Let’s make Babel secure

Juliusz Chroboczek
IRIF, University of Paris-Diderot

17 November 2017

Disclaimer and acknowledgements

None of the ideas in this talk are mine.

Ideas explained to me at different times by:

- Markus Stenberg,
- Denis Ovsienko,
- Toke Høiland-Jørgensen,
- David Schinazi, and
- Antonin Décimo.

If anything is wrong — blame them, not me.

Babel is vulnerable to spoofing

Babel announce (Update):
"I can route packets to that destination".

If you spoof an update for the IETF's prefix,
you can pass yourself for `www.ietf.fr`.

You need to use either of:

- lower metric;
- higher seqno;
- longer prefix.

Babel is vulnerable to replay

Even if you cannot craft a fake Update,
Babel is vulnerable to replay:

- capture a properly formatted/authentified Update;
- wait until it is legal to send it again;
- resend it!

To be suitable for Babel,
a security mechanism needs to protect against replay.

Hop-to-hop, not end-to-end

Limiting the scope

Two approaches to authentication:

- end-to-end: an announcement is authentified by the originator, verified by the destination;
- hop-to-hop: the communcation between neighbours is authentified, any properly authentified node can spoof any data.

End-to-end is a stronger form of authentication, but very difficult. Therefore, out of scope for now.

(But don’t hesitate to experiment with end-to-end approaches.)

Current status

Up to now, our users have used two techniques:

- lower-layer security:
  - physically secure Ethernets;
  - radio links protected by WPA2;
  - VPNs (notably OpenVPN).
- HMAC-based authentication using RFC 7298.

With Babel aiming for Standard Track status, we need to define one or more security mechanisms, and declare one strongly recommended.

Approaches to hop-to-hop security

Serious contenders for strongly recommended status:

- HMAC + replay protection
  (cf. RFCs 2328/5709/7474 and 7298);
- DTLS with Babel over unicast.

Other approaches:

- lower-layer security (VPN, WPA2, etc.);
- dynamically-keyed IPsec with Babel over unicast;
- plain-text password;
- others?

HMAC + replay protection

Denis Ovsienko designed and implemented RFC 7298:

- HMAC-based integrity+authentication;
- algorithm flexibility, two MTI algorithms;
- replay protection :
  - doesn't require persistent storage;
  - doesn't require hardware clocks.

RFC 7298 is:

- reasonably easy to implement;
- requires a new security mechanism.

Great if you don't already have a security stack.

Unfortunately, RFC 7298 has a (rather subtle) flaw, described at IETF-99.

DTLS with Babel over Unicast

- Use multicast for discovery only,
  unicast for all the rest of the protocol;
- protect unicast traffic using DTLS.

DLTS is:

- difficult to implement from scratch;
- an already existing security mechanism.

Great if you already have a DTLS implementation,
horrible if you need to reimplement from scratch.

Explained to me (at different times) by Markus Stenberg, Toke Høiland-Jørgensen and David Schinazi.

Babel over Unicast

Original plan (back in 2010):

- any TLV can be sent over either unicast or multicast and has the same meaning.

But:

1. Hellos can only be sent over multicast;
2. Acks can only be sent over unicast.

RFC 6126bis fixes point 1: all Babel TLVs can now be sent over unicast. It is possible to implement Babel so that multicast is only used for discovery.

- possible to use a unicast-only security mechanism;
- some people have an irrational dislike of multicast.

DTLS with Babel over Unicast

Implementation

Design and implementation started by Antonin Décimo in July 2017. He got to the interesting bits, at which point he ran out of summer.

Antonin identified a number of tricky points:

- which TLVs are allowed in uprotected packets?
- DTLS is client-server, Babel is peer-to-peer;
- DTLS libraries want connected sockets, babeld uses unconnected sockets;
- babeld's buffer management handles unicast inefficiently.

DTLS with Babel over Unicast

Implementation issues

Preliminary ideas (implementation not complete):

- which TLVs are allowed in uprotected packets?
  ⟶ Hello only, vulnerable to DoS?;
- DTLS is client-server, Babel is peer-to-peer
  ⟶ after discovery, smaller router-id is client
  (what about out-of-band discovery?);
- DTLS libraries want connected sockets, babeld uses unconnected sockets
  ⟶ use in-memory buffers;
- babeld's buffer management handles unicast inefficiently
  ⟶ restructure buffer management.

Conclusion

What do we want to do?

Strawman proposal:

- produce a revision of RFC 7298 (HMAC security) that solves the issues with the current version and implement it;
- produce two interoperable implementations of DTLS with Babel over unicast and write up the protocol;
- publish both, recommend one (not necessarily MTI);
- experiment with other approaches?