Melicertes

Cerebrate

Team CIRCL

EC TheHive Training

Current state and identified issues with the tooling

- Melicertes's current implementation relies on re-implementations of exchange protocols
- Massive overhead
- Misalignments with the intents of the underlying tools
- Difficult to extend with new tools as each new tool would mean a new reimplementation
- Trust circle management is complex and awkward
- Tool is complex for complexity's sake

The goal is a full revamping of the management tooling of Melicertes

- New tool to manage Melicertes functionalities: Cerebrate Sync Platform

Goals

- Handle trust group management (based on the MISP sharing group system)
- Handle user and key management for the whole set of Melicertes tooling
- Basic orchestration of the Melicertes platform tools

Goals

- Reusing and adapting elements from the MISP code-base and paradigms shared by both tools
  - Authentication
  - ACL
  - User + role management
  - API handling
  - Organisation and trust circle management
- Reduce the replication of tasks with the various Melicertes tools, rely on native communication channels and instrument the tools via their respective APIs
- Modular, extensible design for supported tools

Cerebrate functionalities

- Internal functionalities (orchestrate my tools, manage my users, contacts)
- External functionalities (Interconnect tools with other orgs, advertise public/trusted information)

Internal functionalities

- Manage users
- Manage signing keys
- Maintain organisation information
- Manage trust circles/sharing groups
- Instrument Melicertes tools

External functionalities (ACL governed, from public to trust circle)

- Organisation registry
- User registry
- signing key registry
- Request access / inbox system

Design principles

- As much code reuse as possible (via MISP 3 core)
  - Reduce development time
  - Assure inherent improvements by upgrades implemented downstream from MISP
- Reliance on built-in APIs, hands-off aproach
  - Do not try to replicate whatś already there
  - Donť open ourselves up to risks from misunderstanding an implementation / building incorrect implementations

Design principles continued

- Modular design
  - Interactions with other tools should happen in modules and not in the core logic of the application
  - Similar to misp export/modules system
  - Built in cerebrate core, allow for implementations in other languages (see MISP STIX export as a design example)
- Tool agnostic design
  - Allow for modules that add new or replace existing tools for given purposes (e.g: I want to use the Hive instead of RT)

Design principles continued

- Build the tool with a generic use-case in mind
  - Organisation/User/Sharing groups outside of the CSIRT network should find the tool just as useful
  - Other communities should be able to find just as much value in the tool as the CSIRT network
  - Bridging communities should be an option
- Configuration and updating should be simplified and no 3rd party should be involved other than granting access to a network

Design principles continued

- User/organisation/trust circle exchange where applicable
- Forwarded authentication method (when possible)
- Instrumentation for org org exchange (MISP sync setup, Jitsi call initiation, etc)
- Instrumentation for intra-tool exchange (Configure RT MISP link, Viper MISP, etc)
- Optional statistics / diagnostics APIs / representation in cerebrate