IOTOPS Working Group   M. Richardson
Internet-Draft   Sandelman Software Works
Intended status: Informational   J. Latour
Expires: May 6, 2021   CIRA Labs
                                                       November 02, 2020

BRSKI enrollment of with disconnected Registrars -- smarkaklink
draft-richardson-anima-smarkaklink-03

Abstract

This document details the mechanism used for initial enrollment using a smartphone of a BRSKI Registrar 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

Copyright Notice

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 user 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 can 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. Intermittent Device connectivity

There is an additional variation which this variation solves: the case where there is one or more devices in a place with no immediate connectivity to a Registrar. An example of this could be a new home construction where a furnace, thermostat or other control systems need to be introduced to each other. If a registrar exists it will have no Internet connectivity (as above), until the home becomes owned by the first owner. There might never be a registrar though.

The basement case is important because the assumption is that the _installer_ may have poor or no LTE connectivity in that location. The installer will have to exit the basement, perhaps even return to their truck, in order to have network connectivity for their provisioning device (a smartphone equivalent).

1.2. Additional Motivation

The Wi-Fi Alliance has released the Device Provisioning Protocol [dpp]. The specification is available only via "free" registation. 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 probably impractical for a vendor an a smart-appliance to independantly produce an application that can do proper DPP in 2019. As one of the common goals of this document and DPP is that there need not be an application-per-device only one DPP application need exist. Until such time as such an application becomes universal it is a goal of this document to lay the groundwork for a transition to full use of DPP by leveraging the QR code infrastructure that DPP depends upon.

In addition to the above concern, DPP is primary concerned about provisioning WiFi credentials to devices. DPP can provision access points themselves, but it lacks any kind of manufacturer integration. BRSKI provides this integration, and therefore an audit trail history for the device.

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

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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

smarkaphone: 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.

smarkaklink: the name of this protocol.

adolescent router (AR): a home router or device containing a registrar. The device does not yet have network connectivity, and has no administrator.

2.1. History and Origin of the name

This document was originally called the "smartpledge" variation of BRSKI. This name was intended to indicate that the variation is one where the BRSKI role of Pledge is taken on by the smartphone device.

While the end-goal is to have the smartphone enrolled into a PKI hosted by the fully-grown Router, the activities of each device do not map into the BRSKI roles at the beginning. In fact, they are reversed with the Adolescent Router being the Pledge. Review of this document suggested that removing the word pledge would help.

The new name "smarkaklink" is intended to sound like the sound that two (wine, beer) glasses make after a toast is made.

3. 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 5.1

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 5.1 is a different public key pair. It MUST be different from the key used in the IDevID. This key is called the DPP-Keypair.

4. 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}

4.1. Scan the QR code

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

4.2. Enroll with the manufacturer

The smartphone uses it's 3G, or other WiFi internet access to connect to the manufacturer with TLS. The manufacturer is identified with the smartpledge URL.

The operator of the smartphone may need to move to another location to get connectivity. It is desireable that an operator be able to scan many QR codes before moving, performing this operation in a batch. There may be multiple devices from the same manufacturer, and the smarkalink application SHOUld enroll with the manufacturer a single time for all devices.

The smartphone does an HTTP POST to the provided URL using it's generated certificate as it's ClientCertificate. As described in Section 6, 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.

4.3. Gather name details for new device

The contents of the scanned QR code may some necessary information to facilititate the connection. After enrolling with the manufacturer, the smartphone makes a request to the manufacturer to get more details.

A POST request is made containing the public key of the device. The public key is used as an index, and this MAY result in a literal reply, or may result in an HTTP 201 response with a Location: header where details on the device may be obtained.

The final result is a JSON object that provides additional details about the device. At present, a key detail that does not fit into the QR code is the full certificate distinguished name that the device will respond with.

This is the Adolescent Router Fully Qualified Domain Name, or AD-FQDN.

Additionally, if the device does not have a certificate that has a public WebPKI anchor, then the manufacturer will include an appropriate trust anchor with which to validate the device. If the device has a self-signed certificate, then it may be returned.

4.4. 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 given ESSID explicitely has no WPA or other security required on it.

There will be no DHCPv4 on this network. This simplifies the operation of the devices that are enrolling, but it also makes the network uninteresting to other random users that may stumble upon the open ESSID.

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)

4.5. Connect to Adolescent Registrar (AR)

The smarkaklink application then makes a direct (no proxy) TLS connection to port 8081 (!To be confirmed!) 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 smartphone uses it's SelfDevID as the TLS ClientCertificate, as the smartphone and smarkaklink will 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.

The Adolescent Registrar is acting here in the role of pledge.

Given typical libraries, the connection will be made initially with all TLS peer name validation turned off, as the connection will not be to an IPv6 Link-Local address, which can not be placed into a certificate. The certificate should still be verified if possible up to a public trust anchor.

After connecting, the certificate presented by the AR MUST contain the AR-FQDN provided above as a subjectAltName rfc822Name extension.

Alternatively, a custom certification verification call back may be made.

4.6. Smartphone Requests Voucher-Request from the Adolescent Registrar

The smartphone generates a random nonce _SPnonce_. (Do we need something time-based too here?)

The smarkaklink client encrypts this to the public key of the AR, which was found in the QR code. If the key is RSA, this is a simple public key operation (we need to reference appropriate padding due to various attacks). In the mandatory to implement ECDSA case, then ECIES is used instead.

The result goes into the Voucher-Request-Request, posted as a JSON containing a single field: _voucher-request challenge_. This is placed in the voucher-challenge-nonce field.

(XXX-should be done with JOSE? Probably)

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

]

4.7. 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 smartphone.

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.

Note: 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 smartphone. (This is in constrast that in ANIMA the voucher-request is sent by the device to the Registrar, or the MASA)

4.7.1. 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.

4.8. Smartphone validates connection

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

The smartphone 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 smartphone moving it's end of the connection from provisional to validated.

4.9. Smart-Phone connects to MASA

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

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

The proximity-registrar-cert of the wrapped voucher-request is set to be the SelfDevID certificate of the smartphone. 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

4.10. 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.

4.11. Smartpledge processing of voucher

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

/.well-known/est/voucher

If an existing TLS connection is still available, it MAY be reused.

If a TLS session-resumption ticket (see [RFC8446] section 2.2 for TLS 1.3, and [RFC5077] for TLS 1.2) has been obtained, it SHOULD be used if the TLS connection needs to be rebuilt. This is particularly useful in the disconnected use case explained in Section 1.1.

4.12. 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].

Again note that the AR is acting in the role of a pledge.

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

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

4.13. Adolescent Registrar (AR) grows up

The roles are now changed.

If necessary, 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. The AR now takes on the role of Registrar.

4.14. Enrollment status

The AR responds to the POST of the voucher with the enrollment status as the reply to the POST, as per the enrollment status object defined in BRSKI section 5.9.4.

The smartphone MAY POST the resulting enrollment status to the MASA, in a manner similar to BRSKI section 5.9.4. Note that the operation described in that section is about telemetry from the pledge to the Registrar. That telemetry was return as part of the POST above.

Returning enrollment status to the MASA is an optional action; while there are privacy implications of doing so, the logicstics feedback of a failure likely will result in a service technician visit, which is hardly privacy enhancing. This telemetry return is strongly encouraged.

The POST to /.well-known/est/enrollstatus MUST include some additional information to tell the MASA which device was involved. This section therefore defines a new element for the status return to be the "voucher" attribute; it is to be filled in with the base64

encoding of the voucher that was provided. While this is excessive, it discloses no information that the MASA does not already have, has been signed and identifies both the Adolescent Router and the Smart Phone client involved.

4.15. Smartphone enrolls

At this stage of the smarkaklink protocol, the typical BRSKI exchange is over. A Secure Transport has been established between the smartphone and the fully-grown AR. The smartphone now takes on the role of secured pledge, or EST client.

The smartphone 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 smartphone has no other way of knowing it.

The smartphone 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 smartphone via the certificate that is issued. This may be done via special attributes in the issued certificate, or it may pin the certificate in a database. Which method to use is a local matter.

The TLS/EST connection MUST remain open at this point. This is connection one.

4.16. Validation of connection

The smartphone 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 smartphone MUST validate that the new connection's TLS Server certificate can be validated by the Registrar's new CA certificate.

The registrar MUST validate that the smartphone's ClientCertificate is validated by the Registrar's CA. The smartphone 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.

Upon success, the original TLS/EST connection (one) MAY now be closed.

5. Protocol Details

5.1. 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 ] [";" mudurl ]
         [";" smarkaklink ] [";" essid ] ";;"
llv6-addr = "L:" 8*hex-octet
essid     = "E:" *(%x20-3A / %x3C-7E) ;   semicolon not allowed
smarkaklink = "S:" *(%x20-3A / %x3C-7E) ; semicolon not allowed
mudurl      = "D:" *(%x20-3A / %x3C-7E) ; semicolon not allowed
```

While the ABNF defined in the [dpp] document assumes a specific order (C:, M:, I:, K:), this specification relaxes this so that 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.

5.1.1. The Smarkaklink Attribute

The _smarkaklink_ attribute indicates that the device is capable of the protocol specified in this document. The contents of the smarkaklink attribute contains part or all of an IRI which identifies the manufacturer of the device.

It SHOULD contain the _iauthority_ of an IRI as specified in section 2.2 of [RFC3987]. The scheme is implicitely "https://", with an ipath of "/.well-known/est/smarkaklink". This implicit form exists to save bytes in the QR code.

If the string contains any "/" characters, then it is not an _iauthority_, but an entire IRI. This takes many more characters,

but is useful in a variety of debugging situations, and also provides for new innovations.

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

5.1.2. 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.

5.1.3. ESSID Name Attribute

The _essid_ attribute provides the name of the 802.11 network on which the enrollment will occur. If this attribute is absent, then it defaults to "BRSKI".

5.2. Enrollment using EST

TBD

6. Smart Pledge enrollment with manufacturer

While it is assumed that there will be many makers of Smarkaklink applications, a goal of this specification is to eliminate the need for an "app" per device, providing onboarding mechanism for a variety of devices from a single app.

Given the secondary goal of a transition to use of Device Provisioning Protocol (DPP), the smarkaklink application may have to be provided as part of the smart phone system, as a system service. This is due to the need to send/receive wifi management frames from DPP. As such each vendor of a smart device will need to produce a smarkaklink app, and it will be impossible for the vendor of the Registrar device (or other DPP capable IoT device) to provide an app on their own.

Having stated this goal, it is understood that initially the app may well come from the manufacturer of the Registrar, but this protocol is designed on the assumption that there is no such vertical integration.

So, there can be no initial relationship between the Smart Pledge and the manufacturer of the Registrar.

But, in a traditional [I-D.ietf-anima-bootstrapping-keyinfra] scenario the pledge would have been provided with an IDevID at manufacturing time. While an IDevID could have been built-in to the SmartPledge "app", such a key would not be private if it was built-in. A key could be generated by the app upon installation. It could be self-signed, it could be signed by the maker of the app, or it could be signed by another party.

- a self-signed certificate is just a container for a public key. For the purposes of the trust relationship with the Registrar, it would be sufficient.

- a certificate signed by the maker of the app (or the maker of the smart-phone) would carry no specific trust beyond what a self-signed certificate would have. Any linking in the certificate to a network expressable identity (such as layer-2 address) would simply be a privacy violation.

- a certificate signed by another party would similarly have little additional relevance, unless the third party is the manufacturer of the Registrar!

The smarkaklink enrollment process uses a combination of the first and third choice. The involvement of the manufacturer at this step affords an opportuntity to do sales-channel integration with the manufacturer. The manufacturer can associate an account with the user using a wide variety of OAUTH2 [RFC6749] processes. In addition, based upon the URL provided the manufacturer can do redirection along a value-added reseller process. For instance, the manufacturer of a home router could redirect the pledge to the ISP that resold the router.

While [RFC7030] describes a Certificate Signing Request in order to have a certificate assigned, the actual contents of the certificate are not interesting at all, and the process of attempting to come up with a meaningful contents tends to cause more interoperability issues than having nothing.

The Smarkaklink takes the _smartpledge attribute_ from the QR code, forming a URL as describe above. An HTTPS POST is performed to this URL, with the JSON body of:

```
{
   "mac" : <mac-address>
}
```

The HTTPS POST MUST be performed with freshly created self-signed certificate. If the smarkaklink application has previously

communicated with this URL, it MAY skip this step and use a previously returned certificate. Doing so has a privacy implication discussed below, but is appropriate when enrolling many devices from the same manufacturer into the same network.

The smarkaklink client should be prepared for three cases:

- o A certificate is immediately returned.
- o A 201 status code is returned, and Location: header is provided. A GET request to that location will retrieve the certificate.
- o A 302 redirection occurs with some initiation of an OAUTH2 process to establish some additional authorization.
- o Any other error (4xx and 5xx) are typically unrecoverable errors.

In the third case, the 302 response SHOULD take the smarkaklink operator to the given URL in an interactive browser. The operator SHOULD be given access to their normal set of cookies and third-party logins such that they can use appropriate third party (Google, Facebook, Github, Live.com, etc.) logins to help validate the operator as a real person, and not a malware. Such logins are optional, and it is a manufacturer choice as to what integrations they want to make.

After the OAUTH2 process, the SmartPledge will be redirected back to the MASA and a 201 status code will be returned when successful as above.

6.1. minimal Smart Pledge enrollment

A manufacturer who has not built-in any restrictions on the identity that the smarkaklink uses, MAY return the same self-signed certificate that the smartpledge used to connect with.

7. Threat Analysis

The following attacks have been considered.

7.1. Wrong Administrator

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

7.2. Rogue Administrator

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

7.3. Attack from Internal device

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

7.4. 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.

7.5. 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.

8. Security Considerations

XXX: 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.

9. IANA Considerations

TBD.

10. Acknowledgements

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

11. References

11.1. 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., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-44 (work in progress), September 2020.

[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>.

[RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, DOI 10.17487/RFC3987, January 2005, <https://www.rfc-editor.org/info/rfc3987>.

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

11.2. 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>.

[RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, DOI 10.17487/RFC5077, January 2008, <https://www.rfc-editor.org/info/rfc5077>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

Appendix A. Resulting DPP QR code specification

This is a merge of the additions from section Section 5.1 and section 5.2.1 of [dpp]:

```
dpp-qr = "DPP:" [channel-list ";"]
         [";" llv6-addr ] [";" mudurl ]
         [";" smarkaklink ] [";" essid ] ";;"
llv6-addr = "L:" 8*hex-octet
essid     = "E:" *(%x20-3A / %x3C-7E) ;   semicolon not allowed
smarkaklink = "S:" *(%x20-3A / %x3C-7E) ; semicolon not allowed
mudurl      = "D:" *(%x20-3A / %x3C-7E) ; semicolon not allowed
pkex-bootstrap-info = [information]
channel-list = "C:" class-and-channels *("," class-and-channels)
class-and-channels = class "/" channel *("," channel)
class = 1*3DIGIT
channel = 1*3DIGIT
mac = "M:" 6hex-octet ; MAC address
hex-octet = 2HEXDIG
information = "I:" *(%x20-3A / %x3C-7E) ; semicolon not allowed
public-key = "K:" *PKCHAR
      ; DER of ASN.1 SubjectPublicKeyInfo encoded in
      ; "base64" as per [14]
PKCHAR = ALPHA / DIGIT / %x2b / %x2f / %x3d
llv6-addr = "L:" 8*hex-octet
essid     = "E:" *(%x21-3A / %x3C-7E) ; semicolon not allowed
smartpledge = "S:" *(%x21-3A / %x3C-7E) ; semicolon not allowed
```

Appendix B. Swagger.IO definition of API

This is a work-in-progress definition of the smarkaklink to MASA API in the form of Swagger.IO format:

```
---
swagger: "2.0"
info:
  description: |
    The smartpledge API is described in detail in
```

```
      draft-richardson-anima-smartpledge. This API is
      a variation of BRSKI (draft-ietf-anima-bootstrapping-keyinfra)
      which provides an initial bootstrap of the
      Secure Home Gateway registrar.
    version: 1.0.0
    title: Secure Home Gateway secure enrollment API (smartpledge-BRSKI)
    contact:
      email: securehomegateway@cira.ca
    license:
      name: Apache 2.0
      url: http://www.apache.org/licenses/LICENSE-2.0.html
  host: virtserver.swaggerhub.com
  basePath: /CIRALabs/smartpledge/1.0.0
  tags:
  - name: est
    description: Enrollment over Secure Transport
  schemes:
  - https
  paths:
    /voucherrequest:
      get:
        tags:
        - developers
        summary: searches inventory
        description: |
          By passing in the appropriate options, you can search for
          available inventory in the system
        operationId: searchInventory
        produces:
        - application/json
        parameters:
        - name: searchString
          in: query
          description: |
            pass an optional search string for looking up inventory
          required: false
          type: string
        - name: skip
          in: query
          description: number of records to skip for pagination
          required: false
          type: integer
          minimum: 0
          format: int32
        - name: limit
          in: query
          description: maximum number of records to return
          required: false
```

```
          type: integer
          maximum: 50
          minimum: 0
          format: int32
        responses:
          200:
            description: search results matching criteria
            schema:
              type: array
              items:
                $ref: '#/definitions/InventoryItem'
          400:
            description: bad input parameter
      post:
        tags:
        - admins
        summary: adds an inventory item
        description: Adds an item to the system
        operationId: addInventory
        consumes:
        - application/json
        produces:
        - application/json
        parameters:
        - in: body
          name: inventoryItem
          description: Inventory item to add
          required: false
          schema:
            $ref: '#/definitions/InventoryItem'
        responses:
          201:
            description: item created
          400:
            description: invalid input, object invalid
          409:
            description: an existing item already exists
  definitions:
    InventoryItem:
      type: object
      required:
      - id
      - manufacturer
      - name
      - releaseDate
      properties:
        id:
          type: string
```

```
          format: uuid
          example: d290f1ee-6c54-4b01-90e6-d701748f0851
        name:
          type: string
          example: Widget Adapter
        releaseDate:
          type: string
          format: date-time
          example: 2016-08-29T09:12:33.001Z
        manufacturer:
          $ref: '#/definitions/Manufacturer'
    Manufacturer:
      required:
      - name
      properties:
        name:
          type: string
          example: ACME Corporation
        homePage:
          type: string
          format: url
          example: https://www.acme-corp.com
        phone:
          type: string
          example: 408-867-5309
```

Authors' Addresses

Michael Richardson
Sandelman Software Works

Email: mcr+ietf@sandelman.ca

Jacques Latour
CIRA Labs

Email: Jacques.Latour@cira.ca