A ZX2C4 Project

pass

the standard unix password manager

Download

Introducing `pass`

Password management should be simple and follow Unix philosophy. With `pass`, each password lives inside of a `gpg` encrypted file whose filename is the title of the website or resource that requires the password. These encrypted files may be organized into meaningful folder hierarchies, copied from computer to computer, and, in general, manipulated using standard command line file management utilities.

`pass` makes managing these individual password files extremely easy. All passwords live in `~/.password-store`, and `pass` provides some nice commands for adding, editing, generating, and retrieving passwords. It is a very short and simple shell script. It's capable of temporarily putting passwords on your clipboard and tracking password changes using `git`.

You can edit the password store using ordinary unix shell commands alongside the `pass` command. There are no funky file formats or new paradigms to learn. There is bash completion so that you can simply hit tab to fill in names and commands, as well as completion for zsh and fish available in the completion folder. The **very active community** has produced many impressive **clients and GUIs for other platforms** as well as **extensions** for `pass` itself.

The `pass` command is extensively documented in its man page.

Tweet

Using the password store

We can list all the existing passwords in the store:

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

And we can show passwords too:

```
zx2c4@laptop ~ $ pass Email/zx2c4.com
sup3rh4x3rizmynam3
```

Or copy them to the clipboard:

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

There will be a nice password input dialog using the standard `gpg-agent` (which can be configured to stay authenticated for several minutes), since all passwords are encrypted.

We can add existing passwords to the store with `insert`:

```
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 handles multiline passwords or other data with `--multiline` or `-m`, and passwords can be edited in your default text editor using `pass edit pass-name`.

The utility can `generate` new passwords using `/dev/urandom` internally:

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

It's possible to generate passwords with no symbols using `--no-symbols` or `-n`, and we can copy it to the clipboard instead of displaying it at the console using `--clip` or `-c`.

And of course, passwords can be removed:

```
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'
```

If the password store is a git repository, since each manipulation creates a git commit, you can synchronize the password store using `pass git push` and `pass git pull`, which call `git-push` or `git-pull` on the store.

You can read more examples and more features in the man page.

Setting it up

To begin, there is a single command to initialize the password store:

```
zx2c4@laptop ~ $ pass init "ZX2C4 Password Storage Key"
mkdir: created directory '/home/zx2c4/.password-store'
Password store initialized for ZX2C4 Password Storage Key.
```

Here, `ZX2C4 Password Storage Key` is the ID of my GPG key. You can use your standard GPG key or use an alternative one especially for the password store as shown above. Multiple GPG keys can be specified, for using pass in a team setting, and different folders can have different GPG keys, by using `-p`.

We can additionally initialize the password store as a git repository:

```
zx2c4@laptop ~ $ pass git init
Initialized empty Git repository in /home/zx2c4/.password-store/.git/
zx2c4@laptop ~ $ pass git remote add origin kexec.com:pass-store
```

If a git repository is initialized, `pass` creates a git commit each time the password store is manipulated.

There is a more detailed initialization example in the man page.

Download

The latest version is 1.7.4.

Ubuntu / Debian

```
$ sudo apt-get install pass
```

Fedora / RHEL

```
$ sudo yum install pass
```

openSUSE

```
$ sudo zypper in password-store
```

Gentoo

```
# emerge -av pass
```

Arch

```
$ pacman -S pass
```

Macintosh

The password store is available through the Homebrew package manager:

```
$ brew install pass
```

FreeBSD

```
# pkg install password-store
```

Tarball

- Version 1.7.4
- Latest Git

The tarball contains a generic makefile, for which a simple `sudo make install` should do the trick.

Git Repository

You may browse the git repository or clone the repo:

```
$ git clone https://git.zx2c4.com/password-store
```

All releases are tagged, and the tags are signed with 0xA5DE03AE.

Data Organization

Usernames, Passwords, PINs, Websites, Metadata, et cetera

The password store does not impose any particular schema or type of organization of your data, as it is simply a flat text file, which can contain arbitrary data. Though the most common case is storing a single password per entry, some power users find they would like to store more than just their password inside the password store, and additionally store answers to secret questions, website URLs, and other sensitive information or metadata. Since the password store does not impose a scheme of it's own, you can choose your own organization. There are many possibilities.

One approach is to use the multi-line functionality of pass (`--multiline` or `-m` in `insert`), and store the password itself on the first line of the file, and the additional information on subsequent lines. For example, `Amazon/bookreader` might look like this:

```
Yw|ZSNH!}z"6{ym9pI
URL: *.amazon.com/*
Username: AmazonianChicken@example.com
Secret Question 1: What is your childhood best friend's most bizarre superhero fantasy? Oh god, Amazon, it's too awful to say...
Phone Support PIN #: 84719
```

This is the preferred organzational scheme used by the author. The `--clip` / `-c` options will only copy the first line of such a file to the clipboard, thereby making it easy to fetch the password for login forms, while retaining additional information in the same file.

Another approach is to use folders, and store each piece of data inside a file in that folder. For example `Amazon/bookreader/password` would hold bookreader's password inside the `Amazon/bookreader` directory, and `Amazon/bookreader/secretquestion1` would hold a secret question, and `Amazon/bookreader/sensitivecode` would hold something else related to bookreader's account. And yet another approach might be to store the password in `Amazon/bookreader` and the additional data in `Amazon/bookreader.meta`. And even another approach might be use multiline, as outlined above, but put the URL template in the filename instead of inside the file.

The point is, the possibilities here are extremely numerous, and there are many other organizational schemes not mentioned above; you have the freedom of choosing the one that fits your workflow best.

Extensions for `pass`

In order to faciliate the large variety of uses users come up with, `pass` supports extensions. Extensions installed to `/usr/lib/password-store/extensions` (or some distro-specific variety of such) are always enabled. Extensions installed to `~/.password-store/.extensions/COMMAND.bash` are enabled if the `PASSWORD_STORE_ENABLE_EXTENSIONS` environment variable is `true` Read the man page for more details.

The community has produced many such 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

Compatible Clients

The community has assembled an impressive list of clients and GUIs for various platforms:

- 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

Migrating to `pass`

To free password data from the clutches of other (bloated) password managers, various users have come up with different password store organizations that work best for them. Some users have contributed scripts to help import passwords from other programs:

- 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

Credit & License

`pass` was written by Jason A. Donenfeld of zx2c4.com and is licensed under the GPLv2+.

Contributing

This is a very active project with a healthy dose of contributors. The best way to contribute to the password store is to join the mailing list and send git formatted patches. You may also join the discussion in `#pass` on Libera.Chat.

Tweet