
# LiME – Linux Memory Extractor

Instructions v1.3

July 7, 2014

# Contents

# 1.0 Introduction

LiME (formerly DMD) is a Loadable Kernel Module (LKM)that allows the acquisition of volatile memory from Linux and Linux-based devices, such as those powered by Android. The tool supports acquiring memory either to the file system of the device or over the network. LiME is unique in that it is the first tool that allows full memory captures from Android devices. It also minimizes its interaction between user and kernel space processes during acquisition, which allows it to produce memory dumps that are more forensically sound than those of other tools designed for Linux memory acquisition.

LiME was first announced at ShmooCon 2012. The video of the presentation can be found here, and the slides are available for download here.

# 2.0 Obtaining LiME

To obtain LiME, please see the instructions at: http://code.google.com/p/lime-forensics.

# 3.0 Compiling LiME

## 3.1 Compiling LiME for Linux

LiME is a Loadable Kernel Module (LKM). LiME ships with a default Makefile that should be suitable for compilation on most modern Linux systems.

For detailed instructions on using LKM see the file Documentation/kbuild/modules.txt included with the Linux kernel source.

## 3.2 Compiling a Debug Version of LiME

When compiling LiME with the default Makefile, using the command "make debug" will compile a LiME module with extra debug output. The output can be read by using the *dmesg* command on Linux.

## 3.3 Cross-Compiling LiME for Android

In order to cross-compile LiME for use on an Android device, additional steps are required.

**PREREQUISITES**

*Disclaimer: This list may be incomplete. Please let us know if we've missed anything.*

- Install the general android prerequisites found here.
- Download and un(zip|tar) the android NDK found here.
- Download and un(zip|tar) the android SDK found here.
- Download and untar the kernel source for your device. This can usually be found on the website of your device manufacturer or by a quick Google search.
- Root your device. In order to run custom kernel modules, you must have a rooted device.
- Plug the device into computer via a USB cable.

**SETTING UP THE ENVIORNMENT**

In order to simplify the process, we will first set some environment variables. In a terminal, type the following commands.

```
$ export SDK_PATH=/path/to/android-sdk-linux/
$ export NDK_PATH=/path/to/android-ndk/
$ export KSRC_PATH=/path/to/kernel-source/
$ export CC_PATH=$NDK_PATH/toolchains/arm-linux-androideabi-
4.4.3/prebuilt/linux-x86/bin/
$ export LIME_SRC=/path/to/lime/src
```

**PREPARING THE KERNEL SOURCE**

We must retrieve and copy the kernel config from our device.

```
$ cd $SDK_PATH/platform-tools
$ ./adb pull /proc/config.gz
$ gunzip ./config.gz
$ cp config $KSRC_PATH/.config
```

Next we have to prepare our kernel source for our module.

```
$ cd $KSRC_PATH
$ make ARCH=arm CROSS_COMPILE=$CC_PATH/arm-eabi- modules_prepare
```

**PREPARING THE MODULE FOR COMPILATION**

We need to create a Makefile to cross-compile our kernel module. A sample Makefile for cross-compiling is shipped with the LiME source. The contents of your Makefile should be similar to the following:

```
obj-m := lime.o
lime-objs := main.o tcp.o disk.o
KDIR := /path/to/kernel-source
PWD := $(shell pwd)
CCPATH := /path/to/android-ndk/toolchains/arm-linux-androideabi-
4.4.3/prebuilt/linux-x86/bin/
default:
 $(MAKE) ARCH=arm CROSS_COMPILE=$(CCPATH)/arm-eabi- -C $(KDIR)
M=$(PWD) modules
```

**COMPILING THE MODULE**

```
$ cd $LIME_SRC
$ make
```

# 4.0 Using LiME

To illustrate the use of LiME, we will now walk through two examples of acquiring memory from an Android device. We will first discuss the acquisition of memory over a TCP connection, followed by a discussion of acquiring a memory dump via the device's SD card. The use of LiME on other Linux devices is similar; however, the use of the Android debug bridge (*adb*) is not needed.

## 4.1 LiME Parameters

Starting in version 1.1, LiME now supports multiple output formats, including a custom lime format which integrates with Volatility's new lime address space. This means that additional parameters are needed when installing the LiME kernel module.

NOTE: There is a bug in the *insmod* utility on some Android devices. Multiple kernel module parameters must be wrapped in quotation marks, otherwise only the first parameter will be parsed. See sections 4.2 and 4.3 for examples.

**LiME Parameters**

- path
  - Either a filename to write on the local system (SD Card) or tcp:<port>
- format
  - raw
    - Simply concatenates all System RAM ranges
  - padded
    - Pads all non-System RAM ranges with 0s, starting from physical address 0
  - lime
    - Each range is prepended with a fixed-sized header which contains address space information
    - Volatility address space developed to support this format
- dio (optional)
  - 1 to enable Direct IO attempt (default), 0 to disable

## 4.2 Acquisition of Memory over TCP

The first step of the process is to copy the kernel module to the device's SD card using the Android Debug Bridge (*adb*), which supports a number of interactions with an Android device tethered via USB. We then use *adb* to setup a port-forwarding tunnel from a TCP port on the device to a TCP port on the local host. The use of *adb* for network transfer eliminates the need

to modify the networking configuration on the device or introduce a wireless peer—all network data is transferred via USB. For the example below, we have chosen TCP port 4444. We then obtain a root shell on the device by using *adb* and *su*. To accomplish this, we run the following commands with the phone plugged into our computer and debugging enabled on the device.

```
$ adb push lime.ko /sdcard/lime.ko
$ adb forward tcp:4444 tcp:4444
$ adb shell
$ su
#
```

Memory acquisition over the TCP tunnel is then a two-part process. First, the target device must listen on a specified TCP port and then we must connect to the device from the host computer. When the socket is connected, the kernel module will automatically send the acquired RAM image to the host device.

In the *adb* root shell, we install our kernel module using the *insmod* command. To instruct the module to dump memory via TCP, we set the path parameter to "tcp", followed by a colon and then the port number that *adb* is forwarding. On our host computer, we connect to this port with *netcat* and redirect output to a file. We also select the "lime" formatting option. When the acquisition process is complete, LiME will terminate the TCP connection.

The following command loads the kernel module via *adb* on the target Android device:

```
# insmod /sdcard/lime.ko "path=tcp:4444 format=lime"
```

On the host, the following command captures the memory dump via TCP port 444 to the file "ram.lime":

```
$ nc localhost 4444 > ram.lime
```

## 4.3 Acquisition of Memory to Disk (SD-Card)

In some cases, such as when the investigator wants to make sure no network buffers are overwritten, disk-based acquisition may be preferred to network acquisition. To accommodate this situation, LiME provides the option to write memory images to the device's file system. On Android, the logical place to write is the device's SD card.

Since the SD card could potentially contain other relevant evidence to the case, the investigator may wish to image the SD card first in order to save unallocated space. Unfortunately, some Android phones, such as the HTC EVO 4G and the Droid series, place the removable SD card to

be either under or obstructed by the phone's battery, making it impossible to remove the SD card without powering off the phone (these phones will power down if the battery is removed, even if they are plugged into a power source!). For this reason, the investigator needs to first image the SD card, and then subsequently write the memory image to it. While this process violates the typical "order of volatility" rule of thumb in forensic acquisition, namely, obtaining the most volatile information first, it is necessary to properly preserve all evidence.

Fortunately, imaging the SD card on an Android device that will be subjected to live forensic analysis (including memory dumping) does not require removal of the SD card. Tethering the device to a Linux machine, for example, and activating USB Storage exposes a /dev/sd? device that can be imaged using traditional means (e.g., using dd on the Linux box). Activating USB Storage mode unmounts the SD card on the Android device, so a forensically valid image can be obtained.

With USB Storage mode deactivated, we copy the LiME kernel module to the device using the same steps described in the last section. When installing the module using *insmod*, we set the path parameter to /sdcard/ram.lime to specify the file in which to write the memory dump. We also select the "lime" format option:

```
# insmod /sdcard/lime.ko "path=/sdcard/ram.lime format=lime"
```

Once the acquisition process is complete, we can power down the phone, remove the SD card from the phone, and transfer the memory dump to the examination machine. If the phone cannot be powered down, *adb* can also be used to transfer the memory dump to the investigator's machine.

# 5.0 LiME Memory Range Header Version 1 Specification

```
typedef struct {
        unsigned int magic;             // Always 0x4C694D45 (LiME)
        unsigned int version;           // Header version number
        unsigned long long s_addr;      // Starting address of physical RAM range
        unsigned long long e_addr;      // Ending address of physical RAM range
        unsigned char reserved[8];      // Currently all zeros
} __attribute__ ((__packed__)) lime_mem_range_header;
```