The Bluepass Password Manager

Geert Jansen - Ravello Systems

 @sysexit

About Me

- Director, Technical Marketing @ Ravello Systems.
- Previously: Shell, Red Hat
- Open Source user / developer since '95
- Married, 1 kid. Live in Catania Area

Why do I care about Passwords?

- Passwords have many problems.
  - Low entropy
  - Can be stolen
  - Server-side implementations often buggy
- Must find better alternatives. But nothing is catching on.
- We will be stuck with passwords for a long time.. Let's make them secure!

Rules of secure passwords

#1 Use long, random passwords.

#2 Use a different password on every site.

Unless you have super-human memory, you will need a *Password Manager*.

Requirements for a modern Password Manager

#zensysexit

- No secret data in the cloud
  - Really. Not even with the best crypto.
  - **Especially** not if encrypted with a password.
- Works on all your devices
  - And keeps data synchronized
- Open Source
  - How can I trust it otherwise?

Introducing ....

Bluepass Overview

- Open Source (GPLv3)
- P2P sync built in.
- Mostly portable code. Currently supports Linux, but other platforms coming.
- Written in Python, Qt
- https://github.com/geertj/bluepass

Bluepass Data Model

- A Bluepass instance manages a set of *Vaults*.
- Vaults contain *Items*.
- Vaults are append-only.
- Vaults are synchronized between *Nodes*.
  - A node is an instance's identity for a Vault.
  - Items are the unit of synchronization
- Items have an envelope + a payload
- Envelope contains routing data and signature

Bluepass Data Model (cont.)

- Item payload type: Certificate, EncryptedSecret
- Certificate:
  - Node X certifies that node Y can do {sign, encrypt, auth}.
  - Valid IFF a cert-chain exists back to my node.
- EncryptedSecret:
  - Encrypted to each node that has a valid cert.
  - Versioned. Conflict resolution by hierarchy and timestamp (eventually consistent).
  - Can contain arbitrary data fields.

Sync. Step 1: Pairing

- Connecting two vaults is called *pairing*.
- Target vault indicates via MDNS that it is visible.
- Initiating vault sends pairing request.
- Target user needs to approve.
- A 6-digit PIN code is shown on Target and needs to be entered on Initiator.
- When OK, certificates are exchanged and a trust relationship is established.

Sync. Step 2: Exchange Data

- Discovery via MDNS/DNS-SD.
- Synchronization happens:
  - Every 10 minutes
  - Whenever there is an update.
  - Whenever a new node comes in range.
- Full-mesh topology.
  - Limits the size to maybe 10-20 nodes.
  - Will move to gossip protocol (more scalable).

Cryptography used

- Recently moved from OpenSSL to (Tweet)NaCl
- Vault master keys
  - Key derivation: scrypt
  - Encryption: XSalsa20 (256-bit)
- Vault Items:
  - Signature: EdDSA over Ed25519
  - Payload encryption: XSalsa20 (256-bit)
  - Payload key sharing: EC-DH over Curve25519

Cryptography used (cont.)

- Pairing: HMAC(CHANNEL_BINDING, PIN)
- Synchronization
  - SSL: AES + ADH
  - Ed25519 signature over channel binding

Bluepass Architecture

- Frontend, Backend split.
- FE and BE are separate processes
- FE/BE Communicate via JSON-RPC
- FE written in Python + PyQt
- BE written in Python
  - Uses gruvi async io (green threads + pyuv/libuv)
  - CFFI binding to TweetNaCl as crypto
  - Local document store based on SQLite

DEMO TIME!

Next Steps

- Some more cleanups.
- Mobile ports
- Windows and Mac
- Browser plugin