

# LiME – Linux Memory Extractor

---

## Contents

---

- [Compiling](#)
  - [Linux](#)
  - [External](#)
  - [Debug](#)
  - [Symbols](#)
  - [Android](#)
- [Usage](#)
  - [Parameters](#)
  - [Acquisition of Memory over TCP](#)
  - [Acquisition of Memory to Disk \(SD-Card\)](#)
- [LiME Memory Range Header Version 1 Specification](#)

## Compiling LiME

---

### 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

<https://www.kernel.org/doc/Documentation/kbuild/modules.txt>.

### External

LiME can be compiled externally from the target in order to provide a more forensically sound and secure method. Follow this [guide](#) to learn how.

### Debug

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.

### Symbols

When compiling LiME with the default Makefile, using the command “make symbols” will compile a LiME module without stripping symbols. This is useful for tools such as Volatility where one can create a profile without loading second module.

## 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 at <http://source.android.com/source/initializing.html>
- Download and un(zip|tar) the android NDK found at <http://developer.android.com/tools/sdk/ndk/index.html>.
- Download and un(zip|tar) the android SDK found at <http://developer.android.com/sdk/index.html>.
- 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 ENVIRONMENT

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/  
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-
default:
$(MAKE) ARCH=arm CROSS_COMPILE=$(CCPATH)/arm-eabi- -C $(KDIR) M=$(PWD) modules
```

## COMPILING THE MODULE

```
cd $LIME_SRC
make
```

## Usage

---

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.

## 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.

```
path      Either a filename to write on the local system (SD Card) or tcp:<port
format    padded: Pads all non-System RAM ranges with 0s, starting from physical
          lime: Each range is prepended with a fixed-sized header which contains
          raw: Simply concatenates all System RAM ranges. Most memory analysis
dio       Optional. 1 to enable Direct IO attempt, 0 to disable (default)
localhostonly Optional. 1 restricts the tcp to only listen on localhost, 0 binds on
timeout   Optional. If it takes longer than the specified timeout (in milliseconds)
          of memory then the range is assumed to be bad and is skipped. To disable
          The default setting is 1000 (1 second).
```

## 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
```

## 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.

## 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;
```