Internet Engineering Task Force (IETF)
Request for Comments: 9853
Updates: 9146, 9147
Category: Standards Track
ISSN: 2070-1721

H. Tschofenig, Ed.
UniBw M.
A. Kraus

T. Fossati
Linaro

March 2026

Return Routability Check for DTLS 1.2 and 1.3

Abstract

This document specifies a Return Routability Check (RRC) subprotocol for use in the context of the Connection ID (CID) construct for the Datagram Transport Layer Security (DTLS) protocol versions 1.2 and 1.3.

Implementations offering the CID functionality described in RFCs 9146 and 9147 are encouraged to also provide the RRC functionality described in this document. For this reason, this document updates RFCs 9146 and 9147.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9853.

Table of Contents

1. Introduction

A Connection ID (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 [RFC9146] for DTLS 1.2 and in [RFC9147] for DTLS 1.3.

Section 6 of [RFC9146] describes how the use of CID increases the attack surface of DTLS 1.2 and 1.3 by providing both on-path and off-path attackers an opportunity for DoS or DDoS. It also describes the steps a DTLS principal must take when a record with a CID is received that has a source address 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. To address the gap, this document defines a Return Routability Check (RRC) subprotocol for DTLS 1.2 and 1.3, inspired by the path validation procedure defined in Section 8.2 of [RFC9000]. As such, this document updates [RFC9146] and [RFC9147].

The return routability check is performed by the receiving endpoint before the CID-address binding is updated in that endpoint's session state. This is done in order to give the receiving endpoint confidence that the sending peer is in fact reachable at the source address indicated in the received datagram. For an illustration of the handshake and address validation phases, see Section 6.

Section 5.1 of this document explains the fundamental mechanism that aims to reduce the DDoS attack surface. Additionally, Section 5.2 discusses a more advanced address validation mechanism. This mechanism is designed to counteract off-path attackers trying to place themselves on-path by racing packets that trigger address rebinding at the receiver. To gain a detailed understanding of the attacker model, please refer to Section 8.1.

Apart from of its use in the context of CID-address binding updates, the path validation capability offered by RRC can be used at any time by either endpoint. For instance, an endpoint might use RRC to check that a peer is still reachable at its last known 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 [RFC9146] and for DTLS 1.3 [RFC9147]. The presentation language used in this document is described in Section 4 of [RFC8446].

In this document, the term "anti-amplification limit" means three times the amount of data received from an unvalidated address. This includes all DTLS records originating from that source address, excluding those that have been discarded. This follows the pattern of [RFC9000], applying a similar concept to DTLS.

The term "address" is defined in Section 1.2 of [RFC9000].

The terms "client", "server", "peer", and "endpoint" are defined in Section 1.1 of [RFC8446].

3. RRC Extension

The use of RRC is negotiated via the rrc extension. The rrc extension is only defined for DTLS 1.2 and 1.3. On connecting, a client wishing to use RRC includes the rrc extension in its ClientHello. If the server is capable of meeting this requirement, it responds with an rrc extension in its ServerHello. The extension_type value for this extension is 61, and the extension_data field of this extension is empty. A client offering the rrc extension MUST also offer the connection_id extension [RFC9146]. If the client includes the rrc extension in its ClientHello but omits the connection_id extension, the server MUST NOT include the rrc extension in its ServerHello. A client offering the connection_id extension SHOULD also offer the rrc extension, unless the application using DTLS has its own address validation mechanism. The client and server MUST NOT use RRC unless both sides have successfully exchanged rrc extensions.

3.1. RRC and CID Interplay

RRC offers an in-protocol mechanism to perform peer address validation that complements the "peer address update" procedure described in Section 6 of [RFC9146]. Specifically, when both CID [RFC9146] and RRC have been successfully negotiated for the session, if a record with CID is received that has the source address of the enclosing UDP datagram different from what is currently associated with that CID value, the receiver SHOULD perform a return routability check as described in Section 5, unless an application-specific address validation mechanism can be triggered instead (e.g., Constrained Application Protocol (CoAP) Echo [RFC9175]).

4. Return Routability Check Message Types

This document defines the return_routability_check content type (Figure 1) to carry Return Routability Check messages.

The RRC subprotocol consists of three message types: path_challenge, path_response, and path_drop. These message types are used for path validation and selection as described in Section 5.

Each message carries a Cookie, an 8-byte field containing 64 bits of entropy (e.g., obtained from the cryptographically secure pseudorandom number generator (CSPRNG) used by the TLS implementation; see Appendix C.1 of [RFC8446]).

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

```
enum {
    invalid(0),
    change_cipher_spec(20),
    alert(21),
    handshake(22),
    application_data(23),
    heartbeat(24),  /* RFC 6520 */
    tls12_cid(25),  /* RFC 9146, DTLS 1.2 only */
    return_routability_check(27), /* NEW */
    (255)
} ContentType;

uint64 Cookie;

enum {
    path_challenge(0),
    path_response(1),
    path_drop(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;
        case path_drop:      Cookie;
    };
} return_routability_check;
```

Figure 1: Return Routability Check Message and Content Type

Future extensions to the RRC subprotocol may define new message types. Implementations MUST be able to parse and understand the three RRC message types defined in this document. In addition, implementations MUST be able to parse and gracefully ignore messages with an unknown msg_type.

5. Path Validation Procedure

A receiver that observes the peer's address change MUST stop sending any buffered application data or limit the data sent to the unvalidated address to the anti-amplification limit. It then initiates the return routability check.

This document describes two kinds of checks: basic (Section 5.1) and enhanced (Section 5.2). The choice of one or the other depends on whether the off-path attacker scenario described in Section 8.1.2 is to be considered. (The decision on what strategy to choose depends mainly on the threat model but may also be influenced by other considerations. Examples of impacting factors include the need to minimise implementation complexity, privacy concerns, and the need to reduce the time it takes to switch path. The choice may be offered as a configuration option to the user of the TLS implementation.)

After the path validation procedure is complete, any pending send operation is resumed to the bound peer address.

Sections 5.3 and 5.4 list the requirements for the initiator and responder roles, broken down per protocol phase.

Please note that the presented algorithms are not designed to handle

nested rebindings, i.e. rebindings that may occur while a path is being validated following a previous rebinding. This should rarely occur, but if it happens, the path_response message is dropped, the address validation times out, and the address will not be updated. A new path validation will start when new data is received.

Also, note that in the event of a NAT rebind, the initiator and responder will have different views of the path: The initiator will see a new path, while the responder will still see the old one.

5.1. Basic

The basic return routability check comprises the following steps:

1. The receiver (i.e., the initiator) creates a return_routability_check message of type path_challenge and places the unpredictable cookie into the message.

2. The message is sent to the observed new address and a timer T (see Section 5.5) is started.

3. The peer (i.e., the responder) cryptographically verifies the received return_routability_check message of type path_challenge and responds by echoing the cookie value in a return_routability_check message of type path_response.

4. When the initiator receives the return_routability_check message of type path_response and verifies that it contains the sent cookie, it updates the peer address binding.

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

5.2. Enhanced

The enhanced return routability check comprises the following steps:

1. The receiver (i.e., the initiator) creates a return_routability_check message of type path_challenge and places the unpredictable cookie into the message.

2. The message is sent to the previously valid address, which corresponds to the old path. Additionally, a timer T is started (see Section 5.5).

3. If the path is still functional, the peer (i.e., the responder) cryptographically verifies the received return_routability_check message of type path_challenge. The action to be taken depends on whether the path through which the message was received remains the preferred one.

   * If the path through which the message was received is preferred, a return_routability_check message of type path_response MUST be returned. (Note that, from the responder's perspective, the preferred path and the old path coincide in the event of a NAT rebind.)

   * If the path through which the message was received is no longer preferred, a return_routability_check message of type path_drop MUST be returned. (Note that the responder must have initiated a voluntary path migration in order to know that this path is no longer the preferred one.)

   In either case, the peer echoes the cookie value in the response.

4. The initiator receives and verifies that the return_routability_check message contains the previously sent

cookie. The actions taken by the initiator differ based on the received message:

* When a return_routability_check message of type path_response is received, the initiator MUST continue using the previously valid address, i.e., no switch to the new path takes place and the peer address binding is not updated.

* When a return_routability_check message of type path_drop is received, the initiator MUST perform a basic return routability check on the observed new address, as described in Section 5.1.

5. If T expires, the peer address binding is not updated. In this case, the initiator MUST perform a basic return routability check on the observed new address, as described in Section 5.1.

5.3. Path Challenge Requirements

* The initiator MAY send multiple return_routability_check messages of type path_challenge to account 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.

  - In general, the number of "backup" path_challenge messages depends on the application, since some are more sensitive than others to latency caused by changes in the path. In the absence of application-specific requirements, the initiator can send a path_challenge message once per round-trip time (RTT), up to the anti-amplification limit.

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

5.4. Path Response/Drop Requirements

* The responder MUST NOT delay sending an elicited path_response or path_drop messages.

* The responder MUST send exactly one path_response or path_drop message for each valid path_challenge it received.

* The responder MUST send the path_response or the path_drop to the address from which the corresponding path_challenge was received. This ensures that the path is functional in both directions.

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

Note that RRC does not account 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.

5.5. Timer Choice

When setting T, implementations are cautioned that the new path could

have a longer RTT than the original.

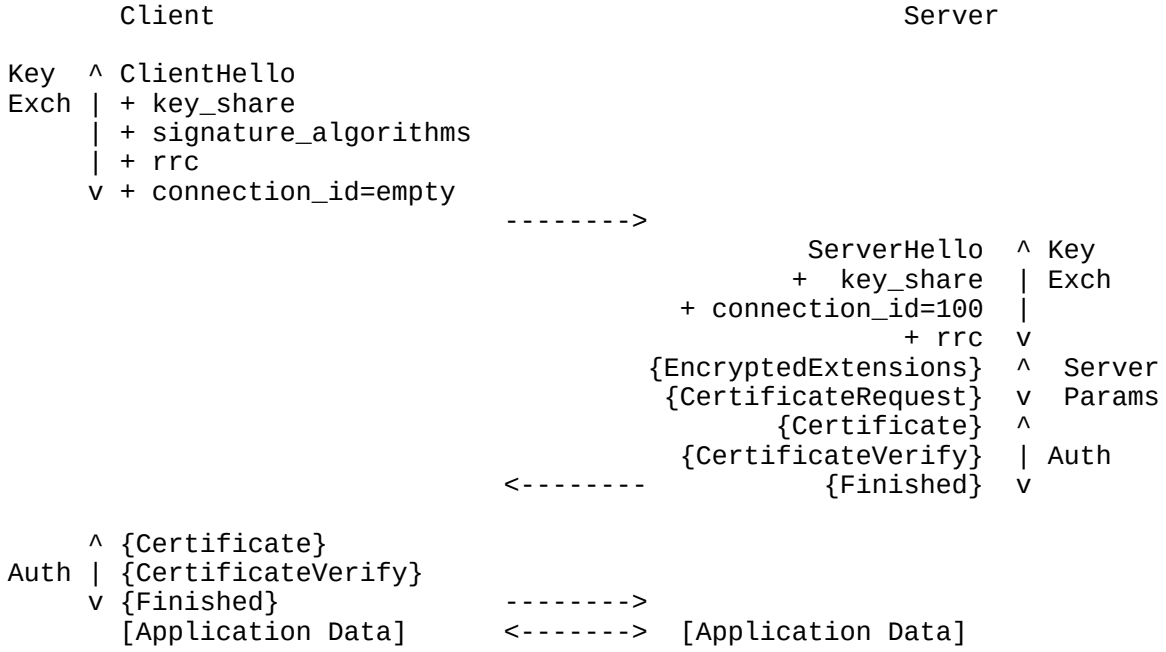
In settings where there is external information about the RTT of the active path (i.e., the old path), implementations SHOULD use T = 3xRTT.

If an implementation has no way to obtain information regarding the RTT of the active path, T SHOULD be set to 1 second.

Profiles for specific deployment environments -- for example, constrained networks [IOT-PROFILE] -- MAY specify a different, more suitable value for T.

6. Example

Figure 2 shows an example of a DTLS 1.3 handshake in which a client and a server successfully negotiate support for both the CID and RRC extensions.



\+ Indicates noteworthy extensions sent in the previously noted message.

{} 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 2: Message Flow for Full DTLS Handshake

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

At some point in the communication interaction, the 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 its new address.

Figure 3 shows the server initiating a basic RRC exchange (see

Section 5.1) that establishes reachability of the client at the new address.

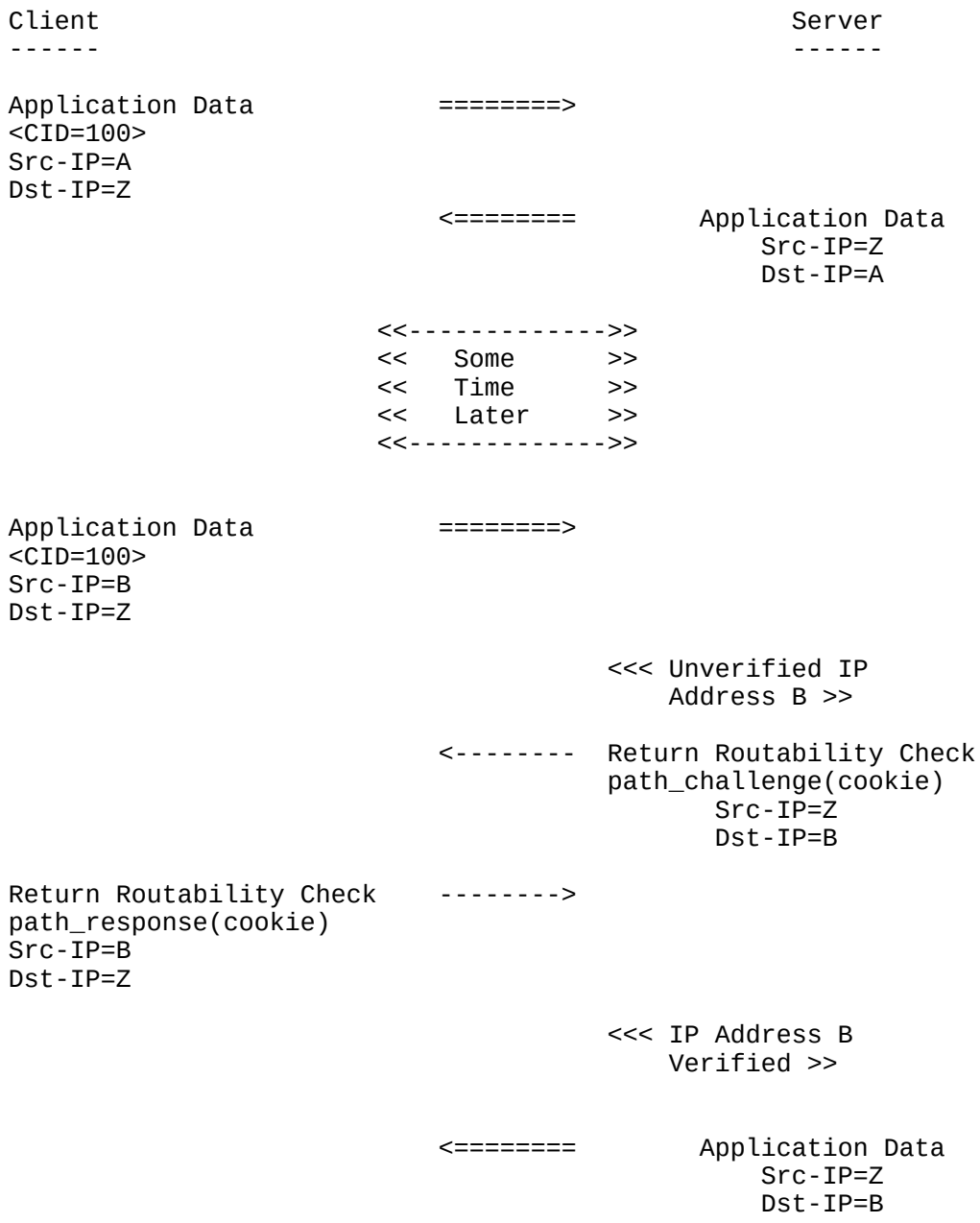


Figure 3: Basic Return Routability Example

7. Operational Considerations

7.1. Logging Anomalous Events

Logging of RRC operations at both ends of the protocol can be generally useful for the users of an implementation. In particular, for Security Information and Event Management (SIEM) and troubleshooting purposes, it is strongly advised that implementations collect statistics about any unsuccessful RRC operations, as they could represent security-relevant events when they coincide with attempts by an attacker to interfere with the end-to-end path. It is also advisable to log instances where multiple responses to a single path_challenge are received, as this could suggest an off-path attack attempt.

In some cases, the presence of frequent path probes could indicate a problem with the stability of the path. This information can be used

to identify any issues with the underlying connectivity service.

7.2. Middlebox Interference

Since the DTLS 1.3 encrypted packet's record type is opaque to on-path observers, RRC messages are immune to middlebox interference when using DTLS 1.3. In contrast, DTLS 1.2 RRC messages that are not wrapped in the tls12_cid record (e.g., in the server-to-client direction if the server negotiated a zero-length CID) have the return_routability_check content type in plain text, making them susceptible to interference (e.g., dropping of path_challenge messages), which would hinder the RRC functionality altogether. Therefore, when RRC is used in DTLS 1.2 and middlebox interference is a concern, it is recommended to enable CID in both directions.

8. Security 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.

If the RRC challenger reuses a cookie that was previously used in the same connection and does not implement anti-replay protection (see Section 4.5.1 of [RFC9147] and Section 4.1.2.6 of [RFC6347]), an attacker could replay a previously sent path_response message containing the reused cookie to mislead the challenger into switching to a path of the attacker's choosing. To prevent this, RRC cookies must be _freshly_ generated using a reliable source of entropy [RFC4086]. See Appendix C.1 of [RFC8446] for guidance.

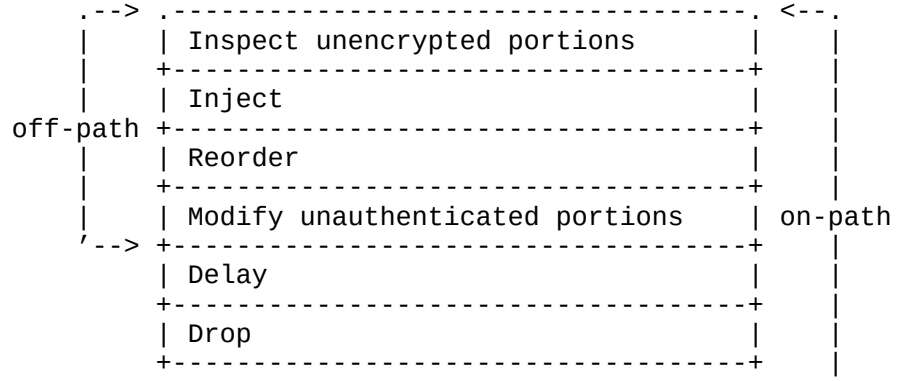
8.1. Attacker Model

Two classes of attackers are considered, off-path and on-path, with increasing capabilities (see Figure 4). The following descriptions of these attackers are based on those introduced in QUIC (Section 21.1 of [RFC9000]):

* An off-path attacker is not on the original path between the DTLS peers, but it is able to observe packets on the original path and has a faster forwarding path compared to the DTLS peers, which allows it to make copies of the observed packets, race its copies to either peer, and consistently win the race.

* An on-path attacker is on the original path between the DTLS peers and is therefore capable, compared to the off-path attacker, to also drop and delay records at will.

Note that, in general, attackers cannot craft DTLS records in a way that would successfully pass verification, due to the cryptographic protections applied by the DTLS record layer.



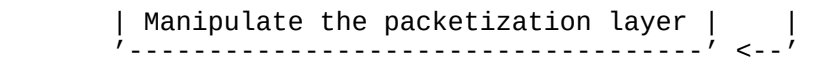


Figure 4: Attacker Capabilities

RRC is designed to defend against the following attacks:

* On-path and off-path attackers that try to create an amplification attack by spoofing the source address of the victim (Section 8.1.1).

* Off-path attackers that try to put themselves on-path (Section 8.1.2), provided that the enhanced path validation algorithm is used (Section 5.2).

8.1.1. Amplification

Both on-path and off-path attackers can send a packet (either by modifying it on the fly or by copying, injecting, and racing it, respectively) with the source address modified to that of a victim host. If the traffic generated by the server in response is larger compared to the received packet (e.g., a CoAP server returning an MTU's worth of data from a 20-byte GET request [AMP-ATTACKS]), the attacker can use the server as a traffic amplifier toward the victim.

8.1.1.1. Mitigation Strategy

When receiving a packet with a known CID that has a source address different from the one currently associated with the DTLS connection, an RRC-capable endpoint will not send a (potentially large) response but instead a small path_challenge message to the victim host. Since the host is not able to decrypt it and generate a valid path_response, the address validation fails, which in turn keeps the original address binding unaltered.

Note that in the case of an off-path attacker, the original packet still reaches the intended destination; therefore, an implementation could use a different strategy to mitigate the attack.

8.1.2. Off-Path Packet Forwarding

An off-path attacker that can observe packets might forward copies of genuine packets to endpoints over a different path. If the copied packet arrives before the genuine packet, this will appear as a path change, like in a genuine NAT rebinding occurrence. Any genuine packet will be discarded as a duplicate. If the attacker is able to continue forwarding packets, it might be able to cause migration to a path via the attacker. This places the attacker on-path, giving it the ability to observe or drop all subsequent packets.

This style of attack relies on the attacker using a path that has the same or better characteristics (e.g., due to a more favourable service level agreements) as the direct path between endpoints. The attack is more effective if relatively few packets are sent or if packet loss coincides with the attempted attack.

A data packet received on the original path that increases the maximum received packet number will cause the endpoint to move back to that path. Therefore, eliciting packets on this path increases the likelihood that the attack is unsuccessful. However, note that, unlike QUIC, DTLS has no "non-probing" packets so this would require application-specific mechanisms.

8.1.2.1. Mitigation Strategy

Figure 5 illustrates the case where a receiver receives a packet with

a new source address. In order to determine that this path change was not triggered by an off-path attacker, the receiver will send an RRC message of type path_challenge (1) on the old path.

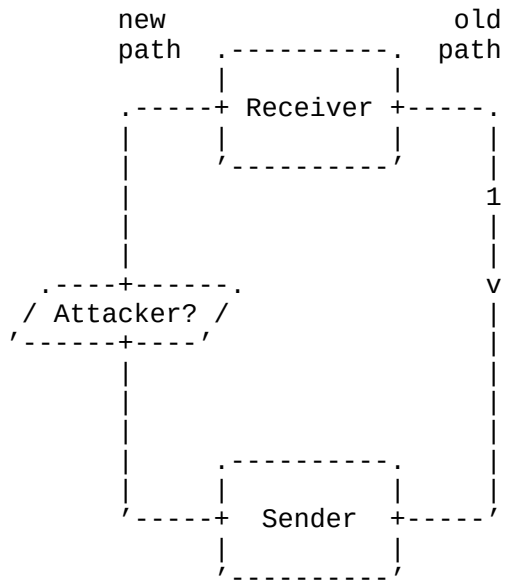


Figure 5: Off-Path Packet Forwarding Scenario

Three cases need to be considered:

Case 1: The old path is dead (e.g., due to a NAT rebinding), which leads to a timeout of (1).

As shown in Figure 6, a path_challenge (2) needs to be sent on the new path. If the sender replies with a path_response on the new path (3), the switch to the new path is considered legitimate.

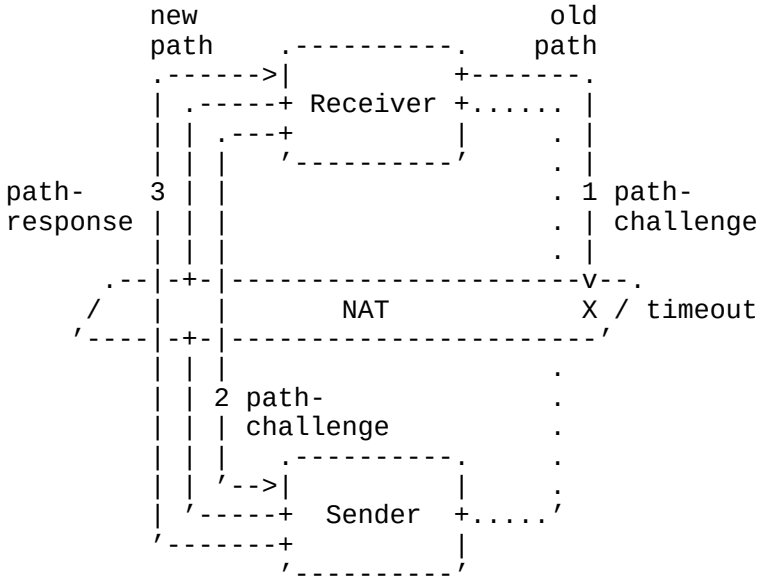


Figure 6: Old Path Is Dead

Case 2: The old path is alive but not preferred.

This case is shown in Figure 7 whereby the sender replies with a path_drop message (2) on the old path. This triggers the receiver to send a path_challenge (3) on the new path. The sender will reply with a path_response (4) on the new path, thus providing confirmation for the path migration.

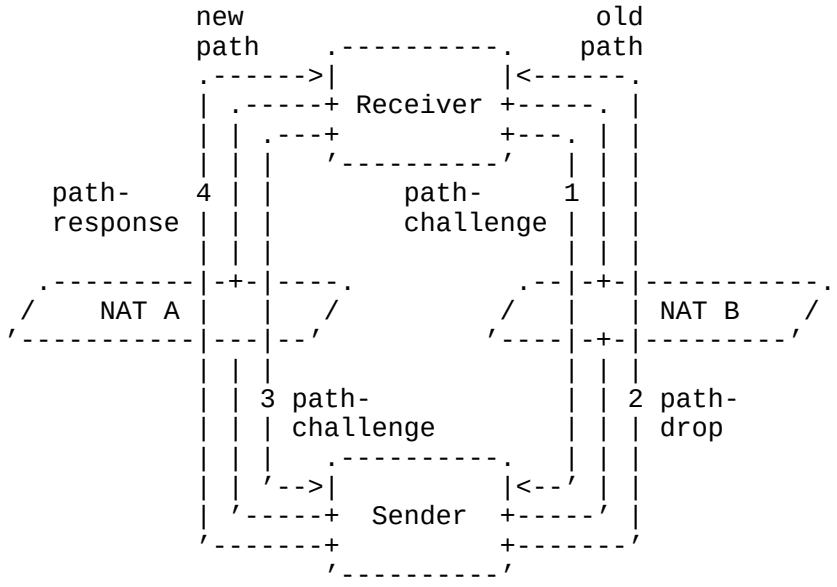


Figure 7: Old Path Is Not Preferred

Case 3: The old path is alive and preferred.

This is most likely the result of an off-path attacker trying to place itself on-path. As shown in Figure 8, the receiver sends a path_challenge (1) on the old path, and the sender replies with a path_response (2) on the old path. This results in the connection not being migrated to the new path, thus thwarting the attack.

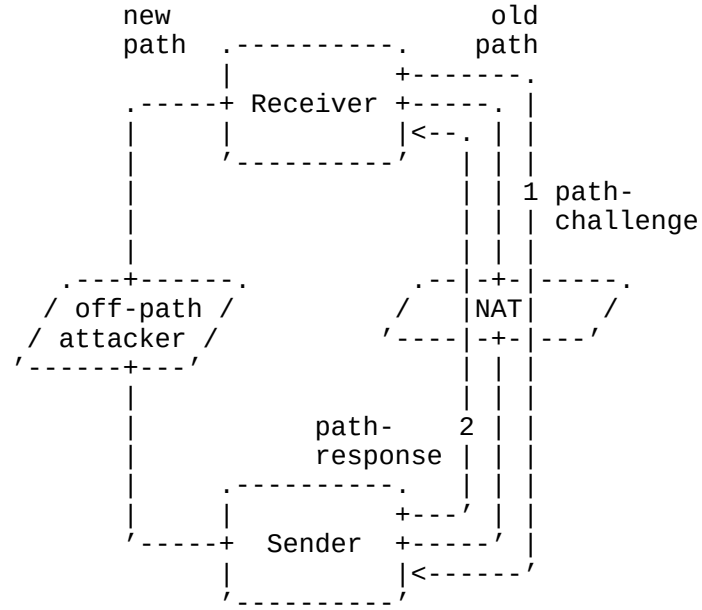


Figure 8: Old Path Is Preferred

Note that this defense is imperfect, but this is not considered a serious problem. If the path via the attacker is reliably faster than the old path despite multiple attempts to use that old path, it is not possible to distinguish between an attack and an improvement in routing.

An endpoint could also use heuristics to improve detection of this style of attack. For instance, NAT rebinding is improbable if packets were recently received on the old path. Endpoints can also look for duplicated packets. Conversely, a change in CID is more

likely to indicate an intentional migration rather than an attack. Note that changes in CIDs are supported in DTLS 1.3 but not in DTLS 1.2.

9. Privacy Considerations

When using DTLS 1.3, peers SHOULD avoid using the same CID on multiple network paths, in particular when initiating connection migration or when probing a new network path, as described in Section 5, as an adversary can otherwise correlate the communication interaction across those different paths. DTLS 1.3 provides mechanisms to ensure that a new CID can always be used. In general, an endpoint should proactively send a RequestConnectionId message to ask for new CIDs as soon as the pool of spare CIDs is depleted (or goes below a threshold). Also, in case a peer might have exhausted available CIDs, a migrating endpoint could include NewConnectionId in packets sent on the new path to make sure that the subsequent path validation can use fresh CIDs.

Note that DTLS 1.2 does not offer the ability to request new CIDs during the session lifetime since CIDs have the same lifespan of the connection. Therefore, deployments that use DTLS in multihoming environments SHOULD refuse to use CIDs with DTLS 1.2 and switch to DTLS 1.3 if the correlation privacy threat is a concern.

10. IANA Considerations

10.1. New TLS ContentType

IANA has allocated an entry in the "TLS ContentType" registry within the "Transport Layer Security (TLS) Parameters" registry group [IANA.tls-parameters] for the return_routability_check (27) message defined in this document. IANA set the DTLS_OK column to "Y" and added the following note to the registry:

| Note: The return_routability_check content type is only applicable
| to DTLS 1.2 and 1.3.

10.2. New TLS ExtensionType

IANA has allocated the extension code point (61) for rrc in the "TLS ExtensionType Values" registry as described in Table 1.

Value	Extension Name	TLS 1.3	DTLS-Only	Recommended	Reference	Comment
61	rrc	CH, SH	Y	N	RFC 9853	

Table 1: New Entry in the TLS ExtensionType Values Registry

10.3. New "TLS RRC Message Type" Registry

IANA has created the "TLS RRC Message Types" registry within the "Transport Layer Security (TLS) Parameters" registry group [IANA.tls-parameters]. This registration procedure is "Expert Review" (Section 4.5 of [RFC8126]).

To submit registration requests, follow the procedures in Section 16 of [RFC9847].

Each entry in the registry must include the following fields:

Value:

A (decimal) number in the range 0 to 253.

Description:
A brief description of the RRC message.

DTLS-Only:
Indication of whether the message only applies to DTLS. Since RRC is only available in DTLS, this column is set to "Y" for all the initial entries in this registry. Future work may define new RRC message types that also apply to TLS.

Recommended:
Indication of whether the message is recommended for implementations to support. The semantics for this field is defined in Section 5 of [RFC8447] and updated in Section 3 of [RFC9847].

Reference:
A reference to a publicly available specification for the value.

Comment:
Any relevant notes or comments that relate to this entry.

Table 2 shows the initial contents of this registry:

Value	Description	DTLS-Only	Recommended	Reference	Comment
0	path_challenge	Y	Y	RFC 9853	
1	path_response	Y	Y	RFC 9853	
2	path_drop	Y	Y	RFC 9853	
3-253	Unassigned				
254-255	Reserved for Private Use	Y		RFC 9853	

Table 2: Initial Entries in TLS RRC Message Type Registry

IANA added the following note to provide additional information regarding the use of RRC message codepoints in experiments:

> Note: As specified in [RFC8126], assignments made in the Private Use space are not generally useful for broad interoperability. Those making use of the Private Use range are responsible for ensuring that no conflicts occur within the intended scope of use. For widespread experiments, provisional registrations (Section 4.13 of [RFC8126]) are available.

10.3.1. Designated Expert Instructions

To enable a broadly informed review of registration decisions, it is recommended that multiple designated experts be appointed to represent the perspectives of both the transport and security areas.

In cases where a registration decision could be perceived as creating a conflict of interest for a particular expert, that expert SHOULD defer to the judgment of the other experts.

11. References

11.1. Normative References

[IANA.tls-parameters]
IANA, "Transport Layer Security (TLS) Parameters", <https://www.iana.org/assignments/tls-parameters>.

[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>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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>.

[RFC8447] Salowey, J. and S. Turner, "IANA Registry Updates for TLS and DTLS", RFC 8447, DOI 10.17487/RFC8447, August 2018, <https://www.rfc-editor.org/info/rfc8447>.

[RFC9146] Rescorla, E., Ed., Tschofenig, H., Ed., Fossati, T., and A. Kraus, "Connection Identifier for DTLS 1.2", RFC 9146, DOI 10.17487/RFC9146, March 2022, <https://www.rfc-editor.org/info/rfc9146>.

[RFC9147] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/info/rfc9147>.

[RFC9847] Salowey, J. and S. Turner, "IANA Registry Updates for TLS and DTLS", RFC 9847, DOI 10.17487/RFC9847, December 2025, <https://www.rfc-editor.org/info/rfc9847>.

11.2. Informative References

[AMP-ATTACKS]
Preu Mattsson, J., Selander, G., and C. Amsss, "Amplification Attacks Using the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-irtf-t2trg-amplification-attacks-05, 18 June 2025, <https://datatracker.ietf.org/doc/html/draft-irtf-t2trg-amplification-attacks-05>.

[IOT-PROFILE]
Tschofenig, H., Fossati, T., Richardson, M., and D. Migault, "TLS/DTLS 1.3 Profiles for the Internet of Things", Work in Progress, Internet-Draft, draft-ietf-uta-tls13-iot-profile-19, 20 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-uta-tls13-iot-profile-19>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[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>.

[RFC9175] Amsss, C., Preu Mattsson, J., and G. Selander, "Constrained Application Protocol (CoAP): Echo, Request-Tag, and Token Processing", RFC 9175, DOI 10.17487/RFC9175, February 2022, <https://www.rfc-editor.org/info/rfc9175>.

Acknowledgments

We would like to thank Colin Perkins, Deb Cooley, Eric Rescorla, ric Vyncke, Erik Kline, Hanno Becker, Hanno Bck, Joe Clarke, Manuel Pgouri-Gonnard, Marco Tiloca, Martin Thomson, Mike Bishop, Mike Ounsworth, Mohamed Boucadair, Mohit Sahni, Rich Salz, Russ Housley, Sean Turner, and Yaron Sheffer for their input to this document.

Authors' Addresses

Hannes Tschofenig (editor)
University of the Bundeswehr Munich
Institute of Distributed Intelligent Systems
Werner-Heisenberg-Weg 39
85577 Neubiberg
Germany
Email: Hannes.Tschofenig@gmx.net

Achim Kraus
Email: achimkraus@gmx.net

Thomas Fossati
Linaro
Email: thomas.fossati@linaro.org