cachy

Cross-Platform Password Protection System

Technical Information

Table of Contents

Version

Version	Author	Date
1.0	Nick Pateman	14/04/2019

Marketing Description

cachy is a cross-platform password protection system that is simple to use, quick and secure. It provides you with all the necessary tools for taking the protection of your credentials into your own hands.

Synchronisation between devices is as simple as configuring a personal cloud storage provider account. You do not need to create a new account with devoctomy or pay for a monthly subscription.

- **Unlimited Vaults** (Create as many vaults as you desire, one for work, one for your personal credentials, one for family, you decide.).
- **Unlimited Credentials** (Store as many credentials in a vault as you require, you aren't limited. If you have a credential, protect it with cachy.).
- **'Military Grade' AES 256bit Encryption** (Your credentials are kept secure using military grade, AES 256 bit encryption using a complex passphrase of your choosing.)
- **Cloud Support** (Use your existing cloud storage accounts to provide synchronisation of your cachy vaults between devices, we currently support Dropbox, OneDrive and Amazon S3 with more to come.).
- **Password Generator** (Automatically generate complex passwords for maximum security, automatically generated to the specification you require.).
- **Password Age Tracking** (Keep track of how old your passwords are to help keep you in the habit of updating them regularly.).
- **Organise** (Keep your credentials organised with customisable glyphs, colours and tags, making it easy to locate a specific credential whenever you need it.).
- Add notes to your credentials (store any additional / relevant information that you may need).
- **Full Offline Support** (Fully functional offline. Internet connectivity is only needed if you have chosen to synchronise your vault with the cloud.).
- **Advanced Cryptographic Options** (Advanced configurable cryptographic options, such as password derivation settings.)
- **Import** (Import your credentials from another password management system. If it's in CSV form, we can import it.).
- **Export** (Export your credentials to another password management system, although we would rather you used cachy, we still don't want to tie you into being stuck with us.).
- **Cross Platform** (cachy runs on both Android and Windows, using the same code so you should always have a consistent experience.)

Introduction

The intention of this document is to describe in detail, how cachy is architected to protect your private account credentials. As the product is not only closed-source, but being released by a new and unknown company, **devoctomy**, it is imperative that all relevant information is disclosed.

devoctomy is the creation of Nick Pateman, a professional software developer from the South East of England. Having worked with several companies developing security themed software for almost 12 years, **altaVENTE**, **CertiVox** and **MIRACL**; I cannot express more, how important it is to look after your account credentials.

In an ideal world, we wouldn't need usernames and passwords, but it has become quite evident that they will remain for a few years yet. With that said, people tend to use things like note keeping services, such as **Google Keep**, **Evernote**, or just plain local **Excel** documents to store what could potentially be their most important, private information.

Although you might find using these methods to be convenient for you, they are not ideal for many reasons and you run the risk of having all of your credentials compromised, potentially causing you a lot of stress and financial issues.

Over the years I have chosen to use **KeePass**, which has been perfect for securing my credentials on my local PC, but as good as it is, **KeePass** was developed long before smartphones were in such widespread use and was not built with multiple device synchronisation in mind. There are of course, Android applications available for accessing KeePass databases now, such as **KeePass Droid**, but nothing to provide a consistent, cross-platform experience.

There are also services such as **LastPass** and **1Password**, but to get the most from these services, you need to pay a monthly subscription, and that's not something that I personally think anyone should need to do.

With that in mind, cachy provides the standard **'military grade' AES 256 bit** encryption (as you would expect), but also the ability to synchronise across multiple devices using your own cloud storage accounts, none of your personal data is stored by us, in fact we don't even have any servers to do so, and we want to keep it that way.

We are also completely transparent as to how the system works and in this document, we will describe exactly how the data is protected and stored.

AES Key Derivation

cachy uses a key derivation function to generate an AES key which is used for encryption and decryption of your vault. The function used is configurable within the application settings and is used for all vaults. You cannot chop and change key derivation function, although we may change this in future versions.

The key derivation functions available are **PBKDF2** or **SCrypt**, these use your master passphrase and system-wide configured parameters (mentioned above) to generate an AES key. This is essentially the most important link in the security of a cachy vault, which is why we recommend creating a relatively long and complex pass "phrase", not a pass "word", but of course the choice is yours and it is merely a recommendation.

Although we do not recommend playing about with the function parameters, doing so can increase the security of your vault (or decrease if you have misconfigured).

The generated key is not stored anywhere by cachy, and once finished with, it is disposed of until the next time that you are prompted to enter your master passphrase, when it is re-generated.

Please note that you must configure the same key derivation function parameters, wherever you choose to synchronise your vault. Failure to do so will result in cachy not being able to open your vault.

The ‘.vault’ File Format

The cachy file format is extremely simple, firstly the vault meta data, and credentials are serialised into JSON (Described in the next section), and then the serialised JSON data is encrypted using a 256 bit (32 byte) AES key derived from your master passphrase. This is all done in-memory, and the resulting encrypted data is stored to disc.

The resulting encrypted data does not reveal any meta data about you, or your credentials, and therefore if you lose the passphrase the only way it could be unlocked again, would be to use brute force, which would take a ridiculously long time, even with todays technology.

devoctomy cannot recover a vault that you have lost the master passphrase for either. Once lost, it’s lost for good.

The encrypted data is structured as follows,

[AES IV (32 bytes)][Encrypted JSON Meta Data][AES Key Salt (16 bytes)]

JSON META Data

The structure of the unencrypted JSON data is as follows,

```
{
  “Header”: {
    “FormatVersion”: “”
  },
  “ID”: “”,
  “Name”: “”,
  “Description”: “”,
  “CreatedAt”: “”,
  “LastUpdatedAt”: “”,
  “Credentials”: [
    {
      “ID”: “”,
      “GlyphKey”: “”,
      “GlyphColour”: “”,
      “Name”: “”,
      “Description”: “”,
      “Website”: “”,
      “CreatedAt”:                                                                                    “”,
      “LastUpdatedAt”:
      “PasswordLastModifiedAt”:
      “Username”: “”,
      “Password”: “”,
      “Tags”: “”,
      “Notes”: “”,
      “AuditLogEntries”: [
        {
          “DateTime”: “”,
          “TypeOfEntry”: “”,
          “Parameters”: [ ... ]
        }
      ]
    }
  ],
  “AuditLogEntries”: [
    {
      “DateTime”: “”,
      “TypeOfEntry”: “”,
      “Parameters”: [ ... ]
    }
  ],
}
```

Application Data

Below is a description of all of the data that cachy stores on your device,

Windows

The root directory that cachy stores its files is,

```
C:\Users\nickp\AppData\Local\Packages\com.devoctomy.cachy_cddyqgaznq
y6j\LocalState\ProgramData
```

- **Config**
    - **cachy.config.json**
        - This is the main application configuration file
    - **cloudproviders.json**
        - This file contains a list of configured cloud providers. Any related API keys granted through OAuth are stored in the Credential Cache that you can access through the control panel for that user.
    - **index.json**
        - This file contains a list of vaults that are displayed in the main vault list of cachy, no matter how they are synchronised, they will be listed here.
    - **logger.config.json**
        - This is the configuration file for the application logging.
- **LocalVaults**
    - Local copies of cloud synced vaults and local vaults are stored here.
- **Log**
    - **cachy.log –** Main application log, this does not contain any sensitive information and is used solely for diagnosing issues with cachy. Logging is disabled by default.

Android

The root directory that cachy stores its files is,

```
/Internal Storage/Documents/devoctomy
```

- cachy
  - Config
    - cachy.config.json
      - This is the main application configuration file
    - cachy.keystore.json
      - This file contains OAuth tokens for the configured cloud providers. The values in this file are encrypted using AES keys generated for the current user via the **AndroidKeyStore** provider. **If you uninstall cachy, all values in this file become irretrievable, requiring you to re-authenticate with your cloud provider(s).**
    - cloudproviders.json
      - This file contains a list of configured cloud providers. Any related API tokens granted through OAuth are stored in the **cachy.keystore.json** file.
    - index.json
      - This file contains a list of vaults that are displayed in the main vault list of cachy, no matter how they are synchronised, they will be listed here.
    - logger.config.json
      - This is the configuration file for the application logging.
  - Exports
    - Due to the over complexities and limitations with the current version of cachy, any exported credentials will be stored in this directory.
  - LocalVaults
    - Local copies of cloud synced vaults and local vaults are stored here.
  - Logs
    - **cachy.log**
      - Main application log, this does not contain any sensitive information and is used solely for diagnosing issues with cachy. Logging is disabled by default.

Biometrics Support

One thing that hasn't been mentioned so far is Biometrics, and why cachy does not (currently) support biometrics to lock/unlock a vault.

Firstly, and most importantly, as mentioned previously, the master passphrase, is the most important part of securing your vault, if you forget it, you will no longer be able to access your vault. You shouldn't store your master passphrase anywhere either, it must exist in one place, that place being your brain.

With that said, you must understand how biometric authentication works in apps; using Android as an example. An application can request an AES encryption key, like the one we used to encrypt the vault, from the Android Key Store, in order to grant the key we can request biometric authentication. The key that is generated is associated with the app that requested it, so if you uninstall / reinstall the app the key is lost, plus Android cannot generate the same key every time, with those limitations in mind, the key would never be used to encrypt the vault itself, rather the master passphrase, herein lies the problem.

So if we were to integrate biometrics, the AES encryption key generated from the authentication process would be used to encrypt the master passphrase so that it can be used to unlock the vault automatically. Doing so prevents the user from having to type in their master passphrase, which saves the user the inconvenience, but after a while it will be forgotten, rendering the vault inaccessible.

This was the main driving force behind deciding not to release cachy with biometrics support. As you will need to remember your master passphrase when transferring your vault between devices, and potentially after certain operating system procedures, such as updates, we felt that it would be too risky to implement.

That's not to say we will never implement it, but we will need to think hard about how we do implement it so that the user doesn't forget their master passphrase because of it.

Cloud Provider OAuth Authentication

cachy currently support 3 cloud providers, those being Dropbox, OneDrive and Amazon S3. Both Dropbox and OneDrive connect to cachy using the OAuth protocol. This section details exactly how cachy implements OAuth.

1. cachy generates an RSA key pair.
2. cachy connects to Azure Function App (owned by devoctomy) and requests a session. Passing the RSA public key generated in step 1.
3. The Azure Function App starts a session with the client and stores the provided RSA public key to Azure Table Storage. The RSA private key stays on the client, and is not transmitted or saved to disk.
4. cachy then makes another request to the Azure Function App which waits for completion of the OAuth authentication process.
5. cachy then begins authentication with either Dropbox or OneDrive, providing the ID of the session created in step 3 as the OAuth 'state' parameter.
6. Once the user has authenticated the cloud provider sends a token back to the Azure Function App along with the session ID, in the form of the 'state' parameter.
7. The Azure Function App then uses the token to request the 'access token' from the OAuth provider (Dropbox / OneDrive).
8. Upon successfully receiving the 'access token' the Azure Function App then generates an AES encryption key and encrypts the token, it then uses the RSA public key associated with the session and encrypts AES encryption key. Both encrypted token and encrypted AES key are then stored in Azure Table storage.
9. The Azure Function App then returns the encrypted data from step 8, to the waiting connection from step 4. Once the data has been sent to the client, it is deleted from Azure Table storage.
10. Now that the client has the OAuth 'access token' it is then stored in an encrypted manor, locked against the current user account, and the current installation of cachy.

The above process is very similar to how OAuth would work on a website, except we are encrypting the data in such a way that only the client can decrypt it.

cachy does not currently use Application URI's to perform all authentication upon the device, with the operating system playing middle-man to the process. It performs the authentication in the way outlined above in order to have a consistent method for all platforms. Whether you are authenticating from the Windows Store App or Android, the same code is used, in the future we are hoping to add iOS, OSX, Linux and Windows Desktop versions which will all use the same process.

At no point is any data used during the authentication process written to disk in plain text.