Network Working Group
Internet-Draft
Intended status: Informational
Expires: July 29, 2018

M. Richardson
SSW

January 25, 2018

Considerations for stateful vs stateless join router in ANIMA bootstrap

draft-richardson-anima-state-for-joinrouter-02

Abstract

This document explores a number of issues affecting the decision to use a stateful or stateless forwarding mechanism by the join router (aka join assistant) during the bootstrap process for ANIMA.

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 https://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 July 29, 2018.

Table of Contents

1. Introduction

The [I-D.pritikin-anima-bootstrapping-keyinfra] defines a process to securely enroll new devices in an existing network. It order to avoid providing globally reachable addresses to the prospective new network member, it assumes that a Join Router. The role of this router is common in this kind of architecture.

1.1. Terminology

EAP [RFC5247], 802.1X and PANA [RFC5191] use the term Authenticator to refer this role.

The Thread architecture [threadcommish] uses the term Joiner Router

The 6tisch architecture ([I-D.ietf-6tisch-terminology]) uses the term JA, short for Join Assistant.

2. Purpose of the Joiner Router/Join Assistant

This device is one layer-2 hop from the new device. In addition to whatever secured networks it might connect to, it runs a sufficiently unprotected network (either physical or wireless) such that a new device can connect at layer-2 without any specific credentials.

The new node runs a discovery protocol as explained in [I-D.pritikin-anima-bootstrapping-keyinfra] to find an address for a registrar to which it can run the Enrollment over Secure Transport (EST, [RFC7030]. EST runs RESTfully over protocols such as HTTP.

The new node does not have a globally routable address, so it can not speak directly outside the current link. This an intentional limitation so that the new node can neither be easily attacked from the general internet, nor can it attack arbitrary parts of the Internet.

The Joiner Router provides a limited channel between the new node, and the Registrar. This document is about the various options and considerations that need to be considered when chosing this limited channel.

An additional goal of this document is to outline which methods could be interchangeably be used by private negotiation between the Joining Router and the Registar, without the knowledge of the New Node.

3. Overview of suggested methods

3.1. method 1: Circuit Proxy method

In response to discovery, the circuit proxy would return a link-local address on the joining router. The joining router would have a TCP (or UDP/CoAP) port open on that interface. It would accept connections on that port, and would turn around and create a new TCP connection to the registrar.

While non-blocking I/O and threading mechanisms permit a single process to handle dozens to thousands of such connections, in effect a new circuit is created for each connection. As a new TCP connection is created to the registrar it might have a different address family (IPv4 vs IPv6), and it might have a different set of TCP options, MSS and windowing properties.

3.2. method 2: NAPT66 method

In response to discovery, the NAT66 would return a link-local address on the joining router. The joining router would establish a NAPT66 mapping between the address/port combination on the join side, with an address/port on the ACP side. The port would be randomly allocated.

The join router would then do a stateful mapping between the pair of link-local addresses and ports, and the ACP GUA and registrar addresses and ports. This method is mostly identical to what is

sometimes called a "port forward"; but is used from the inside to the outside, rather than the converse.

3.3. method 3: HTTP Proxy method

In response to discovery, the proxy would reply with a link-local address and port combination, and possibly also a URL for the registrar.

The new node would then establish an HTTP connection to the proxy, and would use the HTTP CONNECT method with the given URL to establish a connection to the proper registrar. See [RFC7231] section-4.3.6.

Potentially a new node might attempt to other resources than the intended registar. This could be a permitted activity if the connection is to the new node's vendor MASA, but it will in general be difficult to know what URLs are expected, and which are not.

The HTTP proxy would put the normal HTTP proxy headers in, such as the VIA header, which may well help the registrar determine where the New Node has joined.

3.4. method 4: CoAP/DTLS with relay mechanism

In reponse to discovery, the proxy would respond with a link-local address and port combination.

The new node would then initiate a DTLS session over UDP for the purpose of running CoAP on top of it. See [RFC7252] section 9.1.

The Join Router would then use a mechanism such as envisioned by [I-D.kumar-dice-dtls-relay] to mark the real origin of the packets. (Note that this ID did not get to the point of actually specifying the bytes on the wire). Alternatively, the [threadcommish] specifies a way to encapsulate DTLS (that would contain CoAP packets) packets into CoAP, along with a clear origin for the packets.

3.5. method 5: HTTP with IPIP tunnel

In reponse to discovery, the proxy would respond with a link-local address and port combination. The new node would then initiate a regular HTTPS session with the given address and port as in methods 1 and 2.

Rather than create a circuit proxy or NAT66 mapping, the joining router would instead encapsulate the packet in an IPIP header and send it to the registrar.

The registrar (or a device with the registrar's IP in front of it) must then implement the IPIP decapsulation, along with some way to accept the connection to the link-local address of the Joining Router, and route packets back again. The technology to do this is either one of NAT66, or the typical "transparent" application layer proxy technology of the mid-1990s. See [transparentproxy] for a description in an expired patent. The mechanism is simply to #if 0 out the "is dest-IP local" test. This is also supported by as transparent proxying in linux and squid, see [transparentsquid], and is also available on BSD systems' pf and ipf. Also see: [RFC1919]

An issue that arises in IPv6 with link-local addresses is if the joining router has more than non-loopback interface. On such a system, link-local addresses must be qualified by the interface identifier, usually represented as the SMI if_index to software. This is a serious concern, as even on IoT-type/mesh devices where there is only a single radio, there will in general be two logical networks: one secured as part of the production network, and a second one for joining nodes. Alternatives to IPIP encapsulation have so-far been motivated by the need to store this additional context.

A solution to this problem is to simply have the joining router send the IPIP traffic from an IPv6 address that is unique to the interface on which the traffic originates. That is, even if the join network will use link-local addresses, the joining router should allocate additional stable private addresses (via SLACC + [RFC7217] for each interface on which it runs the join protocol. The number of these addresses scales with the number of logical interfaces, not the number of clients that are joining>

3.6. method 6: CoAP/DTLS with IPIP tunnel

In reponse to discovery, the proxy would respond with a link-local address and port combination. The new node would then initiate a regular CoAP/DTLS session with the given address and port as in method 4.

Identically to method 5, the joining router would encapsulate the packet in an IPIP header and send it to the registrar.

This method is otherwise identical to method 4 and method 5.

4. Comparison of methods

The Circuit Proxy and NAT66 methods are mostly indistinguishable from an outside observer. Careful probing with exotic TCP options, or strange MSS values would reveal which is used, but this will otherwise be invisible to a new node.

Method 3 (http-proxy) and methods 1 (circuit), 2(nat66), and 5(ipip) could be made indistinguishable to the new node if methods 1,2, and 5 also included the URL, and instead of running TLS immediately, always used the CONNECT method first. That is, the registar would accept to "proxy" to itself.

While it is possible to proxy between HTTP and CoAP forms in a mechanical fashion, it is not possible to map between DTLS and TLS mechanisms without access to the private keys of both ends. Therefore it is not possible to accept DTLS/CoAP packets on the Joining Router and turn them into an HTTPS session to a registrar that accepts only HTTPS. It is reasonable for a registrar to speak both CoAP and HTTP: this could be done inside the server itself, or could be part of an HTTPS/DTLS front end that normalized both protocols into HTTP. There are channel binding issues that must be addressed within the registrar, but they are well understood in the multi-tier web framework industry.

4.1. State required on Joining Router

Methods 1(circuit), 2(nat66), and 3(proxy) require state on the joining router for each client. Method 3(proxy) will tend to require the most processing and state as it requires re-assembly of TCP packets sufficient to interpret HTTP and perform the CONNECT operation. Methods 3 and 1 both require two TCP socket structures, which are on the order of hundred bytes each.

Method 2(nat66) can require as little as space for 4 IPv6 addresses, plus two TCP port numbers, a total of 68 bytes per client system. Usually there will be some index or hash overhead. Many devices may be able to do this operation for a data-plane (production) network interface at wire speed using a hardware CAM. Joiner Router functionality may not always be able to make use of hardware, as being part of the ACP, it may be implemented entirely in the control plane CPU.

Method 4 (dtls-relay), 5(ipip-http) and 6(ipip-coap) do not require any additional per-client state to be maintained by the joining router.

4.2. Bandwidth required on Joining Router

All the IPIP methods have an additional header cost of 40 bytes for an IPv6 header between the Joining Router and the Registrar.

The DTLS relay method (whether inside DTLS or via CoAP extension), has the cost of an additional CoAP header or DTLS extension, estimated to be around 16 bytes.

The TLS or DTLS headers pass between the New Node and the Registrar in all cases. The DTLS header is bigger than the TLS header, but this is slightly compensated by the UDP vs TCP header cost of 8 vs 20 bytes. The DTLS header is providing much of what the TCP header was providing.

The HTTP proxy mechanism has an initial packet cost to send the CONNECT header.

In Autonomic networks the backhaul from Joining Router to Registrar will be over the ACP. The ACP is not generally as well provisioned as the production data-plane network, but in non-constrained (see [RFC7228] section 2.2 and 2.3) situations, it would be IPv6 tunneled over IPsec across well-provisioned ethernet. The ACP likely capable of at least 1Mb/s of traffic without significant issues.

4.2.1. Bandwidth considerations in constrained networks

In constrained-network situations, there are two situations to examine. The first scenario is where the Joining Router has an interface on a constrained-network, and a backhaul on a non-constrained network. For instance, when the Joining Router is the 6LBR in a mesh-under situation, or is at the top of the DODAG in a route-over situation. In that situation, there are no significant constrained for the cost of backhauled packets, all constrained are on the join network side.

The second scenario is where in the route-over network where the Joining Router is a 6LR within the mesh. In the situation the backhaul network path travels through one or more hops of a LLN, and packet size as well as throughput is constrained.

Note that nothing in the discussion in this section is concerned with the capablities of the Joining Router: the device could well be powered and very capable, but currently not connected by any data-plane networks. For instance two physically adjacent HFRs might use Bluetooth or an in-chassis 802.15.4 sensor network (originally intended to collect temperature readings) to communicate in order to agree on an appropriate lambda for a 100G/bs fiber link.

There are current efforts for optimizing ROLL route-over networks to compress the overhead of IPIP headers out. This is the "Example of Flow from not-RPL-aware-leaf to Internet" in section 5.7 of [I-D.robles-roll-useofrplinfo] and which [I-D.ietf-6lo-paging-dispatch] aims to compress.

4.3. State required on Registrar

All methods require that the registrar maintain an HTTP or CoAP connection with the New Node for duration of each request. HTTP/1.1 clients may use persistent connections if there are multiple request/responses.

CoAP clients are inherently single-request/responses, but it is anticipated that CoAP Block-Transfer Mode [I-D.ietf-core-block]would be required by EST ([RFC7030]) to transfer the certificates and certificate chains, which are likely to be larger than a single UDP packet. The block-transfer mode is designed to be stateless for the server. It could be made more stateless if a 201 Location: header reply was issued in response to a POST for /simplereenroll.

In both HTTP and CoAP cases, the registrar will first have established a TLS or DTLS session with the client. TLS sessions require on the order of a few hundred bytes of storage per client session. The new node will also have a similar expense during the enrollment process. This will take multiple round-trips in general, although the TLS session resumption protocol may be useful in a limited number of re-authentication cases.

5. Security Considerations

STUFF

6. References

6.1. Normative References

[I-D.ietf-6lo-paging-dispatch]
Thubert, P. and R. Cragie, "6LoWPAN Paging Dispatch", draft-ietf-6lo-paging-dispatch-05 (work in progress), October 2016.

[I-D.ietf-6tisch-terminology]
Palattella, M., Thubert, P., Watteyne, T., and Q. Wang, "Terminology in IPv6 over the TSCH mode of IEEE 802.15.4e", draft-ietf-6tisch-terminology-09 (work in progress), June 2017.

[I-D.ietf-core-block]
Bormann, C. and Z. Shelby, "Block-wise transfers in CoAP", draft-ietf-core-block-21 (work in progress), July 2016.

[I-D.kumar-dice-dtls-relay]
Kumar, S., Keoh, S., and O. Garcia-Morchon, "DTLS Relay for Constrained Environments", draft-kumar-dice-dtls-relay-02 (work in progress), October 2014.

[I-D.pritikin-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Behringer, M., and S. Bjarnason, "Bootstrapping Key Infrastructures", draft-pritikin-anima-bootstrapping-keyinfra-02 (work in progress), July 2015.

[I-D.robles-roll-useofrplinfo]
Robles, I., Richardson, M., and P. Thubert, "When to use RFC 6553, 6554 and IPv6-in-IPv6", draft-robles-roll-useofrplinfo-02 (work in progress), October 2015.

[RFC1919] Chatel, M., "Classical versus Transparent IP Proxies", RFC 1919, DOI 10.17487/RFC1919, March 1996, <https://www.rfc-editor.org/info/rfc1919>.

[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>.

[RFC5191] Forsberg, D., Ohba, Y., Ed., Patil, B., Tschofenig, H., and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA)", RFC 5191, DOI 10.17487/RFC5191, May 2008, <https://www.rfc-editor.org/info/rfc5191>.

[RFC5247] Aboba, B., Simon, D., and P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework", RFC 5247, DOI 10.17487/RFC5247, August 2008, <https://www.rfc-editor.org/info/rfc5247>.

[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>.

[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>.

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

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

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

6.2. Informative References

[threadcommish] Thread Group, "Thread Commissioning", Jul 2015, <http://threadgroup.org/Portals/0/documents/whitepapers/Thread%20Commissioning%20white%20paper_v2_public.pdf>.

[transparentproxy] Hung Vu, "CA Patent 2,136,150: Apparatus and method for providing a secure gateway for communication and data exchanges between networks", 1994, <https://www.google.ca/patents/CA2136150C?cl=en>.

[transparentsquid] Daniel Kiracofe, "Transparent Proxy with Linux and Squid mini-HOWTO v1.15", August 2002, <http://www.tldp.org/HOWTO/TransparentProxy-5.html>.

Author's Address

Michael C. Richardson
Sandelman Software Works
470 Dawson Avenue
Ottawa, ON K1Z 5V7
CA

Email: mcr+ietf@sandelman.ca
URI: http://www.sandelman.ca/