UNIVERSITY OF TARTU

Password managers

Rasmus Vahtra
December 20, 2016

1 CONCEPT OF PASSWORD MANAGERS

Password manager is a software that helps to keep track of passwords and allows to use them without remembering all of them. Good password managers store the passwords. In order to gain access a master password is generally required, which is also the only password that has to be memorised.

The idea behind password manager is to allow a user to use a different password everywhere. Remembering all those different passwords in different places is rather difficult, which is why people tend to use the same password on different places, which however, is a security risk. Password manager solves that problem.

The alternative to using a password manager and still using different passwords in different places is to write down the passwords somewhere. In that case there are generally two options – write passwords on paper or write them in notepad in a computer. If passwords are written on paper then at least no one can learn them by hacking the PC and searching around. However, that paper may get lost. In a best case scenario, the owner of that paper just has to change passwords on all the accounts. In worst case scenario if someone should find it then all the accounts might be in danger, especially if the accounts and the location were also written on it. If passwords are stored in notepad then at least it won't get lost physically, but it may be accessible to anyone that gains access to the computer where the notepad-readable file is stored. Both of these options are rather insecure. This is yet another issue that password manager solves. Password managers store passwords in a single file, which is protected by master password and encrypted, so even if someone manages to get their hands on the password database file they won't be able to learn anything from it.

There are many options available on the market ranging from closed source to open source. We will be mostly looking at open source options.

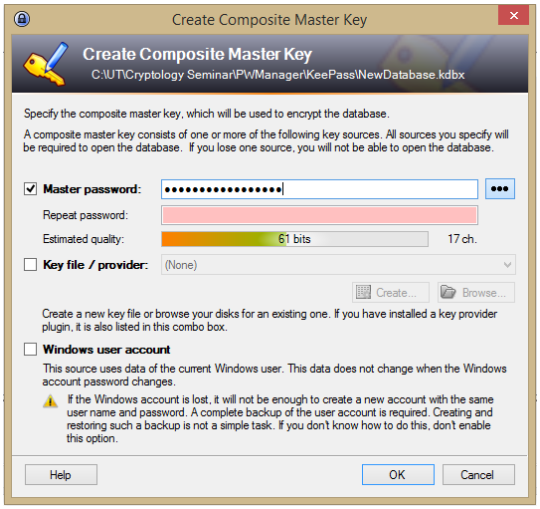
2 Password manager softwares

2.1 Testing methods

There were two primary things considered when testing password managers – usability and security in that order. The whole point of password manager is to make it more simple and more secure for the user to use different passwords at different locations. If the software is not usable or not simple enough then it wouldn't serve the purpose even if it was the most secure password manager in the world. Usability was tested by simply using the software without any prior knowledge. Each manager was installed and then used for awhile to see how intuitive it was. After that, the overall security of the manager was checked by reading available documentation and reading source code to check the implementation.

2.2 KeePass

KeePass is a solution made in C# and the source code comes with a centralised solution file that anyone can use to compile their own version. All the UI and encryption solutions are custom made without using any external libraries. The database is also actually a structure that contains data in variables, so no external nor pre-built solutions are used.



Picture 1. Creating a database

Upon starting the application, user is asked to create a database and set a master password. There's also an option to use key file instead of master password or use them together. Once the database is created, a set of already premade groups are given where people most likely have different accounts with passwords. Of course it's possible to add more and edit the already existing ones. Inserting and saving a password is very similar to how entry system works in databases. Once a desired group is selected, password entry can be made with different variables - the entry can be named, it can contain the user name, URL, notes, attachments, expiration date, icon and of course the password itself. Using the entries is very simple – Ctrl+B wil copy the username and Ctrl+C will copy the password. There are also shortcuts for copying the URL and they are customisable.

2.2.1 KEEPASS SECURITY

KeePass has an entire information section dedicated to security. From there, one can read all about their encryptions and security practices. Their database is encrypted with AES, with 128bit block sizes and 256bit keys. The 256bit keys are generated with SHA-256. If only a password is used for the database then the password plus a 128-bit salt are hashed using SHA-256 to form the final key. When using both password and key file, the final key is derived like this: SHA-256(SHA-256(password), key file contents).

```
private static byte[] GetSystemData(Random rWeak)
                {
                        MemoryStream ms = new MemoryStream();
                        byte[] pb;

                        pb = MemUtil.UInt32ToBytes((uint)Environment.TickCount);
                        ms.Write(pb, 0, pb.Length);

                        pb = TimeUtil.PackTime(DateTime.Now);
                        ms.Write(pb, 0, pb.Length);

#if !KeePassLibSD
                        try
                        {
                                Point pt = Cursor.Position;
                                pb = MemUtil.UInt32ToBytes((uint)pt.X);
                                ms.Write(pb, 0, pb.Length);
                                pb = MemUtil.UInt32ToBytes((uint)pt.Y);
                                ms.Write(pb, 0, pb.Length);
                        }
                        catch(Exception) { }
#endif
```

Code 1. Code sample of gathering system data

When generating random numbers, KeePass uses different sources: current tick count, performance counter, system date/time, mouse cursor position and so on. This method makes sure that the seeds and the resulting numbers are as random as possible. KeePass also has protection against dictionary attacks. No one can prevent those kind of attacks entirely, because there's nothing that would stop an attacker to just enter random passwords and try to decrypt the database. However, it can be made a lot slower to the point where it becomes so slow it simply isn't practical anymore. KeePass uses a method that they call "key transformation rounds". Basically, KeePass first hashes user's password using SHA-256 and then encrypts the result N times using AES and then hashes it again using SHA-256. Since AES transformation isn't pre-computable (key is random), all guesses have to go through the same process. This process can take a lot of time though. If it takes around 1 second then that it barely noticeable

to the legit user, but makes any kind of dictionary attack pretty much useless, because testing 60 passwords in a minute is not feasible. They also bring out in comparison that KeePassX only partially supports this kind of protection. The process memory protection is also described. KeePass keeps the sensitive encrypted in the memory, so even if a memory dump is taken, no sensitive data could be found. In addition, before releasing memory, KeePass will overwrite it. However, there are operations that will keep sensitive data unencrypted in the memory, such as showing password and showing data. KeePass also has protection against keyloggers, which is probably one of the main entry points to decrypt the database. KeePass can be set to show the master key dialog on a secure desktop that keyloggers cannot access [2].

2.3 KeePassX

KeePassX was originally known as KeePassL. It was forked from KeePass, because KeePass didn't support Linux in the past, so the new forked version seeked to solve that problem. However, nowadays both versions support both systems and since the developers felt the naming wasn't appropriate anymore, it got renamed to KeePassX.

KeePassX is a software made in C++. Source code doesn't come with any centralised solution file though, so one has to rely on Make or create the solution themselves. The user interface, being very similar to KeePass, is done using Qt libraries. Throughout the code, the entire software depends heavily on Qt libraries, mainly just datastructures and some custom functions. Database is realised with custom code, as in no third party solution is used.

Similarly to KeePass, at first you can only either open an already existing database or create a new one. Upon creation, there are 4 options – set a master password, set a key, set both or none of them (yes, not setting any is also an option). Once the database is done, there is one primary group Root which will contain all the passwords and sub-groups, which in turn can contain passwords. The idea and logic of inserting and storing passwords is almost identical to KeePass. The entries can contain the username, URL, icons, attachments, notes and other things in addition to password itself. Using passwords is exactly the same as in KeePass – once again Ctrl+B will copy the username and Ctrl+C copies the password.

The security of KeePassX is not that thoroughly explained as it was with KeePass. KeePass used KeePassX for comparsion when talking about dictionary attack prevention, mentioning that methods used by KeePassX are not as good. In terms of security, the whole database is encrypted with AES or Twofish encryption using a 256-bit key, so at least on paper it sounds good. After taking a deeper look at source code, there's a clear implementation for using different symmetric ciphers - AES, Twofish and Salsa20. Taking a closer look at salsa, the encryption looks solid.

```
for (i = 0;i < 16;++i) x[i] = input[i];
  for (i = 20;i > 0;i -= 2) {
    x[ 4] = XOR(x[ 4],ROTATE(PLUS(x[ 0],x[12]), 7));
    x[ 8] = XOR(x[ 8],ROTATE(PLUS(x[ 4],x[ 0]), 9));
    x[12] = XOR(x[12],ROTATE(PLUS(x[ 8],x[ 4]),13));
    x[ 0] = XOR(x[ 0],ROTATE(PLUS(x[12],x[ 8]),18));
    x[ 9] = XOR(x[ 9],ROTATE(PLUS(x[ 5],x[ 1]), 7));
    x[13] = XOR(x[13],ROTATE(PLUS(x[ 9],x[ 5]), 9));
```

```
    x[ 1] = XOR(x[ 1],ROTATE(PLUS(x[13],x[ 9]),13));
    x[ 5] = XOR(x[ 5],ROTATE(PLUS(x[ 1],x[13]),18));
    x[14] = XOR(x[14],ROTATE(PLUS(x[10],x[ 6]), 7));
    x[ 2] = XOR(x[ 2],ROTATE(PLUS(x[14],x[10]), 9));
    x[ 6] = XOR(x[ 6],ROTATE(PLUS(x[ 2],x[14]),13));
    x[10] = XOR(x[10],ROTATE(PLUS(x[ 6],x[ 2]),18));
    x[ 3] = XOR(x[ 3],ROTATE(PLUS(x[15],x[11]), 7));
    x[ 7] = XOR(x[ 7],ROTATE(PLUS(x[ 3],x[15]), 9));
    x[11] = XOR(x[11],ROTATE(PLUS(x[ 7],x[ 3]),13));
    x[15] = XOR(x[15],ROTATE(PLUS(x[11],x[ 7]),18));
    x[ 1] = XOR(x[ 1],ROTATE(PLUS(x[ 0],x[ 3]), 7));
    x[ 2] = XOR(x[ 2],ROTATE(PLUS(x[ 1],x[ 0]), 9));
    x[ 3] = XOR(x[ 3],ROTATE(PLUS(x[ 2],x[ 1]),13));
    x[ 0] = XOR(x[ 0],ROTATE(PLUS(x[ 3],x[ 2]),18));

if (!bytes) return;
  for (;;) {
    salsa20_wordtobyte(output,x->input);
    x->input[8] = PLUSONE(x->input[8]);
    if (!x->input[8]) {
      x->input[9] = PLUSONE(x->input[9]);
      /* stopping at 2^70 bytes per nonce is user's responsibility */
    }
    if (bytes <= 64) {
      for (i = 0;i < bytes;++i) c[i] = m[i] ^ output[i];
      return;
    }
    for (i = 0;i < 64;++i) c[i] = m[i] ^ output[i];
    bytes -= 64;
    c += 64;
    m += 64;
  }
```

Code 2 & 3. Code samples of byte conversion and encryption

2.3.1 AES VS. TWOFISH

KeePassX offers Twofish encryption instead of AES, which is more used with other password manager softwares and seems to be more popular overall. This obviously raises questions on why one should be preferred over the other and in which cases one might perform better. Discussions on the internet state that neither AES nor Twofish are considered as broken as far as public cryptography is concerned. AES has generally a bit more advantage due to being used more, because the implementation of AES is a bit easier. In addition, since it's used more, should there be any exploits found or should the scheme become broken, it will be announced quicker as there are more people working on it. This also increases the chance for patches

and fixes. AES can also benefit on high-end x86 systems and ARM CPUs since they include hardware acceleration. Twofish is considered more secure theoretically as it's unbreakable without brute-force and correct implementation of Twofish makes brute-force attacks really slow. This, however, has an impact on the overall performance, which is why AES is usually considered slightly better as it has higher performance and is secure enough [4] [5] [6].

2.4 LastPass

It's always interesting to find out what people that are using password managers on regular basis prefer. After asking around from my personal circle, it seems that one popular choice for password manager is LastPass [3]. This closed source software is available for free, but also provides a subscription based option with more features. The idea behind LastPass is that the user creates an account with them together with a strong master password. After that, the user can login to LastPass environment and use securely stored passwords anywhere. The only requirement is that the LastPass extension is installed on the desired browser. This is definitely a good option for someone that either uses multiple computers and/or devices, because with previously reviewed softwares user was tied to using their password database on one device only, unless portable option was used. However, portable option would have to be carried on a physical device, such as a USB stick, which might not be convenient enough.

In terms of usability, LastPass is intuitive. At first there's a guide that explains the environment and how to use it. Storing passwords is also very similar to KeePass solutions – the entries can be named, they can contain user names, URLs and notes. In fact, if a URL is provided, the page can be visited straight from the LastPass vault. Once on the webpage, the extension can detect user name and password fields and fill them automatically with correct strings. There's an option to make LastPass automatically fill them every time the site is visited, which makes it even more convenient.

2.4.1 LastPass security

There isn't much officially provided by LastPass like it was with KeePass. However, since LastPass stores passwords on an external vault instead of local device like KeePass softwares, the overall security deserves a more dedicated analysis. According to the information given on LastPass'es website, user data is encrypted and decrypted at device level and kept secret in the vault, meaning that not even LastPass workers know the exact contents of what's on their servers. In addition, the master password keys used to encrypt and decrypt are never sent to LassPass' servers, so in theory it looks good [1]. Users can't do much though about what's happening on the server side – it's the client side where users can make sure that their account and passwords stay safe by practicing good security procedures. For example, it's possible to use two-factor authentication, which has proved to be highly efficient against account hijacks. Furthermore, the master password should be kept as safe as possible, preferably memorised. Whenever visiting LastPass vault, one should pay attention that they really are on the correct site by observing the existing certificates.

A lot depends on the policy of LastPass as well and how they behave with their users' accounts. Any decent web service provider is at least a bit concerned about the security of their

site and their users. Considering the contents of LastPass and what their service is for, they are taking even more extra steps to ensure the safety of what their vaults hold. Back in 2011 they suspected that they might have been compromised, so they forced master password reset on all accounts just in case [7]. In 2015 LastPass announced that hackers managed to download some of the cryptographically protected data. Items downloaded were user passwords, salts, password reminders and e-mail addresses. However, since all that data was encrypted and there was no evidence that hackers managed to open any of the locked vaults, the damage was minimal. The inability to open any vaults is thanks to extremely slow hashing mechanism that requires a lot of computing power. In addition to these breaches, a cross-site scripting bug was also discovered back in 2011 that made acquiring e-mail addresses, password reminders, list of sites and IP addresses possible. The bug was quickly fixed [8].

Despite only having a few breach attempts and bugs with in several years, the debate whether it's smart to store passwords in cloud is wide open. Even if the storage is very tight in security like it seems to be with LastPass, several experts still say that cloud-based password storage is unsuitable.

2.4.2 YUBIKEY

LastPass and several other services offer pairing a Yubikey device for two-step authentication. Usually two-step authentication is handled by software solutions like an e-mail. More advanced solutions use SMS-es, which is more secure. There's still a chance for information leak with both cases, as e-mails can be hacked and phones might get lost or stolen. Yubikey is a purpose built two-step authentication tool that serves as the second doorway for accessing accounts. If a service supports Yubikey and it is set then upon login user will be asked to provide their Yubikey. This will prevent any kind of phishing, malware and other type of attacks, because attackers would also need the physical Yubikey device which they don't have [9].

3 CONCLUSION

Password managers have been around for some time now and the selection is quite wide. Users can select between locally stored managers and cloud storage and also portable options. Whichever is most suitable is up to each person as every option has its benefits. Whether one should use one though is a different question. In most cases it pays to use it, because password managers make it easier to use different passwords at different places, which adds security. However, as long as the user is careful enough and can think of long but easily remembered passwords then it's possible to work without a password manager as well. Most password managers are safe enough for social media accounts and various board accounts. Password managers shouldn't be used for bank accounts or any other high profile locations that have classified information or high level access, because these passwords shouldn't be written down anywhere. When it comes to open and closed source versions, in most cases open source versions might have better proof of security. For example, a thorough research was conducted on KeePass version 1.31, which is done in C++ [10]. This research performed in-depth analysis on the source code and is very different from this report. Although many aspects were checked,

the analysis concluded that there weren't any critical nor high level of security risks in the source code, aside from some medium and minor ones. Such analysis does not exist for every open source password manager though, let alone for closed source ones. Whether it should be used to favour KeePass 1.31 over other managers is up to each individual. Personally I'd choose something that would suit my needs best.

REFERENCES

[1] https://lastpass.com/enterprise/security/ [27.11.2016]

[2] http://keepass.info/help/base/security.html [08.11.2016]

[3] M. Pagel, K. Ilula, personal communication, November 2016

[4] http://crypto.stackexchange.com/questions/24192/twofish-vs-serpent-vs-aes-or-a-combo [02.12.2016]

[5] http://stackoverflow.com/questions/4147451/aes-vs-blowfish-for-file-encryption [07.12.2016]

[6] Ben Joan, "Difference Between AES and Twofish", http://www.differencebetween.net/technology/difference-between-aes-and-twofish/, August 2010 [07.12.2016]

[7] Whitson Gordon, "Is LastPass Secure? What Happens if It Gets Hacked?", http://lifehacker.com/is-lastpass-secure-what-happens-if-it-gets-hacked-1555511389, January 2014 [08.12.2016]

[8] Dan Goodin, "Hack of cloud-based LastPass exposes hashed master passwords", http://arstechnica.com/security/2015/06/hack-of-cloud-based-lastpass-exposes-encrypted-master-passwords/, June 2015 [09.12.2016]

[9] https://www.yubico.com/why-yubico/for-individuals/ [10.12.2016]

[10] Everis, "KeePass Password Safe", https://joinup.ec.europa.eu/sites/default/files/ckeditor_files/files/DLV%20WP6%20-01-%20KeePass%20Code%20Review%20Results%20Report_published.pdf October 2016 [16.12.2016]