



Backing up System...

Backing Up System...



PASEANDO POR LA MEMORIA DE UN ANDROID

Buenaventura Salcedo

Copied recovery log to /sdcard
Fixing permissions...
Loading packages...
Copied recovery log to /sdcard
Fixing /system/app permissions...
Loading packages...
Fixing /system/app permissions...
Fixing /data/app permissions...
Fixing /data/data/ permissions
Done fixing permissions.
Done: [REMOVED] partition details...
* Total number of partitions to back up: 5
* Total size of all data: 1056MB
* Available space: 7559MB
Updating partition details.
* Total size of all data: 1056MB
* Available space: 7559MB
[BACKUP STARTED]
* Backup Folder: /sdcard/TWRP/BACKUPS/509F2
Backing up System...
Backing up System...

- Graduado en Ingeniería Informática (2019)
- CEO Servicio Técnico de Telefonía Movil e Informática (2004)
- Desarrollador de herramientas forenses para smartphones



nomed1



AGRADECIMIENTOS



- ✓ A la Organización de BITUP
- ✓ Universidad de Alicante
- ✓ Colaboradores de BITUP
- ✓ Antonio Sanz y Roberto Hernández



- ✓ Proceso largo: Compilar el kernel, LiME y el profile
- ✓ Deficiencias en el prototipo de ExereWare
- ✓ Estudio y análisis de memoria en Android
- ✓ Desarrollo de procesos automáticos para el análisis de memoria

ANÁLISIS DE RIESGOS



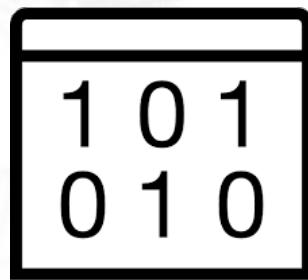
- 1) Preparar el dispositivo (virtual para el POC)**
- 2) Preparar el entorno de compilación**
- 3) Compilar el kernel**
- 4) Compilar LiME**
- 5) Crear profile para Volatility**
- 6) Obtener el dump de memoria**
- 7) Analizar con Volatility**



PROCESO SIMPLIFICADO



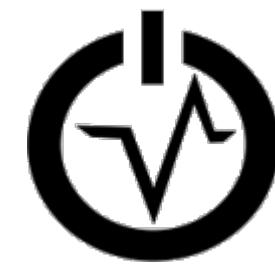
SOURCE
CODE



KERNEL



LIME



PROFILE



ELEMENTOS CLAVE DE ENTRADA



- JDK: Versión de java correcta para la compilación del kernel
- SDK: En el POC, nos dará soporte para que corra la MV
- Código fuente: De la versión del kernel
- NDK (toolchain): Cadena de herramientas para traducir y compilar
- Variables de entorno: ARCH – SUBARCH – CROSS_COMPILE
- .config: El fichero de configuración para generar el kernel



ELEMENTOS CLAVE DE SALIDA



- system.map: Tabla de símbolos(variables o funciones) con sus direcciones.
- zImage (o bzImage): Kernel con LKM que usaremos en el dispositivo.
- lime.ko: El módulo LiME (el LKM) que colgaremos en el kernel
- dump: Fichero que generaremos al volcar la memoria
- profile: El perfil estrictamente necesario para el análisis de memoria

Volatility 3 ???

LKM = Loadable Kernel Module → Necesitará privilegios → Autorizado por la **Magic String**



ENTORNO PARA POC



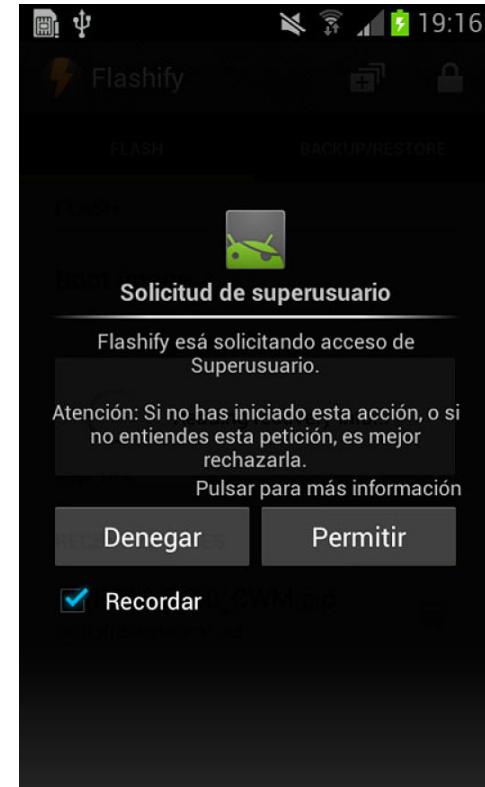
- ✓ **Ubuntu 14.04 (recomendado por Android Developers)**
- ✓ **Emulator de SDK de Android**
- ✓ **Android 4.2.2 arm**
- ✓ **Lime 1.4 (no funciona 1.8, digest parcheado K.O.)**
- ✓ **Volatility 2.3.1**



PREPARAR EL DISPOSITIVO REAL



- ✓ ROOT
- ✓ Después necesitaríamos cargar el kernel nuevo en el boot, cada dispositivo tiene su procedimiento (flasher)



DISPOSITIVO VIRTUAL (POC)

- ✓ Configurar y construir el Android Build Environment

```
$ sudo apt-get install git-core gnupg flex  
bison gperf build-essential zip curl zlib1g-  
dev gcc-multilib g++-multilib libc6-dev-i386  
lib32ncurses5-dev x11proto-core-dev libx11-  
dev lib32z-dev libgl1-mesa-dev libxml2-utils  
xsltproc unzip mesa-utils lib64stdc++6:i386
```



PREPARAR EL DISPOSITIVO VIRTUAL (ii)



DISPOSITIVO VIRTUAL (POC)

- ✓ Instalar JDK 6 (versión no importa para AVD)
- ✓ Instalar SDK 24.0.2 (vieja para que el proceso sea complicado)
- ✓ \$ ~/android-sdk/tools/android (sdk manager)

Platform-tools

Build-tools

System-Image (API)

Android 4.2.2 (API 17)				
<input checked="" type="checkbox"/>	SDK Platform	17	3	Installed
<input type="checkbox"/>	Samples for SDK	17	1	Not installed
<input checked="" type="checkbox"/>	ARM EABI v7a System Image	17	6	Installed
<input checked="" type="checkbox"/>	Intel x86 Atom System Image	17	4	Installed
<input type="checkbox"/>	MIPS System Image	17	1	Not installed
<input type="checkbox"/>	Google APIs ARM EABI v7a System Image	17	6	Not installed
<input type="checkbox"/>	Google APIs Intel x86 Atom System Image	17	6	Not installed
<input type="checkbox"/>	Google APIs	17	4	Not installed
<input type="checkbox"/>	Sources for Android SDK	17	1	Not installed

PREPARAR EL DISPOSITIVO VIRTUAL (iii)



DISPOSITIVO VIRTUAL (POC)

- ✓ Agregar al PATH el SDK (mejor en el .bashrc)
 - tools
 - build-tools/XX.X.X
 - platform-tools

```
nomed@ub1406: ~
nomed@ub1406:~$ echo "export PATH=$PATH:~/android-sdk/tools:~/android-sdk/tools/
bin:~/android-sdk/build-tools/23.0.1:~/android-sdk/platform-tools" >> ~/.bashrc
nomed@ub1406:~$
```



PREPARAR EL DISPOSITIVO VIRTUAL (iv)



DISPOSITIVO VIRTUAL (POC)

✓ Crear los dispositivos virtuales

```
$ android avd
```

- Nexus one (3.7")
- Android 4.2.2
- arm o x86 (es más rápido)
- RAM 512mb
- Internal 512mb
- SD Card 800mb



Si al crear sdcard da error debemos dar permisos a mksdcard de la carpeta tools



PREPARAR EL DISPOSITIVO VIRTUAL (v)



DISPOSITIVO VIRTUAL (POC)

- ✓ Probar el dispositivo

```
$ emulator @dispositivo
```

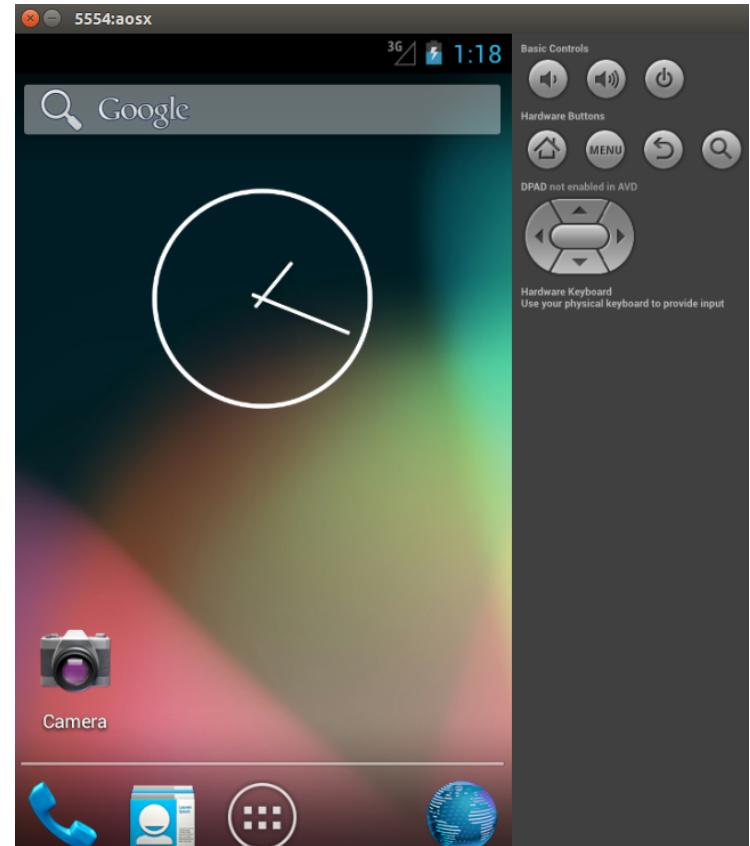
- ✓ Probar y remontar con adb

```
$ adb devices
```

```
$ adb root
```

```
$ adb connect 127.0.0.1:5554
```

```
$ adb remount
```

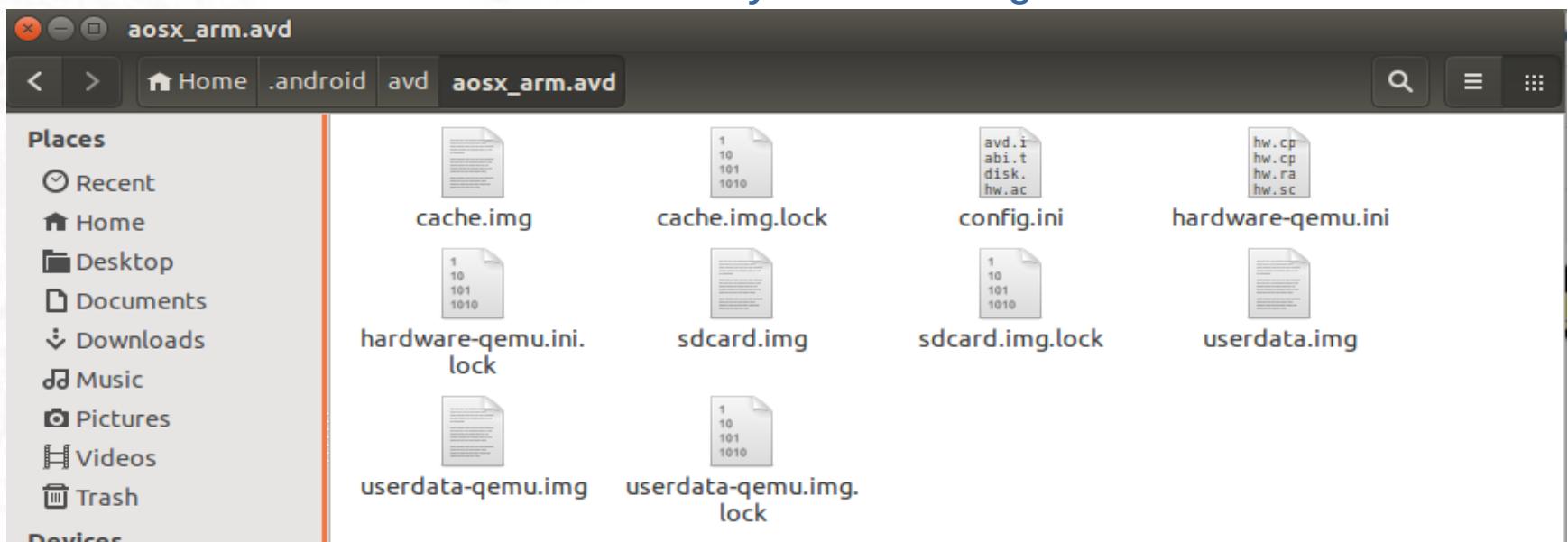


PREPARAR EL DISPOSITIVO VIRTUAL (vi)



DISPOSITIVO VIRTUAL (POC)

- ✓ El dispositivo se creará en ~/.android
- ✓ Los ficheros base estarán en
~/android-sdk/system-images



INFORMACIÓN DEL KERNEL



```
$ adb shell cat /proc/version  
$ adb shell getprop | grep ro.build
```

```
nomed@ub1406: ~/toolchains  
nomed@ub1406:~/toolchains$ adb shell cat /proc/version  
Linux version 3.4.67-01422-gd3ffcc7-dirty (digit@tyrion.par.corp.google.com) (gc  
c version 4.8 (GCC) ) #1 PREEMPT Tue Sep 16 19:34:06 CEST 2014  
nomed@ub1406:~/toolchains$ adb shell getprop | grep ro.build  
[ro.build.characteristics]: [default]  
[ro.build.date.utc]: [1530634644]  
[ro.build.date]: [Tue Jul 3 16:17:24 UTC 2018]  
[ro.build.description]: [sdk-eng 4.2.2 JB_MR1.1 4875371 test-keys]  
[ro.build.display.id]: [sdk-eng 4.2.2 JB_MR1.1 4875371 test-keys]  
[ro.build.fingerprint]: [generic/sdk/generic:4.2.2/JB_MR1.1/4875371:eng/test-key  
s]  
[ro.build.host]: [wped3.hot.corp.google.com]  
[ro.build.id]: [JB_MR1.1]  
[ro.build.product]: [generic]  
[ro.build.tags]: [test-keys]  
[ro.build.type]: [eng]  
[ro.build.user]: [android-build]  
[ro.build.version.codename]: [REL]  
[ro.build.version.incremental]: [4875371]  
[ro.build.version.release]: [4.2.2]  
[ro.build.version.sdk]: [17]  
nomed@ub1406:~/toolchains$
```

Android version
4.2.2

Baseband version
Unknown

Kernel version
3.4.67-01422-gd3ffcc7-dirty
digit@tyrion.par.corp.google.com #1
Tue Sep 16 19:34:06 CEST 2014

Build number
sdk-eng 4.2.2 JB_MR1.1 4875371 test-
keys



CÓDIGO FUENTE (i)



EMULATOR

<https://android.googlesource.com/kernel/goldfish/>

QUALCOMM

<https://android.googlesource.com/kernel/msm/>

SAMSUNG

[http://opensource.samsung.com/reception/receptionSub.d
o?method=sub&sub=T&menu_item=mobile](http://opensource.samsung.com/reception/receptionSub.do?method=sub&sub=T&menu_item=mobile)

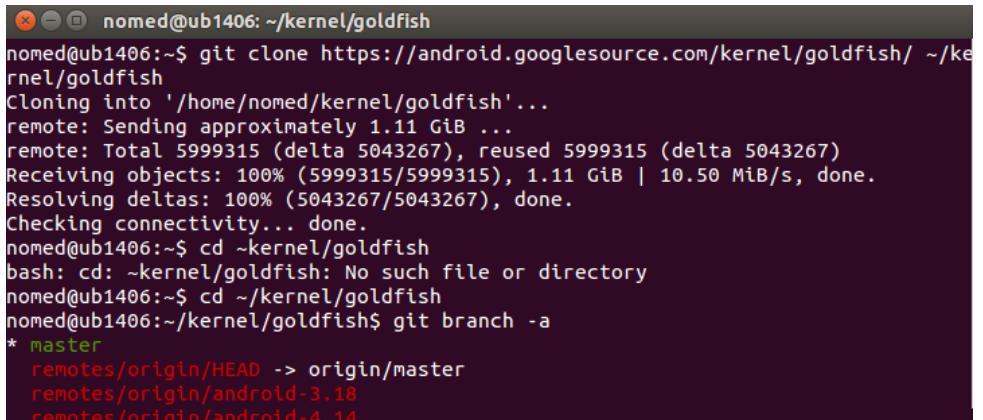
git – repo - zip



CÓDIGO FUENTE (ii)



```
$ git clone  
https://android.googlesource.com/kernel/goldfish/ ~/kernels/goldfish  
y 6 millones de deltas después...  
$ cd ~/kernel/goldfish  
$ git branch -a  
$ git checkout -t remotes/origin/android-goldfish-3.4 -b goldfish34
```



A screenshot of a terminal window titled "nomed@ub1406: ~/kernel/goldfish". The window shows the output of a "git clone" command and a "git branch -a" command. The "git clone" command cloned the "goldfish" kernel from Google's Git repository. The "git branch -a" command shows the local branches "master" and "goldfish34", along with the remote branches "remotes/origin/HEAD", "remotes/origin/master", "remotes/origin/android-3.18", and "remotes/origin/android-4.14".

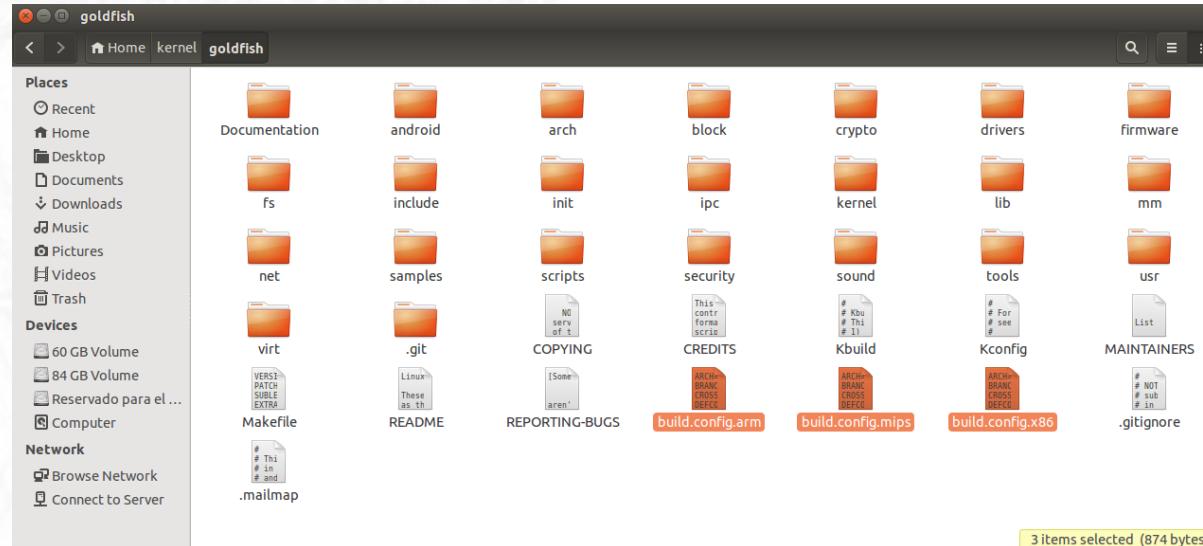
```
nomed@ub1406:~/kernel/goldfish  
nomed@ub1406:~$ git clone https://android.googlesource.com/kernel/goldfish/ ~/kernel/goldfish  
Cloning into '/home/nomed/kernel/goldfish'...  
remote: Sending approximately 1.11 GiB ...  
remote: Total 5999315 (delta 5043267), reused 5999315 (delta 5043267)  
Receiving objects: 100% (5999315/5999315), 1.11 GiB | 10.50 MiB/s, done.  
Resolving deltas: 100% (5043267/5043267), done.  
Checking connectivity... done.  
nomed@ub1406:~$ cd ~kernel/goldfish  
bash: cd: ~kernel/goldfish: No such file or directory  
nomed@ub1406:~$ cd ~/kernel/goldfish  
nomed@ub1406:~/kernel/goldfish$ git branch -a  
* master  
  remotes/origin/HEAD -> origin/master  
  remotes/origin/android-3.18  
  remotes/origin/android-4.14
```



CÓDIGO FUENTE (iii)



Hay que mirar bien los readme o ficheros de ayuda, en busca del **toolchain** que debemos utilizar, en vez de suponer que va a usar cualquiera.



el toolchain debe indicarse en CROSS_COMPILE con la ruta acabada con un guión



CADERAS DE HERRAMIENTAS

Son unas librerías y módulos ejecutables que van a ir realizando la traducción y compilación de los drivers que incluyamos en el kernel que vamos a construir y que nos dejará todo preparado para la arquitectura en la que lo vamos a usar.

- <http://www.codesourcery.com/>
- <https://dn.odroid.com/toolchains/>
- <https://bitbucket.org/matthewdalex/>

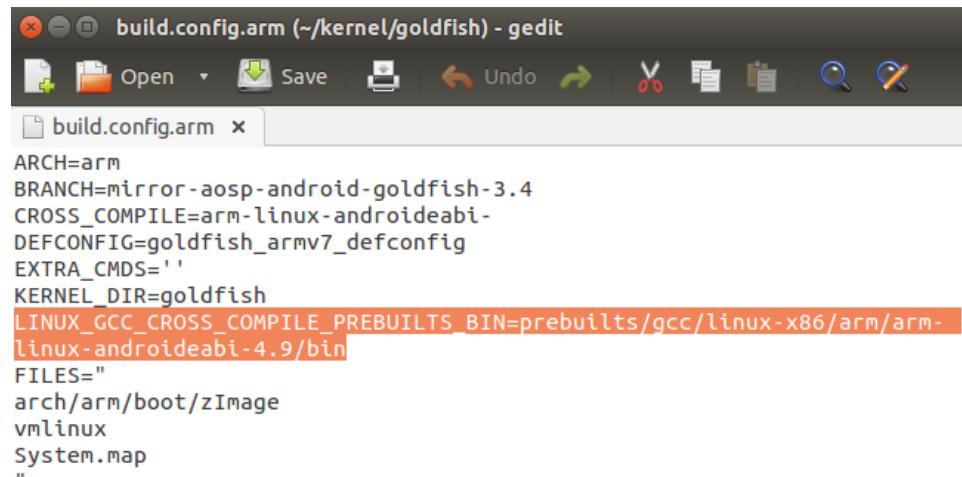
```
$ git clone  
https://android.googlesource.com/platform  
/prebuilts/<ruta_toolchain>
```



CADENAS DE HERRAMIENTAS

```
$ git clone  
https://android.googlesource.com/platform/prebuilts/gcc/linux-  
x86/arm/arm-linux-androideabi-4.9
```

**prebuilts = 3.3 Gb
más deltas??? NO, por favor!!!**



```
build.config.arm (~/kernel/goldfish) - gedit  
build.config.arm x  
ARCH=arm  
BRANCH=mirror-aosp-android-goldfish-3.4  
CROSS_COMPILE=arm-linux-androideabi-  
DEFCONFIG=goldfish_armv7_defconfig  
EXTRA_CMDS=''  
KERNEL_DIR=goldfish  
LINUX_GCC_CROSS_COMPILE_PREBUILTS_BIN=prebuilts/gcc/linux-x86/arm/arm-  
linux-androideabi-4.9/bin  
FILES=""  
arch/arm/boot/zImage  
vmlinuz  
System.map  
"
```

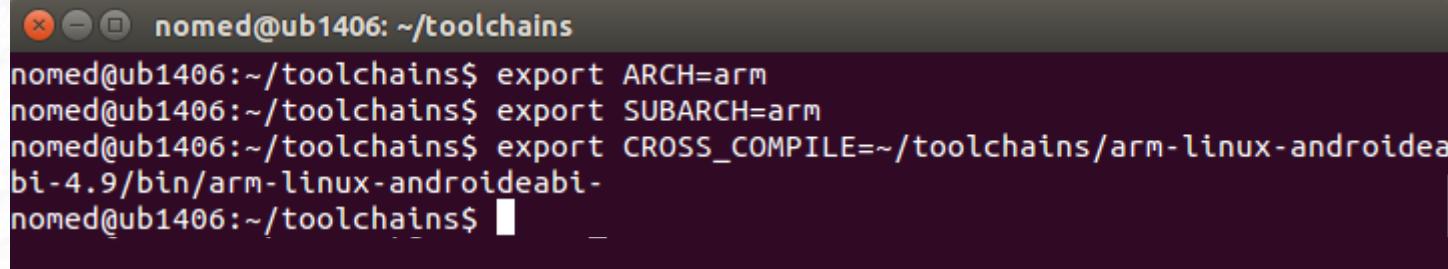


COMPILACIÓN DEL KERNEL (i)



LAS VARIABLES

```
$ export ARCH=arquitectura  
$ export SUBARCH=arquitectura  
$ export CROSS_COMPILE=~/ruta/al/toolchain-
```



A screenshot of a terminal window titled "nomed@ub1406: ~/toolchains". The window shows the user executing three commands to set environment variables: "export ARCH=arm", "export SUBARCH=arm", and "export CROSS_COMPILE=~/toolchains/arm-linux-androideabi-4.9/bin/arm-linux-androideabi-". The terminal has a dark background with light-colored text and standard Linux-style window controls at the top.

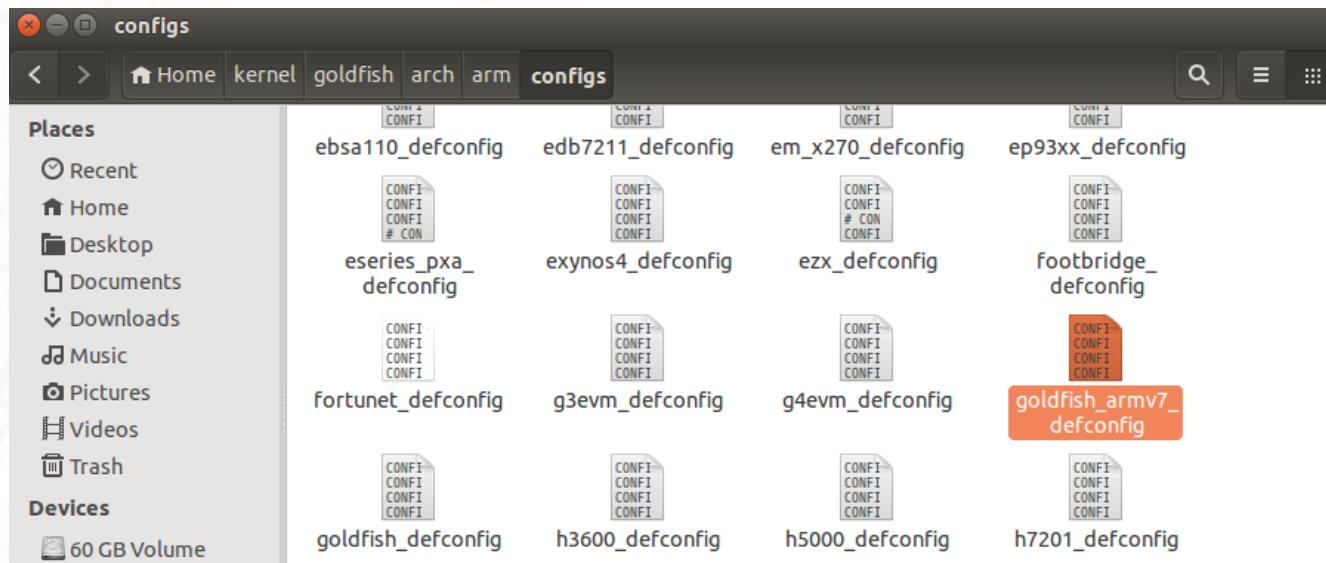
```
nomed@ub1406:~/toolchains$ export ARCH=arm  
nomed@ub1406:~/toolchains$ export SUBARCH=arm  
nomed@ub1406:~/toolchains$ export CROSS_COMPILE=~/toolchains/arm-linux-androideabi-4.9/bin/arm-linux-androideabi-  
nomed@ub1406:~/toolchains$
```



EL PERFIL DE KERNEL

Podemos usar:

- `~/kernel/goldfish/arch/x86/configs`
- podemos usar el `config.gz` del propio dispositivo (renombrarlo)



GENERAR EL .CONFIG

- Seleccionamos nuestro perfil (o los asistentes como menuconfig):

```
$ make goldfish_armv7_defconfig
```

- Agregamos los atributos para poder colgar módulos al kernel en el fichero .config

CONFIG_MODULES=y

CONFIG_MODULES_UNLOAD=y

CONFIG_MODULES_FORCE_UNLOAD=y

```
nomed@ub1406: ~/kernel/goldfish
nomed@ub1406:~/kernel/goldfish$ make goldfish_armv7_defconfig
#
# configuration written to .config
#
nomed@ub1406:~/kernel/goldfish$ █
```



COMPILAR EL KERNEL

\$ make -jX



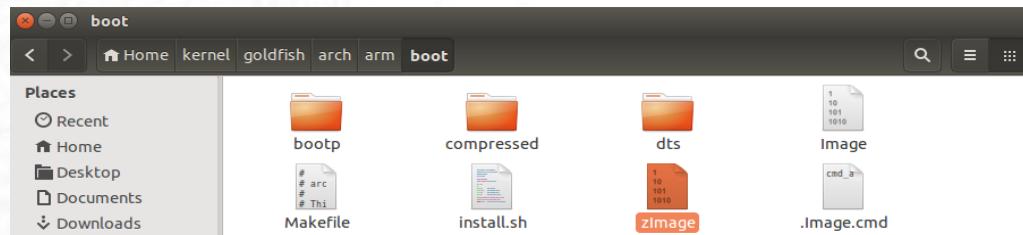
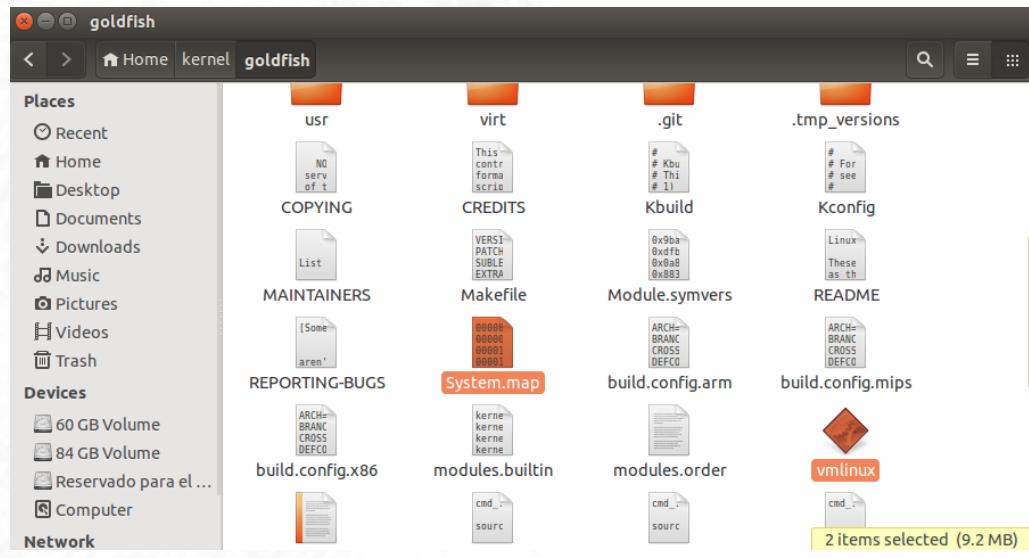
```
./config (~/kernel/goldfish) - gedit
File Open Save Undo .config
build.config.arm *indice.txt .config
CONFIG_ARCH_MMAPP_RND_BITS_MAX=10
CONFIG_ARCH_MMAPP_RND_BITS=16

#
# GCOV-based kernel profiling
#
# CONFIG_GCOV_KERNEL is not set
CONFIG_HAVE_GENERIC_DMA_COHERENT=y
CONFIG_SLABINFO=y
CONFIG_RT_MUTEXES=y
CONFIG_BASE_SMALL=0
# CONFIG_MODULES is not set
CONFIG_MODULES=y
CONFIG_MODULES_UNLOAD=y
CONFIG_MODULES_FORCE_UNLOAD=y
CONFIG_BLOCK=y
CONFIG_LBDAF=y
# CONFIG_BLK_DEV_BSG is not set
# CONFIG_BLK_DEV_BSGLIB is not set
# CONFIG_BLK_DEV_INTEGRITY is not set
```

- KPROBES
- MODULE_FORCE_LOAD
- MODULE_UNLOAD
- MODULE_FORCE_UNLOAD
- MODVERSIONS

```
Enable full-sized data structures for core (BASE_FULL) [Y/n/?] y
Enable futex support (FUTEX) [Y/n/?] y
Enable eventpoll support (EPOLL) [Y/n/?] y
Enable signalfd() system call (SIGNAFLD) [Y/n/?] y
Enable timerfd() system call (TIMERFD) [Y/n/?] y
Enable eventfd() system call (EVENTFD) [Y/n/?] y
Use full shmem filesystem (SHMEM) [Y/n/?] y
Enable AIO support (AIO) [Y/n/?] y
Embedded system (EMBEDDED) [Y/n/?] y
Enable VM event counters for /proc/vmstat (VM_EVENT_COUNTERS) [Y/n/?] y
Disable heap randomization (COMPAT_BRK) [Y/n/?] y
Choose SLAB allocator
> 1. SLAB (SLAB)
    2. SLUB (Unqueued Allocator) (SLUB)
    3. SLOB (Simple Allocator) (SLOB)
choice[1-3?]: 1
Profiling support (PROFILING) [N/y/?] n
Kprobes (KPROBES) [N/y/?] (NEW) y
Optimize very unlikely/likely branches (JUMP_LABEL) [N/y/?] n
Number of bits to use for ASLR of mmap base address (ARCH_MMAPP_RND_BITS) [16] 16
*
* Enable loadable module support
*
Enable loadable module support (MODULES) [Y/n/?] y
Forced module loading (MODULE_FORCE_LOAD) [N/y/?] (NEW) y
Module unloading (MODULE_UNLOAD) [N/y/?] (NEW) y
    Forced module unloading (MODULE_FORCE_UNLOAD) [N/y/?] (NEW) y
) y
Module versioning support (MODVERSIONS) [N/y/?] (NEW) y
```

COMPILACIÓN DEL KERNEL (v)



```
nomed@ub1406: ~/kernel/goldfish$ AS arch/arm/boot/compressed/piggy.gzip.o
Android GCC has been deprecated in favor of Clang, and will be removed from
Android in 2020-01 as per the deprecation plan in:
https://android.googlesource.com/platform/prebuilts/clang/host/linux-x86/+/maste
r/GCC_4_9_DEPRECATION.md

AS arch/arm/boot/compressed/lib1funcs.o
Android GCC has been deprecated in favor of Clang, and will be removed from
Android in 2020-01 as per the deprecation plan in:
https://android.googlesource.com/platform/prebuilts/clang/host/linux-x86/+/maste
r/GCC_4_9_DEPRECATION.md

AS arch/arm/boot/compressed/ashldi3.o
Android GCC has been deprecated in favor of Clang, and will be removed from
Android in 2020-01 as per the deprecation plan in:
https://android.googlesource.com/platform/prebuilts/clang/host/linux-x86/+/maste
r/GCC_4_9_DEPRECATION.md

LD arch/arm/boot/compressed/vmlinux
OBJCOPY arch/arm/boot/zImage
Kernel: arch/arm/boot/zImage is ready
nomed@ub1406:~/kernel/goldfish$
```

PROBANDO EL NUEVO KERNEL



```
$ emulator @aosx_arm -kernel  
~/kernel/goldfish/arch/arm/boot/zImage -show-kernel
```

```
nomed@ub1406:~$ adb devices  
List of devices attached  
* daemon not running. starting it now on port 5037 *  
* daemon started successfully *  
emulator-5554 device  
  
nomed@ub1406:~$ adb root  
adb is already running as root  
nomed@ub1406:~$ adb connect 127.0.0.1:5554  
connected to 127.0.0.1:5554  
nomed@ub1406:~$ adb remount  
remount succeeded  
nomed@ub1406:~$ adb shell cat /proc/version  
Linux version 3.4.67-g880d9af (nomed@ub1406) (gcc version 4.9.x 20150123 (prerel  
ease) (GCC) ) #1 PREEMPT Fri Oct 25 15:32:11 CEST 2019  
nomed@ub1406:~$
```

Android version
4.2.2

Baseband version
Unknown

Kernel version
3.4.67-g880d9af
nomed@ub1406 #1
Fri Oct 25 15:32:11 CEST 2019

Build number
sdk-eng 4.2.2 JB_MR1.1 4875371 test-keys

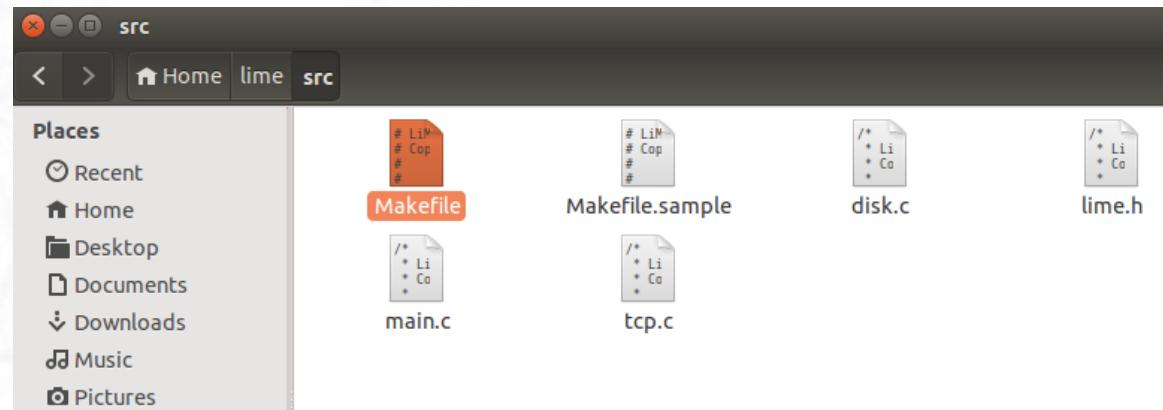


COMPILANDO LiME (i)



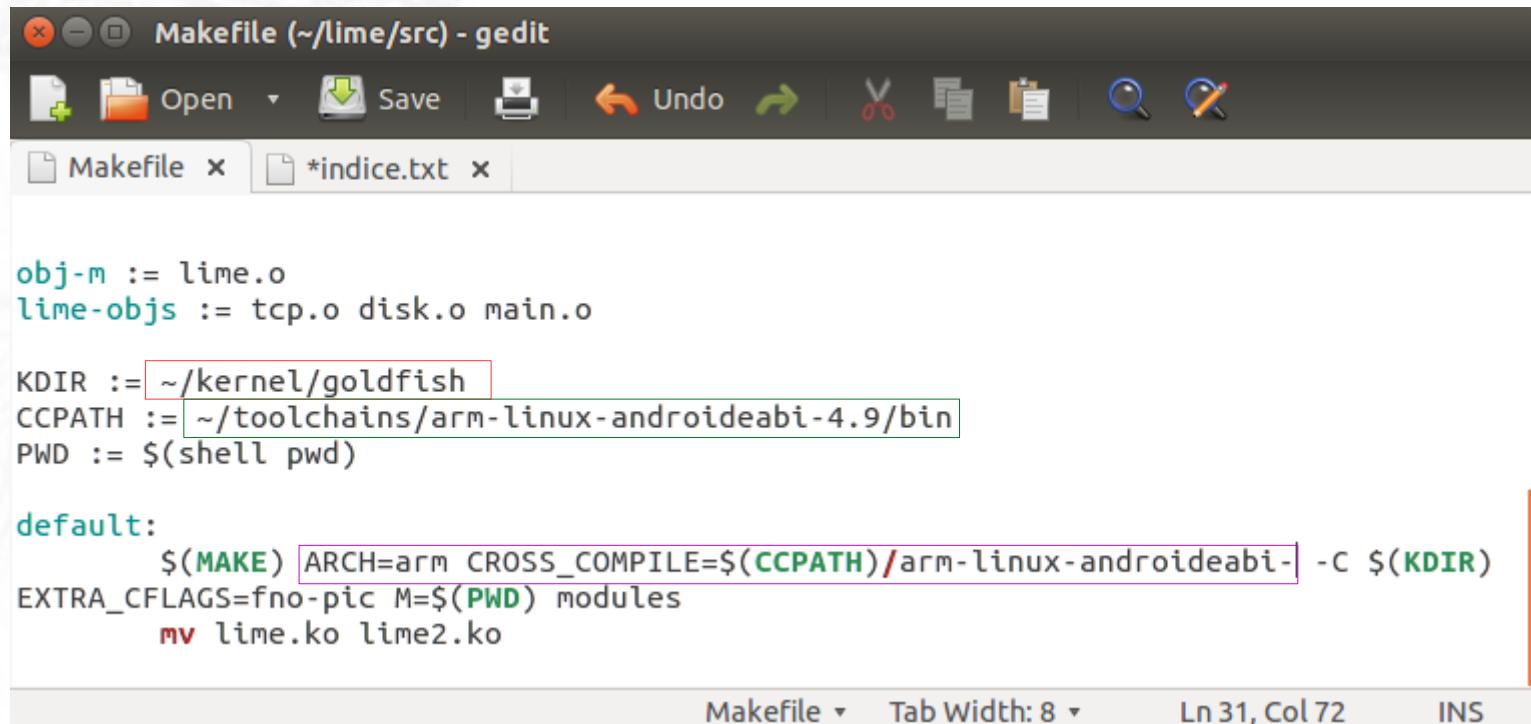
LiME es un extractor de memoria para sistemas basados en Linux. Android es un sistema basado en Linux por tanto nos sirve para realizar adquisiciones de memoria en sistemas vivos.

<https://github.com/504ensicsLabs/LiME>



CONFIGURAR MAKEFILE

- **KDIR**
- **CCPATH**
- **ARCH**
- **CROSS_COMPILE**



```
obj-m := lime.o
lime-objs := tcp.o disk.o main.o

KDIR := ~/kernel/goldfish
CCPATH := ~/toolchains/arm-linux-androideabi-4.9/bin
PWD := $(shell pwd)

default:
    $(MAKE) ARCH=arm CROSS_COMPILE=$(CCPATH)/arm-linux-androideabi-
EXTRA_CFLAGS=fno-pic M=$(PWD) modules
    mv lime.ko lime2.ko

Makefile ▾ Tab Width: 8 ▾ Ln 31, Col 72 ▾ INS
```

errata en la imagen = EXTRA_CFLAGS= -fno-pic

COMPILANDO LiME (iii)



COMPILAR LiME

```
$ cd ~/lime/src
```

```
$ make
```

```
nomed@ub1406:~/lime/src
nomed@ub1406:~$ export ARCH=arm
nomed@ub1406:~$ export SUBARCH=arm
nomed@ub1406:~$ export CROSS_COMPILE=~/toolchains/arm-linux-androideabi-4.9/bin/
arm-linux-androideabi-
nomed@ub1406:~$ cd ~/lime/src
nomed@ub1406:~/lime/src$ make
make ARCH=arm CROSS_COMPILE=~/toolchains/arm-linux-androideabi-4.9/bin/arm-linux-
-androideabi- -C ~/kernel/goldfish EXTRA_CFLAGS=-fno-pic M=/home/nomed/lime/src
modules
```

```
LD [M] /home/nomed/lime/src/lime.ko
make[1]: Leaving directory `/home/nomed/kernel/goldfish'
mv lime.ko lime2.ko
nomed@ub1406:~/lime/src$ ls
Makefile          Module.symvers  lime.h      lime.o      main.o       tcp.o
Makefile.sample   disk.c        lime.mod.c  lime2.ko    modules.order
Makefile~
disk.o          lime.mod.o   main.c     tcp.c
nomed@ub1406:~/lime/src$
```



COLGAR LiME EN EL KERNEL

```
$ adb push ./lime2.ko /sdcard/lime.ko  
$ adb shell  
# insmod /sdcard/lime.ko "path=/sdcard/lime.dump format=lime"
```

```
nomed@ub1406:~/lime/src$ adb push ./lime2.ko /sdcard/lime.ko  
117 KB/s (7600 bytes in 0.062s)  
nomed@ub1406:~/lime/src$ adb shell  
root@android:/ # insmod /sdcard/lime.ko "path=/sdcard/lime.dump format=lime"  
root@android:/ # █
```



formatos: raw - padded - lime

ADQUIRIR EL VOLCADO

- Forma rápida utilizando sleuthkit

```
$ fls -r /ruta/al/fichero/sdcard.img
```

```
$ icat ~/ruta/al/fichero/sdcard.img XX > ~/carpeta_dumps/lime.dmp
```

- Forma lenta, pero válida para cualquier dispositivo

```
$ adb pull /sdcard/lime.dump
```



VOLCADO DE LA MEMORIA (iii)



```
nomed@ub1406:~$ fls -r ~/.android/avd/aosx_arm.avd/sdcard.img
d/d 3: LOST.DIR
d/d 6: .android_secure
d/d 8: Music
d/d 10: Podcasts
d/d 12: Ringtones
d/d 14: Alarms
d/d 16: Notifications
d/d 18: Pictures
d/d 20: Movies
d/d 22: Download
d/d 23: DCIM
r/r 25: lime.ko
r/r 27: lime.dump
v/v 26111683: $MBR
v/v 26111684: $FAT1
v/v 26111685: $FAT2
d/d 26111686: $OrphanFiles
nomed@ub1406:~$ icat ~/.android/avd/aosx_arm.avd/sdcard.img 27 > ~/carpeta_dumps/
/lime.dmp
nomed@ub1406:~$
```



OTRA FORMA DE ADQUIRIR CON LiME

```
$ adb push ./lime2.ko /sdcard/lime.ko  
$ $ adb forward tcp:4444 tcp:4444  
$ adb shell  
# insmod /sdcard/lime.ko "path=tcp:4444 format=lime"
```

desde otra terminal

```
$ nc localhost 4444 > ~/carpeta_dumps/lime.dmp
```



formatos: raw - padded - lime

CREANDO EL PROFILE DE VOLATILITY (i)



Volatility es la herramienta Open Source para analizar la memoria RAM por excelencia. Necesitamos un profile que conozca las direcciones del System.map

profile = fichero zip (modulo dwarf , fichero System.map)

<https://github.com/volatilityfoundation/volatility>



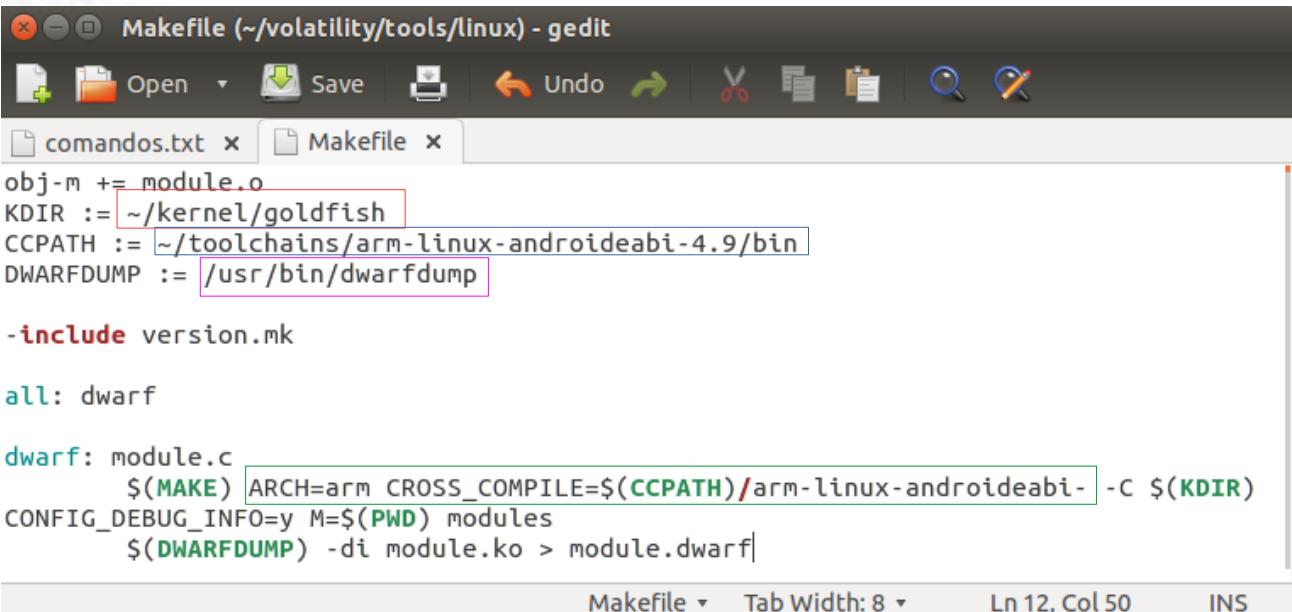
CREANDO EL PROFILE DE VOLATILITY (ii)



CONFIGURAR MAKEFILE

Necesitamos **dwarfdump**, lo que hace es traducir información para que se pueda entender por humanos, es decir, formatea la información de los profiles para presentarla de una forma legible.

- **KDIR**
- **CCPATH**
- **DWARFDUMP**
- **ARCH**
- **CROSS_COMPILE**



A screenshot of the gedit text editor showing a Makefile. The file contains the following configuration:

```
obj-m += module.o
KDIR := ~/kernel/goldfish
CCPATH := ~/toolchains/arm-linux-androideabi-4.9/bin
DWARFDUMP := /usr/bin/dwarfdump

-include version.mk

all: dwarf

dwarf: module.c
    $(MAKE) ARCH=arm CROSS_COMPILE=$(CCPATH)/arm-linux-androideabi- -C $(KDIR)
CONFIG_DEBUG_INFO=y M=$(PWD) modules
    $(DWARFDUMP) -di module.ko > module.dwarf|
```

The variables KDIR, CCPATH, and DWARFDUMP are highlighted with red boxes. The command line at the bottom of the code block is also highlighted with a green box.

CREANDO EL PROFILE DE VOLATILITY (iii)



COMPILAR EL MODULO DWARF

```
$ cd ~/volatility/tools/linux  
$ make
```

```
nomed@ub1406:~/volatility/tools/linux  
nomed@ub1406:~$ export ARCH=arm  
nomed@ub1406:~$ export SUBARCH=arm  
nomed@ub1406:~$ export CROSS_COMPILE=/toolchains/arm-linux-androideabi-4.9/bin/  
arm-linux-androideabi-  
nomed@ub1406:~$ cd ~/volatility/tools/linux  
nomed@ub1406:~/volatility/tools/linux$ make  
make ARCH=arm CROSS_COMPILE=/toolchains/arm-linux-androideabi-4.9/bin/arm-linux  
-androideabi- -C ~/kernel/goldfish CONFIG_DEBUG_INFO=y M=/home/nomed/volatility/  
tools/linux modules
```

```
LD [M] /home/nomed/volatility/tools/linux/module.ko  
make[1]: Leaving directory `/home/nomed/kernel/goldfish'  
/usr/bin/dwarfdump -di module.ko > module.dwarf  
nomed@ub1406:~/volatility/tools/linux$
```

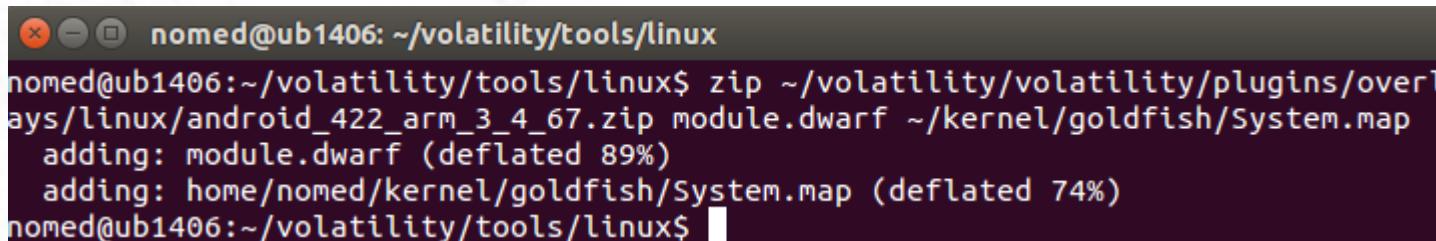


CREANDO EL PROFILE DE VOLATILITY (iv)

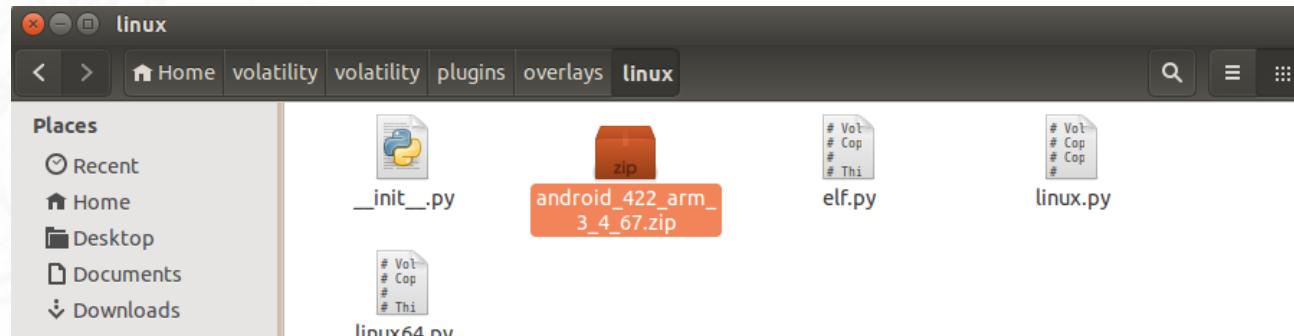


COMPRIMIR EL PROFILE (plugin)

```
$ zip  
~/volatility/volatility/plugins/overlays/linux/android_422_arm_3_4_67.zip  
module.dwarf ~/kernel/goldfish/System.map
```



```
nomed@ub1406:~/volatility/tools/linux$ zip ~/volatility/volatility/plugins/overlays/linux/android_422_arm_3_4_67.zip module.dwarf ~/kernel/goldfish/System.map  
    adding: module.dwarf (deflated 89%)  
    adding: home/nomed/kernel/goldfish/System.map (deflated 74%)  
nomed@ub1406:~/volatility/tools/linux$
```



ANALIZANDO CON VOLATILITY



POC



ANALIZANDO CON VOLATILITY (i)



```
$ python vol.py --info
```

nuestro profile = **Linuxandroid_422_arm_3_4_67ARM**

```
nomed@ub1406: ~/volatility

Profiles
-----
Linuxandroid_422_arm_3_4_67ARM - A Profile for Linux android_422_arm_3_4_67 ARM
VistaSP0x64                      - A Profile for Windows Vista SP0 x64
VistaSP0x86                      - A Profile for Windows Vista SP0 x86
VistaSP1x64                      - A Profile for Windows Vista SP1 x64
VistaSP1x86                      - A Profile for Windows Vista SP1 x86
VistaSP2x64                      - A Profile for Windows Vista SP2 x64
VistaSP2x86                      - A Profile for Windows Vista SP2 x86
```



ANALIZANDO CON VOLATILITY (ii)



linux_arp	- Print the ARP table
linux_banner	- Prints the Linux banner information
linux_bash	- Recover bash history from bash process memory
linux_check_afinfo	- Verifies the operation function pointers of network protocols
linux_check_creds	- Checks if any processes are sharing credential structures
linux_check_evt_arm	- Checks the Exception Vector Table to look for syscall table hooking
linux_check_fop	- Check file operation structures for rootkit modifications
linux_check_idt	- Checks if the IDT has been altered
linux_check_modules	- Compares module list to sysfs info, if available
linux_check_syscall	- Checks if the system call table has been altered
linux_check_syscall_arm	- Checks if the system call table has been altered
linux_check_tty	- Checks tty devices for hooks
linux_cpuid	- Prints info about each active processor
linux_dentry_cache	- Gather files from the dentry cache
linux_dmesg	- Gather dmesg buffer
linux_dump_map	- Writes selected memory mappings to disk
linux_find_file	- Recovers tmpfs filesystems from memory
linux_ifconfig	- Gathers active interfaces
linux_iomem	- Provides output similar to /proc/iomem
linux_keyboard_notifier	- Parses the keyboard notifier call chain



ANALIZANDO CON VOLATILITY (iii)



linux_lsmod	- Gather loaded kernel modules
linux_lsof	- Lists open files
linux_memmap	- Dumps the memory map for linux tasks
linux_moddump	- Extract loaded kernel modules
linux_mount	- Gather mounted fs/devices
linux_mount_cache	- Gather mounted fs/devices from kmem_cache
linux_netstat	- Lists open sockets
linux_pidhashtable	- Enumerates processes through the PID hash table
linux_pkt_queues	- Writes per-process packet queues out to disk
linux_proc_maps	- Gathers process maps for linux
linux_psaux	- Gathers processes along with full command line and start time
linux_pslist	- Gather active tasks by walking the task_struct->task list
linux_pslist_cache	- Gather tasks from the kmem_cache
linux_pstree	- Shows the parent/child relationship between processes
linux_psxview	- Find hidden processes with various process listings
linux_route_cache	- Recovers the routing cache from memory
linux_sk_buff_cache	- Recovers packets from the sk_buff kmem_cache
linux_slabinfo	- Mimics /proc/slabinfo on a running machine
linux_tmppfs	- Recovers tmppfs filesystems from memory
linux_vma_cache	- Gather VMAs from the vm_area_struct cache
linux_volshell	- Shell in the memory image
linux_yarascan	- A shell in the Linux memory image



ANALIZANDO CON VOLATILITY (iv)



```
$ python vol.py --profile=Linuxandroid_422_arm_3_4_67ARM -f ~/carpeta_dumps/lime.dmp linux_pslist
```

Offset	Name	Pid	Uid	Gid	DTB
Start	Time				
-	-	-	-	-	-
0xde81cc00	init	1	0	0	0x1eb3800
0 2019-10-25 15:45:31 UTC+0000					
0xde81c800	kthreadd	2	0	0	-
- 2019-10-25 15:45:31 UTC+0000					
0xde81c400	ksoftirqd/0	3	0	0	-
- 2019-10-25 15:45:31 UTC+0000					
0xde824800	khelper	6	0	0	-
- 2019-10-25 15:45:31 UTC+0000					
0xde824400	sync_supers	7	0	0	-
- 2019-10-25 15:45:31 UTC+0000					
0xde824000	bdi-default	8	0	0	-
- 2019-10-25 15:45:31 UTC+0000					
0xde867c00	kbroadcastd	9	0	0	-
- 2019-10-25 15:45:31 UTC+0000					
0xde867800	rpciod	10	0	0	-
- 2019-10-25 15:45:31 UTC+0000					
0xde867400	kworker/0:1	11	0	0	-



ANALIZANDO CON VOLATILITY (v)



```
$ python vol.py --profile=Linuxandroid_422_arm_3_4_67ARM -f ~/carpeta_dumps/lime.dmp linux_memmap
```

```
nomed@ub1406:~/volatility
nomed@ub1406:~/volatility$ python vol.py --profile=Linuxandroid_422_arm_3_4_67ARM -f ~/carpeta_dumps/lime.dmp linux_memmap
Volatility Foundation Volatility Framework 2.3.1
Task          Pid      Virtual      Physical      Size
-----
init           1 0x00008000 0x1e8ee000      0x1000
init           1 0x00009000 0x1e8ef000      0x1000
init           1 0x0000a000 0x1e920000      0x1000
init           1 0x0000b000 0x1e921000      0x1000
init           1 0x0000c000 0x1e922000      0x1000
init           1 0x0000d000 0x1e923000      0x1000
init           1 0x0000e000 0x1e924000      0x1000
init           1 0x0000f000 0x1e925000      0x1000
init           1 0x00010000 0x1e926000      0x1000
init           1 0x00011000 0x1e927000      0x1000
init           1 0x00012000 0x1e928000      0x1000
init           1 0x00014000 0x1e92a000      0x1000
init           1 0x00015000 0x1e92b000      0x1000
```



ANALIZANDO CON VOLATILITY (vi)



```
$ python vol.py --profile=Linuxandroid_422_arm_3_4_67ARM -f ~/carpeta_dumps/lime.dmp linux_arp
```

```
nomed@ub1406:~/volatility
nomed@ub1406:~/volatility$ python vol.py --profile=Linuxandroid_422_arm_3_4_67AR
M -f ~/carpeta_dumps/lime.dmp linux_arp
Volatility Foundation Volatility Framework 2.3.1
[::] at 00:00:00:00:00:00 on lo
[10.0.2.2] at 52:54:00:12:35:02 on eth0
nomed@ub1406:~/volatility$
```



ANALIZANDO CON VOLATILITY (vii)



```
$ python vol.py --profile=Linuxandroid_422_arm_3_4_67ARM  
-f ~/carpeta_dumps/lime.dmp linux_yarascan -Y "https:"
```

```
Volatility Foundation Volatility Framework 2.3.1  
Task: zygote pid 1241 rule r1 addr 0x977d6e06  
0x977d6e06 68 74 74 70 73 3a 00 66 65 65 64 73 3a 68 74 74 https:.feeds:htt  
0x977d6e16 70 3a 00 66 65 65 64 73 3a 68 74 74 70 73 3a 00 p:.feeds:https:.  
0x977d6e26 66 65 65 64 73 65 61 72 63 68 3a 68 74 74 70 3a feedsearch:http:  
0x977d6e36 00 66 65 65 64 73 65 61 72 63 68 3a 68 74 74 70 .feedsearch:http  
Task: zygote pid 1241 rule r1 addr 0x977d6e1f  
0x977d6e1f 68 74 74 70 73 3a 00 66 65 65 64 73 65 61 72 63 https:.feedsearc  
0x977d6e2f 68 3a 68 74 74 70 3a 00 66 65 65 64 73 65 61 72 h:http:.feedsear  
0x977d6e3f 63 68 3a 68 74 74 70 73 3a 00 52 65 66 72 65 73 ch:https:.Refres  
0x977d6e4f 68 00 5f 73 65 6c 66 00 00 4e 6f 74 20 61 6c 6c h._self..Not.all  
Task: zygote pid 1241 rule r1 addr 0x977d6e42  
0x977d6e42 68 74 74 70 73 3a 00 52 65 66 72 65 73 68 00 5f https:.Refresh._  
0x977d6e52 73 65 6c 66 00 00 4e 6f 74 20 61 6c 6c 6f 77 65 self..Not.allowe  
0x977d6e62 64 20 74 6f 20 6c 6f 61 64 20 6c 6f 63 61 6c 20 d.to.load.local.  
0x977d6e72 72 65 73 6f 75 72 63 65 3a 20 00 00 00 00 20 64 resource:.....d  
Task: zygote pid 1241 rule r1 addr 0x98a545b5  
0x98a545b5 68 74 74 70 73 3a 2f 2f 00 68 74 74 70 3a 2f 2f https://.http://  
0x98a545c5 00 20 64 75 72 69 6e 67 20 53 53 4c 20 72 65 6e ..during.SSL.ren  
0x98a545d5 65 67 6f 74 69 61 74 69 6f 6e 00 30 20 3c 20 62 egotiation.0.<.b  
0x98a545e5 75 66 5f 6c 65 6e 00 20 74 6f 20 43 4f 4e 4e 45 uf_len..to.CONNE  
Task: zygote pid 1241 rule r1 addr 0x9a9fcbe8  
0x9a9fcbe8 68 74 74 70 73 3a 00 08 68 74 74 70 73 3a 2f 2f https:..https://  
0x9a9fcbf8 00 0c 68 74 74 70 73 3a 2f 2f 77 77 77 2e 00 02 ..https://www...  
0x9a9fcc08 68 75 00 03 68 75 65 00 08 68 75 6e 64 72 65 64 hu..hue..hundred  
0x9a9fcc18 73 00 0d 68 77 41 63 63 65 6c 65 72 61 74 65 64 s..hwAccelerated  
Task: zygote pid 1241 rule r1 addr 0x9a9fcbf0
```



ANALIZANDO CON VOLATILITY (viii)



```
$ python vol.py --profile=Linuxandroid_422_arm_3_4_67ARM  
-f ~/carpeta_dumps/lime.dmp linux_pstree
```

```
nomed@ub1406:~/volatility$ volatility --profile=Linuxandroid_422_arm_3_4_67ARM -  
f ~/carpeta_dumps/lime.dmp linux_pstree  
Volatility Foundation Volatility Framework 2.3.1  
Name           Pid     Uid  
init           1        0  
.ueventd       45       0  
.servicemanager 62      1000  
.vold          63       0  
.netd          65       0  
.debuggerd     66       0  
.rild          67      1001  
.surfaceflinger 68      1000  
.zygote         69       0  
.system_server 325     1000  
.ndroid.systemui 388    10038  
.putmethod.latin 450    10021  
.m.android.phone 468    1001  
.ndroid.launcher 480    10022  
.d.process.acore 516    10000  
.m.android.music 536    10026  
.d.process.media 560    10014  
.ndroid.contacts 594    10000  
.viders.calendar 629    10006  
.location.fused 652    10018  
.droid.deskclock 670    10011  
.m.android.email 699    10015  
.ndroid.exchange 718    10016  
.com.android.mms 738    10025  
.ndroid.settings 762    1000  
.ndroid.calendar 770    10005
```



ANALIZANDO CON VOLATILITY (ix)



```
$ python vol.py --profile=Linuxandroid_422_arm_3_4_67ARM  
-f ~/carpeta_dumps/lime.dmp linux_psaux
```

```
nomed@ub1406: ~/volatility  
388 10038 10038 com.android.systemui  
450 10021 10021 com.android.inputmethod.latin  
468 1001 1001 com.android.phone  
480 10022 10022 com.android.launcher  
516 10000 10000 android.process.acore  
536 10026 10026 com.android.music  
560 10014 10014 android.process.media  
594 10000 10000 com.android.contacts  
629 10006 10006 com.android.providers.calendar  
652 10018 10018 com.android.location.fused  
670 10011 10011 com.android.deskclock  
699 10015 10015 com.android.email  
718 10016 10016 com.android.exchange  
738 10025 10025 com.android.mms  
762 1000 1000 com.android.settings  
778 10005 10005 com.android.calendar  
1709 0 0 [flush-179:0]  
1818 0 0 /system/bin/sh -  
1826 0 0 insmod /sdcard/lime.ko path=/sdcard/lime.dump format=lime  
nomed@ub1406:~/volatility$
```



SIEMPRE NOS QUEDARÁ STRINGS



```
$ strings -n 20 -d ~/carpeta_dumps/lime.dmp -o |grep https
```

```
nomed@ub1406:~/volatility
nomed@ub1406:~/volatility$ strings -n 20 -d ~/carpeta_dumps/lime.dmp -o |grep https
2201344425 https://autodiscover.
2335004176 ==https://www.google.com/webhp?client={CID}&source=android-home
2335016226 ))Can only download "http" or "https" URLs.
2335144007 s pot baixar URL "http" o "https".
2335170275 jako http nebo https.
2335212371 CCKan kun downloade webadresser, der starter med "http" eller https..
2335235452 56Nur Download von URLs mit "http" oder "https" m
2335327772 77Solo se pueden descargar URL del tipo "http" o "https".
2335354051 44Solo se pueden descargar URL de "http" o de "https".
2335376023 //Vain http- tai https-URL-osoitteista voi ladata
2335422432 charger que des URL de type "http" et "https".
2335517106 11Mogu se preuzeti samo "http" ili "https" URL-ovi.
2335544123 58Csak "http" vagy "https" URL-ek let
2335567251 possibile scaricare soltanto URL "http" o "https".
2335761534 22Kan alleen URL's met 'http' of 'https' downloaden.
2360673624 accessibility_script_injection_urlhttps://ssl.gstatic.com/accessibility/javascript/android/AndroidVox_v1.js
2367757724 ""https://www.google.com/favicon.ico
```



CONCLUSIONES



- Los pasos son: compilar kernel, lime, el profile, volcar y analizar
- Es importante elegir el JDK correcto para las compilaciones:
 - Android inferiores a 4.4.X debemos utilizar Oracle 6 JDK, no sirve OpenJDK
 - Android 5 y 6 podemos utilizar tanto Oracle 7 JDK como OpenJDK 7
 - Android 7 o superior podemos utilizar OpenJDK 8 o superior
- Hay que usar el toolchain correcto, algunos van a dejar de tener soporte (clang)
- La extracción del dump de un dispositivo virtual es rápida con sleuthkit
- Hay versiones de LiME o Volatility que no funcionan para Android
- Podemos generar máquinas virtuales desde Cli y usarlas en scripts



- Generar los kernels LKM para los dispositivos virtuales
- Realizar procesos automatizados en análisis de malware (versiones nuevas)
- Documentar el impacto del uso de los toolchains clang en versiones viejas
- Uso en cloud de analizadores de malware
- Personalmente: mejorar ExereWare (versión 2) para análisis de riesgos.



REFERENCIAS



- **TFG – Análisis de Malware en Smartphones Android: Prototipo y Estudio de Herramientas**
Buenaventura Salcedo Santos-Olmo
- **Documents Guide Android Developers**
- **Android Security Internals - Nikolay Elenkov**
- **Android & Volatility - Volatility Foundation**
- **Malware en Android: Discovering, Reversing & Forensics- Miguel Ángel García del Moral**
- **Learning Pentesting for Android - Aditya Gupta**
- **Google Group Android-x86**



PREGUNTAS

