ZuluCrypt

https://hacktheplanetwith.me
https://cryptoden.pw

Definitions:
Encrypted File Container (EFC)

Contents:

[1.a] Introduction To ZuluCrypt

ZuluCrypt is a simple encryption program that is currently available on Linux only. It can encrypt entire drives, file containers, and individual files. In this tutorial I’m going to be showing you how to make encrypted file containers and encrypting individual files in ZuluCrypt.

Encrypted file containers act similarly to a folder. This means that you can move files in and out of the container just as you would with a regular folder.

[1.b] Installation of ZuluCrypt

Please Note: You could just do “sudo apt-get install zulucrypt” if running on Debian. However the version in the Debian repository may be out of date.

(1) Open up your web browser and go to https://mhogomchungu.github.io/zuluCrypt/.

(2) Scroll down to “binary packages” (or ctrl+f it) and download your appropriate package. I’m using Debian 8 Jessie so I’ll be downloading the Debian file (currently called: zuluCrypt-5.1.0-debian-8-Jessie.tar.xz).

(3) When the download is done open up your terminal and do the following commands. This is assuming you have the file in your downloads folder.

```
$ cd ~/Downloads
$ tar -xvf zuluCrypt-5.1.0-debian-8-jessie.tar.xz *
$ cd zuluCrypt-5.1.0-debian-8-jessie *

32-Bit:
$ cd i386/
$ ls
$ dpkg -i libzulucrypt1_5.1.0.1483245598.302e6c4_i386.deb **
$ dpkg -i libzulucrypt-dev_5.1.0.1483245598.302e6c4_i386.deb **
$ dpkg -i zulucrypt-cli_5.1.0.1483245598.302e6c4_i386.deb **
$ dpkg -i zulucrypt-gui_5.1.0.1483245598.302e6c4_i386.deb **

64-Bit:
$ cd amd64/
$ ls
$ dpkg -i libzulucrypt1_5.1.0.1483245598.302e6c4_amd64.deb **
$ dpkg -i libzulucrypt-dev_5.1.0.1483245598.302e6c4_amd64.deb **
$ dpkg -i zulucrypt-cli_5.1.0.1483245598.302e6c4_amd64.deb **
$ dpkg -i zulucrypt-gui_5.1.0.1483245598.302e6c4_amd64.deb **

* = dependant on the file you downloaded
** = when you do “ls” it’ll list all files; replace accordingly.
```

[2.a] Creating an Encrypted File Container

Depending on your distro zuluCrypt might be in the your *menu*. However you could also just do *zuluCrypt-gui* in the terminal and it’ll launch zuluCrypt.

(1) In the top bar you’ll see a column called “Create”, click on it and then hit “Encrypted Container in a File.”

(2) A new window will pop up with multiple fields, fill them out accordingly.

Example:

```
File Name: MyContainer
File Path: /home/johnathon/MyContainer
File Size: 2048 MB
```

The progress bar may freeze for a minute, this is normal.

(3) When the progress bar reachs 100% a new window will pop up. The window may appear to freeze, Give it a few moments and a popup timer should show up.

Example 1:

```
Path To File: /home/johnathon/MyContainer
Key
Key: MyPassword123
Repeat Key: MyPassword123
Volume Type: luks
             aes.xts-plain64.512.sha512
File System: ext4
RNG: /dev/urandom
```

Example 2:

```
Path To File: /home/johnathon/MyContainer
Key /or/ Keyfile (Read Below)
Key: MyPassword123
Repeat Key: MyPassword123
Volume Type: Normal+Hidden VeraCrypt
             Aes.xts-plain64.256.sha512
Key
Key: MyPassword1234
Repeat Key: MyPassword1234
Volume Size: 2000MB (must be smaller than master container)
File System: vfat
RNG: /dev/urandom
```

For your key(s) you need to use something complex, if you use a basic password like “MyPassword123” you’re on the wrong site. You could also use a keyfile. A keyfile is a file in place of the password that is usually kept on a USB drive. However if you lose that USB, you’ll lose access to your container; So use at your own discretion. A fix to that issue would be the key + keyfile option, however zuluCrypt sometimes bitchs out when using both.

For Example 2 you get a hidden container inside the main container. The hidden container will take longer to open. I’d recommend only using the hidden container and then putting useless shit in the main container. Do not overfill the two containers (If you fill the main container then put stuff in the hidden container you run the risk of corrupting data.)

[2.b] Mounting an Encrypted File Container

If you used LUKS (Example 1) for your container you’ll click on Open and then “Plain, LUKS, TrueCrypt Container in a File”. It will ask for volume path, mount name, and key.

```
Volume Path: /home/johnathon/MyContainer
Mount Name: MyContainer
Key: MyPassword123
```

If you used VeraCrypt (Example 2) for your container you’ll click on Open and then “VeraCrypt Container in a File”. It is similar however depending on the key you enter you’ll open the main or hidden container.

```
Volume Path: /home/johnathon/MyContainer
Mount Name: MyContainer
Key: MyPassword123 (Main) /or/ MyPassword1234 (Hidden)
```

It will mount the container as if it was a USB drive. It’ll go to /media/johnathon/MyContainer for me.

[3.a] Encrypting a File Using ZuluCrypt

First click zC and then Encrypt A File in your menu bar (Figure 3.1). Upon clicking a window will pop up asking for the file to encrypt. I’ve selected a file for encryption in Figure 3.2 and assigned a password to it and clicked on the create button.

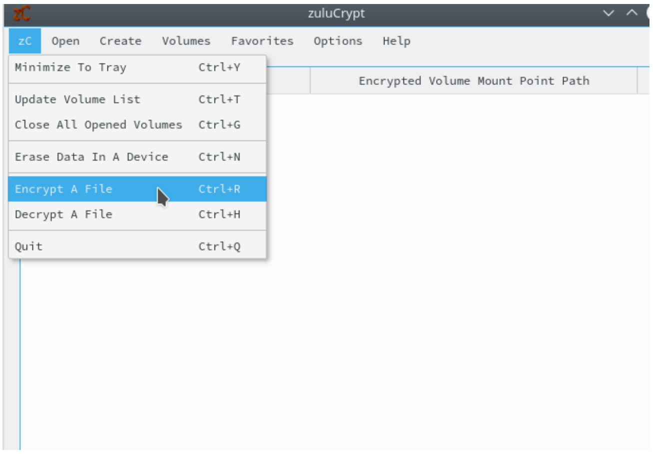
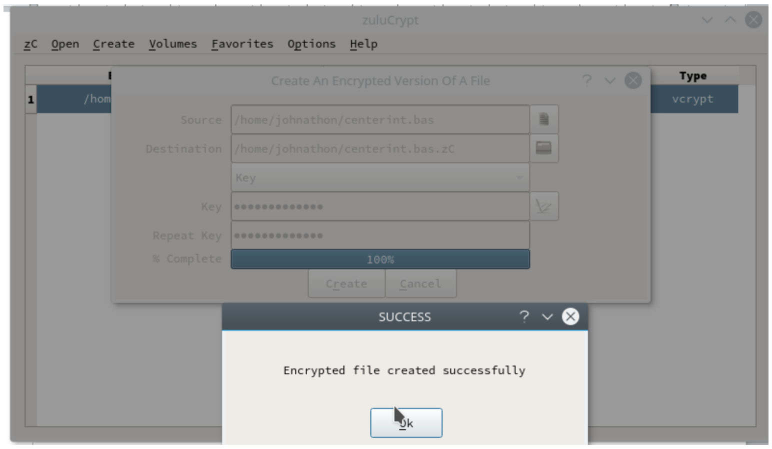


Figure [3.1]



Create An Encrypted Version Of A File

Source: /home/johnathon/centerint.bas

Destination: /home/johnathon/centerint.bas.zC

Key: •••••••••••••

Repeat Key: •••••••••••••

% Complete: 100%

SUCCESS

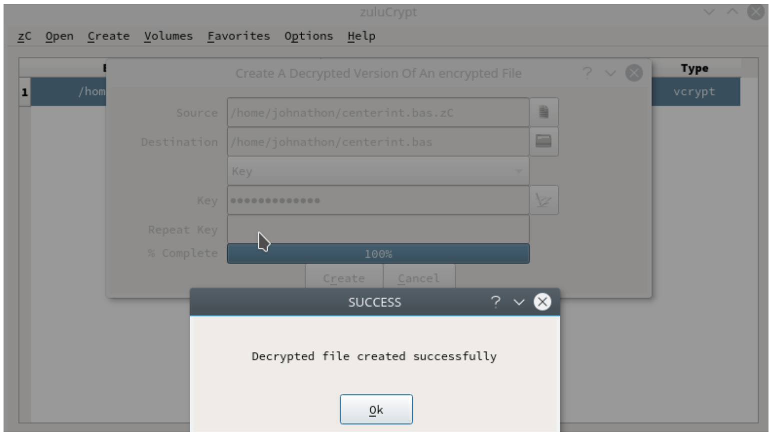
Encrypted file created successfully

Figure [3.2]

When the encrypted file is created it is recommended to remove the plain file. The encrypted file will most be called the original file name with a .zC extension. This can be renamed if desired.

[3.b] Decrypting a File Using ZuluCrypt

First click zC and then “Decrypt A File” in your menu bar (Figure 3.1). Upon clicking a window will pop up asking for the file to decrypt. In Figure 3.3 I’ve selected my encrypted file and entered the key for it. It will decrypt in the same directory as the encrypted file as it’s original form.



Create A Decrypted Version Of An encrypted File

Source: /home/johnathon/centerint.bas.zC

Destination: /home/johnathon/centerint.bas

Key: •••••••••••••

Repeat Key:

% Complete: 100%

SUCCESS

Decrypted file created successfully

Figure [3.3]