Pass

The Standard Unix Password Manager

Why a password manager?

- If you are asking that question, then you are one of the lucky few that have yet to be affected by cybercrime
  - More than ⅔ of adults have
  - 1.5+ Million victims per day
  - 18 victims per second
- Idk about you but I have more passwords then I can remember.
  - At one point I noticed I was getting lazy and my passwords began to look similar for many of the “throw away” sites.
  - The reality is, however, if someone determines one or more of those passwords, your “pattern” can be found, thus breaking all of them.

What constitutes a “safe” password?

- One you can’t remember
  - If you can’t, then no matter the situation, be it unlawful intrusion or torture, they cannot get that password out of you.
- The obvious answer to this then is random generation
  - “ND%^$fgrvRRrtg4%dfg”
- But this introduces another problem...
  - Remembering all those random generated strings

Enter: Password Managers

- Google Smart Lock (Google...yuck)
- Apple Keychain (Apple...even yuckier)
- 1Password (Closed and Proprietary)
- Dashlane (Closed and Proprietary)
- LastPass (Closed and Proprietary)
- KeePassX (Open source finally!)
    - Also, no cloud servers that you don’t have access to!
    - In my opinion still too complex and unnecessary

Pass

- About the simplest you can get beyond encrypting your passwords by hand on paper and putting them in a locked drawer
- Completely written in bash and following “Unix philosophy” (i.e. KISS)
    - It is literally less than 700 lines of code
- Each password lives inside of a gpg encrypted file whose filename is the title of the website or resource (or really whatever you want) that request the file.
- These encrypted files can be organized into meaningful folder hierarchies...or not, copied from computer to computer, and manipulated using standard cl utilities.
- Just a directory of files….nothing more….nothing less

Pass

- All the passwords live in ~/.password-store
- Trackable using git!
- Access to a few very useful built in commands
  - Many of these are just derived from other unix commands like gpg, tree, git.
  - Completion for bash, zsh, and fish
- The community is very strong, producing a lot of GUIs and clients for other platforms, as well as extension.

Pass - Getting Started

```
pass init "GPG Key ID"
mkdir: created directory ‘/home/sworley/.password-store’
Password store initialized for “GPG KEY ID”
```

> If in a team setting you can specify multiple GPG Key IDs and have different folders in the hierarchy encrypted using different keys!

```
pass git init
Initialized empty Git repository in /home/sworley/.password-store/.git/

pass git remote add origin git@github.com:sworley/allmypasswords.git
```

Pass - Usage

```
zx2c4@laptop ~ $ pass (pass ls)
Password Store
├── Business
│   ├── some-silly-business-site.com
│   └── another-business-site.net
├── Email
│   ├── donenfeld.com
│   └── zx2c4.com
└── France
    ├── bank
    ├── freebox
    └── mobilephone
```

Pass - Usage

```
zx2c4@laptop ~ $ pass Email/zx2c4.com
Sup3rh4x3rizmynam3

zx2c4@laptop ~ $ pass -c Email/zx2c4.com
Copied Email/jason@zx2c4.com to clipboard. Will clear in 45 seconds.

zx2c4@laptop ~ $ pass insert Business/cheese-whiz-factory
Enter password for Business/cheese-whiz-factory: omg so much cheese what am i gonna do
```

This also can use multi-line data if you want to specify some more juicy metadata in the file

Pass - Usage

“generate” can create new passwords using /dev/urandom internally

```
zx2c4@laptop ~ $ pass generate Email/gmail.com 15
The generated password to Email/gmail.com is:
$(-QF&Q=IN2nFBx
```

--no-symbols or -n and -c or --clip are options

```
zx2c4@laptop ~ $ pass rm Business/cheese-whiz-factory
rm: remove regular file ‘/home/zx2c4/.password-store/Business/cheese-whiz-factory.gpg’? y
removed ‘/home/zx2c4/.password-store/Business/cheese-whiz-factory.gpg’
```

Pass - Data Organization

- Usernames, Passwords, PINs, Websites, Metadata
- Pass does not force you into any schema
  - It is just a bunch of flat text files
  - A lot of users however do use the multiline approach for passwords:

Amazon/bookreader

Yw|ZSNH!}z"6{ym9pI
URL: *.amazon.com/*
Username: AmazonianChicken@example.com
Secret Question 1: What is your favorite childhood superhero? Spiderman
Phone Support PIN #: 84719

Pass - Data Organization

- Another approach is to use folders as the “site”
  - So in each folder you would have the following:
    - Amazon/bookreader/password
    - Amazon/bookreader/secretquestion1
    - Amazon/bookreader/secretquestion2
    - Amazon/bookreader/pincode
- Or another:
  - Amazon/bookreader for password
  - Amazon/bookreader.meta for the metadata
- The point is you can do whatever the hell you want

Pass - Extensions

- pass-tomb: manage your password store in a Tomb
- pass-update: an easy flow for updating passwords
- pass-import: a generic importer tool from other password managers
- pass-extension-tail: a way of printing only the tail of a file
- pass-extension-wclip: a plugin to use wclip on Windows
- pass-otp: support for one-time-password (OTP) tokens

Pass - Clients

- passmenu: an **extremely useful and awesome** dmenu script
- qtpass: cross-platform GUI client
- Android-Password-Store: Android app
- passforios: iOS app
- pass-ios: (older) iOS app
- passff: Firefox plugin
- browserpass: Chrome plugin
- Pass4Win: Windows client
- pext_module_pass: module for Pext
- gopass: Go GUI app
- upass: interactive console UI
- alfred-pass: Alfred integration
- pass-alfred: Alfred integration
- simple-pass-alfred: Alfred integration
- pass.applescript: OS X integration
- pass-git-helper: git credential integration
- password-store.el: an emacs package
- XMonad.Prompt.Pass: prompt for Xmonad

Pass - Migration Help

- 1password2pass.rb: imports 1Password txt or 1pif data
- keepassx2pass.py: imports KeepassX XML data
- keepass2csv2pass.py: imports Keepass2 CSV data
- keepass2pass.py: imports Keepass2 XML data
- fpm2pass.pl: imports Figaro's Password Manager XML data
- lastpass2pass.rb: imports Lastpass CSV data
- kedpm2pass.py: imports Ked Password Manager data
- revelation2pass.py: imports Revelation Password Manager data
- gorilla2pass.rb: imports Password Gorilla data
- pwsafe2pass.sh: imports PWSafe data
- kwallet2pass.py: imports KWallet data
- roboform2pass.rb: imports Roboform data
- password-exporter2pass.py: imports password-exporter data
- pwsafe2pass.py: imports pwsafe data
- firefox_decrypt: full blown Firefox password interface, which supports exporting to pass

Demo

References

The majority of this talk was taken directly from:

https://www.passwordstore.org