Babel over DTLS

Security in babeld

Antonin Décimo
Joint work with Juliusz Chroboczek

Paris Diderot University

March 22, 2018

Security for Babel

We need security for Babel.

- lower-layer security (WPA2, OpenVPN, physical security)
- Babel security
  - Babel HMAC Cryptographic Authentication [RFC7298]
  - **Babel over DTLS** (TLS for datagrams) (*this talk*)

Why DTLS?

- Authentication and confidentiality (protocol & implementation) are somebody else's problem
- Asymmetric keys
- **Authentication** & Confidentiality

Results

We use mbedTLS with babeld. We have a **working prototype!**

No.	Source	Destination	Protocol	Length	Info
9	fe80::b2d5:…	ff02::1:6	Babel	74	Babel hello
10	fe80::eca3:…	ff02::1:6	Babel	90	Babel hello ihu
11	fe80::b2d5:…	fe80::eca3:…	DTLSv1.2	481	Client Hello
12	fe80::b2d5:…	ff02::1:6	Babel	90	Babel hello ihu
13	fe80::eca3:…	fe80::b2d5:…	DTLSv1.2	122	Hello Verify Request
14	fe80::eca3:…	ff02::1:6	Babel	106	Babel hello ihu ihu
15	fe80::eca3:…	ff02::1:6	Babel	90	Babel hello ihu
16	fe80::b2d5:…	ff02::1:6	Babel	90	Babel hello ihu
17	fe80::eca3:…	ff02::1:6	Babel	90	Babel hello ihu
18	fe80::b2d5:…	fe80::eca3:…	DTLSv1.2	513	Client Hello
19	fe80::eca3:…	fe80::b2d5:…	DTLSv1.2	174	Server Hello
20	fe80::eca3:…	fe80::b2d5:…	DTLSv1.2	484	Server Key Exchange
21	fe80::eca3:…	fe80::b2d5:…	DTLSv1.2	87	Server Hello Done
22	fe80::b2d5:…	fe80::eca3:…	DTLSv1.2	221	Client Key Exchange
23	fe80::b2d5:…	fe80::eca3:…	DTLSv1.2	76	Change Cipher Spec
24	fe80::b2d5:…	fe80::eca3:…	DTLSv1.2	123	Encrypted Handshake Message
25	fe80::eca3:…	fe80::b2d5:…	DTLSv1.2	76	Change Cipher Spec
26	fe80::eca3:…	fe80::b2d5:…	DTLSv1.2	123	Encrypted Handshake Message
27	fe80::eca3:…	ff02::1:6	Babel	106	Babel hello ihu ihu
28	fe80::b2d5:…	ff02::1:6	Babel	106	Babel hello ihu ihu
29	fe80::b2d5:…	fe80::eca3:…	DTLSv1.2	112	Application Data
30	fe80::eca3:…	fe80::b2d5:…	DTLSv1.2	112	Application Data

Discovery

Protected Babel

User interface

babeld configuration file

```
default unicast true
default dtls true
cert-file         ~/cert.pem
private-key-file ~/pkey.pem
cacert-file       ~/cacert.pem
private-key-password 1234
```

Babel + DTLS

Babel is based on UDP, uses unicast and multicast, and is a pure peer-to-peer protocol. The same port (6696) is used for source and destination.
babeld uses a lot multicast, but DTLS can only protect unicast.

1. Juliusz rewrote the buffering mechanism in babeld.
2. Unicast is independent from DTLS.
3. We can protect Babel.

Routing information is protected.
Neighbour discovery and link-quality estimation packets (Hellos & IHU) remain unprotected.

Handshake Asymmetry — Prototype

The DTLS handshake is asymmetric, whereas Babel is symmetric. We have to break the symmetry. Classic technique: the peer with the **lowest link-local** address becomes the DTLS handshake server.

Packet Reception 1 — Prototype

Babel structure is pure peer-to-peer. We would like to **preserve this structure** with Babel-over-DTLS.

- Babel & DTLS traffic is received on the same socket
- We need to differentiate the packets
  $\rightarrow$ The DTLS library can do that for us

Packet Reception 2 — Prototype

insecure: we ignore all TLVs except Hello/IHU.

1. Babel & DTLS traffic is received on the same socket.
2. We try to decrypt the packet.
3. - If we succeed, we tag it as secure.
   - If we fail, we tag is as insecure.
4. We parse the packet.

Multicast is insecure by default.
This behaviour is interleaved with the DTLS handshake.

Packet Emission — Prototype

- All unicast packets are protected
- All multicast packets are sent in the clear
  $\rightarrow$ only Hello/IHU TLVs

Other Approaches

- Pure peer-to-peer on another port.
- Classic client-server model.
- Sub-TLV encapsulating protected data.
  Not a serious proposal.

2 bits of disagreement:

- Is the **server port** the same as the **Babel port**?
- Is the **client port** the same as the **server port**?

What's next?

1. Is parsing insecure packets a good idea?
2. What if a peer reboots after a successful DTLS handshake?
   - Use same port and rely on the SHOULD in DTLS[1] — most implementations don't
   - Use different ports.
   - Client or DTLS lib hacking...
3. PKey/Certificate installation & rollover? PKey password?
4. Will the DTLS overhead cause fragmentation?

- Babel is protected by DTLS.
  We have a running implementation that protects data but not discovery.
  Available soon at `https://github.com/jech/babeld`.

[1] DTLS RFC6347 section-4.2.8