**Título:** **Password Managers**

**Author:** **Pedro Miguel Loureiro Amaral nmec: 93283**

**Date:** **07/07/2021**

## Index

## Introductory Note:

This report presents the result of research about password managers in general. Therefore, it will include how they work and the impacts of their usage on their users in terms of internet security, a theme that is increasingly relevant nowadays. This report will also include an analysis of an existent and widely used password manager.

## 1. Summary / Abstract

This document addresses the use of password managers by first describing them and how they work. Then the widely used password manager LastPass will be analyzed and its architecture explained and replicated as much as possible. Finally, the advantages and disadvantages of the use of password managers will be explored, with a special focus on the consequences to the user's internet security.

## 2. Framework

With the evolution of computing power over the last few decades, there is a growing difficulty in creating a good password that is not easily guessed and stolen. A simple password is easily guessed by a computer program. However, a password too complex can be hard to remember, and if one or two dozen are needed it becomes nearly impossible to remember them all. In this situation the user may feel captivated to write it in a paper or to simply use the same password in every single service, both options being a vulnerability.
The solution to this problem is the usage of a password manager. This service allows users to have all their passwords stored in a secure location and retrieve them when needed. Nowadays, there are several password managers available in the market and many specialists recommend their usage.
However, with security on the web being increasingly more important, the idea of having all passwords in one single place might be unattractive for some people who may think that using this might become a liability.

## 3. What is a Password Manager? [1]

A password manager is a service that provides users with a convenient way to generate, store and manage their passwords for different services. These passwords are stored encrypted and need a key of some sort to be decrypted and used.

## *3.1 Types of Password Managers [2]*

It is possible to find on the market different types of password managers with different features, advantages, and disadvantages. These are the following:

- Desktop-based: in this type of password manager the user credentials are encrypted and stored locally on the user's machine. This has the advantage of being more secure since an attacker can only access the encrypted credentials if he has access to the user machine. However, this comes with the disadvantage that the user also cannot access his credentials in another machine or shared them with other people as other types allow.
- Cloud-based: in this type of password manager the user credentials are encrypted and stored on a remote server belonging to the service provider. This comes with the advantage that the user can access his credentials everywhere. However, the security of this method is dependent on the security of the service provider.
- Browser-based: in this type of password manager the user credentials are encrypted and stored on a browser such as Chrome or Firefox. This method is free and the easiest to use. However, the credentials stored with this type can also only be accessed on one machine. Furthermore, it contains fewer features and is less secure compared to other types.
- Portable: in this type of password manager the user credentials are encrypted and stored on a mobile piece of hardware such as a USB stick, an external hard drive, or even a mobile device. This has the advantage of an attacker needing to access the user's device to steal the credentials. Furthermore, since they are stored in a portable device they also can be used in different devices. However, it comes with the disadvantage that if the portable device is stolen or lost, all the important data stored in the device is lost.
- Token-based: in this type of password manager the user credentials are encrypted and stored and can only be retrieved with a security token in addition to the normal password manager credentials. This type is, therefore, more secure but also more expensive and complex being less recommended for a casual user.
- Stateless: in this type of password manager the user credentials are generated using a master password, a username, and the website the password is for. This type is secure because the passwords that are generated are not stored anywhere but they have a higher vulnerability to brute-force attacks compared to other types.

As we can see in the list above, there are six different types of password managers with different characteristics making each one of them adequate for a different situation. However, in this report, the focus will be on the cloud-based password managers which are the most common, popular, and advised for most users.

# 4. How do Password Managers Work?

## *4.1 General Implementation*

In general, to use a password manager the user needs to have a very secure master password since if the password is not secure the fact that the service is secure will not matter. This password has two uses. First, it will be needed to authenticate the user in the password manager server so that the server sends us the list of encrypted passwords it contains which we will call the vault. After the user receives the vault, it will be decrypted with the help of the master password. After the user finishes using or altering his passwords, the vault is encrypted and sent to the server to be stored. Given this general process, we can see that when the vault is on the server it is always encrypted, and the encryption and decryption process is always made on the client device resulting in greater privacy and safety if the server is attacked. Also, although the master password is used both for authentication and decryption of the vault, it is never stored on the server meaning that the server is unable to decrypt our vault giving an additional layer of security.

## *4.2 LastPass Analysis [3] [4]*

To study an algorithm with more detail, the LastPass Password Manager was chosen to be studied as an example since it is the best overall according to Investopedia [5] and it also has a considerable amount of information about its algorithm. With the information available, a custom implementation was made using Python 3 and cryptography.io [6] which can be accessed in the annexed code or the Github repository [7].
When a user uses LastPass several steps happen behind the scenes. Given that the user has LastPass already configured, he will need to obtain the encrypted vault in the server and decrypt this vault. The first step to this goal is to obtain the encryption key, which is the result of the master password being hashed 100100 times using PBKDF2 (Password-Based Key Derivation Function 2).

```
kdf = PBKDF2HMAC(algorithm=hashes.SHA256(), length=32, salt=username.encode('latin'),
    iterations=100100, backend=backends.default_backend())
encryption_key = kdf.derive(master_password.encode('latin'))
```

**Figure 1 - Client hashing of the master password to obtain the encryption key.**

PBKDF2 is an algorithm used to derive a key from a password and, in this case, will receive as arguments the hash function SHA256, a length of 32 bytes since this is the needed length for the encryption key, a salt which corresponds to the username and 100100 as the number of iterations. The number of iterations is high so that an attacker needs a lot of computing power to generate all possibilities for the password and make a brute force attack. This protection is also reinforced by the existence of a salt which makes it even harder to generate all possibilities.
Then the encryption key will be hashed again to generate the authentication hash.

```
kdf_once = PBKDF2HMAC(algorithm=hashes.SHA256(), length=32, salt=username.encode('latin'),
    iterations=1, backend=backends.default_backend())
authentication_hash = kdf_once.derive(encryption_key)
```

**Figure 2 - Client hashing of the encryption key to obtain the authentication hash.**

After obtaining the authentication hash, a request for the encrypted vault which contains all passwords is made to the server sending the username and the authentication hash so that the server can verify the user identity.
When the server receives a request for a vault it must first verify if the user credentials are valid and to do it, it needs to generate the authentication key which is a key derived from the authentication hash sent by the user.

```
kdf = PBKDF2HMAC(algorithm=hashes.SHA256(), length=32, salt=username.encode('latin'),
    iterations=100100, backend=backends.default_backend())
authentication_key = kdf.derive(auth)
```

**Figure 3 - Server hashing of the authentication hash to obtain authentication key.**

For this custom implementation, it was not possible to obtain the exact number of hashing iterations that happen on the server, so the number 100100 was used again. Regardless, having more hashing iterations will only help to make everything more secure since it increases the effort of an attacker to obtain the password. Now that the server has the authentication key, it checks whether it matches with the username and, if it does, it returns the user's encrypted vault. Once again, it is worth noting that the vault is not changed at all when in the server and the server also cannot decrypt it since the authentication hash is already the result of a hash function on the encryption key making this key hard to obtain.
Back on the client, when the user receives his vault, he will decrypt it using the encryption key mentioned above and a cipher with the encryption algorithm AES256.

```
iv = vault[0:16]
cipher = Cipher(algorithms.AES(encryption_key), mode=modes.CBC(iv), backend=backends.default_backend())
decryptor = cipher.decryptor()
unpadder = padding.PKCS7(128).unpadder()

vault = unpadder.update(decryptor.update(vault[16:]) + decryptor.finalize()) + unpadder.finalize()
vault = json.loads(vault.decode('latin'))
```

**Figure 4 - Client decryption of the password vault.**

AES256 is an encryption algorithm used to both encrypt and decrypt data with a given key. This key is the same in both operations making it a symmetric key. From the available information, it was not clear which mode of AES and which padder is used so in this custom implementation the CBC mode was used as well as the PKCS7 unpadder.
Once the vault is decrypted the user can freely use and manage all the passwords to access all his services. Then if the user wants to add another password or update or remove an existing one, this vault needs to be sent again to the server. To do this the user needs to obtain the encryption key and authentication hash with the same methods as described above. After having the encryption key, the vault needs to be encrypted again.

```
iv = os.urandom(16)
cipher = Cipher(algorithms.AES(encryption_key), mode=modes.CBC(iv), backend=backends.default_backend())
encryptor = cipher.encryptor()
padder = padding.PKCS7(128).padder()

data = iv + encryptor.update(padder.update(json.dumps(vault).encode('latin')) + padder.finalize()) + encryptor.finalize()
```

**Figure 5 - Client encryption of the vault.**

Once again, the AES256 with the CBC mode is used as the encryption algorithm and PKCS7 is used as the padder since encryption and decryption need to be done in the same way or else it becomes impossible to decrypt.
With the vault encrypted, it will be sent to the server alongside the username and authentication hash. In the server, the user will first be authenticated using the previously explained methods and if the username matches with the authentication key, then the new vault will replace the old one.
In the next image, we can see the architecture of the LastPass password manager as described on their website.

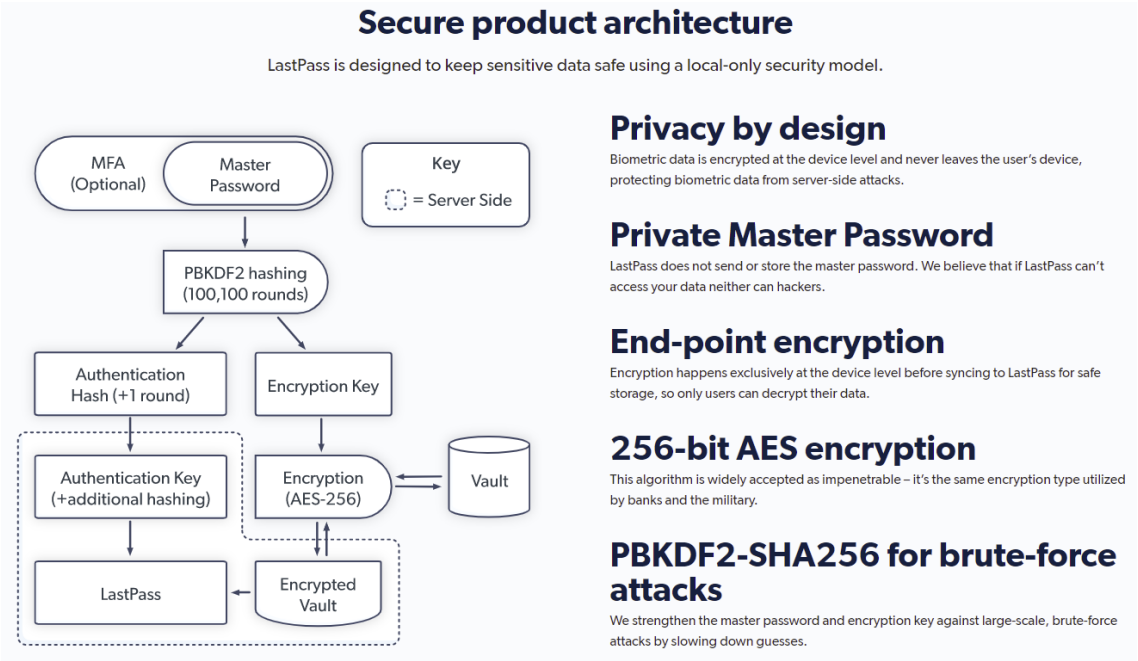


**Figure 6 - LastPass architecture.**

The previous custom implementation attempted to replicate this architecture as much as possible. Despite this, some parameters could not be found and therefore needed to be filled with some valid replacement. In this architecture, we also find an optional item MFA which is a Multi-Factor Authentication. If MFA is present then the user needs to prove his identity in more than one way, being, in this case, the master password not enough. Other ways to authenticate include biometric factors and hardware tokens [8]. However, since it was optional and hard to implement depending on the other authentication factor, the custom implementation described above does not include it.

# 5. Advantages [9]

## *5.1 Better Passwords*

If we check the “Top 200 most common passwords of the year 2020” from NordPass [10] we can observe that most of these are simple passwords many taking less than a second to crack but keep being used and exposed millions of times. Given these statistics, we can conclude that many if not most users are under the illusion that no one is interested in their account enough to protect it better. With many password managers having a secure password generator, this becomes a major advantage of its usage. The password generated will be very secure since it will be random enough to not be guessed with a dictionary attack, which is an attack using commons keywords, and big enough to not be guessed with a brute force attack, which tests all possible combinations even if it takes a lot of time as Figure 7 shows.

| Number of Characters | Numbers Only | Lowercase Letters | Upper and Lowercase Letters | Numbers, Upper and Lowercase Letters | Numbers, Upper and Lowercase Letters, Symbols |
|---|---|---|---|---|---|
| 4 | Instantly | Instantly | Instantly | Instantly | Instantly |
| 5 | Instantly | Instantly | Instantly | Instantly | Instantly |
| 6 | Instantly | Instantly | Instantly | 1 sec | 5 secs |
| 7 | Instantly | Instantly | 25 secs | 1 min | 6 mins |
| 8 | Instantly | 5 secs | 22 mins | 1 hour | 8 hours |
| 9 | Instantly | 2 mins | 19 hours | 3 days | 3 weeks |
| 10 | Instantly | 58 mins | 1 month | 7 months | 5 years |
| 11 | 2 secs | 1 day | 5 years | 41 years | 400 years |
| 12 | 25 secs | 3 weeks | 300 years | 2k years | 34k years |
| 13 | 4 mins | 1 year | 16k years | 100k years | 2m years |
| 14 | 41 mins | 51 years | 800k years | 9m years | 200m years |
| 15 | 6 hours | 1k years | 43m years | 600m years | 15 bn years |
| 16 | 2 days | 34k years | 2bn years | 37bn years | 1tn years |
| 17 | 4 weeks | 800k years | 100bn years | 2tn years | 93tn years |
| 18 | 9 months | 23m years | 6tn years | 100 tn years | 7qd years |

**Figure 7 - Time for Hackers to Brute Force a Password. [12]**

## *5.2 A Password for each Service*

Another problem associated with passwords is the fact that many times users end up using the same password for more than one service or even the same one for all services since it will be easier to remember. This means that if a password from a service is compromised it will not be only that service that is compromised but possibly all that the user is registered. Once again, a password manager is helpful since it can not only generate a strong password for each service but also save it so that the user does not need to remember it.

### 5.3 Convenience

As seen by the two vulnerabilities spoken above, the user is always looking for a convenient process such as creating a password weak and easy-to-remember or using the same password for all services. However, being convenient does not necessarily mean it is insecure. Password managers can also be convenient such as providing the user with a faster login since it automatically populates the needed fields or being able to store other important user data such as credit card information. Depending on the password manager used it might even allow managing shared access to a service by multiple accounts. Being convenient is a major factor in making a password manager safe because that way the user is less likely to make dangerous actions that severely lessen its security.

## 6. Disadvantages [11]

### 6.1 Single Password for Everything

Although one of the advantages of password managers is easily having a different password for each service the fact remains that it also has only one password and with that password, it is possible to access all services. If the user forgets his password or his password gets stolen, he loses access to all services. To reduce the chances of this happening the user must use a strong password and implement multi-factor authentication.

### 6.2 Learning Curve

Despite one of the advantages of using a password manager is its convenience, learning to use it takes some time since the user will have to learn how to access his accounts again. Furthermore, it also takes time to make all the necessary configurations. Although password manager providers created tools like browser extensions to simplify the process, this is still a reason why users may not use them because, again, users like convenience.

### 6.3 Cost

Although there are some free password managers, they only offer fewer features. To increase the security of our accounts a paid password manager is needed [7], which is another reason that diverts users from password managers.

## 7. Conclusions

To conclude, this report has shown that password managers are a helpful tool nowadays that can improve substantially the internet security of the user. Furthermore, it was shown that there are different types of password managers with different advantages and disadvantages and their usage should be considered before using one. Next, a replication of the LastPass password manager was made which helped show how these services are structured to be as safe as possible for the user. Lastly, an analysis of the advantages and disadvantages of the use of a password manager for the user was made showing that the benefits outweigh the costs, but they should still be kept in mind.

## 8. References

[1] Wikipedia, "Password Managers," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Password_manager. [Accessed 06 July 2021].

[2] Zoho, "What are the different types of password managers?," Zoho, [Online]. Available: https://www.zoho.com/vault/educational-content/different-types-of-password-managers.html. [Accessed 05 July 2021].

[3] LastPass, "LastPass - Security," LastPass, [Online]. Available: https://www.lastpass.com/security/zero-knowledge-security. [Accessed 06 July 2021].

[4] LastPass, "What makes LastPass secure?," LastPass, [Online]. Available: https://support.logmeininc.com/lastpass/help/what-makes-lastpass-secure-lp070015. [Accessed 06 July 2021].

[5] M. Kurko, "Best Password Managers," Investopedia, 26 May 2021. [Online]. Available: https://www.investopedia.com/best-password-managers-5080381. [Accessed 05 July 2021].

[6] Criptography.io, "Criptography.io Documentation," Criptograhpy.io, [Online]. Available: https://cryptography.io/en/latest/. [Accessed 06 July 2021].

[7] pedromiglou, "pedromiglou/APSEI_Password_Managers," Github, [Online]. Available: https://github.com/pedromiglou/APSEI_Password_Managers. [Accessed 07 July 2021].

[8] LastPass, "LastPass Multi-Factor Authentication," LastPass, [Online]. Available: https://www.lastpass.com/products/multifactor-authentication. [Accessed 06 July 2021].

[9] K. Gray, "5 Benefits of using a password manager," Envision IT Solutions, 24 April 2018. [Online]. Available: https://blog.envisionitsolutions.com/5-benefits-of-using-a-password-manager. [Accessed 07 July 2021].

[10] NordPass, "Top 200 most common passwords of the year 2020," NordPass, [Online]. Available: https://nordpass.com/most-common-passwords-list/. [Accessed 06 July 2021].

[11] Orange County's Credit Union, "Pros and Cons of using a Password Manager," Orange County's Credit Union, 04 May 2021. [Online]. Available: https://www.orangecountyscu.org/stories/pros-and-cons-of-using-a-password-manager/. [Accessed 07 July 2021].

[12] MILLIONAIRE TREK, "Time for Hackers to Brute Force Your Passwords," MILLIONAIRE TREK, 30 December 2020. [Online]. Available: https://millionairetrek.com/top-ways-and-time-for-hackers-to-brute-force-your-passwords-in-2020/. [Accessed 07 July 2021].