Cubbit C++ developer test project

This document is part of the Cubbit HR process and is meant to be a tool to prove the candidate's ability to structure a brand new project that meets the company standards.

Cubbit horcrux project description

Cubbit is designing a brand new product to let its users encrypt, split and save copies of their files in the filesystem (the name "horcrux" is inspired by the Harry Potter saga). To do so, a new Command Line Interface application has to be designed so that it reflects Cubbit's standards in terms of:

- User experience
- Security
- Performances

Therefore the perfect candidate can implement a solution that reflects the standards above, quickly and with particular attention to details.

Requirements

With Cubbit horcrux, users can:

- **save a file** with the following steps:
  - The user selects a file from his/her filesystem
  - The user specifies the number of horcrux he/she wants to create
  - The application encrypts the file with a randomly generated key
  - The application splits the resulting buffer in n chunks where n is the number of horcrux specified by the user
  - The application saves the horcruxes on the disk in a folder specified by the user
  - The application shows the key used for encryption in **base64** format
- **load a file** with the following steps:
  - The user selects the horcruxes he/she created before
  - The user inserts the encryption key he/she got during the saving process above
  - The application reads the horcruxes from the disk
  - The application joins the encrypted buffers
  - The application decrypts the joined buffer with the provided key
  - The application saves back the plain file in a folder specified by the user

The CLI interface

The CLI should expose the following commands

1. `user@hostname:~$ horcrux create -n <horcrux count> <input path> <output path>`
2. `user@hostname:~$ horcrux load -k <decryption key> <input files> <output_file>`

Bonus points

- When designing the application try to put care in designing the system to be **easily extendable**. What if we want to add a new encryption algorithm or change the way the application splits the chunks in the future? What if we want to use a different transport instead of saving the horcruxes on the disk.
- Bonus points will be applied for solutions that provide **unit tests** as they help maintaining the software.
- Bonus points will be applied for candidates providing **compatibility for Windows, MacOS and Linux**

Hard technical requirements

- C++
- AES256

Nice technologies to see in action

- OpenSSL
- Boost
- CMake/Bazel
- GTest/GMock

Where to put the code

Feel free to create your own git repository to host the code. It is important for us to examine your git proficiency as well. When you feel ready, just email us with a link to it. It's that simple.