ANIMA WG
Internet-Draft
Intended status: Informational
Expires: August 17, 2015

M. Pritikin
M. Behringer
S. Bjarnason
Cisco

February 13, 2015

Bootstrapping Key Infrastructures
draft-pritikin-anima-bootstrapping-keyinfra-01

Abstract

This document specifies automated bootstrapping of an key infrastructure using vendor installed IEEE 802.1AR manufacturing installed certificates, in combination with a vendor based service on the Internet. Before being authenticated, a new device has only link-local connectivity, and does not require a routable address. When a vendor provides an Internet based service, devices can be forced to join only specific domains but for constrained environments we describe a variety of options that allow bootstrapping to proceed.

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 August 17, 2015.

Table of Contents

1. Introduction

To literally "pull yourself up by the bootstraps" is an impossible action. Similarly the secure establishment of a key infrastructure without external help is also an impossibility. Today it is accepted that the initial connections between nodes are insecure, until key distribution is complete, or that domain-specific keying material is pre-provisioned on each new device in a costly and non-scalable manner. This document describes a zero-touch approach to bootstrapping an entity by securing the initial distribution of key material using third-party generic keying material, such as a manufacturer installed IEEE 802.1AR certificate [IDevID], and a corresponding third-party service on the Internet.

The two sides of an association being bootstrapped authenticate each other and then determine appropriate authorization. This process is described as four distinct steps between the existing domain and the new entity being added:

- New entity authentication: "Who is this? What is its identity?"
- New entity authorization: "Is it mine? Do I want it? What are the chances it has been compromised?"
- Domain authentication: "What is this domain's claimed identity?"
- Domain authorization: "Should I join it?"

A precise answer to these questions can not be obtained without leveraging an established key infrastructure(s). The domain's decisions are based on the new entity's authenticated identity, as established by verification of previously installed credentials such as a manufacturer installed IEEE 802.1AR certificate, and verified back-end information such as a configured list of purchased devices or communication with a trusted third-party. The new entity's decisions are made according to verified communication with a trusted third-party or in a strictly auditable fasion.

Optimal security is achieved with IEEE 802.1AR certificates on each new entity, accompanied by a third-party Internet based service for verification. The concept also works with less requirements, but is then less secure. A domain can choose to accept lower levels of security when a trusted third-party is not available so that bootstrapping proceeds even at the risk of reduced security. Only the domain can make these decisions based on administrative input and known behavior of the new entity.

The result of bootstrapping is that a domain specific key infrastructure is deployed. Since IEEE 802.1AR PKI certificates are used for identifying the new entity and the public key of the domain identity is leveraged during communiciations with an Internet based service, which is itself authenticated using HTTPS, bootstrapping of a domain specific Public Key Infrastructure (PKI) is fully described. Sufficient agility to support bootstrapping alternative key infrastructures (such as symmetric key solutions) is considered although no such key infrastructure is described.

1.1. 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].

The following terms are defined for clarity:

2. Architectural Overview

The logical elements of the bootstrapping framework are described in this section. Figure 1 provides a simplified overview of the components. Each component is logical and may be combined with other components as necessary.

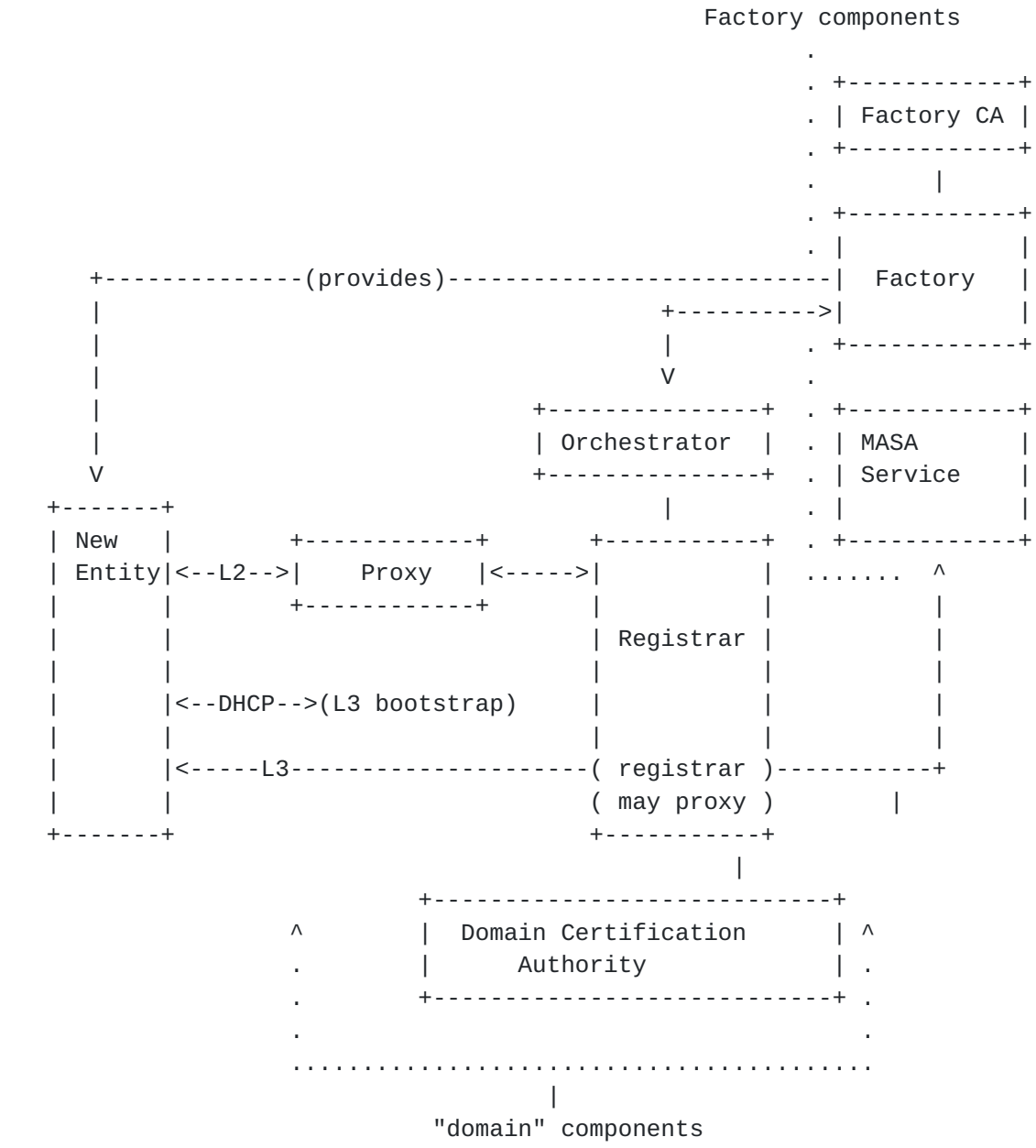

Figure 1

Domain: The set of entities that trust a common key infrastructure trust anchor.

Domain CA: The domain Certification Authority (CA) provides certification functionalities to the domain. At a minimum it provides certification functionalities to the Registrar and stores

the trust anchor that defines the domain. Optionally, it certifies all elements.

Domain Identity: The domain identity is the 160-bit SHA-1 hash of the BIT STRING of the subjectPublicKey of the domain trust anchor that is stored by the Domain CA. This is consistent with the RFC5280 Certification Authority subject key identifier of the Domain CA's self signed root certificate. (A string value bound to the Domain CA's self signed root certificate subject and issuer fields is often colloquially used as a humanized identity value but during protocol discussions the more exact term as defined here is used).

Orchestrator: Although bootstrapping of an individual device is automated and requires zero administrative involvement (particularly on the New Entity) the orchestrator drives general operations of the domain. This can be an automated process or a human administrator, see Section 3.3 for more details.

Factory: This instantiates the New Entity. For physical devices this can be representative of third-party vendor manufacturing, ordering and shipping process(es) that results in a physical hardware device with an IEEE 802.1AR identity being drop shipped to a destination domain for physical installation. In a virtual machine environment this can be the virtual machine hypervisor control software that initiates a virtual machine instance, in which case the factory is a "virtual factory" and might be managed by the domain itself.

Factory CA: This Certification Authority is leveraged by the Factory to issue IEEE 802.1AR identities to each New Entity. For a virtual factory it may be reasonable to assume the domain certification authority is directly used but in a complex environment it is assumed the Factory does not have direct access to the Domain Certification Authority.

Registrar: A representative of the domain that is configured, perhaps autonomically, to decide whether a new device is allowed to join the domain. The administrator of the domain interfaces with a Registrar to control this process. Typically a Registrar is "inside" its domain.

New Entity: A new device or virtual machine or software component that is not yet part of the domain.

Proxy: A domain entity that helps the New Entity join the domain. A Proxy facilitates communication for devices that find themselves

in an environment where they are not provided L3 connectivity until after they are validated as members of the domain.

MASA Service: A Manufacturer Authorized Signing Authority (MASA) service on the global Internet. At a minimum the MASA provides a trusted repository for audit information concerning privacy protected bootstrapping events. As a service offering the MASA can incorporate many of the bootstrapping elements (such as the Registrar and the Domain CA) into the overall service. The MASA is not a mandatory component, but it enables the new device to validate which domain it is joining. This allows for a completely secure zero-touch bootstrap of domain certificates with mutual authentication (device <-> domain).

We assume a multi-vendor network. In such an environment, there could a MASA for each vendor that supports devices following this document's specification, or an integrator could provide a MASA service for all devices which he supplies. Note again that the MASA is not mandatory. Also, this approach describes a secure zero-touch approach to bootstrapping a key infrastructure; if certain devices in a network do not support this approach, they can still be bootstrapped manually.

3. Operational Overview

This section describes how an operator interacts with a domain that supports the bootstrapping as described in this document.

3.1. Instantiating the Domain Certification Authority

This is a one time step by the domain administrator. This is an "off the shelf" CA with the exception that it is designed to work as an integrated part of the security solution. This precludes the use of 3rd party certification authority services that do not provide support for delegation of certificate issuance decisions to a domain managed Registration Authority.

3.2. Instantiating the Registrar

This is a one time step by the domain administrator. One or more devices in the domain are configured take on a Registrar function.

A device can be configured to act as a Registrar or a device can auto-select itself to take on this function, using a detection mechanism to resolve potential conflicts and setup communication with the Domain Certification Authority. Automated Registrar selection is outside scope for this document.

3.3. Accepting New Entities

For each New Entity the Registrar is informed the unique identifier (e.g. serial number) along with the manufacturer's identifying information (e.g. manufacturer root certificate). This can happen in different ways:

1. Default acceptance: In the simplest case, the new device itself provides its identity to the registrar, which then accepts any device blindly, without validating its identity. This mode does not provide any security against intruders and is not recommended.

2. Per device acceptance: Also here the device provides its identity directly to the registrar during enrollment. A non-technical human validates the identity, for example by comparing the identity displayed by the registrar (for example using a smartphone app) with the identity shown on the packaging of the device. Acceptance may be triggered by a click on a smartphone app "accept this device", or by other forms of pairing. See also [I-D.behringer-homenet-trust-bootstrap] for how the approach could work in a homenet.

3. Whitelist approach: In larger networks, neither of the previous approaches is acceptable. Default acceptance is not secure, and a manual real-time acceptance per device does not scale. Here, the registrar is provided a priori with a list of identifiers of devices that belong to the network. This list can be for example extracted from an inventory database, or sales records. If a device is detected that is not on the list of known devices, it can still be manually accepted or declined.

4. Automated Orchestrator: an automated process that queries the MASA service or an inventory database either a priori for all devices, or in real time for each new device. It feeds this information into the Registrar. Once set up, no human intervention is required in this process.

None of the approaches requires the network to have permanent Internet connectivity. Even when the Internet based MASA service is used, it is possible to pre-fetch the required information from the MASA a priori, for example at time of purchase. In this case devices can enrol later even in a completely isolated network.

Additional policy can be stored for future authorization decisions. For example an expected deployment time window or that a certain Proxy must be used.

3.4. Automatic Enrolment of Devices

The approach outlined in this document provides a secure zero-touch method to enrol new devices without any pre-staged configuration. New devices communicate with already enrolled devices of the domain, which proxy between the new device and a Registrar. As a result of this completely automatic operation, all devices obtain a domain based certificate.

3.5. Operating the Network

The certificate installed in the previous step can be used for all subsequent operations. For example, to determine the boundaries of the domain: If a neighbor has a certificate from the same trust anchor it can be assumed "inside" the same organization; if not, as outside. See also Section 4.5.1. The certificate can also be used to securely establish a connection between devices and central control functions. Also autonomic transactions can use the domain certificates to authenticate and/or encrypt direct interactions between devices. The usage of the domain certificates is outside scope for this document.

4. Functional Overview

Entities behave in an autonomic fashion. They discover each other and autonomically establish a key infrastructure deliminating the autonomic domain. See [I-D.irtf-nmrg-autonomic-network-definitions] for more information.

The overall flow is shown in Figure 2:

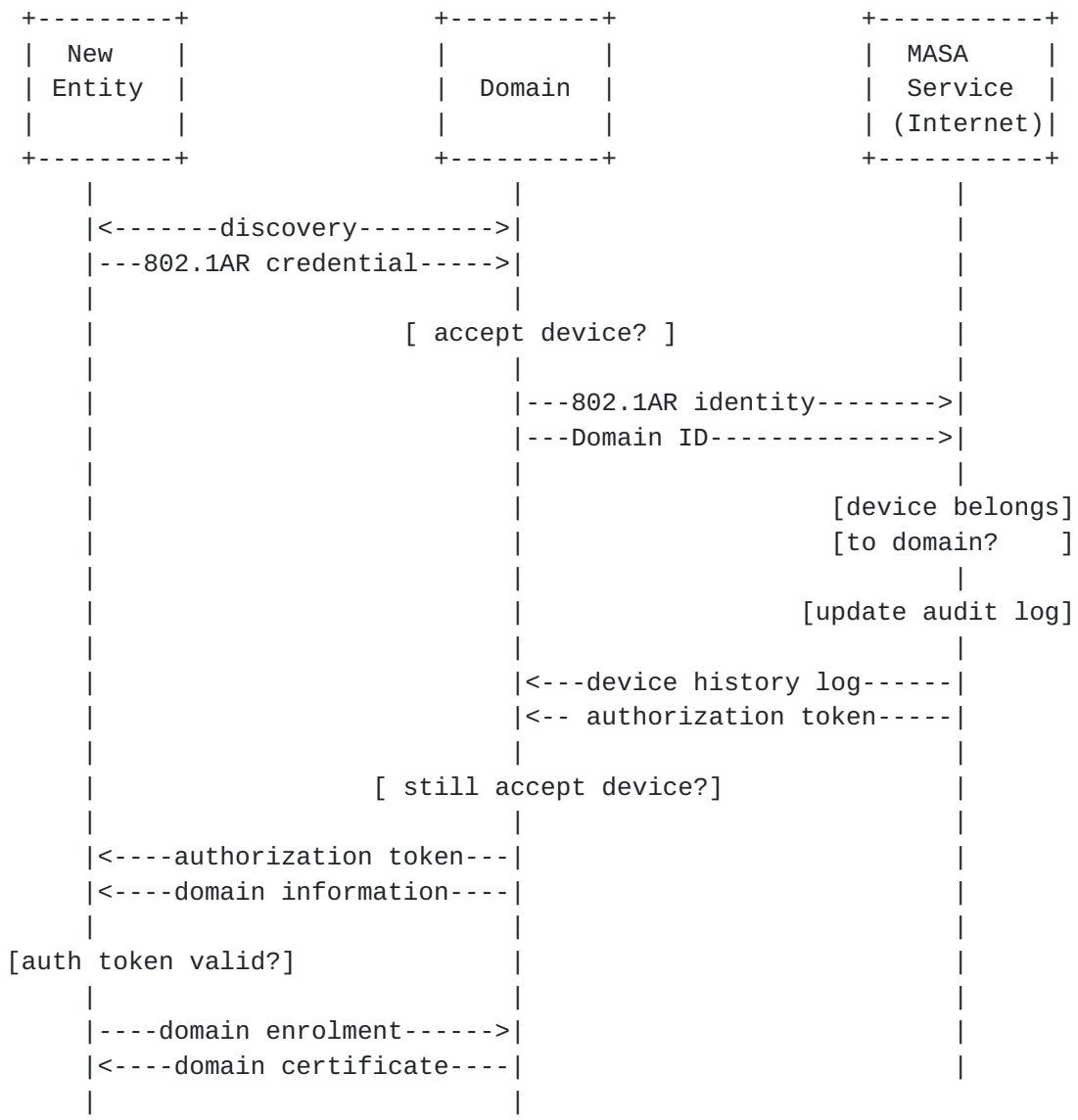

Figure 1

4.1. Behavior of a new entity

A New Entity that has not yet been bootstrapped attempts to find a local domain and join it. A number of methods are attempted for establishing communications with the domain in a specified order.

Client behavior is as follows:

1. Discover a communication channel to the "closest" Registrar by trying the following steps in this order:

A. Search for a Proxy on the local link using a link local discovery protocol (no routable addresses are required for this approach). If multiple local proxies are discovered attempt communications with each before widening the search to other options. The proxy relays information to the registrar. If this fails:

B. Obtain an IP address using existing methods, such as SLAAC or DHCPv6, and search for a local registrar using DNS service discovery. If this fails:

C. Obtain an IP address (as above), and search for the domain registrar using a pre-defined Factory provided Internet based re-direct service. Various methods could be used, such as DNS or RESTful APIs.

2. Present IEEE 802.1AR credentials to the discovered Registrar (via a Proxy if necessary). Included is a generated nonce that is specific to this attempt.

3. Verify the MASA service generated authorization token as provided by the contacted Registrar. The authorization token contains the valid domain(s) for this device and is signed by the MASA service. The device uses a pre-installed certificate of the MASA service to validate the signature of the MASA. The nonce information previously provided is also checked, if it was not removed by the Registrar.

4. If and only if step three is successful: Join Domain, by accepting the domain specific information from the registrar, and by enrolling a domain certificate from the registrar.

5. The New Entity is now a member of the domain and will only repeat the discovery aspects of bootstrapping if it is returned to factory default settings.

The following sections describe each of these steps in more detail.

4.1.1. Proxy Discovery

Existing protocols provide the appropriate functionality for both discovering the Proxy and facilitating communication through the Proxy:

IEEE 802.1X Where the New Entity can be cast as the "supplicant" and the Proxy is the "authenticator". The bootstrapping protocol messages are encapsulated as EAP methods. The "authenticator"

reencapsulates the EAPOL frames and forwards them to the "Authentication Server", which provides Registrar functionalities.

PANA [RFC5191] [[EDNOTE: TBD]]

ND [RFC2461] / [RFC4861] [[EDNOTE: TBD]] NOTE: Neighbor Discovery protocols do not describe a mechanism for forwarding messages.

Each provides a method for the New Entity to discover and initiate communication with a local neighbor. In each protocol methods are available to support encapsulation of the bootstrapping protocol messages described elsewhere in this document. Other protocols for transporting bootstrapping messages can be added in future references.

All security assocaitions established are between the new device and the Registrar regardless of proxy operations.

If multiple proxies are available the New Entity tries each until a successful bootstrapping occurs. The New Entity may prioritize proxies selection order as appropriate for the anticipated environment.

If Proxy discovery fails the New Entity moves on to discovering a Registrar directly.

4.1.2. Receiving and accepting the Domain Identity

The domain trust anchor is received by the New Entity during the boostrapping protocol exchange.

An enrollment protocol such as EST [RFC7030] details a set of non-autonomic bootstrapping methods such as:

- using the Implicit Trust Anchor database (not an autonomic solution because the URL must be securely distributed),
- engaging a human user to authorize the CA certificate using out-of-band data (not an autonomic solution because the human user is involved),
- using a configured Explicit TA database (not an autonomic solution because the distribution of an explicit TA database is not autonomic),
- and using a Certificate-Less TLS mutual authentication method (not an autonomic solution because the distribution of symmetric key material is not autonomic).

This document describes an additional autonomic method:

MASA authorization token Authorization tokens are obtained by the Registrar from the MASA service and presented to the New Entity for validation.

If the autonomic methods fails the New Entity returns to discovery state and attempts bootstrapping with the next available discovered Registrar.

4.1.3. Enrollment

As the final step of bootstrapping a Registrar helps to issue a domain specific credential to the New Entity. For simplicity in this document, a Registrar primarily facilitates issuing a credential by acting as an RFC5280 Registration Authority for the Domain Certification Authority.

Enrollment proceeds as described in Enrollment over Secure Transport (EST) [RFC7030]. The New Entity contacts the Registrar using EST as indicated:

- The New Entity is authenticated using the IEEE 802.1AR credentials. (EST support for .
- The EST section 4.1.3 CA Certificates Response is verified using the MASA authorization token provided domain identity.

4.1.4. After Enrollment

Functionality to provide generic "configuration" is supported. The parsing of this data and any subsequent use of the data, for example communications with a Network Management System is out of scope but is expected to occur after bootstrapping enrollment is complete.

See Section 4.5.

4.2. Behavior of a proxy

The role of the Proxy is to facilitate communications. The Proxy forwards messages between the New Entity and a Registrar. Where existing protocols, as detailed in Section 4.1.1, already provide this functionality nothing additional is defined.

4.3. Behavior of the Registrar

Once a registrar is established it listens for new entities and determines if they can join the domain. The registrar delivers any necessary authorization information to the new device and facilitates enrollment with the domain PKI.

Registrar behavior is as follows:

4.3.1. Authenticating the Device

The applicable authentication methods detailed in EST [RFC7030] are:

- the use of an IEEE 802.1AR IDevID credential,
- or the use of a secret that is transmitted out of band between the New Entity and the Registrar (this use case is not autonomic).

4.3.2. Accepting the Entity

In a fully automated network all devices must be securely identified.

A Registrar accepts or declines a request to join the domain, based on the authenticated identity presented and other policy defined criteria such as Proxy identity. Automated acceptance criteria include:

- allow any device of a specific type (as determined by the IEEE 802.1AR device identity),
- allow any device from a specific Factory (as determined by the IEEE 802.1AR identity),
- allow a specific device from a Factory (as determined by the IEEE 802.1AR identity)

In all cases a Registrar must use the globally available MASA service to verify that the device's history log does not include unexpected Registrars. Because if a device had previously registered with another domain, the registrar of that domain would show in the log.

If a device is accepted into the domain, it is then invited to request a domain certificate through a certificate enrolment process. The result is a common trust anchor and device certificates for all autonomic devices in a domain. These certificates can subsequently be used to determine the boundaries of the homenet, to authenticate other domain nodes, and to autonomically enable services on the homenet.

For each entity that will be accepted a Registrar maintains the Factory CA identity and the entity's unique identifier. The Factory CA identity could be implemented as the Factory CA root certificate keyIdentifier (the 160-bit SHA-1 hash of the value of the BIT STRING subjectPublicKey). For user interface purposes the keyIdentifier information can be mapped to a colloquial Factory name (Registrars can be shipped with the keyIdentifier of a significant number of third-party manufacturers).

4.3.3. Claiming the new entity

During initial bootstrapping the New Entity provides a nonce specific to the particular bootstrapping attempt. The registrar should include this nonce when claiming the New Entity from the Internet based MASA service. If a nonce is provided by the Registrar, then claims from an unauthenticated Registrar are serviced by the MASA resource.

The Registrar can claim a New Entity that is not online by forming the request using the entities unique identifier but not including a nonce in the claim request. MASA authorization tokens obtained in this way do not have a lifetime and they provide a permanent method for the domain to claim the device. Evidence of such a claim is provided in the audit log entries available to any future Registrar. Such claims reduce the ability for future domains to secure bootstrapping and therefore the Registrar MUST be authenticated by the MASA service.

Claiming an entity establishes an audit log at the MASA server and provides the Registrar with proof, in the form of a MASA authorization token, that the log entry has been inserted. As indicated in Section 4.1.2 a New Entity will only proceed with bootstrapping if a validated MASA authorization token has been recieved. The New Entity therefore enforces that bootstrapping only occurs if the claim has been logged.

4.4. Behavior of the MASA Service

The MASA service is provided by the Factory provider on the global Internet. The URI of this service is well known. The URI should be provided as an IEEE 802.1AR IDevID X.509 extension (a "MASA authorization token Distribution Point" extension).

The MASA service provides the following functionalities to Registrars:

4.4.1. Issue Authorization Token and Log the event

A Registrar POSTs a claim message optionally containing the bootstrap nonce to the MASA server.

If a nonce is provided the MASA service responds to all requests. The MASA service verifies the Registrar is representative of the domain and generates a privacy protected log entry before responding with the authorization token.

If a nonce is not provided then the MASA service MUST authenticate the Registrar as a valid customer. This prevents denial of service attacks. The specific level of authentication provided by the customer is not defined here. An MASA Practice Statement (MPS) similar to the Certification Authority CPS, as defined in RFC5280, is provided by the Factory such that Registrar's can determine the level of trust they have in the Factory.

4.4.2. Retrieve Audit Entries from Log

When determining if a New Entity should be accepted into a domain the Registrar retrieves a copy of the audit log from the MASA service. This contains a list of privacy protected domain identities that have previously claimed the device. Included in the list is an indication of the time the entry was made and if the nonce was included.

4.5. Leveraging the new key infrastructure / next steps

As the devices have a common trust anchor, device identity can be securely established, making it possible to automatically deploy services across the domain in a secure manner.

Examples of services:

- Device management.
- Routing authentication.
- Service discovery.

4.5.1. Network boundaries

When a device has joined the domain, it can validate the domain membership of other devices. This makes it possible to create trust boundaries where domain members have higher level of trusted than external devices. Using the autonomic User Interface, specific devices can be grouped into to sub domains and specific trust levels can be implemented between those.

5. Protocol Details

For simplicity the bootstrapping protocol is described as extensions to EST [RFC7030].

EST provides a bootstrapping mechanism for new entities that are configured with the URI of the EST server such that the Implicit TA database can be used to authenticate the EST server. Alternatively EST clients can "engage a human user to authorize the CA certificate using out-of-band data such as a CA certificate". EST does not provide a completely automated method of bootstrapping the PKI as both of these methods require some user input (either of the URI or authorizing the CA certificate).

This section details additional EST functionality that support automated bootstrapping of the public key infrastructure. These additions provide for fully automated bootstrapping. These additions are to be optionally supported by the EST server within the same .well-known URI tree as the existing EST URIs.

The "New Entity" is the EST client and the "Registrar" is the EST server.

The extensions for the client are as follows:

- The New Entity provisionally accept the EST server certificate during the TLS handshake as detailed in EST section 4.1.1 ("Bootstrap Distribution of CA Certificates").

- The New Entity request and validates a "bootstrap token" as described below. At this point the New Entity has sufficient information to validate domain credentials.

- The New Entity calls the EST defined /cacerts method to obtain the current CA certificate. These are validated using the "bootstrap token".

- The New Entity completes bootstrapping as detailed in EST section 4.1.1.

These extensions could be implemented as an independent protocol from EST but since the overlap with basic enrollment is extensive, particularly with respect to client authorization, they are presented here as additions to EST.

In order to obtain a validated bootstrap token and history logs the Registrar contacts the MASA service Service using REST calls.

5.1. EAP-EST

In order to support Proxy environments EAP-EST is defined.

[[EDNOTE: TBD. EST is TLS with some data. EAP-TLS and other similar protocols provide an example framework for filling out this section]]

5.2. Request bootstrap token

When the New Entity reaches the EST section 4.1.1 "Bootstrap Distribution of CA Certificates" state but wishes to proceed in a fully automated fashion it makes a request for a MASA authorization token from the Registrar.

This is done with an HTTPS POST using the operation path value of "/requestbootstraptoken".

The request format is JSON object containing a nonce.

Request media type: application/masanonce

Request format: a json file with the following:

{"nonce":"<64bit nonce value>"}

[[EDNOTE: exact format TBD. There is an advantage to having the client sign the nonce (similar to a PKI Certification Signing Request) since this allows the MASA service to confirm the actual device identity. It is not clear that there is a security benefit from this.]]

The Registrar validates the client identity as described in EST [RFC7030] section 3.3.2. The registrar performs authorization as detailed in Section 4.3.2. If authorization is successful the Registrar obtains a MASA authorization token from the MASA service (see Section 5.3).

The recieved MASA authorization token is returned to the New Entity.

5.3. Request MASA authorization token

A registrar requests the MASA authorization token from the MASA service using a REST interface.

This is done with an HTTP POST using the operation path value of "/requestMASAauthorization".

The request format is a JSON object optionally containing the nonce value (as obtained from the bootstrap request) and the IEEE 802.1AR identity of the device as a serial number (the full certificate is not needed and no proof-of-possession information for the device identity is included). The New Entity's serial number is extracted from the subject name :

{"nonce":"<64bit nonce value>", "serialnumber", "<subjectname/ subjectaltname serial number>"}

Inclusion of the nonce is optional because the Registar might request an authorization token when the New Entity is not online, or when the target bootstrapping environment is not on the same network as the MASA server.

This information is encapsulated in a PKCS7 signed data structure that is signed by the Registrar. The entire certificate chain, up to and including the Domain CA, is included in the PKCS7.

The MASA service checks the internal consistency of the PKCS7 but is unable to actually authenticate the domain identity information. The domain is not know to the MASA server in advance and a shared trust anchor is not implied. The MASA server verifies that the PKCS7 is signed by a Registrar (by checking for the cmc-idRA field in the Registrar certificate) certificate that was issued by the root certificate included in the PKCS7.

The domain ID is extracted from the root certificate and is used to generate the MASA authorization token and to update the audit log.

[[EDNOTE: This assumes the Registrar can extract the serial number successfullly from the cilent certificate. The RFC4108 hardwareModuleName is likely the best known location.]]

5.4. Request MASA authorization log

A registrar requests the MASA authorization log from the MASA service using this EST extension.

This is done with an HTTP GET using the operation path value of "/requestMASAlog".

The log data returned is a file consisting of all previous log entries. For example:

```
"log":[
  {"date":"<date/time of the entry>"},
   "domainID":"<domainID as extracted from the root
               certificate within the PKCS7 of the
               authorization token request>",
   "nonce":"<any nonce if supplied (or NULL)>"},

  {"date":"<date/time of the entry>"},
   "domainID":"<domainID as extracted from the root
               certificate within the PKCS7 of the
               authorization token request>",
   "nonce":"<any nonce if supplied (or NULL)>"},
]
```

Distribution of a large log is less than ideal. This structure can be optimized as follows: only the most recent nonce'd log entry is required in the response. All nonce-less entries for the same domainID can be condensed into the single most recent nonceless entry.

The Registrar uses this log information to make an informed decision regarding the continued bootstrapping of the New Entity.

[[EDNOTE: certificate transparency might offer an alternative log entry method]]

6. Reduced security operational modes

A common requirement of bootstrapping infrastructures is often that they support less secure operational modes. To support these operational modes the Registrar can choose to accept devices using less secure methods. For example:

1. The registrar may choose to accept all devices, or all devices of a particular type, at the administrator's discretion. This may occur when: Informing the Registrar of unique identifiers of new entities might be operationally difficult.

2. The registrar may choose to accept devices that claim a unique identity without the benefit of authenticating that claimed identity. This may occur when: The New Entity does not include an IEEE 802.1AR factory installed credential.

3. A representative of the Registrar (e.g. the Orchestrator) may request nonce-less authorization tokens from the MASA service when network connectivity is available. These tokens can then be transmitted to the Registrar and stored until they are needed during bootstrapping operations. This may occur when: The target

network is protected by an air gap and therefore can not contact the MASA service during New Entity deployment.

4. The device may have an operational mode where it skips authorization token validation. For example if a physical button is depressed during the bootstrapping operation. This may occur when: A device Factory goes out of business or otherwise fails to provide a reliable MASA service.

5. The device may not require the MASA service authorization token. An entity that does not validate the domain identity is inherently dangerous as it may contain malware. This risk should be mitigated using attestation and measurement technologies. In order to support an unsecured imprint the New Entity MUST support remote attestation technologies such as is defined by the Trusted Computing Group. [[EDNOTE: How to include remote attestation into the boostrapping protocol exchange is TBD]]. This may occur when: The device Factory does not provide a MASA service.

7. Security Considerations

In order to support a variety of use cases, devices can be claimed by a registrar without proving possession of the device in question. This would result in a nonceless, and thus always valid, claim. The MASA service is required to authenticate such Registrars but no programmatic method is provided to ensure good behavior by the MASA service. Nonceless entries into the audit log therefore permanently reduce the value of a device because future Registrars, during future bootstrap attempts, must now be configured with policy to ignore previously (and potentially unknown) domains.

Future registrars are recommended to take the audit history of a device into account when deciding to join such devices into their network.

It is possible for an attacker to send an authorization request to the MASA service directly after the real Registrar obtains an authorization log. If the attacker could also force the bootstrapping protocol to reset there is a theoretical opportunity for the attacker to use the authorization token to take control of the New Entity but then proceed to enrol with the target domain. To prevent this the MASA service is rate limited to only generate authorization tokens at a rate of 1 per minute. The Registrar therefore has at least 1 minute to get the response back to the New Entity. [[EDNOTE: a better solution can likely be found. This text captures the issue for now.]] Also the Registrar can double check the log information after enrolling the New Entity.

The MASA service could lock a claim and refuse to issue a new token. Or the MASA service could go offline (for example if a vendor went out of business). This functionality provides benefits such as theft resistance, but it also implies an operational risk. This can be mitigated by Registrars that request nonce-less authorization tokens.

7.1. Trust Model

[[EDNOTE: (need to describe that we need to trust the device h/w. To be completed.)]]

8. Acknowledgements

We would like to thank the various reviewers for their input, in particular Markus Stenberg, Michael Richardson, Brian Carpenter, Fuyu Eleven.

9. References

9.1. Normative References

[IDevID] IEEE Standard, , "IEEE 802.1AR Secure Device Identifier", December 2009, <http://standards.ieee.org/findstds/standard/802.1AR-2009.html>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC7030] Pritikin, M., Yee, P., and D. Harkins, "Enrollment over Secure Transport", RFC 7030, October 2013.

9.2. Informative References

[I-D.behringer-homenet-trust-bootstrap]
Behringer, M., Pritikin, M., and S. Bjarnason, "Bootstrapping Trust on a Homenet", draft-behringer-homenet-trust-bootstrap-02 (work in progress), February 2014.

[I-D.irtf-nmrg-autonomic-network-definitions]
Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking - Definitions and Design Goals", draft-irtf-nmrg-autonomic-network-definitions-05 (work in progress), December 2014.

Authors' Addresses

Max Pritikin
Cisco

Email: pritikin@cisco.com

Michael H. Behringer
Cisco

Email: mbehring@cisco.com

Steinthor Bjarnason
Cisco

Email: sbjarnas@cisco.com