Password Safes by pmateti@wright.edu • Work in progress. This article is exclusively for the personal use of Prabhaker Mateti. It is not ready even for my own students. 2019/06/28 00:52:59

Contents

Password Safes
Internal Memorandum – Work in Progress

Prabhaker Mateti
Wright State University, Dayton, OH 45435, USA
pmateti@wright.edu

2019/06/28

Abstract

(This article is exclusively for the personal use of Prabhaker Mateti. It is not ready even for my own students. This article hopes to come across as a proposal. Currently, it is a collection of notes and thoughts. Implementation has begun, but going very slow.)

This article is a survey of digital vault design, and how it could be distributed across machines, as in, e.g., IPFS. It also fully documents the architecture, design and implementations two examples of password safes: KeePassXC and Password-Safe. We include intuitive explanations as well as design-by-contract assertions.

Keywords: Password Safes, KeePassXC, Password-Server, Cloud Storage,

1 Comments by PM

This document is being written as if it is a paper for submission to a conference/ journal. It is **not** ready for submission anywhere **yet**. Of course, it is perfectly OK to request others to read this draft.

1. This version is not focused on good English. But, do be careful about lower-case/Upper-Case. E.g., Scala, not scala; Linux not linux.

   My focus, for now, is only on outlining what content this paper ought to have.

2. TBD is a place marker for To Be Done.

3. Note the usage of `\citep` versus `\cite`.

4. Spell check before sending this paper off. Also, submit the pdf for plagiarism check.

5. It is OK to leave the hyperrefs. Inserted several LaTeXmacros and packages. They do not change what Springer expects.

6. Only short segments of sentences or words are struck out. Larger sections are cut out but saved in `cuts.tex`.

7. Citations should never be present in an abstract.

8. It is traditional that names of files and such are in `typewriter` font.

9. Switching to bibliographystyle LaTeX/ ACM-Reference-Format-Journals-annote.bst has problems. Loses sorting some times.

2 Cuts

Only short segments of sentences or words are ~~struck out~~. Larger sections are cut out but saved in `cuts.tex`, this section.
End of Cuts.

1 Introduction

We are building a next-gen Android ROM [?] that assumes significantly powerful CPU and local storage capacity. Even with ever increasing capacities of local storage, this collected data will have to migrate to the cloud. In this paper, we are ignoring how large this can be, and how we may compress it. We also ignore the issues of upload/ download costs, possible lag, and energy consumption. We do enable permissions and encryption so that the data collected is "secure and private".

The present article limits itself to extending Android to include (large) collections of passwords, how to store them securely and privately, and how they can be shared across devices and individuals. To keep it better focused we even omit discussing the smart phone issues and focus on how to accomplish this on Linux.

1.1 Contributions

1.2 Requirements of Password Safes

1.2.1 Why Have Password Safes?

"Today you need to remember many passwords. You need a password for the Windows network logon, your e-mail account, your website's FTP password, online passwords (like website member account), etc. etc. etc. The list is endless. Also, you should use different passwords for each account. Because if you use only one password everywhere and someone gets this password you have a problem... A serious problem. He would have access to your e-mail account, website, etc. Unimaginable." [1]

"Security starts with you, the user. Keeping written lists of passwords on scraps of paper, or in a text document on your desktop is unsafe and is easily viewed by prying eyes (both cyber-based and human). Using the same password over and over again across a wide spectrum of systems and web sites creates the nightmare scenario where once someone has figured out one password, they have figured out all

[1] https://keepass.info/help/base/index.html

your passwords and now have access to every part of your life (system, e-mail, retail, financial, work)."[2]

1.2.2 What is a Vault?

A vault is an Encoded Database of Passwords

1.2.3 Master Key

1.2.4 Expected Usage

KeePass and Password Server shape our expected usage. Only the additional elements are described below.

1. Be paranoid about the collection of passwords and the master key. That is, assume that master key will be compromised, and the database may become corrupted.

1.3 Security and Privacy

Rodwald [2019] title=Attack on Students' Passwords, Findings and Recommendations,

2 Background

This section describes background ideas and technologies needed for password safes.

2.1 Expected Demise of Passwords

2.2 Password Safe Terminology

All password safes work with the following "data types" and functionality.

2.2.1 Collection of Passwords

An *entry* is a triplet: a context, a username and a (plain-text) password. All three are non-empty strings of bytes, possibly "encoded" or encrypted. A collection of passwords is actually a collection of entries. We may have multiple such collections. Each collection can be thought of as a table of three columns – that is, there should not be duplicate contexts.

[2]https://pwsafe.org/

2.2.2 Vault: Encoded Database of Passwords

2.2.3 Master Key

2.2.4 Watching for Password Input Events

2.3 Encryption Technology

We need to be aware of encryption technology and proper use of it. In this project, we do not expect make any contributions to this area.

2.4 DPAPI

"DPAPI (Data Protection Application Programming Interface) is a simple cryptographic application programming interface available as a built-in component in Windows operating systems." `https://en.wikipedia.org/wiki/Data_Protection_API`

On Linux, there is `libsecret`, "which seems to be much wider in scope than DPAPI. Whereas DPAPI has 2 methods (protect and unprotect), libsecret has a lot of moving pieces, ..."

2.5 ARC4

ARC4 is an RC4 cipher implementation for use in Python. `https://pypi.org/project/arc4/` "A small and insanely fast ARCFOUR (RC4) cipher implementation of Python. Strongly focused on performance; entire source code is written in C. Easily installable; single file with no dependency."

Cryptography with ARC4 (PIC18) by Brennen Ball, 2007

2.6 Yubi Key

`https://support.yubico.com/`
More to be written.

2.7 Proxy Re-Encryption

TBD Put together from a couple of web pages. Must rewrite. `https://coinremix.com/reviews/nucypher` has decent explanation, terminology,

Figure 1: YubiKey 5 Nano – $50 in 2019

and diagrams. https://www.smithandcrown.com/asset/nucypher/ Explain it even better.

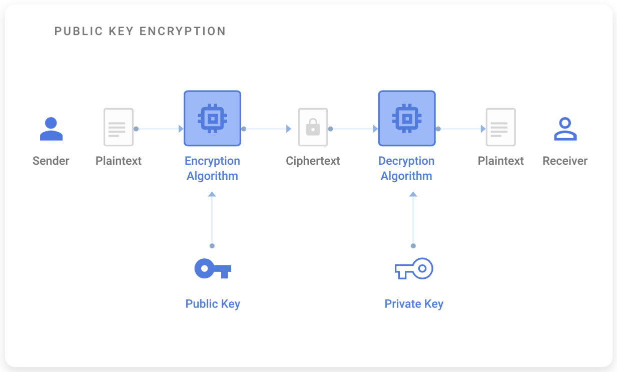

Figure 2: public-private encr

Public key encryption allows the sender to encrypt plaintext with the receiver's public key. The receiver can then decrypt the encrypted text with receiver's private key. The encrypted message can be viewed by others, but not decrypted except by the holder of the receiver's private key.

Proxy re-encryption is a way of encrypting a file for one recipient and then having a middle-party re-key that encrypted file, so that different recipients are able to decrypt it. This process allows for decryption rights to be granted or revoked as needed.

This has some use for sharing normal files, however once a third party has accessed a file, even if their rights to access it are revoked, they may have already copied over the data. The real benefits of access control are to streams of information.

How this works is perhaps best understood with an analogy. Bob buys a bunch of dogecoin. A year later, Bob is a dogecoin millionaire running a 24 x 7 livestream out of room 1001 at the Bellagio in Vegas. Alice decides that she wants to make an appearance on the livestream. Bob only has one key to the room, and he doesn't want Alice to be able to access the livestream all the time. So Bob asks Alice to send him a lock that she has the key to.

Bob's room, 1001, has an adjoining door to room 1002. Bob books room 1002 as well. He places the key for room 1002 inside a brief case, locks the briefcase with Alice's lock, and leaves it with reception. Bob then opens the door between the two rooms.

On arrival Alice collects the brief case, unlocks it, and gets the key to room 1002. Alice enters room 1002 with the key that was in the briefcase, walks through the adjoining door in to room 1001, and is able to appear on the livestream – all without access to Bob's room key.

The room key to 1002 (intermediate private key) on its own is not enough for Alice to get access to the livestream. The adjoining door between the two rooms needs to be opened by Bob as well. Bob can revoke Alice's access by closing the adjoining door. If room 1002 has a adjoining door to 1003, Alice can grant and revoke access to another person as well (multi-hop proxy re-encryption). This way she can invite other talented guests on to Bob's livestream, and Bob can still retain control of the show by closing and opening his adjoining door. That, in essence, is how proxy re-encryption works.

Enterprises have huge amounts of data in their "data lakes". They want to share their encrypted files in a data lake with other organizations but have each organization bring their own key. Multi-tenant, multi-source data lakes enable different organizations to collaborate.

This method allows for a number of applications such as e-mail forwarding, law-enforcement monitoring, and content distribution.

NuCypher is a proxy re-encryption network. Besafe is a centralized proxy re-encryption service This technology enables NuCypher to build a decentralised, trustless Key Management System (KMS). At its core, this helps developers and organisations manage access to their secure data more efficiently, and that opens up all sorts of possibilities.

2.8 KeePassXC

We have begun with the KeePassXC, which is a free and open-source password manager. It started as a community fork of KeePassX, itself a cross-platform port of KeePass. KeePass uses mono (C# on Linux), KeepassX uses mono, keepassXC uses C++, does not use mono.

2.9 Password Safe pwsafe

`PW-Safe` is designed by security technologist Bruce Schneier. For a comparison with KeePassXC, see the Section on Related Work. `https://en.wikipedia.org/wiki/Password_Safe` `https://privacyaustralia.net/passwordsafe-review/` `https://www.slant.co/versus/2824/7876/~keepass_vs_password-safe`

“Password Safe protects passwords with the Twofish encryption algorithm, a fast, free alternative to DES. The program’s security has been thoroughly verified by Counterpane Labs under the supervision of Bruce Schneier, author of Applied Cryptography and creator of the Twofish algorithm.”

2.10 Design by Contract

TBD Cite Bertrand Meyer.

TBD cite Mateti CEG 7140 DbC `https://cecs.wright.edu/~pmateti/Courses/7140/Lectures/OODesign/design-by-contract.html` `https://web1.cs.wright.edu/~pmateti/Courses/7140/Lectures/Design/correct-by-design.html`

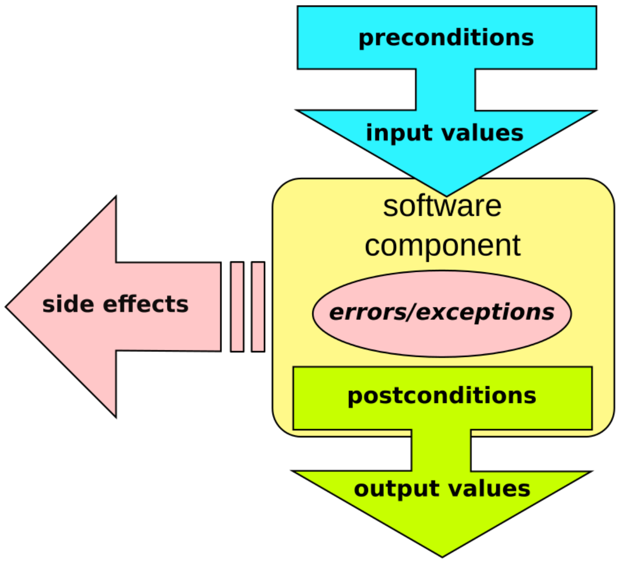

Figure 3: Design by Contract idea of Bertand Meyer. Source: Web

2.11 Replication

3 Architecture of General Vaults

3.1 Password Safe Data Vaults

All password safes work with the following “data types” and functionality.

3.1.1 Collection of Passwords

An *entry* is a triplet: a context, a username and a (plain-text) password. All three are non-empty strings of bytes, possibly “encoded” or encrypted. A collection of passwords is actually a collection of entries. We may have multiple such collections. Each collection can be thought of as a table of three columns – that is, there should not be duplicate contexts.

3.1.2 Vault: Encoded Database of Passwords

3.1.3 Master Key

3.1.4 Watching for Password Input Events

3.1.5 Security of the Vault

What would guarantee?

3.1.6 Privacy of the Vault

4 Comprehension of KeePassXC

Below is our deduced comprehension of KeePassXC from its source code of 121,510 lines.

4.0.1 Doxygen

It is helpful to browse the source using the doxygen-generated web site: `/usr/local/src/keepassxc-2.4.2/KeePassXC-Doxygen`

4.0.2 Source Files

The following is a sketch/ exploration of a design that one can deduce from the src code of KeePassXC and its roots.

1. Not much is written up by the authors of KeePass or KeePassXC. Dominik Reichl, author of original KeePass, writes this: `https://sourceforge.net/p/keepass/discussion/329220/thread/85359f2a/` 2011-12-14; "Parts of the code are documented within the source code files by XML comments, but there is no overall documentation of the architecture." However, a recent (Jan 2019) does help: `https://www.codeproject.com/Articles/5489/KeePass-Password-Safe-2` Dominik Reichl, Jan 2019.

2. What is presented below is gathered from multiple sources, especially from forums. Note that nearly all docs of KeePass are expected to be valid for KeePassXC.

   (a) `https://en.wikipedia.org/wiki/KeePass`
   (b) `https://keepassxc.org/docs/`
   (c) `https://keepass.info/help/base/index.html`
   (d) `https://keepass.info/help/v2_dev/plg_index.html`
   (e) `https://keepass.info/help/v2_dev/customize.html`
   (f) `https://keepass.info/help/v2_dev/plg_keyprov.html`
   (g) `https://keepass.info/help/kb/kdbx_4.html`
   (h) `https://keepass.info/help/base/security.html`
   (i) `https://keepass.info/help/base/firststeps.html`
   (j) `https://keepass.info/help/v2_dev/scr_kps_index.html` Scripting
   (k) `https://github.com/keepassxreboot/keepassxc/issues/2272` The issues files often discuss internals.

Build 34383 build cpp=33849,ansic=534

zxcvbn ansic=26267; Low-Budget Password Strength Estimation[3]

src-gui cpp=16250,xml=23

tests cpp=11539,xml=635

src-core cpp=8882

src-format cpp=4311

src-browser cpp=3366

src-crypto cpp=2500,ansic=634

src-autotype cpp=3057,python=56

src-keeshare cpp=2679

[3]`https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/wheeler`

src-cli cpp=1796

src-streams cpp=1154

top-dir sh=946

src-keys cpp=810

src-sshagent cpp=678

share xml=508,sh=60

src-totp cpp=234

src-proxy cpp=179

src-qrcode cpp=143

src-updatecheck cpp=142

utils sh=127

src-top-dir cpp=115

src-touchid cpp=29

4.0.3 Templates

More to be written.

4.0.4 Design by Contract Assertions

The KeePassXC code has no assertions. We intend to supply for major methods entry- and exit assertions.

4.1 Password Safe

In the words of Dr Schneier, the main designer of Password Safe, it "isn't as popular as the others. This is for two reasons: 1) I don't publicize it very much, and 2) it doesn't have an easy way to synchronize passwords across devices or otherwise store password data in the cloud."

We intend to provide design document for PWSafe also.

Password Safe Product Review by Nomad Mobile Research Centre, Aug 17, 1999.

5 Design of Password Safes

We would like to present a design description and also enhance the design of `KeePassXC` with Design-by-Contract, and other rigorous assertions stated using mathematical logic and discrete mathematics. [TBD cite Mateti CEG7140 Lecture Notes]

5.1 Terminology

An *entry* is a triplet: a context, a username and a (plain-text) password. All three are non-empty strings of bytes, possibly "encoded" or encrypted. A collection of passwords is actually a collection of entries. We may have multiple such collections. Each collection can be thought of as a table of three columns – that is, there should not be duplicate contexts.

5.2 Assertions

5.3 Replication of Vault

5.4 Synchronizing the Vaults

5.5 Security of the Vaults

1. Determine validity of a vault by consensus algorithm.

2. Never replicate the master key. Never reveal the location of the master key.

5.6 Privacy of the Vaults

6 Architecture of General Vaults

6.1 Password Safe Data Vaults

All password safes work with the following "data types" and functionality.

6.1.1 Collection of Passwords

An *entry* is a triplet: a context, a username and a (plain-text) password. All three are non-empty strings of bytes, possibly "encoded" or encrypted. A collection of passwords is actually a collection of entries. We may have multiple such collections. Each collection can be thought of as a table of three columns – that is, there should not be duplicate contexts.

6.1.2 Vault: Encoded Database of Passwords

6.1.3 Master Key

6.1.4 Watching for Password Input Events

6.1.5 Security of the Vault

What would guarantee?

6.1.6 Privacy of the Vault

7 Implementation of Password Safes

The following is a sketch of an implementation plan. We intend to use much of the existing src code of KeePassXC as-is in our implementation of KeePassXC-on-IPFS.

7.1 KeePassXC Source Code

```
Total Source Lines of Code (SLOC) = 121,510

Totals grouped by language
cpp:          91713 (75.48%)
ansic:        27435 (22.58%)
xml:           1166 (0.96%)
sh:            1140 (0.94%)
python:          56 (0.05%)

SLOC    Directory       SLOC-by-Language
34383   build           cpp=33849,ansic=534
26267   src_zxcvbn      ansic=26267
16273   src_gui         cpp=16250,xml=23
12174   tests           cpp=11539,xml=635
8882    src_core        cpp=8882
4311    src_format      cpp=4311
3366    src_browser     cpp=3366
3134    src_crypto      cpp=2500,ansic=634
3113    src_autotype    cpp=3057,python=56
2679    src_keeshare    cpp=2679
1796    src_cli         cpp=1796
1154    src_streams     cpp=1154
946     top_dir         sh=946
810     src_keys        cpp=810
678     src_sshagent    cpp=678
568     share           xml=508,sh=60
234     src_totp        cpp=234
179     src_proxy       cpp=179
143     src_qrcode      cpp=143
142     src_updatecheck cpp=142
127     utils           sh=127
115     src_top_dir     cpp=115
29      src_touchid     cpp=29
7       snap            sh=7
0       ci              (none)
0       cmake           (none)
0       docs            (none)
```

7.2 KeePassXC Refactored

7.3 Password Vault in IPFS

7.3.1 Role of NuCypher

8 Attacks

Schneier: We tried to code Password Safe not to leave plaintext passwords lying around in memory.
https://www.securityevaluators.com/casestudies/password-manager-hacking/
https://www.schneier.com/blog/archives/2019/02/on_the_security_1.html

8.1 Past Attacks

1. https://www.consumerreports.org/digital-security/everything-you-need-to-know-about-password-managers/
   Andrew Chaikivsky February 07, 2017

2. https://www.zdnet.com/article/critical-vulnerabilities-uncovered-in-popular-passwor
   Charlie Osborne — February 20, 2019

3. https://www.securityevaluators.com/casestudies/password-manager-hacking/ 2019

4. https://www.schneier.com/blog/archives/2019/02/on_the_security_1.html Bruce Schneier, 2019

8.2 Possible Attacks

Password Managers: Attacks and Defenses David Silver1 , Suman Jana1 , Eric Chen2 , Collin Jackson2 , and Dan Boneh1 1Stanford University 2Carnegie Mellon University.

https://sourceforge.net/p/keepass/discussion/329220/thread/d1bff5a460/?limit=250#1302 Selections below.

“The KDBX file format differs from how data is stored in the process memory. In KDBX files, everything is encrypted (using AES or ChaCha20), and sensitive data is additionally encrypted using ChaCha20 (Salsa20 in KDBX 3). This additional encryption prevents plain-text data being processed by the XML parser, i.e. it helps with memory protection while loading/saving the file. When loading a database, everything from the file is decrypted and sensitive data is immediately encrypted again using DPAPI (i.e. after loading the file, there is no second stage anymore, it’s just DPAPI).

Best regards, Dominik” 2019

While a database is open, KeePass remembers the master key components, encrypted using DPAPI. Other keys are erased immediately after the file has been loaded. Best regards, Dominik 2019

Well, given that KeePass encrypts sensitive data in its process memory using DPAPI (whose key is protected by Windows), crash dumps aren’t really interesting for an attacker. Furthermore, native crashes occur very rarely with KeePass 2.x, and when one happens, users typically have to confirm sending a report.

Anyway, excluding KeePass from Windows Error Reporting probably doesn’t have any disadvantages. Therefore, I’ve added this now (KeePass calls the WerAddExcludedApplication function [1]).

Here are the latest development snapshots for testing: KeePass 2.x: https://keepass.info/filepool/KeePass-190612-2.zip KeePass 1.x: https://keepass.info/filepool/KeePass-190612-1.zip

If successful, the WerAddExcludedApplication function seems to create an item in the `HKEY-CURRENT-USER\Software\Microsoft\Windows\Windows` Error ReportingExcludedApplications registry key. An item called ’KeePass.exe’ with value 1 seems to indicate a successful exclusion from Windows Error Reporting.

Note that this registry item is permanent, i.e. it’s not removed when KeePass exits (otherwise there would be problems when the user runs multiple KeePass instances).

Thanks and best regards, Dominik

[1] https://docs.microsoft.com/en-us/windows/desktop/api/Werapi/nf-werapi-weraddexcludedapplication

DPAPI only encrypts passwords due to performance constraints. KeePass natively will store the masterpassword in plain text. luckily Dominik added an option to not store it in plain text(hash will still be stored in memory but encrypted)

The option is on: Options -¿ Security - ¿ the last option.

I invoked stack overflow in KeePass (through a malformed plugin), and a dump was created.

Is the natively master password really stored in the plain text? As I understand, it is stored just encrypted by DPAPI. But some tools, like KeeThief, can invoke decryption of this data by injecting a shellcode in the KeePass process.

8.3 What Makes Password Safe Secure?

“So we claim to be secure. Why should you trust us? What steps do we take to back this claim? Well, here are a few.” The following is a mildly edited version of some paragraphs from https://pwsafe.org/readmore.shtml.

Open Source - Most important, you don’t have to take our word for it. You can download the source code and inspect it yourself, or have someone else check it for you. If you're really concerned, you can build the program from the sources you've downloaded and reviewed, instead of the binary files we build for each release.

For the files we provide , you can check that they're the ones that we've uploaded, and not tampered with, by checking the GPG cryptographic signature that's generate for each file.

Designed by Bruce Schneier - the original version was designed by renowned security expert Bruce Schneier, and we have his permission to say so.

No back door / recovery mechanism - there's no way for users (or developers, for that matter) to access the passwords without the master key.

Hard to brute-force - In the absence of back doors, an attacker can try a brute-force attack, e.g., using a dictionary. Password Safe has safeguards in place to make this as hard as possible.

The master passphrase is never stored the clear. We store something that derived from the master passphrase, but hard to calculate. When you enter your passphrase, we duplicate the calculation and compare the results. Only if the comparison succeeds do we continue to derive the encryption key from your master passphrase.

Sensitive memory is kept from swapping to disk.

All user data is encrypted in memory.

Memory with sensitive data is wiped as soon as possible.

File integrity checks : Even if the file's encrypted, it's not necessarily protected against unauthorized modification. Password Safe implements integrity checks on the file so that an attacker cannot modify it without knowing the master passphrase.

Reliability : Backups of previous databases are kept by default. The user can configure how many backups to keep and where to keep them.

TBD Is this a complete list??

8.4 Active Functionality Concerns

8.4.1 Security and Privacy of the Vault

8.4.2 Watching for Password Input Events

8.4.3 Supplying the Password

8.4.4 Synchronizing the Vault across Devices

9 Evaluation

This article is a proposal. Its implementation is progressing, but slowly.

Below we list a few items whose code complexity and performance we expect to evaluate. At a minimum, these are as follows.

9.1 Code Complexity

1. sloccount, numbers of methods, classes, etc
2. subjective rating of documentation
3. subjective rating of architecture diagrams
4. assertion count, subjective rating of entry- and exit assertion complexity

9.2 Performance

We intend to measure the

1. Time taken in the operation, as fine as possible
2. Peak usage of memory
3. Total amount network data sent/recd
4. Total change in persistent storage

9.3 Authentication Module

9.4 Security of the Vault

"While a database is open, KeePass remembers the master key components, encrypted using DPAPI. Other keys are erased immediately after the file has been loaded." – Dominik 2019 Methodically analyze this statement by carefully examining the code.

9.5 Security of the Password Safe Processes

10 Related Work

Arias-Cabarcos et al. [2016] title=Comparing Password Management Software: Toward Usable and Secure Enterprise Authentication,

10.1 Encryption Mechanisms

Zhang et al. [2016] title=Analysis of Encryption Mechanism in KeePass Password Safe 2.30,
https://www.securityevaluators.com/casestudies/password-manager-hacking/ we propose security guarantees password managers should offer

10.1.1 Proxy Re-Encryption

10.2 KeePass

10.3 KeePassXC

10.4 Password Server

10.5 Misc

https://developers.facebook.com/docs/messenger-platform/discovery/; https://www.quora.com/How-do-you-monitor-your-mobile-app-traffic; https://lp.smartlook.com/mobile-recordings/; https://www.linuxjournal.com/content/monitoring-android-traffic-wireshark; https://www.eviltester.com/2014/09/using-wireshark-to-observer-mobile-http.htmlShallow. Wants you take Technical Web Testing 101 $10 course.; https://support.google.com/admanager/answer/6206401 Capture mobile session traffic;

11 Conclusion

11.1 Acknowledgements

2019: Jayal Shah is working on bringing the vault to IPFS.

References

(????). Rodwald P. (2020) Attack on Students' Passwords, Findings and Recommendations. In: Zamojski W., Mazurkiewicz J., Sugier J., Walkowiak T., Kacprzyk J. (eds) Engineering in Dependability of Computer Systems and Networks. DepCoS-RELCOMEX 2019. Advances in Intelligent Systems and Computing, vol 987. Springer, Cham.

Alaa Alharbi, Khaled Elleithy, Rowida Alharbi, and Wafa Elmannai. 2015. A Highly Portable Enhanced Password Protection Environment. (2015).

Patricia Arias-Cabarcos, Andrés Marín, Diego Palacios, Florina Almenárez, and Daniel Díaz-Sánchez. 2016. Comparing Password Management Software: Toward Usable and Secure Enterprise Authentication. *IT Professional* 18, 5 (2016), 34–40. https://e-archivo.uc3m.es/bitstream/handle/10016/23827/comparing_IEEEITP_2016_ps.pdf?sequence=1.

Derk Barten. 2019. Client-Side Attacks on the LastPass Browser Extension. *Unknown* (2019). https://uvalight.net/~delaat/rp/2018-2019/p59/report.pdf.

Manuel Egele, David Brumley, Yanick Fratantonio, and Christopher Kruegel. 2013. An empirical study of cryptographic misuse in android applications. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 73–84.

Elizabeth A Gallagher. 2019. Choosing the Right Password Manager. *Serials Review* (2019), 1–4.

Michal Kavan. 2018. *Bezpečnostní analỳza programu KeePassXC*. B.S. thesis. České vysoké učení technické v Praze. Vypočetní a informační centrum.

Jaeho Lee, Ang Chen, and Dan S Wallach. 2019. Total Recall: Persistence of Passwords in Android.. In *NDSS*. `https://www.cs.rice.edu/~jl128/papers/ndss19-total_recall_jaeho_ang_dan.pdf`.

Avinay Mehta, Sayan Chakraborty, and Chetan Nagar. 2019. Sharing Of Key Using Geometric Progression. (02 2019).
:: Course Project; In this "KeePass" 2.30 sign in authentication [9] takes place four times of 'SHA-256' and by default there is 6000 AES algorithm rounds that will increase the cracking time of password ... It is due to the reason of secured ... ;;

Satyan G Pitroda. 2015. System and Method for Electronic Vault to Manage Digital Contents. (July 23 2015). US Patent App. 14/598,456.

Przemysław Rodwald. 2019. Attack on Students' Passwords, Findings and Recommendations. In *International Conference on Dependability and Complex Systems*. Springer, 425–434.

Vishal Shah, V Sushma, Shweta Hiremath, PC Manjunath, and others. 2018. File Synchronization between Digital Safes. *International Journal of Advanced Research in Computer Science* 9, Special Issue 3 (2018), 330.

Jinliang Shen, Shiming Gong, and Wencong Bao. 2018. Analysis of Network Security in Daily Life. *Information and Computer Security* 1, 1 (2018).

Jani Tenhivaara. 2018. Matkailijan tietoturvapaketti. (2018).

Hengwei Zhang, Jingxin Hong, and Jun Hu. 2016. Analysis of Encryption Mechanism in KeePass Password Safe 2.30. In *2016 10th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID)*. IEEE, 43–46.

Klaudia Zotzmann-Koch. 2019. Alternativen zu WhatsApp, Google, Facebook & Co. *Medienimpulse* 57, 1 (2019).

Ivan Albert Zudic and Neil Patrick Adams. 2018. Method and System for Master Password Recovery in a Credential Vault. (Aug. 30 2018). US Patent App. 15/445,308.