LiME ~ Linux Memory Extractor

A Loadable Kernel Module (LKM) which allows for volatile memory acquisition from Linux and Linux-based devices, such as Android. This makes LiME unique as it is the first tool that allows for full memory captures on Android devices. It also minimizes its interaction between user and kernel space processes during acquisition, which allows it to produce memory captures that are more forensically sound than those of other tools designed for Linux memory acquisition.

This is the Github repo link for LiME:-

504ensicsLabs/LiME
LiME (formerly DMD) is a Loadable Kernel Module (LKM), which allows the acquisition of volatile memory from Linux and…
github.com

We can simply download the source code and compile it to binary files with make.

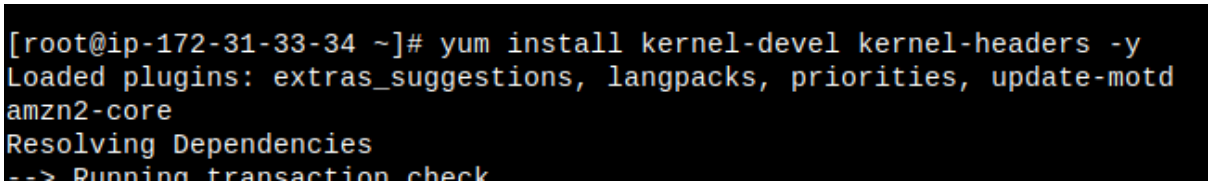
Note that I will be compiling the source code on the same machine where I want to read ram but in the real world, specifically on the crime scene where you want to do ram acquisition of criminals machine you should not compile the source code on criminals machine because if you do ram data might be overridden and you may lose critical data for proof.

I am using AWS ec2 instance with Amazon Linux 2 AMI to perform ram acquisition but you can do this on any Linux based O.S

We will also need to install kernel headers to do ram acquisition.

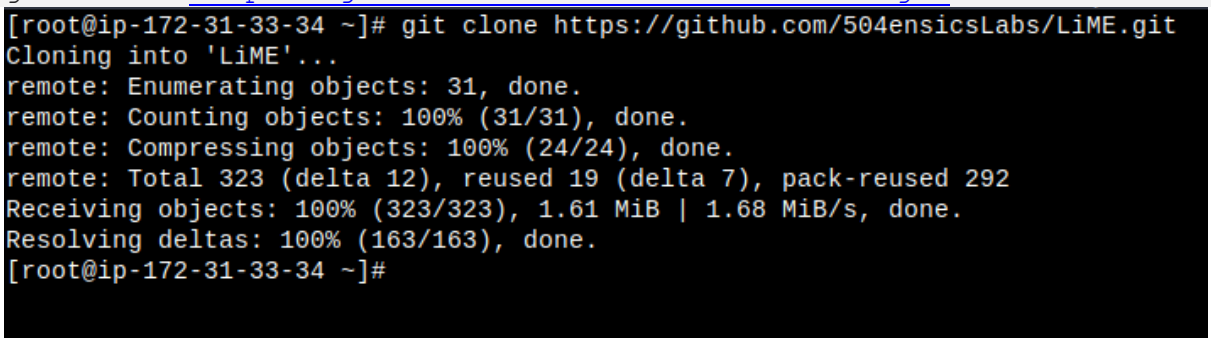
```
yum install kernel-devel kernel-headers -y
```

You can use the above command to install kernel-headers



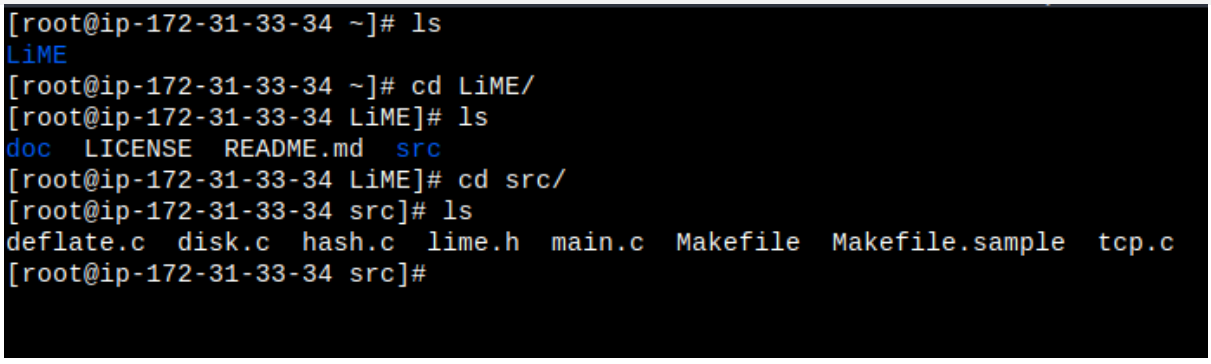
Now we have to clone the GitHub repo of LiME

```
git clone https://github.com/504ensicsLabs/LiME.git
```



Now we can compile the source code of LiME... first, we need to navigate to the src directory

```
cd LiME/src
```



Now we can simply type the "**make**" command it will compile the source code and give us a loadable kernel object file

```
make
```

```
[root@ip-172-31-43-238 src]# make
make -C /lib/modules/4.14.198-152.320.amzn2.x86_64/build M="/root/LiME/src" modules
make[1]: Entering directory `/usr/src/kernels/4.14.198-152.320.amzn2.x86_64'
  CC [M]  /root/LiME/src/tcp.o
  CC [M]  /root/LiME/src/disk.o
  CC [M]  /root/LiME/src/main.o
  CC [M]  /root/LiME/src/hash.o
  CC [M]  /root/LiME/src/deflate.o
  LD [M]  /root/LiME/src/lime.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /root/LiME/src/lime.mod.o
  LD [M]  /root/LiME/src/lime.ko
make[1]: Leaving directory `/usr/src/kernels/4.14.198-152.320.amzn2.x86_64'
strip --strip-unneeded lime.ko
mv lime.ko lime-4.14.198-152.320.amzn2.x86_64.ko
[root@ip-172-31-43-238 src]#
```

Source code has been compiled and we get a .ko extension file that is the nothing but a kernel object now we need to insert or load this kernel object but first let generate some data in ram so once we dump ram data we can verify it

```
[root@ip-172-31-43-238 src]# ls
deflate.c  hash.c                                lime.mod.c  main.o          Module.symvers
deflate.o  hash.o                                lime.mod.o  Makefile        tcp.c
disk.c     lime-4.14.198-152.320.amzn2.x86_64.ko lime.o      Makefile.sample tcp.o
disk.o     lime.h                                main.c      modules.order
[root@ip-172-31-43-238 src]#
```

We can start Python REPL and can create a list variable, because every book, teachers, article says that variable resides in RAM but no one show today we will verify if that's true.

I am creating a list with my name in python REPL

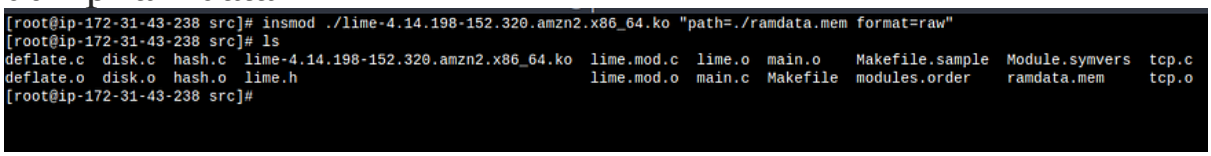
```
[root@ip-172-31-43-238 ~]# python
Python 2.7.18 (default, Aug 27 2020, 21:22:52)
[GCC 7.3.1 20180712 (Red Hat 7.3.1-9)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> kuldeep = ["i resides in RAM",10,20,30,40,50]
>>> test = 10
>>>
```

Now let insert or load the kernel object...

```
insmod ./lime-4.14.198-152.320.amzn2.x86_64.ko
"path=./ramdata.mem format=raw"
```

insmod command will insert the kernel object and it will dump the ram data at the path we specified and there are different formats for memory file I am here using the raw format.

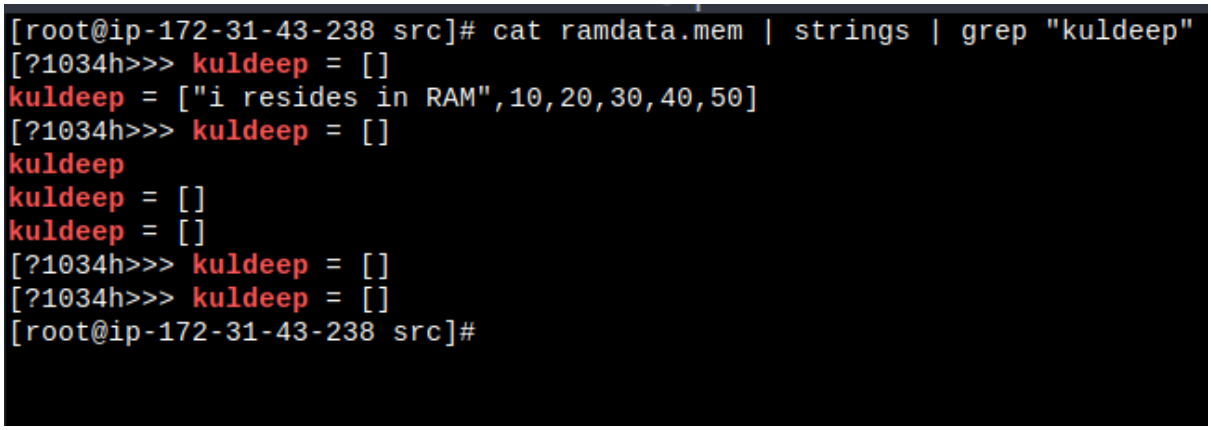
Depending on the ram size and disk I/O speed it will take time to dump ram data



In the above image a ***ramdata.mem*** file is created that contain all the ram data at that point in time now we can verify it that the python variable we created earlier resides in ram or not

```
cat ramdata.mem | strings | grep "kuldeep"
```

we can cat the ramdata.mem and pipe it to strings because ram contains data in binary or other encodings so strings will convert it into a string and then we can grep with the variable name



You see that we verified that the variable we created earlier present in RAM, now we have the memory file we can do a more detail analysis on it like we can get all photos which reside in ram using ***potorec*** tool or we can use the ***volatility*** framework for more granular analysis with volatility you can get detailed about every process, can get details about CPU caches or every network connection details, socket information, website info, caches, tokens, passwords, usernames, encrypted disk data and a lot of other things