

# Proposal for resolving the “Password Problem”

in Fingerprint GUI

## The Problem

All \*NIX systems are based on a name/password scheme for identifying and authenticating a user to be logged in. Usernames and hashed passwords are stored in a file (e.g. /etc/passwd, /etc/shadow) or in some kind of a database (e.g. LDAP). When a user performs his login the given password is hashed and then compared to the stored hashed password.

On almost all Linux systems a special modular library PAM is used for user authentication by the login procedure or any other application when user identification or authentication is required. In addition to perform the authentication mechanism this PAM library is able to hold the given password as clear text in its session environment. Any application running in this session can request the password from PAM if PAM is configured to give the password to this application. This mechanism is used by applications like gnome-keyring for encrypting/decrypting something like the users keyring store.

If the authentication mechanism of PAM is enhanced by some alternative methods like iris scanning, smart card authentication or fingerprint scanning a password is not needed to authenticate the user. In these cases the clear text password is missing in the session environment of PAM. This causes an additional request for a password by the application that needs it for encryption/decryption after login. The user might wonder why he is asked for a password when he is successfully authenticated already and will see this procedure as an inconvenience.

## The Solution Principle

The problem could be solved by the PAM authentication mechanism (e.g. the fingerprint PAM module) when the user's password would be known even if the login procedure would not request a password. The authentication mechanism then could hand over this password to PAM after the user was identified and authenticated. This requires some place (e.g. a file) where the password is accessible by the authentication mechanism. To protect it against reading by unauthorized persons this password has to be stored somewhere in encrypted form and has to be decrypted with some decryption key after the user is authenticated.

### *The Problem with Open Source*

The decryption key mentioned above might be either a user specific key to be used for this user's password only or a global key to be used for decrypting all those passwords. If a user specific key is used it must be stored somewhere. Unauthorized persons who have access to the encrypted password and the stored key would be able to decrypt the password. If a global key is used this key could be included in the authentication mechanism module (e.g. the fingerprint PAM module). Proprietary software may be able to hide this key in their program code in a form that is not easy to reverse engineer. But open source software can hide nothing in its program code. So the only solution is to save the encrypted password and the decryption key for it in different places.

## Solution for Fingerprint GUI

The idea for a solution to be implemented in Fingerprint GUI is to save the encrypted password in a remote place (e.g. a USB memory stick) while a user specific decryption key is saved on the local machine inside the user's home directory structure.

## Configuration

The configuration procedure should be implemented in “fingerprintGUI”. If the user decides to make use of the “Password Feature” (e.g. by a check box), he is requested for a UUID of some external media (e.g. USB stick SD card etc.) to save the encrypted password to. Maybe we can provide him with a drop down list of UUIDs from currently connected media. The “fingerprintGUI” will then try to mount this media (if not done already) and ask for a path of some directory there. This path must be accessible and writable while running “fingerprintGUI”. The program will store a file named “<username@machinename>.xml” containing the encrypted password there.

After that the user is asked to enter and reenter his password. The “fingerprintGUI” will then generate a private/public key pair, use the private key to encrypt the given password, save the encrypted password to the file “<username@machinename>.xml” in the given path and then immediately throw the private key away. The public key will be used by “fingerprintPAM” to decrypt the password. This public key along with the given UUID and path is saved by “fingerprintGUI” to a file “~/.fingerprints/.settings.xml”.

## Authentication

Authentication is done by “libpam\_fingerprint.so” if the service in question is configured in PAM. After the user is identified and authenticated by his fingerprint the “libpam\_fingerprint.so” module looks for the file “~/.fingerprints/.settings.xml”. If found it looks for a connected media with the given UUID and tries to mount it. If the “<username@machinename>.xml” file is accessible there it decrypts the password with the public key given in “~/.fingerprints/.settings.xml” and hands the clear text password over to the PAM environment. The unmounts the external media.

## Vulnerabilities

Everyone who has access to the external media and the computer can decrypt the password. The file “~/.fingerprints/.settings.xml” should have mode 0600 to be accessible by the user only. However root will be able to read this file and thus know the path and the public key. As long as the external media is connected, root can mount it and decrypt the user's password. This should be taken in account by the user for his decision whether or not to make use of the “Password Feature”. If using the feature the user should observe the following:

- Hold the external media always apart from the computer; never leave both at the same place.
  - Use the external media for login only; don't use it for other data because it should never be mounted except for the login procedure.
  - Keep the password in mind in case the external media is lost.
  - Never store the content of the external media to the computer.
-