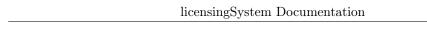# licensingSystem Documentation

nopreserveroot

v0.0.1

Abstract

This project is not to be used for malicious or illegal purposes. Project is available on a github repository `https://github.com/nopreserveroot`

Table of contents

Chapter 1

Overview

The goal of this project is to create a tool that allows for easy and secure method of giving someone a bunch of tools and commands to run, collecting results from it (potentially in the future wits some sort of scripting language aloowing for things like conditional commads) and giving a single encrypted file for the person being helped to transfer to you for review

Disclaimer: This project is not intended to be used for malicious purposes and is only meant to make helping others easier, more efficient and protecting your knowledge.

Chapter 2

Development rules

Suggestions:

- Server can be done in python3 flask framework (I can provide hosting after we leave dev phase) (needs more research into the framework itself)

- I'm still working on specs, If you have any idea let me know/add them to the docs

- Any suggestions? Say it

Suggested development rules:

- use Issues on GitHub

- keep docs up to date whenever you change something (if you don't want to play with LaTeX just leave a txt file describing changes, I can add it later)

- keep it nice and clean

- Make sure to periodically check documentations. Especially dev hints and specs

Chapter 3

Embedded Tools

This chapter explains the format for embedded executables for the client.

3.1 Format

Embedded executables and scripts are appended to the end of a client binary by the injector script.

The format of that section is as follows:

- Main Header
- Item 1 Header
- Item 1
- Item 2 Header
- Item 2
- ...

3.2 Main Header

The main header of embedded tools is defined as follows:

```
struct embeddedToolsMainHeader {
    uint32_t totalSize;
    uint16_t numberOfItems;
    char contentHash[32];
};
```

Explanation:

totalSize - the full size of embedded tools in bytes (including main header and individual headers)

numberOfItems - the number of embedded Items

contentHash - the SHA256 hash of the embedded part (Individual headers + items)

3.3 Individual headers

The individual header is separate for each embedded item and placed directly before the item content starts.

It is defied as follows:

```
struct individualHeader {
    uint16_t ID;
    uint32_t length;
    char fileName[64];
    char itemHash[32];
    uint32_t permissions;
};
```

Explanation:

ID - numeric ID of the item. Starts from 0 and is being incremented by 1.

length - length of the item (excluding individual header)

fileName - the file name of the item including extension

itemHash - SHA256 hash of the item after unpacking and decrypting

permissions - item permissions to be set by *chmod*

3.4 Items

The item itself is going to be encrypted using one of the individual keys generated and injected using the Injector script. It'll be decrypted when extracting. It'll only be extracted when will have to be used.

Chapter 4

FAQ

- **Will you add a function to run it quietly/fully automatically/send results automatically in the background?** No, this would make it much easier to use it for malicious purposes and that's not a goal of this project. This way at least a 2nd targeted party has to make some actions which makes for some form of a consent I guess.

- **What is the actual goal of this project?** No idea. Fun? Gaining knowledge?

- **Why aren't you using a library for that thing? Why don't you use X or Y instead of Z, it's easier?** The goal of this project is to learn how things work under the hood. You won't learn using just ready-made tools.