(EMT) Advanced File Encryption and Management Tool Documentation

By -
Aman Singh - aman20100@iiitnr.edu.in
Vinayak Shrivastava - vinayak20100@iiitnr.edu.in

I. Set Up

We will install the Pass program, set up a GPG key pair, and then set up a create repository to store everything in. Once that's done, we can walk through how to use Pass on a day-to-day basis.

II. Install Pass

The installation process is different for each operating system.

macOS: You'll need to have the Brew Package Manager installed first. Once you've got that installed, run the following:

```
brew install pass
```

Windows: You can install it into the Linux subsystem.

III. Create a GPG Key

The GPG key is what encrypts your passwords and keeps them safe. You need to have access to your GPG key to read any of your passwords. Following is the sample run of the application.

1. Start the key generation:

```
gpg --gen-key
> gpg (GnuPG) 1.4.20; Copyright © 2017 Free Software Foundation, Inc.
> This is free software: you are free to change and redistribute it.
> There is NO WARRANTY, to the extent permitted by law.
```

2. Select the RSA and RSA (default) option.

```
> Please select what kind of key you want:
> (1) RSA and RSA (default)
> (2) DSA and Elgamal
> (3) DSA (sign only)
> (4) RSA (sign only)
> Your selection?
1
```

Set the key size to 4096:

```
> RSA keys may be between 1024 and 4096 bits long.
> What key size do you want? (3048):
4096
```

4. Set when you want the key to expire. While having a key that doesn't expire is insecure, it does mean that you won't have to repeat this process all over again in a month.

```
> Please specify how long the key should be valid.
> 0 = key does not expire
> <n> = key expires in n days
> <n>w = key expires in n weeks
> <n>m = key expires in n months
> <n>y = key expires in n years
1m
```

You'll be asked to confirm that the expiration date is correct if you selected anything other than
0:
> Key expires at Mon 05 Dec 2022 03:12:25 PM IST
> Is this correct? (y/N)
Y
5. Next up is to add your details to the key. This gets handy when you have more than one key on your system, and you need to keep track of which one's which:
> Real name: Aman
> Email address: aman@email.com
> Comment: Key for Unix Pass
> You selected this USER-ID:
> "Aman (Key for Unix Pass) <aman@email.com>"
6. Enter O to select that everything's okay:
> Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit?
O

7. The last thing you've got to do is enter a strong password to protect your key. This way, if anyone steals your key, they can't use it without this password. Remember this, as you'll need to enter it every time you want to view or edit anything within Pass.

8. After a few seconds, you should see something like the following:
> gpg: key CA987727 marked as ultimately trusted
> public and secret key created and signed.
>
> gpg: checking the trustdb
> gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
> gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u
> gpg: next trustdb check due at 2019–08–05
> pub 4096R/CA987727 2019–07–06 [expires: 2019–08–05]
> Key fingerprint = 027E 5752 0AF7 C0B5 5E89 59B3 F873 743F CA98 7727
> uid Johnny (Unix Pass Manager) <johnny@email.com>
> sub 4096R/05750380 2019–07–06 [expires: 2019–08–05]
So that's your GPG key sorted, keep it safe! You won't be able to access your passwords without it. GPG keys are stored in different locations, depending on your operating system. So take a look into how your system deals with them.

Initialise your Password Store

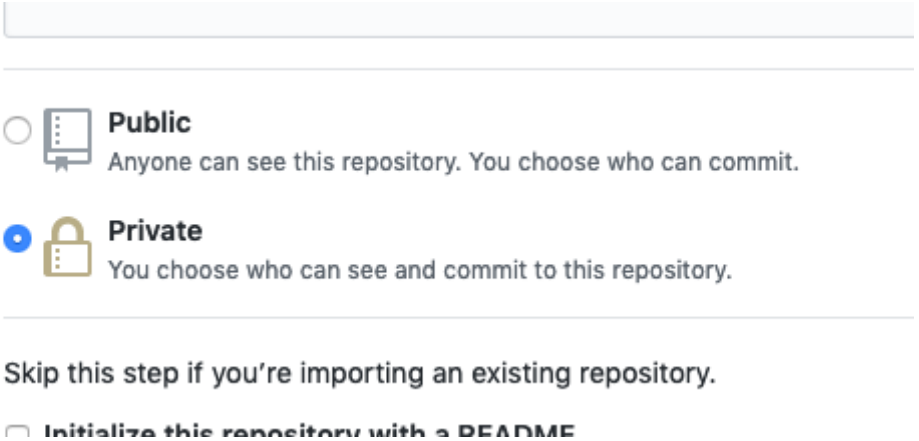
This step creates a blank database for you to store your passwords in. To initialise your password store, you'll need the name you gave to your GPG key. I used *Johnny* as my key name in the example above.

pass init Aman
> mkdir: created directory '/home/johnny/.password-store/'
> Password store initialized for Aman
Pass will make a new directory ~/.password-store and put your passwords in there.

Set Up GitHub Repository

Now that you've got your GPG key created and ready to use, we will create a private repository on GitHub to store your keys. This step is optional but incredibly useful if you use your password manager on multiple devices.

1. Go to GitHub (or whatever Git service you use) and create a new **private repository**.



Create a Private repository on GitHub.

2. Copy the repository URL to your clipboard.
3. Back in the terminal, add your Git repository to your password store:

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

4. Tell Git and Pass where your repository is stored by running this command and adding your repository's URL onto the end:

```
pass git remote add origin YOUR_GIT_URL
```

For example:

```
pass git remote add origin https://github.com/simplyuix/test-repo
```

Your password store is now linked to that repository. You can run Git commands within Pass by adding `pass` to the start of the command. We'll go into this a bit later.

Manage your Passwords

So you've got your password manager set up, but how do you use it? It might help to understand how Pass manages your passwords. Each password is defined as a single file, and all those files live within the default `password-store` folder. On Ubuntu and macOS, this is `~/.password-store`. Windows stores it in different places depending on which installation option you follow. If you're using the Linux subsystem, everything is likely stored within `~/.password-store` within that subsystem.

Because all the passwords are stored in files, they can be organized into folders for easier management. I like to keep my work life and personal life separate, so I have one folder for work and one for personal things:

```
tree ~/.password-store
> ~/password-store
> ├── personal
> └── work
```

The tree command is just a function that lists all the files and folders within a directory. It's found on most Unix systems. macOS users can install it by running brew install tree. At the moment, I don't have any passwords stored in Pass, so let's add one.

Add Password

1. Create a new password for GitHub:

```
pass add personal/github
```

2. Pass will ask you to enter the password twice.

```
> Enter password for personal/github:
> Retype password for personal/github:
```

3. You now have a password stored in Pass! You can see it by listing all the passwords in your password store with the pass command:

```
pass
> Password Store
> └── personal
> └── github
```

Pass can create passwords for you, too, so you don't have to think of a hard-to-crack password every time!

```
pass generate personal/github 18
The generated password for personal/github is
UWR9F$2Q"3E,l2={xG
```

The number at the end of the command is the length of the password. You can supply a bunch of arguments to the generate command. Here are a couple of helpful ones:

-c | Copy the password to your clipboard once it's been created. For example: pass generate -c personal/github 18
-n | Create a password using only alpha-numeric characters. For example: pass generate -n personal/github

View Password

To view a password, simply call the location of the password file with the passcommand:
pass personal/github
> UWR9F$2Q”3E,l2={xG

If you just want to copy your password straight to your clipboard without it showing in the terminal, add the —c argument.

pass -c personal/github

Edit Password

If you need to edit a password file, call the edit command, followed by the file you want to edit:
pass edit personal/github

The first time you try to edit a file, Pass will likely ask you which text editor you want to use. I use Vim because I hate myself, but you can use whatever you want. Except for Emacs. Never use Emacs.

When editing your file, remember that your password should **always be stored at the very top of the file**. The pass -c command will always copy the first line. You can add any other information to the file, such as your username or the URL of a website below the first line.

UWR9F$2Q”3E,l2={xG
login: aman@example.com
URL: https://github.com/login

Delete Password

If you have decided you no longer need a password, run the delete command and select the password or folder you want to delete:

pass delete personal/github
> Are you sure you would like to delete personal/github? [y/N]
Y
> removed ‘/home/johnny/.password-store/personal/github.gpg’
> [master ac89faa] Remove personal/github from store.
> 1 file changed, 0 insertions(+), 0 deletions(-)
> delete mode 100644 personal/github.gpg

You can also delete the file directly with the rm command:

rm ~/.password-store/personal/github

Using the integrated pass delete command is safer since there is less chance that you will corrupt your password store.

Save your changes to GitHub

To save your changes to your Git repository, run the regular `git` commands with the `pass` prefix:

```
pass git add .
pass git commit -m "New password"
pass git push
```

I added the following line to my `~/.bash_profile` to speed things up a little bit:

```
alias passpush="pass git add . && pass git commit -m 'New account.' && pass git push"
```

Now all I have to do is run `passpush` when I make any changes. I'm not too fussed about having proper commit messages since I'm the only person looking at this repository anyway.

Import from LastPass

Many people use LastPass as their password manager. I used to! Luckily, some Unix Pass community members have created a migration tool to help you import everything into Pass.

First up, you need to export your LastPass password library:

1. Go to LastPass.com and log in.
2. Click **More Options** at the bottom left.
3. Click **Advanced** > **Export**.
4. Enter your password again.
5. Copy the text and save it as a `.csv` file somewhere on your computer, preferably your `~/Downloads` folder.

Now that you've downloaded all your passwords, we can import them using a script.

1. Download the script to your root `~` folder:

```
cd ~/
```

2. Call the script, and enter the destination of your LastPass export file as an argument:

```
./lastpass2pass.rb path/to/passwords_file.csv
```

For example:

```
./lastpass2pass.rb ~/Downloads/passwords_from_lastpass.csv
```

3. The script should output something like this:

```
Reading '/home/johnny/Downloads/passwords_from_lastpass.csv'…
6 records found!
Records parsed: 6
…
done!
6 records were successfully imported!
```

You should be able to view all those records within `Pass`.