DICE Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 23, 2015

S. Kumar, Ed.
Philips Research
S. Keoh
University of Glasgow Singapore
O. Garcia-Morchon
Philips Research
October 20, 2014

DTLS Relay for Constrained Environments
draft-kumar-dice-dtls-relay-02

Abstract

The 6LoWPAN and CoAP standards defined for resource-constrained devices are fast emerging as the de-facto protocols for enabling the Internet-of-Things (IoTs). Security is an important concern in IoTs and the DTLS protocol has been chosen as the preferred method for securing CoAP messages. DTLS is a point-to-point protocol relying on IP routing to deliver messages between the client and the server. However in some low-power lossy networks (LLNs) with multi-hop, a new "joining" device may not be initially IP-routable. Moreover, it exists in a separate, unauthenticated domain at the point of first contact and therefore cannot be initially trusted. This puts limitations on the ability to use DTLS as an authentication and confidentiality protocol at this stage. These devices being Resource-constrained often cannot accommodate more than one security protocol in their code memory. To overcome this problem we suggest DTLS as the single protocol and therefore, we present a DTLS Relay solution for the non-IP routable "joining" device to enable it to establish a secure DTLS connection with a DTLS Server. Furthermore we present a stateful and stateless mode of operation for the DTLS Relay.

Status of This Memo

This Internet-Draft will expire on April 23, 2015.

Table of Contents

1. Introduction

For the Internet of Things (IoT) to become a reality, it will require the participation of constrained nodes in constrained networks [RFC7228]. These constrained nodes typically implement the IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) [RFC4944] and Constrained Application Protocol (CoAP) [RFC7252] standards. The 6LoWPAN adaptation layer allows for transmission of IPv6 Packets over IEEE 802.15.4 networks [ieee802.15.4], thereby enabling end-to-end IPv6 connectivity between constrained nodes and other devices on the Internet. CoAP is a web protocol based on REST architecture designed for constrained node networks. It supports binding to UDP [RFC0768], which has advantages over TCP [RFC0793] when used in low-power lossy networks (LLNs) such as IEEE 802.15.4 [ieee802.15.4].

Security is an important concern in such a constrained node network, which could be used in various application domains such as smart energy and building automation. However, security protocols are often heavy-weight in terms of both code and network processing. Use of multiple security protocols for different purposes and at different networking layers is problematic in constrained devices, therefore the use of a single security protocol to fulfil multiple security requirements is greatly preferred.

CoAP has chosen Datagram Transport Layer Security (DTLS) [RFC6347] as the preferred security protocol for authenticity and confidentiality of the messages. It is based on Transport Layer Security (TLS) [RFC5246] with modifications to run over UDP. DTLS makes use of additional reliability mechanisms in its handshake due to the lack of TCP reliable transmission mechanisms that are available to TLS.

DTLS is a client-server protocol relying on the underlying IP layer to perform the routing between the DTLS Client and the DTLS Server. However in some LLNs with multi-hop, a new "joining" device may not be initially IP routable until it is authenticated to the network. A new "joining" device can only initially use a link-local IPv6 address to communicate with a neighbour node using neighbour discovery [RFC6775] until it receives the necessary network configuration parameters. However, before the device can receive these configuration parameters, it may need to authenticate itself or wish to authenticate the network to which it connects. Although DTLS is a suitable protocol for such authentication and secure transfer of configuration parameters, it would not work due to the lack of IP routability of DTLS messages between DTLS Client and DTLS Server.

We present a DTLS Relay solution to overcome this problem for the "joining" device to establish a DTLS connection with a DTLS Server. This draft is inspired by the Protocol for carrying Authentication for Network Access (PANA) Relay Element [RFC6345] which is intended to solve a similar problem when PANA [RFC5191] is used as the transport protocol for Extensible Authentication Protocol (EAP) [RFC3748] based network access. Recently there has been interest in transporting EAP over CoAP [I-D.marin-ace-wg-coap-eap][I-D.ohba-core-eap-based-bootstrapping] and presented DTLS Relay solution can be used to secure these messages. Further, we present a stateful and stateless mode of operation for the DTLS Relay.

This draft is an early description of the solutions and does not provide the complete details yet. This draft is structured as follows: we present a use-case for the DTLS Relay in Section 2, then present the DTLS Relay solution in Section 3 for stateful and stateless mode of operation. We compare these two solutions in

Section 3.3. Further we present some security considerations in Section 5.

2. Use Case

We present here a target usecase based on [I-D.jennings-core-transitive-trust-enrollment] describing a rendezvous protocol that allows a constrained IoT device to securely connect into a system or network. The main idea is that the joining Device has a pre-established trust relationship with a "Transfer Agent" entity, for e.g. Pre-Shared Keys provisioned during manufacturing. This "Transfer Agent" provides the needed trust credentials to the Device and/or a Controller in the system to establish a secured connection to perform further authentication and transfer of system/network configuration parameters. This step is enabled by an "Introducer" entity which informs the "Transfer Agent" about the details of Controller to which the joining Device should connect, and provide to the Controller the identity including one-time credentials for enable secure connection to the Device. The transitive trust trust establishment procedure is explained in detail in [I-D.jennings-core-transitive-trust-enrollment] and we focus here on how to enable this using DTLS.

As depicted in the Figure 1, the joining Device (D) is more than one hop away from the Controller (C) and not yet authenticated into the network. At this stage, it can only communicate one-hop to its nearest neighbour (N) using their link-local IPv6 addresses. However, the Device needs to communicate with end-to-end security with a Transfer Agent (T) or a Controller (C) to authenticate and get the relevant system/network parameters. If the Device (D) initiates a DTLS connection to the Transfer Agent whose IP address has been pre-configured, then the packets are dropped at the neighbour (N) since the Device (D) is not yet admitted to the network or there is no IP routability to Device (D) for any returned messages.

```
             Transfer Agent
                 ++++
                 |T |
                 |  |             +--+
                 ++++             |N'|
                   |             --+--+
                   |  ++++    /
                   |  |C |----       +--+        +--+
                    --|  |    \      |N |........|D |
                      ++++     \-----|  |        |  |
                   Controller        +--+        +--+
                                  Neighbour   "Joining" Device
```

Figure 1: Use case depiction in a multi-hop network

Furthermore, the Device (D) may wish to establish a secure connection to the Controller (C) in the network assuming appropriate credentials are exchanged out-of-band, e.g. a hash of the Device (D)'s raw public key could be provided to the Controller (C). However, the Device (D) is unaware of the IP address of the Controller (C) to initiate a DTLS connection and perform authentication with.

To overcome these problems with non-routability of DTLS packets and/or discovery of the destination address of the DTLS Server to contact, we define a DTLS Relay solution. This DTLS Relay ability is configured into all authenticated devices in the network which may act as the Neighbour (N) device for newly joining nodes. The DTLS Relay allows for relaying of the packets from the Neighbour (N) using IP routing to the intended DTLS Server. Furthermore, if the DTLS Server address is not known to the joining Device (D), then messages are delivered to a pre-configured DTLS Server address (most likely the Controller (C)) known to the DTLS Relay.

3. DTLS Relay

In this section, we describe how the DTLS Relay functionality can be achieved. When a joining device as a client attempts a DTLS connection (for example to a "Transfer Agent"), it uses its link-local IP address as its IP source address. This message is transmitted one-hop to a neighbour node. Under normal circumstances, this message would be dropped at the neighbour node since the joining device is not yet IP routable or it is not yet authenticated to send messages through the network. However, if the neighbour device has the DTLS Relay functionality enabled, it relays the DTLS message to a specific DTLS Server. Additional security mechanisms need to exist to prevent this relaying functionality being used by rogue nodes to

bypass any network authentication procedures. These mechanisms are discussed in Section 5.

The DTLS Relay can operate in two different modes: stateful and stateless. We present here both modes, however for inter-operability, only one of the modes should be mandated. Within each mode, the DTLS Relay can further relay packets based on the client-defined DTLS Server address or a DTLS Server address that has been configured into the DTLS Relay.

3.1. DTLS Relay in Stateful mode

On receiving a DTLS message from a joining device, the neighbour node enters into DTLS Relay stateful mode. In this mode, the neighbour node has the additional DTLS Relay functionality to send DTLS messages further to the end-point DTLS Server the joining device wishes to contact. In the stateful mode of operation, the message is transmitted to the end-point DTLS Server as if it originated from the DTLS Relay, by replacing the IP address and port to the DTLS Relay's own IP address and a randomly chosen port. The DTLS message itself is not modified.

Additionally, the DTLS Relay must track the ongoing DTLS connections based on the following 4-tuple stored locally:

- o DTLS Client source link-local IP address (IP_C)

- o DTLS Client source port (p_C)

- o DTLS Server IP address (IP_S)

- o DTLS Relay source port (p_R)

The DTLS Server communicates with the DTLS Relay as if it were communicating with the DTLS Client, without any modification required to the DTLS messages. On receiving a DTLS message from the DTLS Server, the DTLS Relay looks up its locally stored 4-tuple array to identify to which DTLS Client (if multiple exist) the message belongs. The DTLS message's destination address and port are replaced with the link-local address and port of the corresponding DTLS Client respectively and the DTLS message is then forwarded to the DTLS Client. The DTLS Relay does not modify the DTLS packets and therefore the normal processing and security of DTLS is unaffected.

The following message flow diagram indicates the various steps of the process where the DTLS Server address in known to the joining device:

```
+---------------+---------------+----------------+---------------------------+
|  DTLS Client  |   DTLS Relay  |   DTLS Server  |          Message          |
|     (C)       |       (R)     |       (S)      | Src_IP:port | Dst_IP:port |
+---------------+---------------+----------------+-------------+-------------+
|      --ClientHello-->                          |   IP_C:p_C  | IP_S:5684   |
|                          --ClientHello-->      |   IP_R:p_R  | IP_S:5684   |
|                                                |             |             |
|                          <--ServerHello--      |   IP_S:5684 | IP_R:p_R    |
|                                  :             |             |             |
|            <--ServerHello--                    |   IP_S:5684 | IP_C:p_C    |
|                    :                           |             |             |
|                        ::                      |       :     |    :        |
|                        ::                      |       :     |    :        |
|        --Finished-->                           |   IP_C:p_C  | IP_S:5684   |
|                            --Finished-->       |   IP_R:p_R  | IP_S:5684   |
|                                                |             |             |
|                            <--Finished--       |   IP_S:5684 | IP_R:p_R    |
|        <--Finished--                           |   IP_S:5684 | IP_C:p_C    |
|                        ::                      |      :      |     :       |
+-------------------------------------------------+-------------+-------------+
IP_C:p_C = Link-local IP address and port of DTLS Client
IP_S:5684 = IP address and coaps port of DTLS Server
IP_R:p_R = IP address and port of DTLS Relay
```

Figure 2: Message flow in Stateful mode with DTLS Server defined by DTLS Client

In the situation where the joining device is unaware of the IP address of the DTLS Server it needs to contact, for e.g. the Controller of the network, the DTLS Relay can be configured with IP destination of the default DTLS Server that a DTLS client (joining device) needs to contact. The DTLS client initiates its DTLS request as if the DTLS Relay is the intended end-point DTLS Server. The DTLS Relay changes the IP packet (without modifying the DTLS message) as in the previous case by modifying both the source and destination IP addresses to forward the message to the intended DTLS Server. The DTLS Relay keeps a similar 4-tuple array to enable translation of the DTLS messages received from the DTLS Server and forward it to the DTLS Client. The following message flow indicates this process:

```
+---------------+---------------+----------------+---------------------------+
|  DTLS Client  |   DTLS Relay  |   DTLS Server  |          Message          |
|     (C)       |       (R)     |       (S)      | Src_IP:port | Dst_IP:port |
+---------------+---------------+----------------+-------------+-------------+
|      --ClientHello-->                          |   IP_C:p_C  | IP_Ra:5684  |
|                         --ClientHello-->       |   IP_Rb:p_Rb| IP_S:5684   |
|                                                |             |             |
|                         <--ServerHello--       |   IP_S:5684 | IP_Rb:p_Rb  |
|                                 :              |             |             |
|            <--ServerHello--                    |   IP_Ra:5684| IP_C:p_C    |
|                    :                           |             |             |
|                        ::                      |       :     |    :        |
|                        ::                      |       :     |    :        |
|        --Finished-->                           |   IP_C:p_C  | IP_Ra:5684  |
|                           --Finished-->        |   IP_Rb:p_Rb| IP_S:5684   |
|                                                |             |             |
|                           <--Finished--        |   IP_S:5684 | IP_Rb:p_Rb  |
|        <--Finished--                           |   IP_Ra:5684| IP_C:p_C    |
|                       ::                       |      :      |     :       |
+------------------------------------------------+-------------+-------------+
IP_C:p_C = Link-local IP address and port of DTLS Client
IP_S:5684 = IP address and coaps port of DTLS Server
IP_Ra:5684 = Link-local IP address and coaps port of DTLS Relay
IP_Rb:p_Rb = IP address (can be same as IP_Ra) and port of DTLS Relay
```

Figure 3: Message flow in Stateful mode with DTLS Server defined by DTLS Relay

3.2. DTLS Relay in Stateless mode

In the alternative mode of operation for the DTLS Relay, a stateless approach is applied where the DTLS Relay does not need to store a local 4-tuple array. Just as in the previous case, if an untrusted DTLS Client that can only use link-local addressing wants to contact a trusted end-point DTLS Server, it send the DTLS message to the DTLS Relay. Instead of changing the IP addresses and port of the IP packet, the DTLS Relay encapsulates this message into a new type of message called DTLS Relay (DRY) message. The DRY message consists of two parts:

- o Header (H) field: consisting of the source link-local address and port of the DTLS Client device, and

- o Contents (C) field: containing the original DTLS message.

On receiving the DRY message, the DTLS Server decapsulates it to retrieve the two parts. It uses the Header field information to

transiently store the DTLS Client's address and port. The DTLS Server then performs the normal DTLS operations on the DTLS message from the Contents field. However, when the DTLS Server replies, it also encapsulates its DTLS message in a DRY message back to the DTLS Relay. The Header contains the original source link-local address and port of the DTLS Client from the transient state stored earlier (which can now be discarded) and the Contents field contains the DTLS message.

On receiving the DRY message, the DTLS Relay decapsulates it to retrieve the two parts. It uses the Header field to relay the DTLS message retrieved from the Contents field to the right DTLS Client.

The following figure depicts the message flow diagram when the DTLS Server end-point address is known only to the Relay:

```
+----------------+---------------------+---------------------
+---------------------------+
|  DTLS Client   |      DTLS Relay     |      DTLS Server    |
Message           |
|     (C)        |          (R)        |          (S)        | Src_IP:port |
Dst_IP:port |
+----------------+---------------------+---------------------+-------------
+-------------+
|         --ClientHello-->                                   |   IP_C:p_C  |
IP_Ra:5684  |
|                     --DRY[H(IP_C:p_C),C(ClientHello)]-->  |   IP_Rb:p_Rb|
IP_S:5684   |
|                                                            |
|             |
|                     <--DRY[H(IP_C:p_C),C(ServerHello)]--  |   IP_S:5684 |
IP_Rb:p_Rb  |
|                                          :                 |
|             |
|         <--ServerHello--                                   |   IP_Ra:5684|
IP_C:p_C    |
|                 :                                          |
|             |
|                              ::                            |       :
|    :        |
|                              ::                            |       :
|    :        |
|          --Finished-->                                     |   IP_C:p_C  |
IP_Ra:5684  |
|                        --DRY[H(IP_C:p_C),C(Finished)]-->  |   IP_Rb:p_Rb|
IP_S:5684   |
|                                                            |
```

```
|             |
|                         <--DRY[H(IP_C:p_C),C(Finished)]--  |   IP_S:5684 |
IP_Rb:p_Rb  |
|          <--Finished--                                     |   IP_Ra:5684|
IP_C:p_C    |
|                               ::                           |      :
|     :       |
+-------------------------------------------------------------+-------------
+-------------+
IP_C:p_C = Link-local IP address and port of DTLS Client
IP_S:5684 = IP address and coaps port of DTLS Server
IP_Ra:5684 = Link-local IP address and coaps port of DTLS Relay
IP_Rb:p_Rb = IP address(can be same as IP_Ra) and port of DTLS Relay

DRY[H(),C()] = DTLS Relay message with header H and content C
```

Figure 4: Message flow in Stateless mode with DTLS Server defined by DTLS Relay

The message flow for the case in which the DTLS Client is aware of the end-point DTLS Server's IP address is similar and not described further. It can be derived based on Figure 2 and Figure 4.

3.3. Comparison between the two modes

The stateful and stateless mode of operation for the DTLS Relay have their advantages and disadvantages. This comparison should enable to make a good choice between the two based on the available device resources and network bandwidth in a given deployment.

Properties	Stateful mode	Stateless mode
State information	The Relay needs additional storage to maintain mapping of the joining device's address with the port number being used to communicate with the Server.	No information is maintained by the Relay.
Packet size	The size of the relayed message is the same as the original message .	The size of the relayed message is bigger than the original, it includes additional source and destination addresses.
Standardization requirements	The additional functionalities of the Relay to maintain state information, and modify the source and destination addresses of the	New DRY message to encapsulate DTLS message. The Server and the Relay have to understand the DRY message in order to process it.

```
|                   |DTLS handshake messages.
|                               |
+-------------------+-----------------------------------
+--------------------------------+
```

Table 1: Comparison between stateful and stateless mode DTLS Relay

Figure 5

4. IANA Considerations

tbd

Note to RFC Editor: this section may be removed on publication as an RFC.

5. Security Considerations

Additional security considerations need to be taken into account when forwarding messages from devices through a network to which it has not yet been admitted since this can lead to denial-of-service (DoS) attacks or misuse of network resources without proper authentication. There are various solution options by which one could try to limit the damage that an attacker can cause by DoS. One way to overcome any large scale misuse of the network is to have a management message from the Controller that initiates already authenticated devices in the network to enable or disable the DTLS Relay mode. This is often possible since the administrator of the network is aware when new devices join the network either because of the "Introduction" phase or commissioning phase. Alternatively the management message can be used to control a different networking layer on the relay nodes that disable new nodes from joining. Such solution options are orthogonal to the DTLS relay functionality and should be built in based on the underlying network capabilities and deployment scenario.

In terms of the two different modes, Stateful mode has additional security issues since the DTLS Relay has to store state from an unauthenticated node and then relay a message, expecting a corresponding reply sometime in the future. Furthermore, the DTLS Server has to store state as well but it is more transient. This could lead to a simple localised attack on a DTLS Relay whereby a rogue device could use up state storage on a DTLS Relay quite easily, thus denying a legitimate device from being able to gain access.

In comparison, in the Stateless mode the DTLS Relay does not store any state, and therefore an attack as described above is not possible. Also a DTLS Server can legitimately silently discard a DTLS message without concern as the DTLS Relay has no further knowledge or state stored of the DTLS Client. The DTLS cookie mechanism is a good addition to a stateless transaction which improves the likelihood a DTLS Server is talking to a genuine DTLS Client.

6. Acknowledgements

The authors would like to thank Sahil Sharma, Ernest Ma, Dee Denteneer, Peter Lenoir and Martin Turon for the valuable discussions. Also thank Robert Craige for his valuable comments and edits.

7. References

7.1. Normative References

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

7.2. Informative References

[I-D.jennings-core-transitive-trust-enrollment]
Jennings, C., "Transitive Trust Enrollment for Constrained Devices", draft-jennings-core-transitive-trust-enrollment-01 (work in progress), October 2012.

[I-D.marin-ace-wg-coap-eap]
Garcia, D., "EAP-based Authentication Service for CoAP", draft-marin-ace-wg-coap-eap-01 (work in progress), October 2014.

[I-D.ohba-core-eap-based-bootstrapping]
Das, S. and Y. Ohba, "Provisioning Credentials for CoAP Applications using EAP", draft-ohba-core-eap-based-bootstrapping-01 (work in progress), March 2012.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.

[RFC5191] Forsberg, D., Ohba, Y., Patil, B., Tschofenig, H., and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA)", RFC 5191, May 2008.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC6345] Duffy, P., Chakrabarti, S., Cragie, R., Ohba, Y., and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA) Relay Element", RFC 6345, August 2011.

[RFC6775] Shelby, Z., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, November 2012.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, May 2014.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, June 2014.

[ieee802.15.4] IEEE Computer Society, , "IEEE Std. 802.15.4-2003", October 2003.

Authors' Addresses

Sandeep S. Kumar (editor)
Philips Research
High Tech Campus 34
Eindhoven 5656 AE
NL

Email: ietf@sandeep.de

Sye Loong Keoh
University of Glasgow Singapore
Republic PolyTechnic, 9 Woodlands Ave 9
Singapore 838964
SG

Email: SyeLoong.Keoh@glasgow.ac.uk

Oscar Garcia-Morchon
Philips Research
High Tech Campus 34
Eindhoven 5656 AE
NL

Email: oscar.garcia@philips.com