Network Working Group
Internet-Draft
Updates: 6126bis (if approved)
Intended status: Standards Track
Expires: August 10, 2019

A. Decimo
IRIF, University of Paris-Diderot
D. Schinazi
Google LLC
J. Chroboczek
IRIF, University of Paris-Diderot

February 6, 2019

Babel Routing Protocol over Datagram Transport Layer Security
draft-ietf-babel-dtls-04

Abstract

The Babel Routing Protocol does not contain any means to authenticate neighbours or protect messages sent between them. This documents specifies a mechanism to ensure these properties, using Datagram Transport Layer Security (DTLS). This document updates RFC 6126bis.

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 August 10, 2019.

Table of Contents

1. Introduction

The Babel Routing Protocol [RFC6126bis] does not contain any means to authenticate neighbours or protect messages sent between them. Because of this, an attacker is able to send maliciously crafted Babel messages which could lead a network to route traffic to an attacker or to an under-resourced target causing denial of service. This documents specifies a mechanism to prevent such attacks, using Datagram Transport Layer Security (DTLS) [RFC6347].

1.1. Specification of Requirements

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.

1.2. Applicability

The protocol described in this document protects Babel packets with DTLS. As such, it inherits the features offered by DTLS, notably authentication, integrity, replay protection, confidentiality and asymmetric keying. It is therefore expected to be applicable in a wide range of environments.

There exists another mechanism for securing Babel, namely Babel HMAC authentication [BABEL-HMAC]. HMAC only offers basic features, namely authentication, integrity and replay protection with a small number of symmetric keys.

Since HMAC authentication is simpler, requires fewer changes to the Babel protocol, and avoids a dependency on DTLS, its use is RECOMMENDED in deployments where both protocols are equally applicable.

2. Operation of the Protocol

Babel over DTLS requires some changes to how Babel operates. First, DTLS is a client-server protocol, while Babel is a peer-to-peer protocol. Second, DTLS can only protect unicast communication, while Babel packets can be sent over to both unicast and multicast destinations.

2.1. DTLS Connection Initiation

Babel over DTLS operates on a different port than unencrypted Babel. All Babel over DTLS nodes MUST act as DTLS servers on a DTLS port, and MUST listen for unencrypted Babel traffic on an unencrypted port, which MUST be distinct from the DTLS port. The default port for Babel over DTLS is registered with IANA as the "babel-dtls" port (UDP port TBD, see Section 4), and the unencrypted port is registered as the "babel" port (UDP port 6696). Nodes SHOULD use these default ports.

When a Babel node discovers a new neighbor (generally by receiving an unencrypted multicast Babel packet), it compares the neighbour's IPv6 link-local address with its own, using network byte ordering. If a node's address is lower than the recently discovered neighbor's address, it acts as a client and connects to the neighbor. In other words, the node with the lowest address is the DTLS client for this pairwise relationship. As an example, fe80::1:2 is considered lower than fe80::2:1.

The node acting as DTLS client initiates its DTLS connection from an ephemeral UDP port. Nodes SHOULD ensure that new client DTLS

connections use different ephemeral ports from recently used connections to allow servers to differentiate between the new and old DTLS connections. Alternatively, nodes MAY use DTLS connection identifiers [DTLS-CID] as a higher-entropy mechanism to distinguish between connections.

When a node receives a new DTLS connection, it MUST verify that the source IP address is an IPv6 link-local address; if it is not, it MUST reject the connection. Nodes use mutual authentication (authenticating both client and server); servers MUST send a CertificateRequest message and subsequently authenticate the client. Implementations MUST support authenticating peers against a local store of credentials. If either node fails to authenticate its peer against its local policy, it MUST abort the DTLS handshake. Nodes MUST only negotiate DTLS version 1.2 or higher. Nodes MUST use DTLS replay protection to prevent attackers from replaying stale information. Nodes SHOULD drop packets that have been reordered by more than several IHU intervals, to avoid letting attackers make stale information last longer.

2.2. Protocol Encoding

Babel over DTLS sends all unicast Babel packets protected by DTLS. The entire Babel packet, from the Magic byte at the start of the Babel header to the last byte of the Babel packet trailer, is sent protected by DTLS.

2.3. Transmission

When sending packets, Babel over DTLS nodes MUST NOT send any TLVs over the unprotected "babel" port, with the exception of Hello TLVs without the Unicast flag set. Babel over DTLS nodes MUST NOT send any unprotected unicast packets. This ensures the confidentiality of the information sent in Babel packets (e.g. the network topology) by only sending it encrypted by DTLS. Unless some out-of-band neighbor discovery mechanism is available, nodes SHOULD periodically send unprotected multicast Hellos to ensure discovery of new neighbours. In order to maintain bidirectional reachability, nodes can either rely entirely on unprotected multicast Hellos, or send protected unicast Hellos in addition to the multicast Hellos.

Since Babel over DTLS only protects unicast packets, implementors may implement Babel over DTLS by modifying an implementation of Babel without DTLS support, and replacing any TLV previously sent over multicast with a separate TLV sent over unicast for each neighbour. TLVs previously sent over multicast can be replaced with the same contents over unicast, with the exception of Hellos as described above. Some implementations could also change the contents of IHU

TLVs when converting to unicast in order to remove redundant information.

2.4. Reception

Babel over DTLS nodes can receive Babel packets either protected over a DTLS connection, or unprotected directly over the "babel" port. To ensure the security properties of this mechanism, unprotected packets are treated differently. Nodes MUST silently ignore any unprotected packet sent over unicast. When parsing an unprotected packet, a node MUST silently ignore all TLVs that are not of type Hello. Nodes MUST also silently ignore any unprotected Hello with the Unicast flag set. Note that receiving an unprotected packet can still be used to discover new neighbors, even when all TLVs in that packet are silently ignored.

2.5. Neighbour table entry

It is RECOMMENDED for nodes to associate the state of their DTLS connection with their neighbour table. When a neighbour entry is flushed from the neighbour table (Appendix A of [RFC6126bis]), its associated DTLS state SHOULD be discarded. The node SHOULD send a DTLS close_notify alert to the neighbour if it believes the link is still viable.

While DTLS provides protection against an attacker that replays valid packets, DTLS is not able to detect when an active on-path attacker intercepts valid packets and resends them at a later time. This attack could be used to make a node believe it has bidirectional reachability to a neighbour even though that neighbour has disconnected from the network. To prevent this attack, nodes MUST discard the DTLS state associated with a neighbour after a finite time of not receiving valid DTLS packets. This can be implemented by, for example, discarding a neighbour's DTLS state when its associated IHU timer fires. Note that relying solely on the receipt of Hellos is not sufficient as multicast Hellos are sent unprotected.

2.6. Simultaneous operation of both Babel over DTLS and unprotected Babel

Implementations MAY implement both Babel over DTLS and unprotected Babel. However, accepting unprotected Babel packets (other than multicast Hellos) loses the security properties of Babel over DTLS. A node MAY allow configuration options to allow unprotected Babel on some interfaces but not others; this effectively gives nodes on that interface the same access as authenticated nodes, and SHOULD NOT be done unless that interface has a mechanism to authenticate nodes at a lower layer (e.g. IPsec).

3. Interface Maximum Transmission Unit Issues

Compared to unprotected Babel, DTLS adds header, authentication tag and possibly block-size padding overhead to every packet. This reduces the size of the Babel payload that can be carried. This document does not relax the packet size requirements in Section 4 of [RFC6126bis], but recommends that DTLS overhead be taken into account when computing maximum packet size.

More precisely, nodes SHOULD compute the overhead of DTLS depending on the ciphers in use, and SHOULD NOT send Babel packets larger than the interface maximum transmission unit (MTU) minus the overhead of IP, UDP and DTLS. Nodes MUST NOT send Babel packets larger than the attached interface's MTU adjusted for known lower-layer headers (at least UDP and IP) or 512 octets, whichever is larger, but not exceeding 2^16 - 1 adjusted for lower-layer headers. Every Babel speaker MUST be able to receive packets that are as large as any attached interface's MTU adjusted for UDP and IP headers or 512 octets, whichever is larger. Note that this requirement on reception does not take into account the overhead of DTLS because the peer may not have the ability to compute the overhead of DTLS and the packet may be fragmented by lower layers.

4. IANA Considerations

If this document is approved, IANA is requested to register a UDP port number, called "babel-dtls", for use by Babel over DTLS. The IANA registry will include a reference to this document.

5. Security Considerations

Confidential interaction between two Babel peers requires Datagram Transport Layer Security (DTLS) with a cipher suite offering confidentiality protection. The guidance given in [RFC7525] MUST be followed to avoid attacks on DTLS.

A malicious client might attempt to perform a high number of DTLS handshakes with a server. As the clients are not uniquely identified by the protocol and can be obfuscated with IPv6 temporary addresses, a server needs to mitigate the impact of such an attack. Such mitigation might involve rate limiting handshakes from a given subnet or more advanced denial of service avoidance techniques beyond the scope of this document.

6. References

6.1. Normative References

[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>.

[RFC6126bis] Chroboczek, J. and D. Schinazi, "The Babel Routing Protocol", Internet Draft draft-ietf-babel-rfc6126bis-07, November 2018.

[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>.

[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>.

6.2. Informative References

[BABEL-HMAC] Do, C., Kolodziejak, W., and J. Chroboczek, "Babel Cryptographic Authentication", Internet Draft draft-ietf-babel-hmac-03, November 2018.

[DTLS-CID] Rescorla, E., Tschofenig, H., Fossati, T., and T. Gondrom, "Connection Identifiers for DTLS 1.2", Internet Draft draft-ietf-tls-dtls-connection-id-02, October 2018.

[RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <https://www.rfc-editor.org/info/rfc7250>.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <https://www.rfc-editor.org/info/rfc7525>.

[RFC7918] Langley, A., Modadugu, N., and B. Moeller, "Transport Layer Security (TLS) False Start", RFC 7918, DOI 10.17487/RFC7918, August 2016, <https://www.rfc-editor.org/info/rfc7918>.

[RFC7924] Santesson, S. and H. Tschofenig, "Transport Layer Security (TLS) Cached Information Extension", RFC 7924, DOI 10.17487/RFC7924, July 2016, <https://www.rfc-editor.org/info/rfc7924>.

[RFC8094] Reddy, T., Wing, D., and P. Patil, "DNS over Datagram Transport Layer Security (DTLS)", RFC 8094, DOI 10.17487/RFC8094, February 2017, <https://www.rfc-editor.org/info/rfc8094>.

Appendix A. Performance Considerations

To reduce the number of octets taken by the DTLS handshake, especially the size of the certificate in the ServerHello (which can be several kilobytes), Babel peers can use raw public keys [RFC7250] or the Cached Information Extension [RFC7924]. The Cached Information Extension avoids transmitting the server's certificate and certificate chain if the client has cached that information from a previous TLS handshake. TLS False Start [RFC7918] can reduce round trips by allowing the TLS second flight of messages (ChangeCipherSpec) to also contain the (encrypted) Babel packet.

Appendix B. Acknowledgments

The authors would like to thank Donald Eastlake, Thomas Fossati, Gabriel Kerneis, Antoni Przygienda, Barbara Stark, Markus Stenberg, Dave Taht, Martin Thomson, and Sean Turner for their input and contributions. The performance considerations in this document were inspired from the ones for DNS over DTLS [RFC8094].

Authors' Addresses

Antonin Decimo
IRIF, University of Paris-Diderot
Paris
France

Email: antonin.decimo@gmail.com

David Schinazi
Google LLC
1600 Amphitheatre Parkway
Mountain View, California 94043
USA

Email: dschinazi.ietf@gmail.com

Juliusz Chroboczek
IRIF, University of Paris-Diderot
Case 7014
75205 Paris Cedex 13
France

Email: jch@irif.fr