Box

Transport Agnostic Protocol for Encrypted Communication

Alexander Færøy

The Camp 2015

July 22, 2015

Table of Contents

Cryptographic Primitives

Define a set of primitives that we will use throughout the system.

Cryptographic Primitives

ed25519

ed25519 is a public key signature scheme, used for long-term and short-term identity key pairs.

Cryptographic Primitives

curve25519

curve25519 is a Diffie-Hellman function used to compute a shared secret.

Cryptographic Primitives

ed25519 and curve25519

It is possible to create a deterministic curve25519 key pair from an ed25519 key pair.

Cryptographic Primitives

Sodium

We use `libsodium`, which contains all the primitives we need thus completely avoid any of the stupid OpenSSL variants.

Cryptographic Primitives

Sodium

We may have to get another library in to implement the Socialist Millionaire Protocol, but we are hoping that researchers will come up with something more simple than the current SMP algorithm.

Identity

- The identity keys are all ed25519.
- From this key, we can derive our curve25519 key, which is used for ECDH.
- Are stored encrypted on disk, encrypted with a passphrase.

Passphrase Identities

- Security minded people are using systems like Tails, where you want to avoid persistent storage.
- Use the human brain to store the information necessary to consistently (re)create the Box identity.
- Define a set of profiles to avoid requiring the human brain to store more data than necessary to (re)create the Box identity. Each profile have its own static key to harden the system against brute force attacks.

Passphrase Identities

Profiles

2015.desktop
: Uses Colin Percival's *scrypt*() function together with Daniel J. Bernstein's Salsa20 stream cipher and SHA256.
This will most likely change in the final version of the Box specification now that the BLAKE2b-based Argon2i won the Password Hashing Competition.

2015.mobile
: Same as the `2015.desktop` profile, but with lower memory and CPU requirements, which makes it more suitable for mobile targets.

Passphrase Identities

Identity Generation Algorithm

- Verify that the passphrase is of high enough security level.
- Let the user select a profile.
- Compute a salt value:

$$BLAKE2b(Profile_{Name}, Profile_{StaticKey})$$

- Compute the seed value:

$$scrypt\_salsa208\_sha256(Salt, Passphrase, Profile_{CPU}, Profile_{Memory})$$

- Compute the ed25519 keypair using the seed value.

Transport Protocols

- Focus on IRC and Jabber support.
- Focus on Irssi for IRC, written in C and Jackline for Jabber, written in Ocaml.
- Have two implementations at release day and ensure they work together.
- Test everything.

Transport Protocol Restrictions

- Disallow non-UTF-8 character sets.
- Disallow HTML in chat messages.
- All timestamps are 32-bit integers (`uint32_t`), the system is sensitive to time changes so watch out for changes from the system clock.
- Every piece of information with variable length must have a length before the value is extracted from the byte-stream.

The Box Protocol

- Every frame is BASE64 encoded. We don't strip padding.
- We allow multiple messages in a single transport message.

The Box Protocol

Presence Broadcast

- The client broadcasts its presence either explicitly or periodically. You can optionally decide to "leak" your long-term identity key. Most users will do this but it wont be the default option.
- Always sign using long-term identity key, but include the short-term session key.
- This allows for anonymous rendezvous channels.
- Uses an $O(n)$ algorithm for identifying remote peers.

The Box Protocol

Peer-to-Peer

- Both parties decide if they want to announce their presence to each other.
- Derive curve25519 keys from the ID keys and establish the shared secret.
- Start sending encrypted frames to each other.

The Box Protocol

Peer-to-Channel

- A user receives an invitation to a channel with a key.
- If the user joins, we invoke a channel key rotation, which creates a new key for everyone in the session, thus not leaking former parts of the conversation in the channel.
- Use poly1305 MAC to quickly see if you're a member of a channel or if we can ignore the message.
- You can be in multiple channels that all uses one IRC channel as transport :-)
- Also periodically change the encryption key.

User Interface

- Be loud when something is off.
- Authenticated encrypted messages are green, encrypted messages are yellow and plain-text messages are red.
- Use the native view's of the client for communications.

Testing

- We have two implementations: Ocaml and C, for two different client's that can communicate with each other.
- Unit tests for everything, try to share as many error cases as possible.
- Experimental: Using EQC to test C code against Erlang code.

Undecided Problems

- Storing information for friends.
- Remote authentication of users (SMP).

Questions?