Workgroup: TLS
Internet-Draft: draft-ietf-tls-dtls-rrc-04
Updates: 6347 (if approved)
Published: 21 December 2021
Intended Status: Standards Track
Expires: 24 June 2022

Authors: H. Tschofenig, Ed.   T. Fossati, Ed.
         Arm Limited   Arm Limited

Return Routability Check for DTLS 1.2 and DTLS 1.3

Abstract

   This document specifies a return routability check for use in
   context of the Connection ID (CID) construct for the Datagram
   Transport Layer Security (DTLS) protocol versions 1.2 and 1.3.

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 24 June 2022.

Table of Contents

1. Introduction

In "classical" DTLS, selecting a security context of an incoming
DTLS record is accomplished with the help of the 5-tuple, i.e.
source IP address, source port, transport protocol, destination IP
address, and destination port. Changes to this 5 tuple can happen
for a variety reasons over the lifetime of the DTLS session. In the
IoT context, NAT rebinding is common with sleepy devices. Other
examples include end host mobility and multi-homing. Without CID, if
the source IP address and/or source port changes during the lifetime
of an ongoing DTLS session then the receiver will be unable to
locate the correct security context. As a result, the DTLS handshake
has to be re-run. Of course, it is not necessary to re-run the full
handshake if session resumption is supported and negotiated.

A CID is an identifier carried in the record layer header of a DTLS
datagram that gives the receiver additional information for
selecting the appropriate security context. The CID mechanism has
been specified in [I-D.ietf-tls-dtls-connection-id] for DTLS 1.2 and
in [I-D.ietf-tls-dtls13] for DTLS 1.3.

Section 6 of [I-D.ietf-tls-dtls-connection-id] describes how the use
of CID increases the attack surface by providing both on-path and
off-path attackers an opportunity for (D)DoS. It then goes on
describing the steps a DTLS principal must take when a record with a
CID is received that has a source address (and/or port) different
from the one currently associated with the DTLS connection. However,
the actual mechanism for ensuring that the new peer address is
willing to receive and process DTLS records is left open. This
document standardizes a return routability check (RRC) as part of
the DTLS protocol itself.

The return routability check is performed by the receiving peer
before the CID-to-IP address/port binding is updated in that peer's
session state database. This is done in order to provide more
confidence to the receiving peer that the sending peer is reachable
at the indicated address and port.

Note however that, irrespective of CID, if RRC has been successfully
negotiated by the peers, path validation can be used at any time by
either endpoint. For instance, an endpoint might use RRC to check
that a peer is still in possession of its address after a period of
quiescence.

2. Conventions and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and
"OPTIONAL" in this document are to be interpreted as described in
BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all
capitals, as shown here.

This document assumes familiarity with the CID format and protocol
defined for DTLS 1.2 [I-D.ietf-tls-dtls-connection-id] and for DTLS
1.3 [I-D.ietf-tls-dtls13]. The presentation language used in this
document is described in Section 4 of [RFC8446].

This document reuses the definition of "anti-amplification limit"
from [RFC9000] to mean three times the amount of data received from
an unvalidated address. This includes all DTLS records originating
from that source address, excluding discarded ones.

3. RRC Extension

The use of RRC is negotiated via the rrc DTLS-only extension. On
connecting, the client includes the rrc extension in its ClientHello
if it wishes to use RRC. If the server is capable of meeting this
requirement, it responds with a rrc extension in its ServerHello.
The extension_type value for this extension is TBD1 and the
extension_data field of this extension is empty. The client and
server MUST NOT use RRC unless both sides have successfully
exchanged rrc extensions.

Note that the RRC extension applies to both DTLS 1.2 and DTLS 1.3.

4. The Return Routability Check Message

When a record with CID is received that has the source address of
the enclosing UDP datagram different from the one previously
associated with that CID, the receiver MUST NOT update its view of
the peer's IP address and port number with the source specified in
the UDP datagram before cryptographically validating the enclosed
record(s) but instead perform a return routability check.

```
enum {
    invalid(0),
    change_cipher_spec(20),
    alert(21),
    handshake(22),
    application_data(23),
    heartbeat(24),   /* RFC 6520 */
    return_routability_check(TBD2), /* NEW */
    (255)
} ContentType;

uint64 Cookie;

enum {
    path_challenge(0),
    path_response(1),
    reserved(2..255)
} rrc_msg_type;

struct {
    rrc_msg_type msg_type;
    select (return_routability_check.msg_type) {
        case path_challenge: Cookie;
        case path_response:  Cookie;
    };
} return_routability_check;
```

The newly introduced return_routability_check message contains a
cookie. The cookie is a 8-byte field containing arbitrary data.

The return_routability_check message MUST be authenticated and
encrypted using the currently active security context.

5. Path Validation Procedure

The receiver that observes the peer's address or port update MUST
stop sending any buffered application data (or limit the data sent
to the unvalidated address to the anti-amplification limit) and
initiate the return routability check that proceeds as follows:

   1. An unpredictable cookie is placed in a return_routability_check
      message of type path_challenge;

   2. The message is sent to the observed new address and a timer T
      (see Section 5.3) is started;

   3. The peer endpoint, after successfully verifying the received
      return_routability_check message responds by echoing the cookie
      value in a return_routability_check message of type
      path_response;

   4. When the initiator receives and verifies the
      return_routability_check message contains the sent cookie, it
      updates the peer address binding;

   5. If T expires, or the address confirmation fails, the peer
      address binding is not updated.

After this point, any pending send operation is resumed to the bound
peer address.

Section 5.1 and Section 5.2 contain the requirements for the
initiator and responder roles, broken down per protocol phase.

5.1. Path Challenge Requirements

   *The initiator MAY send multiple return_routability_check messages
    of type path_challenge to cater for packet loss on the probed
    path.

      -Each path_challenge SHOULD go into different transport
       packets. (Note that the DTLS implementation may not have
       control over the packetization done by the transport layer.)

      -The transmission of subsequent path_challenge messages SHOULD
       be paced to decrease the chance of loss.

-Each path_challenge message MUST contain random data.

   *The initiator MAY use padding using the record padding mechanism
    available in DTLS 1.3 (and in DTLS 1.2, when CID is enabled on
    the sending direction) up to the anti-amplification limit to
    probe if the path MTU (PMTU) for the new path is still
    acceptable.

5.2. Path Response Requirements

   *The responder MUST NOT delay sending an elicited path_response
    message.

   *The responder MUST send exactly one path_response messages for
    each received path_request.

   *The responder MUST send the path_response on the network path
    where the corresponding path_challenge has been received, so that
    validation succeeds only if the path is functional in both
    directions.

      -The initiator MUST NOT enforce this behaviour

   *The initiator MUST silently discard any invalid path_response it
    receives.

   Note that RRC does not cater for PMTU discovery on the reverse path.
   If the responder wants to do PMTU discovery using RRC, it should
   initiate a new path validation procedure.

5.3. Timer Choice

   When setting T, implementations are cautioned that the new path
   could have a longer round-trip time (RTT) than the original.

   In settings where there is external information about the RTT of the
   active path, implementations SHOULD use T = 3xRTT.

   If an implementation has no way to obtain information regarding the
   RTT of the active path, a value of 1s SHOULD be used.

   Profiles for specific deployment environments - for example,
   constrained networks [I-D.ietf-uta-tls13-iot-profile] - MAY specify
   a different, more suitable value.

6. Example

   The example TLS 1.3 handshake shown in Figure 1 shows a client and a
   server negotiating the support for CID and for the RRC extension.

```
          Client                                            Server

Key  ^ ClientHello
Exch | + key_share
     | + signature_algorithms
     | + rrc
     v + connection_id=empty
                              -------->
                                                  ServerHello  ^ Key
                                               +  key_share  | Exch
                                          + connection_id=100  |
                                                     + rrc  v
                                          {EncryptedExtensions}  ^  Server
                                           {CertificateRequest}  v  Params
                                                  {Certificate}  ^
                                            {CertificateVerify}  | Auth
                              <--------              {Finished}  v


     ^ {Certificate}
Auth | {CertificateVerify}
     v {Finished}              -------->
       [Application Data]      <------->  [Application Data]


           +   Indicates noteworthy extensions sent in the
               previously noted message.

           *   Indicates optional or situation-dependent
               messages/extensions that are not always sent.

           {} Indicates messages protected using keys
               derived from a [sender]_handshake_traffic_secret.

           [] Indicates messages protected using keys
               derived from [sender]_application_traffic_secret_N.

           Figure 1: Message Flow for Full TLS Handshake
```

Once a connection has been established the client and the server
exchange application payloads protected by DTLS with an unilaterally
used CIDs. In our case, the client is requested to use CID 100 for
records sent to the server.

At some point in the communication interaction the IP address used
by the client changes and, thanks to the CID usage, the security
context to interpret the record is successfully located by the
server. However, the server wants to test the reachability of the
client at his new IP address.

```
Client                                        Server
------                                        ------

Application Data              ========>
<CID=100>
Src-IP=A
Dst-IP=Z
                              <========            Application Data
                                                        Src-IP=Z
                                                        Dst-IP=A


                        <<------------->>
                        <<    Some      >>
                        <<    Time      >>
                        <<    Later     >>
                        <<------------->>


Application Data              ========>
<CID=100>
Src-IP=B
Dst-IP=Z

                                         <<< Unverified IP
                                               Address B >>

                              <--------  Return Routability Check
                                         path_challenge(cookie)
                                                   Src-IP=Z
                                                   Dst-IP=B
Return Routability Check      -------->
path_response(cookie)
Src-IP=B
Dst-IP=Z

                                         <<< IP Address B
                                               Verified >>


                              <========            Application Data
                                                        Src-IP=Z
                                                        Dst-IP=B
```

Figure 2: Return Routability Example

7. Security and Privacy Considerations

Note that the return routability checks do not protect against
flooding of third-parties if the attacker is on-path, as the
attacker can redirect the return routability checks to the real peer
(even if those datagrams are cryptographically authenticated). On-
path adversaries can, in general, pose a harm to connectivity.

8. IANA Considerations

IANA is requested to allocate an entry to the TLS ContentType
registry, for the return_routability_check(TBD2) defined in this
document. The return_routability_check content type is only
applicable to DTLS 1.2 and 1.3.

IANA is requested to allocate the extension code point (TBD1) for
the rrc extension to the TLS ExtensionType Values registry as
described in Table 1.

Value	Extension Name	TLS 1.3	DTLS-Only	Recommended	Reference
TBD1	rrc	CH, SH	Y	N	RFC-THIS

Table 1: rrc entry in the TLS ExtensionType Values registry

9. Open Issues

Issues against this document are tracked at https://github.com/
tlswg/dtls-rrc/issues

10. Acknowledgments

We would like to thank Achim Kraus, Hanno Becker, Hanno Boeck,
Manuel Pegourie-Gonnard, Mohit Sahni and Rich Salz for their input
to this document.

11. References

11.1. Normative References

[I-D.ietf-tls-dtls-connection-id] Rescorla, E., Tschofenig, H.,
           Fossati, T., and A. Kraus, "Connection Identifiers for
           DTLS 1.2", Work in Progress, Internet-Draft, draft-ietf-
           tls-dtls-connection-id-13, 22 June 2021, <https://
           www.ietf.org/archive/id/draft-ietf-tls-dtls-connection-
           id-13.txt>.

[I-D.ietf-tls-dtls13] Rescorla, E., Tschofenig, H., and N. Modadugu,
           "The Datagram Transport Layer Security (DTLS) Protocol
           Version 1.3", Work in Progress, Internet-Draft, draft-

ietf-tls-dtls13-43, 30 April 2021, <https://www.ietf.org/
archive/id/draft-ietf-tls-dtls13-43.txt>.

[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>.

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

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

11.2. Informative References

[I-D.ietf-uta-tls13-iot-profile] Tschofenig, H. and T. Fossati,
"TLS/DTLS 1.3 Profiles for the Internet of Things", Work
in Progress, Internet-Draft, draft-ietf-uta-tls13-iot-
profile-03, 25 October 2021, <https://www.ietf.org/
archive/id/draft-ietf-uta-tls13-iot-profile-03.txt>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based
Multiplexed and Secure Transport", RFC 9000, DOI
10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/
info/rfc9000>.

Appendix A. History

RFC EDITOR: PLEASE REMOVE THIS SECTION

draft-ietf-tls-dtls-rrc-04

  *Re-submitted draft to fix references

draft-ietf-tls-dtls-rrc-03

  *Added details for challenge-response exchange

draft-ietf-tls-dtls-rrc-02

  *Undo the TLS flags extension for negotiating RRC, use a new
   extension type

draft-ietf-tls-dtls-rrc-01

  *Use the TLS flags extension for negotiating RRC

*Enhanced IANA consideration section

   *Expanded example section

   *Revamp message layout:

      -Use 8-byte fixed size cookies

      -Explicitly separate path challenge from response

   draft-ietf-tls-dtls-rrc-00

   *Draft name changed after WG adoption

   draft-tschofenig-tls-dtls-rrc-01

   *Removed text that overlapped with draft-ietf-tls-dtls-connection-
    id

   draft-tschofenig-tls-dtls-rrc-00

   *Initial version

Authors' Addresses

   Hannes Tschofenig (editor)
   Arm Limited

   Email: hannes.tschofenig@arm.com

   Thomas Fossati (editor)
   Arm Limited

   Email: thomas.fossati@arm.com