

# 高通平台下使用crash解析ramdump

[具体操作参考第6点，其他都是过程记录](#)

## 1. 下载crash工具

github官网: <https://github.com/crash-utility/crash>

解析64位Android kernel使用如下指令编译:

```
make target=ARM64
```

```
make install
```

## 2. 解压ramdump文件

安装:

```
apt-get install p7zip-full
```

解压7z:

```
7z x file.7z
```

解压出来就是文件夹.

## 3. gdb依赖包

```
texinfo bison libncurses-dev
```

## 4. 高通ramdump内存解析脚本适配【记录过程，不用care~】

(1) 尝试使用github上python2版本 (<https://github.com/emonti/qualcomm-opensource-tools>)

python2依赖包

```
enum pyelftools prettytable
```

需要安装0.29版本的pyelftools

<https://stackoverflow.com/questions/77067937/python-modulenotfounderror-no-module-named-elftools-common-py3compat>

python2没有collections.abc模块，修改为import collections

local\_settings报错

[https://blog.csdn.net/m0\\_46250244/article/details/112428261](https://blog.csdn.net/m0_46250244/article/details/112428261)

sched\_info.py

from utils.anomalies import Anomaly

修改为 from anomalies import anomaly

pip报错sys.stderr.write(f"ERROR: {exc}")

curl <https://bootstrap.pypa.io/pip/2.7/get-pip.py> --output get-pip.py

python get-pip.py

【参考：<https://www.cjavapy.com/article/1605/>】

python2不太行~~~

【参考：<https://www.cnblogs.com/rainey-forrest/p/12162216.html>】

报错 supportid

## (2) 尝试用源码中的tools来进行解析

sxr2130p\_repo/emdoor/LINUX/android/vendor/qcom/opensource/tools/linux-ramdump-parser-v2

python3依赖包

numpy anomalies prettytable

## python3遇到问题

```
root@long-ThinkPad-E15:/home/zyr# python3
Python 3.5.2 (default, Jan 26 2021, 13:30:48)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from prettytable import PrettyTable
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/local/lib/python3.5/dist-packages/prettytable/__init__.py", line 3, in <module>
    from .prettytable import (
  File "/usr/local/lib/python3.5/dist-packages/prettytable/prettytable.py", line 334
    raise IndexError(f"Index {index} is invalid, must be an integer or slice")
    ^
SyntaxError: invalid syntax
```

解决方法：重装prettytable即可

直接使用crash解析还有bug😓

尝试了2个dump和vmlinux都有类似的问题...

```
crash ../vmlinux --kaslr 0x1b2d400000
DDRCS0_0.BIN@0x80000000,DDRCS1_0.BIN@0x200000000,DDRCS0_1.BIN@0x100000000,DDRCS1_1.BIN@
0x280000000
```

### GNU gdb (GDB) 10.2

```
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "--host=x86_64-pc-linux-gnu --target=aarch64-elf-linux".
Type "show configuration" for configuration details.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...

please wait... (gathering task table data)
crash: read error: kernel virtual address: fffffffe0fb8eedc8 type: "radix_tree_node.slot[off]"
root@long-ThinkPad-E15:/home/zyr/Desktop/GOM2#
```

```
crash ../EM-B652-V1.0.0-20230905-033443-userdebug/vmlinux --kaslr 0x222c000000
DDRCS0_0.BIN@0x80000000,DDRCS1_0.BIN@0x200000000,DDRCS0_1.BIN@0x100000000,DDRCS1_1.BIN@
0x280000000
```

```

root@long-ThinkPad-E15:/home/zyr/Port_COM77# crash ../EM-B652-V1.0.0-20230905-033443-userdebug/vmlinux --kaslr 0x222c000000 DDRCS0_0.BIN@0
x80000000,DDRCs1_0.BIN@0x200000000,DDRCs0_1.BIN@0x100000000,DDRCs1_1.BIN@0x280000000

crash 7.2.8
Copyright (C) 2002-2020 Red Hat, Inc.
Copyright (C) 2004, 2005, 2006, 2010 IBM Corporation
Copyright (C) 1999-2006 Hewlett-Packard Co
Copyright (C) 2005, 2006, 2011, 2012 Fujitsu Limited
Copyright (C) 2006, 2007 VA Linux Systems Japan K.K.
Copyright (C) 2005, 2011 NEC Corporation
Copyright (C) 1999, 2002, 2007 Silicon Graphics, Inc.
Copyright (C) 1999, 2000, 2001, 2002 Mission Critical Linux, Inc.
This program is free software, covered by the GNU General Public License,
and you are welcome to change it and/or distribute copies of it under
certain conditions. Enter "help copying" to see the conditions.
This program has absolutely no warranty. Enter "help warranty" for details.

GNU gdb (GDB) 7.6
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=x86_64-unknown-linux-gnu --target=aarch64-elf-linux"...

WARNING: kernel relocated [139968MB]: patching 127929 gdb minimal_symbol values

please wait... (gathering task table data)
crash: read error: kernel virtual address: ffffffffe77662a6c8 type: "radix_tree_node shift"

```

#crash/tools源码

```

readmem(node + OFFSET(radix_tree_node_slots) + sizeof(void *) * off, KVADDR, &slot,
sizeof(void *), "radix_tree_node.slot[off]", FAULT_ON_ERROR);

```

看上去是读取内存信息错误

```

#define READ_ERRMSG      "read error: %s address: %llx  type: \"%s\"\\n"

readmem {
switch (READMEM(fd, bufptr, cnt, (memtype == PHYSADDR) || (memtype == XENMACHADDR) ? 0
: addr, paddr))
case READ_ERROR:
if (PRINT_ERROR_MESSAGE)
error(INFO, READ_ERRMSG, memtype_string(memtype, 0), addr, type);
}

#define READMEM  pc->readmem

pc->readmem = read_ramdump;

```

查看源码，paddr不在内存起始范围内，可能是少读取了内存块

```

int
read_ramdump(int fd, void *bufptr, int cnt, ulong addr, physaddr_t paddr)
{
    off_t offset;
    int i, found;
    struct ramdump_def *r = &ramdump[0];

    offset = 0;

    for (i = found = 0; i < nodes; i++) {
        r = &ramdump[i];

        if ((paddr >= r->start_paddr) &&
            (paddr <= r->end_paddr)) {
            offset = (off_t)paddr - (off_t)r->start_paddr;
            found++;
            break;
        }
    }

    if (!found) {
        if (CRASHDEBUG(8))
            fprintf(fp, "read_ramdump: READ_ERROR: "
                    "offset not found for paddr: %llx\n",
                    (ulonglong)paddr);
        error(INFO, "read_ramdump1!!!!\n");
        return READ_ERROR;
    }

    if (CRASHDEBUG(8))
        fprintf(fp,
            "read_ramdump: addr: %lx paddr: %llx cnt: %d offset: %llx\n",
            addr, (ulonglong)paddr, cnt, (ulonglong)offset);
}

```

修改为以下成功👉 (增加了DDRC0\_2.BIN和DDRC1\_2.BIN)

```

crash ../vmlinux-2 --kaslr=0x2eea000000
DDRC0_0.BIN@0x80000000,DDRC1_0.BIN@0x200000000,DDRC0_1.BIN@0x100000000,DDRC1_1.BIN@
0x280000000,DDRC0_2.BIN@180000000,DDRC1_2.BIN@300000000

```

## 5.高通内存解析脚本操作方法

### (1) 下载工具链

包括aarch64-linux-android-gdb, aarch64-linux-android-nm, aarch64-linux-android-objdump

<https://gitlab.com/TeeFirefly/prebuilts/-/tree/master/gcc/linux-x86/aarch64/aarch64-linux-android-4.9/bin>

## (2) 编辑bash脚本

```
ramdump=$1
vmlinux=$2
ramparse_dir=$3/ramparse.py
outdir=$4

gdb="/home/zyr/prebuilts-master-gcc-linux-x86-aarch64/gcc/linux-x86/aarch64/aarch64-
linux-android-4.9/bin/aarch64-linux-android-gdb"
nm="/home/zyr/prebuilts-master-gcc-linux-x86-aarch64/gcc/linux-x86/aarch64/aarch64-
linux-android-4.9/bin/aarch64-linux-android-nm"
objdump="/home/zyr/prebuilts-master-gcc-linux-x86-aarch64/gcc/linux-
x86/aarch64/aarch64-linux-android-4.9/bin/aarch64-linux-android-objdump"

echo $1,$2,$ramparse_dir,$4
python3 $ramparse_dir --vmlinux $vmlinux -g $gdb -n $nm -j $objdump -a $ramdump -o
$outdir -x
```

## (3) 执行脚本，举例如下：

```
#参数1： 内存dump文件目录
#参数2： vmlinux文件目录
#参数3： 高通平台自带内存解析脚本
#参数4： 输出文件夹名（存放解析结果）
./dumpparse.sh Port_COM77 EM-B652-V1.0.0-20230905-033443-userdebug/vmlinux linux-
ramdump-parser-v2 ./out77/
```

## (4) 解析结果

输出文件夹里最重要的是dmesg\_TZ.txt文件，可以根据调用栈快速定界是哪个模块的问题；

除此之外，还可以看到其他信息，如：

tasks.txt 所有进程调用栈信息；

pagetypeinfo.txt 页块信息；

memory.txt 各进程内存占用信息

mem\_stat.txt 整体内存占用信息

## 6.crash解析操作方法

(1) 解压ramdump文件

```
7z x *.7z
```

(2) 获取kaslr地址

```
hexdump -e '16/4 "%08x " "\n"' -s 0x03f6d4 -n 8 OCIMEM.BIN
```

```
root@long-ThinkPad-E15:/home/zyr/Port_COM61# hexdump -e '16/4 "%08x " "\n"' -s 0x03f6d4 -n 8 OCIMEM.BIN
ea000000 0000002e
```

由于arm架构下大小端，组合为00000002eea000000

(3) 确认ramdump的加载偏移量，借助dump\_info.txt确认

```
root@long-ThinkPad-E15:/home/zyr/Port_COM61# cat dump_info.txt | grep DDRCS
1 0x00000000080000000 0000002147483648 DDR CS0 part0 Memo DDRCS0_0.BIN
1 0x00000000100000000 0000002147483648 DDR CS0 part1 Memo DDRCS0_1.BIN
1 0x00000000180000000 0000002147483648 DDR CS0 part2 Memo DDRCS0_2.BIN
1 0x00000000200000000 0000002147483648 DDR CS1 part0 Memo DDRCS1_0.BIN
1 0x00000000280000000 0000002147483648 DDR CS1 part1 Memo DDRCS1_1.BIN
1 0x00000000300000000 0000002147483648 DDR CS1 part2 Memo DDRCS1_2.BIN
```

(4) 组合命令即可

```
crash ../vmlinux-2 --kaslr=0x00000002eea000000
DDRCS0_0.BIN@0x80000000,DDRCS1_0.BIN@0x200000000,DDRCS0_1.BIN@0x100000000,DDRCS1_1.BIN@
0x280000000,DDRCS0_2.BIN@180000000,DDRCS1_2.BIN@300000000
```

## 7.参考链接

1.基于crash工具搭建分析ramdump的平台

<https://blog.csdn.net/u014175785/article/details/112868957>

2.高通平台RAMDUMP分析

[https://www.tsz.wiki/tools/optimize/crash/qcom-ramdump/qcom-ramdump.html#\\_3-ramdump](https://www.tsz.wiki/tools/optimize/crash/qcom-ramdump/qcom-ramdump.html#_3-ramdump)