Network Working Group
Internet-Draft
Updates: 6126bis (if approved)
Intended status: Standards Track
Expires: April 11, 2019

A. Decimo
IRIF, University of Paris-Diderot
D. Schinazi
Apple Inc.
J. Chroboczek
IRIF, University of Paris-Diderot

October 8, 2018

Babel Routing Protocol over Datagram Transport Layer Security
draft-ietf-babel-dtls-01

Abstract

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

Status of This Memo

Copyright Notice

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 describes 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 current two main mechanisms for securing Babel are Babel over DTLS (as described in this document) and Babel Cryptographic Authentication [BabelHMAC]. The latter has the advantages of being simpler and not requiring a dependency on DTLS, therefore implementers are encouraged to consider it in preference to the

mechanism defined in this document whenever both are applicable to a given deployment. Both mechanisms ensure integrity of messages and prevent message replay.

However, DTLS offers several features that are not provided by Babel Cryptographic Authentication, therefore Babel over DTLS is applicable in cases where those features are needed. Examples of such features include:

- o Asymmetric keys. DTLS allows authentication via asymmetric keys, which allows a finer granularity of trust per-peer, and allows for revocation.

- o Confidentiality of data. DTLS encrypts payloads, preventing an on-link attacker from observing the routing table.

2. Operation of the Protocol

Babel over DTLS requires changes to how Babel is operated, for two reasons. Firstly, because DTLS introduces the concepts of client and server, while Babel is a peer-to-peer protocol. Secondly, DTLS can only protect unicast, while Babel TLVs can be sent over both unicast and multicast.

2.1. DTLS Connection Initiation

All Babel over DTLS nodes MUST act as DTLS servers on the "babel-dtls" port (UDP port TBD), and MUST listen for multicast traffic on the unencrypted "babel" port (UDP port 6696). 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. When a node receives a new DTLS connection, it MUST verify the source IP address, and reject the connection if the address is not an IPv6 link-local address.

2.2. Protocol Encoding

Babel over DTLS sends all unicast Babel packets encrypted 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 packet. 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 on unprotected multicast Hellos, or also send protected unicast Hellos.

Since Babel over DTLS only protects unicast packets, implementors may implement Babel over DTLS by modifying an unprotected implementation of Babel, and replacing any TLV sent over multicast with a separate TLV sent over unicast for each neighbour.

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 MAY send a DTLS close_notify alert to the neighbour.

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. 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 lower layers (IP, UDP and DTLS). This helps reduce the likelihood of lower-layer fragmentation which would negatively impact performance and reliability. 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. Babel packets MUST NOT be sent in IPv6 Jumbograms.

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.

5. Security Considerations

The 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. The DTLS client SHOULD use the TLS Certificate Status Request extension (Section 8 of [RFC6066]).

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 IPv4 address sharing and 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-05, May 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

[BabelHMAC] Do, C., Kolodziejak, W., and J. Chroboczek, "Babel Cryptographic Authentication", Internet Draft draft-ietf-babel-hmac-00, August 2018.

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <https://www.rfc-editor.org/info/rfc6066>.

[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.

These performance considerations 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
Apple Inc.
One Apple Park Way
Cupertino, California 95014
USA

Email: dschinazi@apple.com

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

Email: jch@irif.fr