Network Working Group
Internet-Draft
Intended status: Experimental
Expires: April 16, 2013

C. Jennings
Cisco

October 13, 2012

Transitive Trust Enrollment for Constrained Devices
draft-jennings-core-transitive-trust-enrollment-01

Abstract

This document provides a sketch of a rendezvous protocol that allows constrained internet devices such as sensors to securely connect into a system that uses them. The solution is based on the idea that each device will be manufactured with a one time password that can be used by the customer to tell the device which controller to enroll with, and the device will be manufactured to contact a given Transfer Server that is used to tell the device which system to connect to. The administrator of the device will be able to get this one time password from the device using a QR code, and then will be able to use that one time password to inform a Transfer Server which system the device should connect to. The device will contact the Transfer Agent, get this information, and then connect to the appropriate system.

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 April 16, 2013.

Table of Contents

1. Introduction

Secure enrollment of devices into internet-based systems has never been easy. The constrained devices that need to be enrolled into systems today face many challenges. Typically, simple devices such as keyboards and screens have no user interface; they may have only a single button or LED. At the time they are installed, there may not be a working network or even power. However, these devices are being used for applications that are increasingly important and safety-critical, so they need to have reasonable security and privacy characteristics. This documents specifies an enrollment system for such devices.

In many systems, there is a need to configure a Device, such as a sensor or actuator, so that it is controlled by some specific Controller. With Devices like a switch and light, it may be that all the Controller does is later configure the switch to control the light. To make this happen, both Devices need to be under the control of a common Controller that is authorized to make changes to the Devices.

The simplified high-level information flow is illustrated in the following figure. The goal is to get to the point where the Device knows that it should be talking to the Controller.

TODO - See PDF Version of draft

When the Manufacturer builds the Device, it includes a One Time Password (OTP) that the Introducer can use to enroll the Device with the Controller. The Manufacturer also runs a website known as the Transfer Agent that knows the OTP for every device that uses that Transfer Agent. The Device can include the OTP as a QR code on the outside of the Device. When the Device is installed, the network administrator or installer uses a software agent known as the Introducer. The Introducer would typically be simply a normal browser running on a smart phone with a camera that can read QR codes. When the Device is installed, the Introducer can scan the QR code on the Device. This QR code includes a URL to the Transfer Agent along with the OTP and a separate Device secret DevSecret. (Message 1). The Introducer then contacts the Transfer Agent and uses the OTP to tell the Transfer Agent which Controller this Device should use (Message 2). The Introducer can also tell the Controller the DevSecret (Message 3) so that the Controller and Device can authenticate each other. Later, the first time the Device boots up and gets network connectivity, it contacts the Transfer Agent, and the Transfer Agent tells the Device which Controller to talk to (Message 4). From that point on, any time the Device boots, the Device can communicate directly with the Controller (Message 5). The

actual message flow is slightly more complicated and shown in Section 2, but it uses the same basic idea as this simplified flow.

The system is designed to achieve several desirable properties:

- Can work for Devices with very limited memory and processing power.
- Does not require network or power to be available when the Device is installed.
- Is fairly secure (see more in the security section).
- Minimal addition to manufacturing costs.
- The installer can detect if the OTP has already been used.
- Provides a work flow in which a Device does not need to be taken out of the box to be enrolled. This can be very important to enable consumers themselves to enroll devices they buy from a service provider.
- Works with common firewall and NAT network topologies.

One of the key steps in making this system work is getting the OTP from the Device to the Introducer. The approach used here is to use a QR code representing the URL. The QR code is printed on the Device and/or box it comes in.

The Device uses HTTP or COAP [I-D.ietf-core-coap] to communicate with the Controller. The Transfer Agent and Introducer use HTTPS to communicate with each other. There are three pieces of keying material used for cryptographic operations. The first is the One Time Password (OTP) that is passed via a QR code from the device to the Introducer and that the Introducer then uses to authorize itself to the Transfer Agent. There is also a DevSecret that is used to secure communications between the Device and the Controller. Finally there is a TaSecret that is used to secure communications between the Device and the TransferAgent. The Transfer Agent needs a normal certificate to use HTTPS.

It is assumed that the Device may have a NAT between it and the Controller and that the Device is on the inside of the NAT. The Transfer Agent is assumed to be a generally accessible server on the internet, but the Controller and Device can be on the inside of a firewall or NAT between them and the Transfer Agent.

The semantic level information in each message is discussed in Section 2 and the syntax of the messages is discussed in Section 5 and Section 6. The security properties of the system are described in Section 7.

2. Enrollment Information Flow

In the following message flow we use the following definitions:

TaURL An http URL that can be used to reach the root resource on the Transfer Agent.

DevURN A globally unique URN assigned by the Manufacturer to uniquely identify this Device.

OTP The One Time Password created by the Manufacturer for enrolling the Device.

TaSecret The secret created by the Manufacturer for the Device to communicate with the Transfer Agent.

DevSecret The secret created by the Manufacturer for the device to communicate with the Controller.

ContURL This is a URL that provides the address to reach the controller. It can have a scheme of http, https, coap, or coaps.

DevLabel A locally significant string that the Introducer can assign to a Device. For example, the convention for a thermostat in building 30, floor 2, office 361 might be assign the string "BLD30/2/361 - Thermostat". This string is provided purely as a way to let the Introducer and Controller exchange information that may be useful for the user installing the system.

The information flow is illustrated in the following figure. The goal is get to the point where the Device knows that it should be talking to the Controller, the Controller knows it should be talking the Device, and the Device and Controller can communicate and authenticate each other using the DevSecret.

TODO - See PDF Version of draft

When the Manufacturer builds the Device, it includes a TaSecret on the Device, a DevSecret, and the URN for the Device (DevURN). It also creates a QR code on the Device that contains the URL to the transfer agent (TaURL), the URN for the Device (DevURL), the OTP, and the DevSecret. This QR codes is described in detail in section TODO. The Manufacturer also tells the Transfer Agent the OTP, TaSecret and DevURN for this Device.

When the Device is installed, the Introducer reads OTP, DevSecret, DeviceURN, and the URL for the Transfer Agent (TaURL) from the Device by scanning the QR code on the device (Message 1). If the Introducer is a web browser, it uses the Transfer Agent URL to fetch an HTML user interface to perform the next steps (Message 2). The user interface on the Introducer allows the user to user to enter the URL for the Controller (ContURL) and an optional label for the device (DevLabel).

Next the controller tells the Transfer Agent the Controller URL to

use for this DeviceURN. This request is authenticated by the Transfer Agent using the OTP (Message 3). Open Issue: right now the OTP is transfered in the request (which is over HTTPS). A better design might be to have the Introducer prove procession of the OTP to the Transfer Agent and not send the OTP over the wire.

The Introducer also tells the controller the DevSecret for this Device and the optional DevLabel (message 4).

Later the Device contacts the Transfer Agent and the Transfer Agent tells the Device the URL of the Controller to talk to (ContURL) in message 5 and 6). The information from the Transfer Agent to Device is encrypted and signed with the TaSecret.

From that point on, any time the Device boots, it can directly communicate with the Controller (Messages 7 and 8). The Controller and Device both know the DevSecret for the Device and can use that to authenticate and encrypt communications between them. It is suggested that the first thing the Controller and Device should do is to use this DevSecret to securely replace it with some different secret that is not known to anyone that saw the QR code.

Open Issue: should we add in an additional ContSecret that is picked by the Controller, passed to Introducer, then passed to the Trust Agent, and finally passed to Device?

3. Terminology

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

When this draft says Base64, it means the URL safe Base64 encoding from TODO.

4. QR Code

The QR code for the Device MUST be an HTTPS URL that points at the appropriate Transfer Agent. The authority MUST be formed by using the authority from the TaURL. The path is formed by concatenating ".well-known/tte1/" followed the DevURN followed by "/s". The DevURN SHOULD be one of the URNs defined in [I-D.arkko-core-dev-urn]. It MUST include the OTP as a Base64 encoded value for a parameter called otp. The secret MUST be encoded in Base64 and used as the fragment identifier of the URL. The secret is put as a fragment so that if the Introducer scans the QR code and dereferences the URL with a web

browser, the fragment identifier will not be transferred in the request to the Transfer Agent.

As an example, if the Transfer Agent's domain is example.net, a valid URL for the QR code would be:

TOOD - change hex to base64
https://example.net/.well-known/tte1/urn:dev:mac:90a2da001a0c/s
?otp=88F5EC91493E473B758159C7792C#00004DCFDCDBD9F54C1B2E71FC22

The QR code SHOULD use an error coding level of "H". This would generate the following QR code:

QR code in ASCII art left as an exercise to
the reader but there is one in the PDF version.

5. Transfer Agent API

Note that future version of the API that needed to increment a version number would do it by changing the tte1 to tte2.

TODO - still need to define all the error responses but basic approach will be a simple JSON object with the error responses.

5.1. Create

The HTTP REST API allows the manufacturer to tell the Transfer Device about the DevURN and OTP for a given device the Manufacturer has created.

Path: .well-known/tte1/d/{DevURN}
Methods: POST
Parameters:
otp: Base64 encoded version of the OTP
Return Values: TODO

This creates an entry for the device in the database and stores the OTP associated with this Device. The Transfer Agent SHOULD authenticate this request to authorize it. Note the "d" in the path is short for "device"; having this path segment allows for future extensibility.

Example Request:

POST https://example.net/.well-known/tte1/d/urn:dev:mac:90a2da001a0c
?otp=88F5EC91493E473B758159C7792C

Example Response:

TODO

5.2. Setup

The Transfer Agent MUST return a web page that allows the user to provide the information needed for the bind, and then the Introducer must call the bind and add methods.

Path: .well-known/tte1/d/{DevURN}/s
Methods: GET
Parameters:
otp: Base64 encoded version of the OTP
Return Values: HTML5 web page

This MUST return a HTML web page that has a suitable user interface to allow the user to enter the address of the the Controller. It is suggested that the page validate that this controller entry is correct using the "alive" API in Section TODO. Once the Controller is verified, the web page MUST tell the Transfer Agent the Controller address using the "bind" API in Section TODO. The page MUST tell the controller the DevURN and DevSecret for the Device using the "add" API in Section TODO. MUST be done over HTTPS.

Example Request:

```
GET https://example.net/.well-known/tte1/d/urn:dev:mac:90a2da001a0c/s
          ?otp=88F5EC91493E473B758159C7792C
```

5.3. Bind

TODO MUST be sent over TLS, and the Introducer MUST verify that the HTTPS certificate of the Transfer Agent matches the URL.

Once the Transfer Agent has successfully stored the Controller's address for a given OTP, it MUST NOT allow that OTP to be used again to store an address for that Device.

Path: .well-known/tte1/d/{DevURN}/c
Methods: PUT
Parameters:
otp: Base64 encoded version of the OTP
conturl: URL to controller escaped as necessary for placement in a URL

Return Values: TODO

This request using the

Example Request:

TODO
PUT https://example.net/.well-known/tte1/d/urn:dev:mac:90a2da001a0c/c
?otp=88F5EC91493E473B758159C7792C

5.4. Fetch

This API allows a Device to get the information about the controller it should connect to. It is provided in a JSON object which is encrypted using the OTP.

Path: .well-known/tte1/{DevURN}/c
Methods: GET
Parameters: None
Success Return Values: JSON object as defined in TODO that contains the encrypted URL to the Controller.
Error Return Values: Returns HTTP 404 if the DevID can not be found or if the controller URL has not yet been set for this DevURN.

The Transfer Agent looks up the OTP and ContURL for the requested DevURN. If the DevURN cannot be found, or the ContURL has not yet been set for this DevURN, then the Transfer Agent returns an HTTP 404 error. If they are found, it then the Authenticated Encryption with Associated Data (AEAD) algorithm described in Appendix A (TODO ref) to form the JSON object that is returned. The TaSecret is used as the key for the AEAD, the ContURL is the input data to be encrypted, and the DevURN is used as Associated Data for the authentication.

Example Request:

GET https://example.net/.well-known/tte1/d/urn:dev:mac:90a2da001a0c/c

Example Response:

TODO

6. Controller API

The Alive and Add API need to include a CORS (TODO REF) header to allow AJAX from the Transfer Agent to call these APIs. They MUST include an HTTP header in the response that sets the header field Access-Control-Allow-Origin to a value of "*". TODO security section

needs to discuss implications.

6.1. Test Alive

This method allows the Introducer to validate that a valid Controller address has been entered. It simply returns an HTTP 200 if the controller is operational.

Path: .well-known/tte1/alive
Methods: GET
Parameters: None
Return Values: TODO

6.2. Add

This method is used by the Introducer to add a new Device to the Controller. (Open issues - should it result in redirect to a controller web page to configure device?)

Path: .well-known/tte1/c/{DevURN}/k
Methods: PUT
Parameters:
devSecret: Base64 encoded version of the secret
devLabel:
Return Values: TODO

6.3. Sensor Update

TODO

Path: .well-known/tte1/s/{DevURN}/v
Methods: PUT
Parameters: None
Body: Encrypted SENML
Return Values: TODO

The body is a Encrypted JOSE object, as specified in Appendix A (TODO REF). The secret for this Device is used as the key to encrypt the data. The DevURN is used as the associated data. The decrypted data is a SENML sensor reading for this Device as described in [I-D.jennings-senml].

7. Security Considerations

This section has not really been started and needs lots of work.

TODO - Discuss how one can replace a dead Controller with a new one

in an operational network. The short answer is likely that one needs to back up the keys of the old Controller and move these to the new Controller.

What happens if the OTP is stolen during Device transit? The short answer is that the Device is compromised at this point and needs to be discarded or returned to the manufacturer to get a new OTP. The Introducer needs to detect that this has happened and warn the user.

There are additional concerns about Devices that may be operational without ever being introduced to a Controller. For example, if a light switch supported this protocol but could also be used just as a stand alone light switch, there would be a risk that the OTP could be stolen by an attacker, with the attacker enrolling the Device to the attacker's Controller. If the correct user installed the light switch but did not Introduce it to anything, the fact it had been compromised would go undetected. One way to mitigate this risk might be to include some manual configuration on the Device to indicate that it is to be used in stand-alone mode, such as a jumper that can be cut.

Network topology consideration - Introducer can install firewall rules that allow Devices to contact Transfer Agent.

Explain why works with NATs / FWs.

8. Variations

8.1. LED Based Enrollment

An alternative to QR codes is to have an LED on the Device flash out the relevant information to the Introducer. The output string is formed by concatenating a 16-bit start of message constant value of 0x0001, followed by the 8 bit length of TaURN, TaURN, 8 bit length of DevURN, the DevURN, 8 bit length of OTP, OTP, 8 bit length of DevSect, DevSecret and then an 8-bit two's compliment checksum value computed over the previous bytes, including the start of message constant. All values are in network byte order. The resulting string is output using Non-Return-to-Zero Inverted (NRZI) encoding on the LED at a baud rate of 15 bps. This allows a Device such as a smartphone with video capture to detect the signal and recover the information.

TODO - see if this works at 30 bps. See about encoding multiple intensity levels or colors in the LED. Initial experiments indicate this does not work very well, as auto contrast in the video camera tends to saturate LED range. Would an Adler-32 checksum be better?

Could multiple colors of intensity improve the speed of this as this is very slow.

8.2. Bulk Enrollment

Imagine one wants to enroll a whole box of sensors. We should define some scheme where one could simply bar code something on the outside of a box so that all the sensors in the box could be bulk enrolled. Perhaps there could be a master secret and start and end DevURN on the outside of the box bar code. Then the OTP for a given Device would be generated using the master secret and DevURN of that Device. Work is needed to sort out details of a scheme like this.

8.3. Public Key Crypto

This specification assumes that COAP is being used with DTLS in Pre Shared Key (PSK) mode. It would also be possible to use DTLS with self signed certificates with a very similar flow, where the Introducer provided the Transfer Agent with the fingerprint of the certificate or public key of the Controller.

9. Implementation Notes

The system described here can be implemented on a very small device. An implementations for Arduino with ethernet was done that includes all the parts described here, including SENML, JSON, the encryption and signing, HTTP, DNS, and DHCP. It also included libraries for reading a 1-wire temperature sensor. This fits in under 32k of flash, and uses less than 4k of ram on an 8 bit AVR processor. That means that the cost for an embedded processor in volume with this much flash, ram, etc. is very roughly $1.50 USD in 2012. A key part of getting this to be small is the extremely small crypto footprint from using SHA224-CFB.

9.1. Random Numbers

Note: This section would be better in a separate draft.

TODO - Explain how to use SHA224_DRBG as defined in NIST SP800-90A. TODO REF. Store reseed counter in eeprom every 24 hours. Explain what to store in eeprom on reseed. TODO REF RFC 4086 and XKCD 221.

Todo - Discuss sources of randomness in use: 16 bytes of random data created during manufacturing. A 32 bit boot counter that increments every time the device boots. 8 byte pool of random data from sensor readings. 8 byte pool of random data from timing of receiving or sending network packets. A 32 bit counter that increments each time

a random number is generated but resets to 0 on reboot.

10. Open Issues

The references section is in serious need of work - let me know stuff that should be added to it.

Does QR encoding of L work out better than H?

Is there any advantage in having the HTTP URL in well-known space?

Is there some clever way (perhaps zeroconf) for the Introducer to discover the ContURL?

11. IANA Considerations

TODO register .well-known/tte1

12. Acknowledgments

Some of the fundamental ideas in this draft were inspired by Max Pritikin's work on Transitive Trust Introduction. Randy Bush provided crisp and excellent advice on what the security properties of the solutions should be. I'd like to thank the following people for review comments: Eric Rescorla, Jari Arkko, Lyndsay Campbell, and Zach Shelby.

13. Appendix A: JOSE SHA224-CFB

Note: This section will eventually be moved to an experimental draft submitted to JOSE WG.

This section describes how to create a JOSE object as described by [I-D.barnes-jose-jsms] that is encrypted and signed with SHA224-CFB as specified in [HashCFB].

This adds a new ENCRYPTION algorithm called sha224-cfb to [I-D.barnes-jose-jsms]. This takes one parameter named "n" which represents the nonce as defined in [HashCFB]. It is RECOMMENDED that the key be 14 bytes long and that the nonce be 24 bytes long. The authentication information from the algorithm is stored in the "mac" field.

13.1. Example

TODO example. Todo fix to base64 instead of hex encoding. TOOD talk to Barnes about keyID and case with no key wrap. TODO - state of sha224-cfb and this is all experimental.

```
Input Key (Hex) = 88F5EC91493E473B758159C7792C
Input Associated Data = "urn:dev:mac:90a2da001a0c"
Input plain text = "http://example.com" - TODO

Result  =
{
  TODO
}
```

14. References

14.1. Normative References

[HashCFB] Forler, C., McGrew, D., Lucks, S., and J. Wenzel, "Hash-CFB: Authenticated Encryptions without a Block Cipher", Directions in Authenticated Ciphers Workshop , July 2012.

[I-D.barnes-jose-jsms] Barnes, R., "JavaScript Message Security Format", draft-barnes-jose-jsms-00 (work in progress), June 2012.

[I-D.ietf-core-coap] Shelby, Z., Hartke, K., Bormann, C., and B. Frank, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-08 (work in progress), October 2011.

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

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, April 2010.

14.2. Informative References

[I-D.arkko-core-dev-urn]
Arkko, J., Jennings, C., and Z. Shelby, "Uniform Resource Names for Device Identifiers", draft-arkko-core-dev-urn-01 (work in progress), October 2011.

[I-D.jennings-senml]
Jennings, C., Shelby, Z., and J. Arkko, "Media Types for Sensor Markup Language (SENML)", draft-jennings-senml-07 (work in progress), October 2011.

Author's Address

Cullen Jennings
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Phone: +1 408 421-9990
Email: fluffy@iii.ca