. The goal in this section is to consisely explain the interaction among the components. Clearly this text currently fails in that regard}

5.1. Scan the QR code

The operator of the smartphone invokes the smartpledge application, and scans the QR code on the AR. The smartpledge learns the ESSID, Public-Key, mac-address, smartpledge URL, and link-local address of the AR.

5.2. Enroll with the manufacturer

The smartpledge uses it's 3G, or other WiFi internet access to connect to the manufacturer with TLS. The smartpledge does an HTTP POST to the provided URL using it's generated certificate as it's ClientCertificate. As described in Section 7, the manufacturer MAY respond with a 302 result code, and have the end user go through a web browser based process to enroll. After that process, a redirection will occur using OAUTH2.

The result should finally be a 201 result code, and at that URL is a new certificate signed by the manufacturer.

5.3. Connect to BRSKI join network

The application then reconnects the Wi-Fi interface of the smartphone to the ESSID of the AR. This involves normal 802.11 station attachment. The ESSID explicitely has no WPA or other security required on it.

There will be no DHCPv4. A IPv6 Router Solicitation may elicit an answer (confirming the device is there), but it is acceptable for there to be no prefix information. An IPv6 Neighbour Discovery is done for the IPv6 Link-Local address of the AR. Receipt of an answer confirms that the ESSID is correct and present.

(XXX - not using GRASP here. Could use GRASP, but QR code is better)

5.4. Connect to Adolescent Registrar (AR)

The smartpledge application then makes a direct (no proxy) TLS connection to port 443 of the AR, on the IPv6 Link-Local address given. This is as in section 5.1 of [I-D.ietf-anima-bootstrapping-keyinfra]. The smartpledge uses it's SelfDevID as the TLS ClientCertificate, as the smartpledge does not have a manufacturer signed IDevID.

Additionally, the AR will use it's IDevID certificate as the ServerCertificate of the TLS conncetion. As with other BRSKI IDevID, it will have a MASA URL extension, as described in [I-D.ietf-anima-bootstrapping-keyinfra] section 2.3.2.

5.5. Pledge Requests Voucher-Request from the Adolescent Registrar

The smartpledge generates a random nonce _SPnonce_. To this is adds SOMETHING-that-is-time-unique, to create a _voucher-request challenge_. This is placed in the voucher-challenge-nonce field.

Using the public-key of the AR that was scanned from the QR code, the smartpledge encrypts the challenge using CMS (or COSE?).

NOTE: DPP has a round with the SHA256 of the device's key to make sure that the correct device has been chosen. The TLS connection effectively provides the same privacy that the Bx keys provided.

The resulting object is POST'ed to the new BRSKI endpoint:

/.well-known/est/requestvoucherrequest

[or should it be named: /.well-known/est/requestvoucherchallenge

]

5.6. AR processing of voucher-request, request.

The AR processes this POST. First it uses the private key that is associated with it's QR printed public key to decrypt the voucher-request challenge. Included in this challenge is a nonce, and also the link-local address of the smartpledge.

The AR SHOULD verify that the link-local address matches the originating address of the connection on which the request is received.

The AR then forms a voucher-request identically to as described in section 5.2 of [I-D.ietf-anima-bootstrapping-keyinfra]. Note that

the AR uses it's IDevID to sign the voucher-request. This is the same key used to terminate the TLS connection. It MUST be different from the public key printed in the QR code.

In addition to the randomly generated nonce that the AR generates to place in the the voucher-request, into the nonce field, it also includes the _SPnonce_ in a new _voucher-challenge-nonce_ field. {EDNOTE: hash of nonce?}

This voucher-request is then _returned_ during the POST operation to the smartpledge. (This is in constrast that in ANIMA the voucher-request is sent by the device to the Registrar, or the MASA)

5.7. Smartpledge validates connection

The smartpledge then examines the resulting voucher-request. The smartpledge validates that the voucher-request is signed by the same public key as was seen in the TLS ServerCertificate.

The smartpledge then examines the contents of the voucher-request, and looks for the _voucher-challenge-nonce_. As this nonce was encrypted to the AR, the only way that the resulting nonce could be correct is if the correct private key was present on the AR to decrypt it. Succesful verification of the _voucher-challenge-nonce_ (or the hash of it, see below) results in the smartpledge moving it's end of the connection from provisional to validated.

5.8. Smart-Pledge connects to MASA

The smartpledge application then examines the MASA URL provided in the TLS ServerCertificate of the AR. The smartpledge application then connects to that URL using it's 3G/LTE connection, taking on the role of Registrar.

A wrapped voucher-request is formed by the smartpledge in the same way as described in section 5.4 of [I-D.ietf-anima-bootstrapping-keyinfra]. The inner prior-signed-voucher-request is filled in with the voucher-request that was created by the AR in the previous step.

The pinned-domain-cert of this voucher-request is set to be the SelfDevID certificate of the smartpledge. The voucher-request is to be signed by the SelfDevID.

The voucher-request is POST'ed to the MASA using the same URL that is used for Registrar/MASA operation:

/.well-known/est/requestvoucher

5.9. MASA processing

The MASA processing occurs as specified in section 5.5 of [I-D.ietf-anima-bootstrapping-keyinfra] as before. The MASA MUST also copy the voucher-challenge-nonce into the resulting voucher.

5.10. Smartpledge processing of voucher

The smartplege will receive a voucher that contains it's IDevID as the pinned-domain-cert, and the voucher-challenge-nonce that it created will also be present. The smartpledge SHOULD verify the signature on the artifact, but may be unable to validate that the certificate used has a relationship to the TLS ServerCertificate used by the MASA. (This limitation exists in ANIMA as well).

The smartpledge will then POST the resulting voucher to the AR using the URL

/.well-known/est/voucher

5.11. Adolescent Registrar (AR) receives voucher

When the AR receives the voucher, it validates that it is signed by it's manufacturer. This process is the same as section 5.5.1 of [I-D.ietf-anima-bootstrapping-keyinfra]. Note that this is the future Registrar that is performing what in ANIMA is a pledge operation.

Inside the voucher, the pinned-domain-cert is examined. It should match the TLS ClientCertificate that the smartpledge used to connect. This is the SelfDevID.

At this point the AR has validated the identity of the smartpledge, and the AR moves it's end of the connection from provisional to validated.

5.12. Adolescent Registrar (AR) grows up

The roles are now slightly changed. The AR generates a new key pair as it's Domain CA key. It MAY generate intermediate CA certificates and a seperate Registrar certificate, but this is discouraged for home network use.

The AR is now considered a full registrar.

5.13. Smartpledge enrolls

The smartpledge MUST now request the full list of CA Certificates, as per [RFC7030] section 4.1. As the Registrar's CA certificate has just been generated, the smartpledge has no other way of knowing it.

The smartpledge MUST now also generate a CSR request as per [I-D.ietf-anima-bootstrapping-keyinfra] section 5.8.3. The smartpledge MAY reuse the SelfDevID key pair for this purpose. (XXX - maybe there are good reasons not to reuse)

The Registrar SHOULD grant administrator privileges to the smartpledge via the certificate that is issued. This may be done via special attributes in the issued certificate, or it may pin the certificate into a database. Which method to use is a local matter.

The EST connection MUST remain open at this point.

5.14. Validation of connection

The smartpledge MUST now open a new HTTPS connection to the Registrar (AR), using it's newly issued certificate. (XXX should this be on a different IP, or a different port? If so, how is this indicated?)

The smartpledge MUST validate that the new connection has a certificate that is validated by the Registrar's new CA certificate.

The registrar MUST validate that the smartpledge's ClientCertificate is validated by the Registrar's CA. The smartpledge SHOULD perform a POST operation on this new connection to the [I-D.ietf-anima-bootstrapping-keyinfra] Enrollment Status Telemetry mechanism, see section 5.8.3. The EST connection MAY not be closed.

Should the validations above fail, then the original EST connection MUST be used to GET a value from the

/.well-known/est/enrollstatus

from the Registrar. The contents of this value SHOULD then be send to the MASA, using a POST to the enrollstatus, and including the reply from the AR in a new attribute, "adolescent-registrar-reason".

6. Protocol Details

6.1. Quick Response Code (QR code)

Section 5.3 of [dpp] describes the contents for an [iso18004] image. It specifies content that starts with DPP:, and the contains a series of semi-colon (;) deliminated section with a single letter and colon. This markup is terminated with a double semi-colon.

Although no amending formula is defined in DPP 1.0, this document is defining two extensions. This requires amending the ABNF from section 5.2.1 as follows:

```
dpp-qr = "DPP:" [channel-list ";"] [channel-list ";"]
         [mac ";"] [information ";"] public-key
         [";" llv6-addr ] [";" smartpledge ] [";" essid ] ";;"
llv6-addr = "L:" 8*hex-octet
essid     = "E:" *(%x20-3A / %x3C-7E) ; semicolon not allowed
smartpledge = "S:" *(%x20-3A / %x3C-7E) ; semicolon not allowed
```

While the ABNF defined in the [dpp] document assumes a specific order (C:, M:, I:, K:) the tags can come in any order. However, in order to make interoperation with future DPP-only clients as seamless as possible, the extensions suggested here are placed at the end of the list. This is consistent with the Postel Principle.

It is intended that parts of this protocol could be performed by an actual DPP implementation, should it become possible to implement DPP using current smartphone operating systems in an unprivileged way.

6.1.1. The SmartPledge Attribute

The _smartpledge_ attribute indicates that the device is capable of the protocol specified in this document. The contents of the smartpledge attribute contains part of a URL which identifies the manufacturer of this device, along with a unique token enabling service access to the device.

Short URLs are essential to fit into typical QR code space.

The smartpledge application prepends the text "https://" to the value provided to form the full address of the smartpledge enrollment.

6.1.2. Link-Layer Address Attribute

The _llv6-addr_ attribute is optional. When present, it specifies the IPv6 Link-Local address at which the adolescent router is listening. If not specified, then the link-local address may be formed according to the historical (privacy-violating) process described in [RFC4291] Appendix A. The _llv6-addr_ attribute is

present so that devices that have implemented [RFC7217] stable addresses can express that address clearly.

6.1.3. ESSID Name Attribute

The _essid_ attribute provides the name of the 802.11 network to which the _smartpledge_ SHOULD join in order to reach the AR. If this attribute is absent, then it defaults to "BRSKI".

6.2. Artifacts

6.2.1. Voucher-Request Challenge

The smartpledge generates a random nonce _SPnonce_. To this is adds SOMETHING-that-is-time-unique, to create a _voucher-request challenge_. This is placed in the voucher-challenge-nonce field.

Using the public-key of the AR that was scanned from the QR code, the smartpledge encrypts the challenge using CMS (or COSE?).

6.2.2. Additions to Voucher-Request

QUESTION: should the _voucher-challenge-nonce_ be provided directly in the voucher-request, or should only a hash of the nonce be used? The nonce is otherwise not disclosed, and a MITM on the initial TLS connection would get to see the nonce. A hash of the nonce validates the nonce as easily.

6.3. Enrollment using EST

TBD

7. Smart Pledge enrollment with manufacturer

TBD.

8. Threat Analysis

The following attacks have been considered.

8.1. Wrong Administrator

Neighbours with similar setups wind up managing each other's network (by mistake).

8.2. Rogue Administrator

Uninitialized networks can be adopted by 'wardrivers' who search for networks that have no administrator.

8.3. Attack from Internal device

A compromised device inside the home can be used by an attack to take control of the home router.

8.4. Attack from camera enabled robot

A robot (such as a home vacuum cleaner) could be compromised, and then used by an attacker to observe and/or scan the router QRcode.

8.5. Attack from manipulator enabled robot

A robot (for instance, a toy) could be compromised, and then used by an attacker to push the WPA and/or factory reset button on the router.

9. Security Considerations

Go through the list of attacks above, and explain how each has been mitigated.

Go through the list of concerns in ANIMA and EST-RFC7030 and indicate if there are additional concerns, or if a concern does not apply.

10. IANA Considerations

TBD.

11. Acknowledgements

This work was supported by the Canadian Internet Registration Authority https://cira.ca/blogs/cira-labs/about-cira-labs.

12. References

12.1. Normative References

[dpp] "Device Provisioning Protocol Specification", n.d., <https://www.wi-fi.org/downloads-registered-guest/Device_Provisioning_Protocol_Draft_Technical_Specification_Package_v0_0_23_0.zip/31255>.

[I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Behringer, M., Bjarnason, S., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-16 (work in progress), June 2018.

[iso18004]
"Information technology --- Automatic identification and data capture techniques --- Bar code symbology --- QR Codes (ISO/IEC 18004:2015)", n.d., <https://github.com/yansikeim/QR-Code/blob/master/ISO%20IEC%2018004%202015%20Standard.pdf>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://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, <https://www.rfc-editor.org/info/rfc7030>.

12.2. Informative References

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/info/rfc4291>.

[RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <https://www.rfc-editor.org/info/rfc7217>.

Authors' Addresses

Michael Richardson
Sandelman Software Works

Email: mcr+ietf@sandelman.ca

Jacques Latour
CIRA Labs

Email: Jacques.Latour@cira.ca

Faud Khan
Twelve Dot Systems

Email: faud.khan@twelvedot.com