|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| D:\02-资料分类\04-文档模板\扉页图1-new.jpg | | | | |
|  | AndroidR\_ADB可连定屏问题调试指南 | | |  |
| 文档版本 V1.1 |  | |
| 发布日期 2020-08-17 |  | |
|  | | | | |
|  | 紫光展锐科技有限公司 | |  |  |

|  |
| --- |
| 版权所有 © 紫光展锐科技有限公司。保留一切权利。  本文件所含数据和信息都属于紫光展锐所有的机密信息，紫光展锐保留所有相关权利。本文件仅为信息参考之目的提供，不包含任何明示或默示的知识产权许可，也不表示有任何明示或默示的保证，包括但不限于满足任何特殊目的、不侵权或性能。当您接受这份文件时，即表示您同意本文件中内容和信息属于紫光展锐机密信息，且同意在未获得紫光展锐书面同意前，不使用或复制本文件的整体或部分，也不向任何其他方披露本文件内容。紫光展锐有权在未经事先通知的情况下，在任何时候对本文件做任何修改。紫光展锐对本文件所含数据和信息不做任何保证，在任何情况下，紫光展锐均不负责任何与本文件相关的直接或间接的、任何伤害或损失。  请参照交付物中说明文档对紫光展锐交付物进行使用，任何人对紫光展锐交付物的修改、定制化或违反说明文档的指引对紫光展锐交付物进行使用造成的任何损失由其自行承担。紫光展锐交付物中的性能指标、测试结果和参数等，均为在紫光展锐内部研发和测试系统中获得的，仅供参考，若任何人需要对交付物进行商用或量产，需要结合自身的软硬件测试环境进行全面的测试和调试。非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。 |



|  |
| --- |
| 紫光展锐科技有限公司 |

前 言

概述

本文档主要讲述adb可连定屏问题出现时如何进行现场调试、现场需要抓取哪些关键log以及如何分析。

读者对象

本文档适用于分析adb可连定屏问题的同事，及其他对此感兴趣的同事。

缩略语

|  |  |  |
| --- | --- | --- |
| 缩略语 | 英文全名 | 中文解释 |
| ss | systemserver | 系统核心服务 |
| sf | surfaceflinger | 显示服务 |
| adb | android debug bridge | 调试工具 |
| wms | windowmanagerservice | 窗口管理服务 |
| fw | framework | 框架层 |

变更信息

| 文档版本 | 发布日期 | 作者 | 修改说明 |
| --- | --- | --- | --- |
| V1.0 | 2020-08-17 | Bo.han | 初稿 |
|  |  |  |  |

关键字

adb可连

目 录

[前 言 ii](#_Toc51142696)

[目 录 3](#_Toc51142697)

[1 简介 4](#_Toc51142698)

[1.1 adb可连定屏现场识别 4](#_Toc51142699)

[1.2 adb可连定屏产生原因 4](#_Toc51142700)

[1.3 adb可连定屏问题测试人员处理步骤 4](#_Toc51142701)

[1.4 adb可连定屏rootcause分类 4](#_Toc51142702)

[1.5 adb可连定屏现场分析 5](#_Toc51142703)

[1.5.1 分析流程图 5](#_Toc51142704)

[1.5.2 adb可连定屏现场调试 5](#_Toc51142705)

[1.5.3 adb可连定屏离线log分析 6](#_Toc51142706)

[2 Adb可连定屏问题分析实例 8](#_Toc51142707)

[2.1 关键进程阻塞导致的定屏 8](#_Toc51142708)

[2.2 Security异常导致的定屏 10](#_Toc51142709)

[2.3 应用频繁异常导致的定屏 12](#_Toc51142710)

[2.4 Binder泄漏导致定屏 13](#_Toc51142711)

[2.5 显示相关进程异常导致的定屏 15](#_Toc51142712)

[2.6 低内存导致的定屏 16](#_Toc51142713)

# 简介

造成定屏的因素比较复杂, 但主要涉及两类, 一类是底层kernel异常导致的定屏，一类是android上层异常导致的定屏/卡屏, 对于第二类归结为Adb可连定屏问题, 此类定屏问题则需要更多的现场Log信息进行分析、定位, 而本文主要关注这类定屏问题的调试与处理。

## adb可连定屏现场识别

1.Adb可连识别:

adb devices可识别设备(列表), adb/uart shell 可进入终端。

2.Adb可连定屏涉及场景:

1).测试/使用过程中出现的定屏、黑屏、白屏, 冻屏、花屏等现象;

2).触发或点击物理按键、虚拟按键、触屏等操作均无响应均认定为定屏问题。

备注:

注意此时系统在kernel态是良好的, 无异常的。

## adb可连定屏产生原因

对于Android层面的系统级异常,原生代码在Framework层有watchog机制用于系统自救,可以保证系统能够重启进行恢复, 那为何还会发生Adb可连定屏问题呢?

1. 系统反复重启表现为定屏现象

系统重要进程/线程被Blocked, 或者发生Crash等造成的系统反复重启表现为定屏现象, 此种情况下, watchdog机制仍是有效的。

2. watchdog机制失效导致的定屏

造成watchdog失效的原因主要是watchdog线程被Bolck,诸如monkey测试中, watchdog与monkey进程交互, 由于monkey进程中的相关子线程等主线程的锁(主线程持锁和system server通信)导致的Block,或者由于monkey测试脚本缺失--kill-process-after-error参数导致的定屏等。

3.应用自身异常导致的定屏

此种情况下,系统在android以及kernel层面均是正常的,但只是由于应用自身异常导致应用不再响应按键/触屏事件,或者其他类型的消息事件等,对于用户来说表现的就是定屏现象。

4.systemui/launcher频繁发生crash/anr等问题造成的定屏现象

此两种应用进程属于系统关键进程,所以单独描述,用于区别普通应用进程,当出现anr/crash等问题时进程异常退出,进而无法响应Input相关事件造成定屏。

5.假定屏

系统关键进程/线程并没有发生阻塞,watchdog机制也是有效的, 但出现界面不刷新, 而造成不刷新的原因诸如LCD异常,或者是由于特殊窗口覆盖(诸如FallbackHome、动画窗口)导致无法响应任何操作,这些现象可归结为假定屏。

## adb可连定屏问题测试人员处理步骤

在出现Adb可连定屏现象时,对于测试人员或者使用者该如何操作呢? 首先为避免定屏现场的丢失, 可先按照如下步骤进行处理:

1. 首先, 记录下发生定屏时的时间以及详细的操作步骤, 并进行截图、拍照;

2. 然后, 为避免log的缺失,使用 log工具 Logs4Android2PC导出完整的ylog信息;

3. 接着, 点击Power key 、Volume key以及滑动屏幕查看是否有反应;

4. 最后, 抓取 bugreport;

执行命令:

adb bugreport bugrpt(自定义名称)

待执行完成后, 以1分钟为间隔，反复做3次。

备注: 不要轻易破坏现场,尤其在客户产线以及客退机上出现的定屏问题现场尤为重要。

下面在介绍定屏现场log抓取与现场调试前, 先来看下造成定屏现象的”问题原因”分类。

## adb可连定屏rootcause分类

造成定屏的原因是多种多样的, 但大概率的情况下是系统出现了严重的异常, 因此首先需要关注系统核心进程的运行状态,其次低概率的情况下是由于应用自身的异常导致,需要关注应用进程运行时的调用栈信息。

下面看下造成定屏原因的分类：

1.关注Zygote、Systemserver关键进程或线程被Blocked、Crash;

2.关注Systemserver、Surfaceflinger、Systemui等Binder线程的相关异常(如线程池是否达上限、buffer是否满等);

3.与显示相关的进程如Surfaceflinger、hwComposer是否出现异常(诸如进程频繁重启、D/T/Z状态);

4.驱动异常: 主要按键/触屏驱动、显示驱动、背光驱动的检测;

5.文件系统是否异常;

6.Security是否异常;

7.是否有低内存现象。

## adb可连定屏现场分析

### adb可连定屏分析流程图



### adb可连定屏现场调试

在确认Adb可连后, 可以进行定屏现场的调试与实验,有现场的前提下,对于造成定屏问题原因的定位有很大帮助. 具体现场调试步骤如下:

|  |  |
| --- | --- |
| 检查项 | 描述 |
| 1. 按键事件(是否正常) | 1). 点击Power或者Volume键, 确认按键事件是否在kernel层/驱动层能够正常接收;  2). 按键事件是否有正常上报到FW层并进行分发, 后续系统唤醒亮屏以及背光设置是否正常; |
| 2.进程状态确认 | 1).首先关注 D/Z等状态的进程 以及 对应的trace调用栈信息;  2).查看关键进程运行状态并抓取对应的 trace栈; |
| 3.启动Activity是否正常 | 检验FW层服务工作是否正常 |
| 4.截屏,启动Activity前后分别截屏并进行截图比较 | 检验SF工作是否正常 |
| 5.检测窗口焦点是否有正常切换 | wms服务是否正常 |
| 6.检测背光节点是否有正常设置,其值是否有效 | 背光驱动是否正常 |
| 7.检测df以及mount信息 | data分区是否已满,文件系统是否有正常挂载; |
| 8.检测memory信息 | 是否存在内存不足的现象; |
| 9.触发Sysrq | 查看kernel态主要是D状态进程的运行状况; |
| 10.抓取bugreport | 建议多抓取若干次,此操作由于耗时,放置最后一步执行,bugreport涵盖的内容比较广泛,体现了当时定屏时系统整体的一个运行状态; |

以上各项操作已经开发成Python脚本: devicehung\_logcat,在出现定屏现象时,可通过执行此Python脚本能够自动识别连接的定屏设备,并达到log的自动抓取,此脚本抓取的log命名为 scene\_log,这样在没有定屏现场的情况下,通过脚本抓取的log能够对造成定屏的原因进行分析与定位。

### adb可连定屏离线log分析

其实在大多数情况下出现定屏现象后, 在通过log工具导出的Ylog后, 并不会保留定屏现场的, 也有些Ylog是直接通过adb命令pull出来的, 此方式抓取到Ylog更不完整。在实际处理定屏问题中, 发现这些Ylog, 对于分析、定位定屏原因的帮助比较有限, 除非一些明显的进程异常, 否则较难定位, 而离线log需要关注的信息其实也如前面介绍, 但总体来说可以分为两部分进行分析定位:

1. 由系统方面入手进行分析定位, 看能否确认系统状态是否良好;
2. 由关键应用入手, 分析应用是否有明显的异常。
3. 系统关键进程的运行情况

|  |  |
| --- | --- |
| 进程 | 描述 |
| zygote | 是否有正常启动、运行 |
| system\_server | 检查 trace信息, activity的启动，窗口焦点的切换，按键事件, 广播堆积等状态检测 |
| systemui | 是否有anr、crash出现 |
| launcher | 是否有anr、crash出现 |
| Surfaceflinger/ hwcomposer | 检查trace是否有阻塞在surface的调用;  检测是否有异常log输出 |

2. 文件系统检查，确认是否挂载正常, 存储器件的异常信

3. 系统Common信息的确认,检测Sysinfo

|  |  |
| --- | --- |
| 检查项 | 描述 |
| 内存信息 | Meminfo、vmstat、buddyinfo等, 是否有内存泄漏 |
| Df信息 | Df, 关注 /data 分区 |
| IO信息 | Sgm信息: cpu 使用率, iowait, procs\_running, procs\_blocked等 |
| Binder信息 | 关注关键进程的Binder信息:  1.failed\_transaction\_log/ free async space/ node等  2. Binder buffer /node等是否正常 |

4. Sysrq的确认, 同样关注D 状态进程trace信息

5. Bugreport 的确认

如果bugreport有抓取的话, 除上述关键信息外, 其他需要的关注项如下示.

|  |  |
| --- | --- |
| 关注项 | 描述 |
| log | Android log/kernel log, 记录的log有限, 可参考 |
| Dumpsys activity | 是否有窗口泄露 |
| PAGETYPEINFO /BUDDYINFO /FRAGMENTATION | 是否有内存泄漏/碎片化 |
| LIST OF OPEN FILES | 是否有fd泄漏 |
| BLOCKED PROCESS WAIT-CHANNELS | 线程是否发生阻塞 |
| PROCESSES AND THREADS | 进程&线程信息(Ｄ状态) |

# Adb可连定屏问题分析实例

本章是将结合相关Bug实例来介绍如何分析、定位 “adb可连定屏”的原因。

## 关键进程阻塞导致的定屏

[Bug 1339044](https://bugzilla.unisoc.com/bugzilla/show_bug.cgi?id=1339044) - [RDTest][AndroidR][9863A]0096机进行Monkey测试出现定屏adb可连



1.由于出现定屏问题时, 有保留现场, 因此可以按照”1.3.2 节定屏现场调试” 中的调试步骤/实验手段对定屏现场当时的系统状态进行确认;

2.可参见附件抓取的现场定屏log : scene\_log;

3.定屏可优先关注 systmserver的运行状态;

4.此问题原因是 系统上层关键进程systemserver主线程被阻塞, 等待tid=141线程持有的 0x01abad15 锁的释放, 而tid=141线程等待tid=153线程持有的0x07156fef 锁的释放, 而线程153又等待线程141锁的释放, 进而形成两个Binder线程的互锁造成定屏, 后经确认此互锁问题属于AndroidR原生问题。

|  |
| --- |
| //主线程被Block:  "main" prio=5 tid=1 Blocked  | group="main" sCount=1 dsCount=0 flags=1 obj=0x71a577c0 self=0xb62e3e00  | sysTid=2808 nice=-2 cgrp=default sched=0/0 handle=0xb6812470  | state=S schedstat=( 41541415664 21271252643 47605 ) utm=2622 stm=1531 core=4 HZ=100  | stack=0xbe0d7000-0xbe0d9000 stackSize=8192KB  | held mutexes=  at com.android.server.am.ActivityManagerService.isAppBad(ActivityManagerService.java:6113)  - waiting to lock <0x01abad15> (a com.android.server.am.ActivityManagerServiceEx) held by thread 141  at com.android.server.am.ActivityManagerService.access$1600(ActivityManagerService.java:425)  at com.android.server.am.ActivityManagerService$LocalService.isAppBad(ActivityManagerService.java:19824)  at com.android.server.job.JobSchedulerService.isComponentUsable(JobSchedulerService.java:2262)  at com.android.server.job.JobSchedulerService.areComponentsInPlaceLocked(JobSchedulerService.java:2309)  at com.android.server.job.controllers.StateController.wouldBeReadyWithConstraintLocked(StateController.java:130)  at com.android.server.job.controllers.TimeController.evaluateStateLocked(TimeController.java:210)  at com.android.server.job.JobSchedulerService.evaluateControllerStatesLocked(JobSchedulerService.java:2273)  at com.android.server.job.JobSchedulerService$MaybeReadyJobQueueFunctor.accept(JobSchedulerService.java:2121)  at com.android.server.job.JobSchedulerService$MaybeReadyJobQueueFunctor.accept(JobSchedulerService.java:2069)  at com.android.server.job.JobStore$JobSet.forEachJob(JobStore.java:1247)  at com.android.server.job.JobStore.forEachJob(JobStore.java:299)  at com.android.server.job.JobSchedulerService.maybeQueueReadyJobsForExecutionLocked(JobSchedulerService.java:2161)  at com.android.server.job.JobSchedulerService.access$600(JobSchedulerService.java:144)  at com.android.server.job.JobSchedulerService$JobHandler.handleMessage(JobSchedulerService.java:1900)  - locked <0x05c641d0> (a java.lang.Object)  at android.os.Handler.dispatchMessage(Handler.java:106)  at android.os.Looper.loop(Looper.java:223)  at com.android.server.SystemServer.run(SystemServer.java:646)  at com.android.server.SystemServer.main(SystemServer.java:430)  at java.lang.reflect.Method.invoke(Native method)  at com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.java:603)  at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:925)  //0x01abad15锁被线程 153持有,而此线程同样被Block:  "Binder:2808\_D" prio=5 tid=141 Blocked  | group="main" sCount=1 dsCount=0 flags=1 obj=0x13dcaaf8 self=0x7ee09400  | sysTid=12044 nice=-2 cgrp=default sched=0/0 handle=0x7ac311c0  | state=S schedstat=( 15267427263 12926055779 32775 ) utm=911 stm=615 core=4 HZ=100  | stack=0x7ab36000-0x7ab38000 stackSize=1008KB  | held mutexes=  at com.android.server.uri.UriGrantsManagerService$LocalService.checkGrantUriPermissionFromIntent(UriGrantsManagerService.java:1346)  - waiting to lock <0x07156fef> (a java.lang.Object) held by thread 153  at com.android.server.am.ActiveServices.startServiceLocked(ActiveServices.java:605)  at com.android.server.am.ActiveServices.startServiceLocked(ActiveServices.java:466)  at com.android.server.am.ActivityManagerService.startService(ActivityManagerService.java:15179)  - locked <0x01abad15> (a com.android.server.am.ActivityManagerServiceEx)  at android.app.IActivityManager$Stub.onTransact(IActivityManager.java:2489)  at com.android.server.am.ActivityManagerService.onTransact(ActivityManagerService.java:2925)  at android.os.Binder.execTransactInternal(Binder.java:1159)  at android.os.Binder.execTransact(Binder.java:1123)  //0x07156fef锁被线程 141持有, 此线程同样被Block, 形成互锁:  "Binder:2808\_16" prio=5 tid=153 Blocked  | group="main" sCount=1 dsCount=0 flags=1 obj=0x13dcb168 self=0x7f054a00  | sysTid=16021 nice=0 cgrp=default sched=0/0 handle=0x7b6911c0  | state=S schedstat=( 10397802761 9108427424 23347 ) utm=627 stm=412 core=0 HZ=100  | stack=0x7b596000-0x7b598000 stackSize=1008KB  | held mutexes=  at com.android.server.am.ActivityManagerService.removeContentProviderExternalUnchecked(ActivityManagerService.java:7760)  - waiting to lock <0x01abad15> (a com.android.server.am.ActivityManagerServiceEx) held by thread 141  at com.android.server.am.ActivityManagerService.checkContentProviderUriPermission(ActivityManagerService.java:8233)  at com.android.server.am.ActivityManagerService$LocalService.checkContentProviderUriPermission(ActivityManagerService.java:19076)  at com.android.server.uri.UriGrantsManagerService.checkHoldingPermissionsInternal(UriGrantsManagerService.java:957)  at com.android.server.uri.UriGrantsManagerService.checkHoldingPermissions(UriGrantsManagerService.java:873)  at com.android.server.uri.UriGrantsManagerService.checkGrantUriPermission(UriGrantsManagerService.java:1079)  at com.android.server.uri.UriGrantsManagerService.checkGrantUriPermissionFromIntent(UriGrantsManagerService.java:601)  at com.android.server.uri.UriGrantsManagerService$LocalService.checkGrantUriPermissionFromIntent(UriGrantsManagerService.java:1348)  - locked <0x07156fef> (a java.lang.Object)  at com.android.server.wm.ActivityStarter$Request.resolveActivity(ActivityStarter.java:530)  at com.android.server.wm.ActivityStarter.execute(ActivityStarter.java:647)  at com.android.server.wm.ActivityTaskManagerService.startActivityAsUser(ActivityTaskManagerService.java:1117)  at com.android.server.wm.ActivityTaskManagerService.startActivityAsUser(ActivityTaskManagerService.java:1089)  at android.app.IActivityTaskManager$Stub.onTransact(IActivityTaskManager.java:1498)  at android.os.Binder.execTransactInternal(Binder.java:1159)  at android.os.Binder.execTransact(Binder.java:1123) |

## Security异常导致的定屏

[Bug 1376000](https://bugzilla.unisoc.com/bugzilla/show_bug.cgi?id=1376000) - [PSST][Android11][Sharkl5Pro][T610][3G][CTCC][Native]进行渐进式掉电测试, #0734测试168小时出现定屏，adb可连，累计出现10台



1.按照现场调试步骤分析现场, 具体现场log请参见附件抓取的”1376000\_现场log.rar”;

2.定屏原因android层发生反复重启表现为定屏现象, 查看df命令时发现 data分区满, 并且现场看并未保存下来任何的 ylog, 因此通过现场清除一些临时文件用于抓取log, 并最终定位到造成定屏的原因是加密失败导致的data挂载失败。

|  |
| --- |
| 1.现场df并没有看到 data目录 ,但可以 cd /data 进入, 但目录下显示空 ,说明data并没有完成挂载:  ums512\_1h10:/ $ df  Filesystem 1K-blocks Used Available Use% Mounted on  /dev/block/dm-4 971480 968536 0 100% /  tmpfs 1886080 960 1885120 1% /dev  tmpfs 1886080 0 1886080 0% /mnt  /dev/block/dm-5 538840 537224 0 100% /system\_ext  /dev/block/dm-6 833364 830840 0 100% /vendor  /dev/block/dm-7 346888 345848 0 100% /product  tmpfs 1886080 0 1886080 0% /apex  /dev/block/loop6 4740 4712 0 100% /apex/com.android.conscrypt@300000000  /dev/block/loop7 6400 6368 0 100% /apex/com.android.cellbroadcast@300000000  /dev/block/loop8 8004 7976 0 100% /apex/com.android.permission@300000000  /dev/block/loop9 131356 127692 1036 100% /apex/com.android.vndk.v30@1  /dev/block/loop10 2384 2352 0 100% /apex/com.android.resolv@300000000  /dev/block/loop11 844 816 12 99% /apex/com.android.tzdata@300000000  /dev/block/loop12 560 532 20 97% /apex/com.android.ipsec@300000000  /dev/block/loop13 3668 3640 0 100% /apex/com.android.mediaprovider@300000000  /dev/block/loop14 332 300 28 92% /apex/com.android.sdkext@300000000  /dev/block/loop15 5652 5624 0 100% /apex/com.android.neuralnetworks@300000000  /dev/block/loop16 860 832 12 99% /apex/com.android.tethering@300000000  /dev/block/loop17 4932 4904 0 100% /apex/com.android.wifi@300000000  /dev/block/loop18 24640 24608 0 100% /apex/com.android.i18n@1  /dev/block/loop19 1696 1668 0 100% /apex/com.android.os.statsd@300000000  /dev/block/loop20 232 92 136 41% /apex/com.android.apex.cts.shim@1  /dev/block/loop21 3932 3904 0 100% /apex/com.android.extservices@300000000  /dev/block/loop22 7728 7700 0 100% /apex/com.android.adbd@300000000  /dev/block/loop23 131340 118768 9944 93% /apex/com.android.art@1  /dev/block/loop24 5108 5072 0 100% /apex/com.android.runtime@1  /dev/block/loop25 18572 18540 0 100% /apex/com.android.media.swcodec@300000000  /dev/block/loop26 4668 4640 0 100% /apex/com.android.media@300000000  cgroup\_root 1886080 8 1886072 1% /sys/fs/cgroup  //data目录为空:  ums512\_1h10:/ # cd data/  ums512\_1h10:/data # ls  2.并且在此种情况下,没有任何log生成, 因此抓取了现场kernel log,只看到 credstore频繁的发生crash,而其原因也是与data相关:  [17292.246942] DEBUG: Timestamp: 1970-01-01 04:48:12+0000  [17292.250325] DEBUG: pid: 20107, tid: 20107, name: credstore >>> /system/bin/credstore <<<  [17292.255669] DEBUG: uid: 1076  [17292.264080] DEBUG: signal 6 (SIGABRT), code -1 (SI\_QUEUE), fault addr --------  [17292.267219] DEBUG: Abort message: 'Check failed: chdir(data\_dir.c\_str()) != -1 chdir: /data/misc/credstore: No such file or directory'  3. 查看/cache log, 发现元数据加密的key没找到，导致data挂载失败:  1925:00784 <15> [ 5.242332][01-01 00:00:04.242] vdc: Waited 0ms for vold  1926:00785 <15> [ 5.242798][01-01 00:00:04.242] vold: fscrypt\_mount\_metadata\_encrypted: /data 0  1927:00786 <15> [ 5.242970][01-01 00:00:04.242] vold: metadata\_key\_dir/key: /metadata/vold/metadata\_encryption/key  1928:00787 <14> [ 5.246360][01-01 00:00:04.246] vold: List of Keymaster HALs found:  1929:00788 <14> [ 5.246666][01-01 00:00:04.246] vold: Keymaster HAL #1: TrustyKeymaster from Unisoc SecurityLevel: TRUSTED\_ENVIRONMENT HAL: android.hardware.keymaster@4.1::IKeymasterDevice/default  1930:00789 <15> [ 5.247740][01-01 00:00:04.247] vold: Computing HMAC with params { (seed: , nonce: a7044f4f2851ea52e374c4b392cdaff20d04d5ded9a54c7de777b6c9338fa) }  1931:0078A <15> [ 5.247771][01-01 00:00:04.247] vold: Computing HMAC for TrustyKeymaster from Unisoc SecurityLevel: TRUSTED\_ENVIRONMENT HAL: android.hardware.keymaster@4.1::IKeymasterDevice/default  1932:0078B <14> [ 5.248625][01-01 00:00:04.248] vold: Using TrustyKeymaster from Unisoc for encryption. Security level: TRUSTED\_ENVIRONMENT, HAL: android.hardware.keymaster@4.1::IKeymasterDevice/default  >>>  >1933:0078C <11> [ 5.249188][01-01 00:00:04.249] vold: No key found in /metadata/vold/metadata\_encryption/key  1934:0078D <11> [ 5.249633][01-01 00:00:04.249] vdc: Command: cryptfs mountFstab /dev/block/platform/soc/soc:ap-ahb/20600000.sdio/by-name/userdata /data Failed: Status(-8, EX\_SERVICE\_SPECIFIC): '0:  4.进一步根据上述异常log, 现场查看加密data所需的 key文件确实不存在:  ums512\_1h10:/metadata/vold/metadata\_encryption/tmp # ls -al  total 68  drwx------ 2 root root 4096 1970-01-01 00:00 .  drwx------ 3 root root 4096 1970-01-01 00:00 ..  -rw------- 1 root root 92 1970-01-01 00:00 encrypted\_key  -rw------- 1 root root 231 1970-01-01 00:00 keymaster\_key\_blob  -rw------- 1 root root 16384 1970-01-01 00:00 secdiscardable  -rw------- 1 root root 10 1970-01-01 00:00 stretching  -rw------- 1 root root 1 1970-01-01 00:00 version |

## 应用频繁异常导致的定屏

[Bug 1391301](https://bugzilla.unisoc.com/bugzilla/show_bug.cgi?id=1391301) (SPCSS00718516) - [UMS312][AGENEWTECH][9190SWO][7323]系统无法正常显示，power键灭屏背光亮

此问题并没有保留现场, 在导出ylog后定屏现场就被破坏掉了,只能通过ylog进行分析, 因此可按照”1.3.3 adb可连定屏离线log分析”来定位问题，由两部分入手分析:

1.由系统方面进行分析定位, 需排除系统是否有异常;

2.重点应用方面的异常检测;

3.详细分析如下:

|  |
| --- |
| 系统方面: 发生定屏时,系统负载不大,各CPU的使用率也不高,可用内存也不紧张,IO也无阻塞,因此系统状态确认良好;  应用方面：systemui进程频繁发生InflateExceptioncrash异常导致窗口无法正常显示, 按键事件无法响应进而造成定屏：  C2CB684 08-15 17:32:09.609 30380 30380 E AndroidRuntime: Process: com.android.systemui, PID: 30380  C2CB684 08-15 17:32:09.609 30380 30380 E AndroidRuntime: android.view.InflateException: Binary XML file line #21: Attempt to invoke virtual method 'void android.widget.TextView.setVisibility(int)' on a null object reference  C2CB684 08-15 17:32:09.609 30380 30380 E AndroidRuntime: Caused by: java.lang.NullPointerException: Attempt to invoke virtual method 'void android.widget.TextView.setVisibility(int)' on a null object reference  C2CB684 08-15 17:32:09.609 30380 30380 E AndroidRuntime: at com.android.keyguard.KeyguardStatusView.updateDark(KeyguardStatusView.java:541)  C2CB684 08-15 17:32:09.609 30380 30380 E AndroidRuntime: at com.android.keyguard.KeyguardStatusView.onFinishInflate(KeyguardStatusView.java:235)  C2CB684 08-15 17:32:09.609 30380 30380 E AndroidRuntime: at android.view.LayoutInflater.rInflate(LayoutInflater.java:876)  C2CB684 08-15 17:32:09.609 30380 30380 E AndroidRuntime: at android.view.LayoutInflater.rInflateChildren(LayoutInflater.java:824)  C2CB684 08-15 17:32:09.609 30380 30380 E AndroidRuntime: at android.view.LayoutInflater.inflate(LayoutInflater.java:515)  C2CB684 08-15 17:32:09.609 30380 30380 E AndroidRuntime: at android.view.LayoutInflater.inflate(LayoutInflater.java:423)  C2CB684 08-15 17:32:09.609 30380 30380 E AndroidRuntime: at com.android.systemui.statusbar.phone.NotificationPanelView.onThemeChanged(NotificationPanelView.java:395)  C2CB684 08-15 17:32:09.609 30380 30380 E AndroidRuntime: at com.android.systemui.statusbar.phone.StatusBar.onThemeChanged(StatusBar.java:1331)  C2CB684 08-15 17:32:09.609 30380 30380 E AndroidRuntime: at com.android.systemui.statusbar.phone.StatusBar.updateTheme(StatusBar.java:4490)  C2CB684 08-15 17:32:09.609 30380 30380 E AndroidRuntime: at com.android.systemui.statusbar.phone.StatusBar.setBarState(StatusBar.java:4673)  C2CB684 08-15 17:32:09.609 30380 30380 E AndroidRuntime: at com.android.systemui.statusbar.phone.StatusBar.hideKeyguardImpl(StatusBar.java:4276)  C2CB684 08-15 17:32:09.609 30380 30380 E AndroidRuntime: at com.android.systemui.statusbar.phone.StatusBar.updateIsKeyguard(StatusBar.java:4116)  C2CB684 08-15 17:32:09.609 30380 30380 E AndroidRuntime: at com.android.systemui.statusbar.phone.StatusBar.access$3000(StatusBar.java:279)  C2CB684 08-15 17:32:09.609 30380 30380 E AndroidRuntime: at com.android.systemui.statusbar.phone.StatusBar$16.onStartedWakingUp(StatusBar.java:5146)  C2CB684 08-15 17:32:09.609 30380 30380 E AndroidRuntime: at com.android.systemui.keyguard.-$$Lambda$TPhVA13qrDBGFKbgQpRNBPBvAqI.accept(Unknown Source:2)  C2CB684 08-15 17:32:09.609 30380 30380 E AndroidRuntime: at com.android.systemui.keyguard.Lifecycle.dispatch(Lifecycle.java:39)  C2CB684 08-15 17:32:09.609 30380 30380 E AndroidRuntime: at com.android.systemui.keyguard.WakefulnessLifecycle.dispatchStartedWakingUp(WakefulnessLifecycle.java:48)  C2CB684 08-15 17:32:09.609 30380 30380 E AndroidRuntime: at com.android.systemui.keyguard.KeyguardLifecyclesDispatcher$1.handleMessage(KeyguardLifecyclesDispatcher.java:69)  C2CB684 08-15 17:32:09.609 30380 30380 E AndroidRuntime: at android.os.Handler.dispatchMessage(Handler.java:106)  C2CB684 08-15 17:32:09.609 30380 30380 E AndroidRuntime: at android.os.Looper.loop(Looper.java:193) |

## Binder泄漏导致定屏

Bug 1275032 - [PSST][Android9.0][SharkLE][GMS][PTR3Block]进行Monkey4System测试, #1430测试22.7145小时出现定屏,adb可连



1.此定屏有保留现场, 依然按照”定屏现场调试”来分析;

2.定屏问题, 对于Fw层来说, 仍然是首先关注 systemserver进程的运行状态, 分析trace调用栈信息, 待排除systemserver异常后, 再进行其他项的检查。

|  |
| --- |
| 目前看出现的10台定屏现象均与Binder node泄漏相关,主线程在内的若干systemserver线程均被阻塞, 由于watchdog也被Block住进而使得watchdog失效没有触发系统重启而出现定屏现象, 而提供的 1430 ylog中有发生重启,但其根因也是与Binder泄漏有关,现以408为例, 在BpBinder create时检测到 system\_server进程中持有的binder proxy已达水位线,其主要来自uid=10033 systemui进程的Binder node对象, 进而触发BpBinder的sLimitCallback,调用到BinderProxy.dumpProxyInterfaceCounts()，对多达6000个BpBinder调用bp.getInterfaceDescriptor()耗时较长导致:  ./03-0102\_155041--0102\_195322/android/0000.log:403116:M7F7E68 01-02 17:03:15.176 677 1034 E BpBinder: Too many binder proxy objects sent to uid 1000 from uid 10024 (6000 proxies held)  ./03-0102\_155041--0102\_195322/android/main.log:2423334:7F7E68 01-02 17:03:15.176 677 1034 E BpBinder: Too many binder proxy objects sent to uid 1000 from uid 10024 (6000 proxies held  "main" prio=5 tid=1 Blocked  | group="main" sCount=1 dsCount=0 flags=1 obj=0x77a38998 self=0x7b5d214c00  | sysTid=677 nice=-2 cgrp=default sched=0/0 handle=0x7be2d03548  | state=S schedstat=( 2100403017136 4821644493180 6060703 ) utm=126795 stm=83245 core=0 HZ=100  | stack=0x7ff0166000-0x7ff0168000 stackSize=8MB  | held mutexes=  at com.android.server.am.ActivityManagerService.getMemoryTrimLevel(ActivityManagerService.java:16283)  - waiting to lock <0x09e5e93b> (a com.android.server.am.ActivityManagerServiceEx) held by thread 139  at com.android.server.job.JobSchedulerService.assignJobsToContextsLocked(JobSchedulerService.java:2106)  at com.android.server.job.JobSchedulerService.maybeRunPendingJobsLocked(JobSchedulerService.java:2065)  at com.android.server.job.JobSchedulerService.access$800(JobSchedulerService.java:131)  at com.android.server.job.JobSchedulerService$JobHandler.handleMessage(JobSchedulerService.java:1672)  - locked <0x09799822> (a java.lang.Object)  at android.os.Handler.dispatchMessage(Handler.java:106)  at android.os.Looper.loop(Looper.java:193)  at com.android.server.SystemServer.run(SystemServer.java:469)  at com.android.server.SystemServer.main(SystemServer.java:309)  at java.lang.reflect.Method.invoke(Native method)  at com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.java:504)  at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:838)    "watchdog" prio=5 tid=102 Blocked  | group="main" sCount=1 dsCount=0 flags=1 obj=0x13f53de0 self=0x7b3a8af800  | sysTid=1045 nice=0 cgrp=default sched=0/0 handle=0x7b39f004f0  | state=S schedstat=( 9345764295 20030447544 16056 ) utm=174 stm=760 core=3 HZ=100  | stack=0x7b39dfd000-0x7b39dff000 stackSize=1041KB  | held mutexes=  at com.android.server.am.ActivityManagerService.setProcessLimit(ActivityManagerService.java:8870)  - waiting to lock <0x09e5e93b> (a com.android.server.am.ActivityManagerServiceEx) held by thread 139  at com.android.server.Watchdog.autoAdjustProcessLimit(Watchdog.java:796)  at com.android.server.Watchdog.run(Watchdog.java:503)    "Binder:677\_1D" prio=5 tid=139 Native  | group="main" sCount=1 dsCount=0 flags=1 obj=0x13f57a20 self=0x7b3a8b1c00  | sysTid=5573 nice=-2 cgrp=default sched=0/0 handle=0x7b33ba24f0  | state=S schedstat=( 1068485431907 1984563929728 2490980 ) utm=67407 stm=39441 core=3 HZ=100  | stack=0x7b33aa7000-0x7b33aa9000 stackSize=1009KB  | held mutexes=  kernel: \_\_switch\_to+0xa4/0xd4  kernel: futex\_wait\_queue\_me+0xcc/0x144  kernel: futex\_wait+0xec/0x20c  kernel: do\_futex+0x174/0xb3c  kernel: SyS\_futex+0x120/0x1ac  kernel: \_\_sys\_trace\_return+0x0/0x4  native: #00 pc 000000000001f02c /system/lib64/libc.so (syscall+28)  native: #01 pc 000000000002217c /system/lib64/libc.so (\_\_futex\_wait\_ex(void volatile\*, bool, int, bool, timespec const\*)+140)  native: #02 pc 0000000000082824 /system/lib64/libc.so (NonPI::MutexLockWithTimeout(pthread\_mutex\_internal\_t\*, bool, timespec const\*)+216) //等待 AutoMutex \_l(mLock);  native: #03 pc 000000000007a380 /system/lib64/libbinder.so (android::ProcessState::getStrongProxyForHandle(int)+64)  native: #04 pc 0000000000061aac /system/lib64/libbinder.so (android::acquire\_object(android::sp<android::ProcessState> const&, flat\_binder\_object const&, void const\*, unsigned long\*)+252)  native: #05 pc 00000000000657a8 /system/lib64/libbinder.so (android::Parcel::writeObject(flat\_binder\_object const&, bool)+292)  native: #06 pc 0000000000061e58 /system/lib64/libbinder.so (android::flatten\_binder(android::sp<android::ProcessState> const&, android::sp<android::IBinder> const&, android::Parcel\*)+252)  native: #07 pc 0000000000064c88 /system/lib64/libbinder.so (android::Parcel::writeStrongBinder(android::sp<android::IBinder> const&)+56)  native: #08 pc 0000000000120960 /system/lib64/libandroid\_runtime.so (android::android\_os\_Parcel\_writeStrongBinder(\_JNIEnv\*, \_jclass\*, long, \_jobject\*)+72)  at android.os.Parcel.nativeWriteStrongBinder(Native method)  at android.os.Parcel.writeStrongBinder(Parcel.java:737)  at android.app.servertransaction.ClientTransaction.writeToParcel(ClientTransaction.java:173)  at android.app.IApplicationThread$Stub$Proxy.scheduleTransaction(IApplicationThread.java:1772)  at android.app.servertransaction.ClientTransaction.schedule(ClientTransaction.java:129)  at com.android.server.am.ClientLifecycleManager.scheduleTransaction(ClientLifecycleManager.java:47)  at com.android.server.am.ClientLifecycleManager.scheduleTransaction(ClientLifecycleManager.java:100)  at com.android.server.am.ActivityManagerService.updateGlobalConfigurationLocked(ActivityManagerService.java:22959)  at com.android.server.am.ActivityManagerService.updateDisplayOverrideConfigurationLocked(ActivityManagerService.java:23072)  at com.android.server.am.ActivityManagerService.updateDisplayOverrideConfigurationLocked(ActivityManagerService.java:23049)  at com.android.server.am.ActivityStackSupervisor.ensureVisibilityAndConfig(ActivityStackSupervisor.java:1693)  at com.android.server.am.ActivityStack.resumeTopActivityInnerLocked(ActivityStack.java:2657)  - locked <0x016c52b3> (a com.android.server.wm.WindowHashMap) |

## 显示相关进程异常导致的定屏

[Bug 1374488](https://bugzilla.unisoc.com/bugzilla/show_bug.cgi?id=1374488) (SPCSS00709145) - [SC9863A][SENWA][LX6530][7912]请问基于s9863a1h10\_go\_32b如何配置使用LPDDR4

1.定屏现场没有保留, 提供了离线 串口log以及ylog;

2.通过离线log可以明显的看到与显示相关的服务进程hwcomposer连续发生crash, 而crash原因是由于lcd初始化失败导致。

|  |
| --- |
| //lcd初始化失败:  [ 4.656364] c0 sprd-mipi-panel-drv 63100000.dsi.0: 63100000.dsi.0 supply power not found, using dummy regulator  [ 4.666486] c0 [drm:sprd\_panel\_probe] \*ERROR\* /soc/mm/dsi@63100000/panel: could not find lcd\_gc9503v\_mipi\_fwvga node  [ 4.676931] c0 [drm:sprd\_panel\_probe] \*ERROR\* parse panel info failed  [ 22.093609] c4 init: Command 'start console' action=ro.debuggable=1 (/init.rc:843) took 574ms and succeeded  [ 22.108840] c4 init: Service 'vendor.hwcomposer-2-1' (pid 395) exited with status 1  [ 22.116497] c4 init: Sending signal 9 to service 'vendor.hwcomposer-2-1' (pid 395) process group...  [ 22.125850] c4 libprocessgroup: Successfully killed process cgroup uid 1000 pid 395 in 0ms  [ 22.135245] c0 init: Sending signal 9 to service 'surfaceflinger' (pid 433) process group...  [ 22.149989] c1 libprocessgroup: Successfully killed process cgroup uid 1000 pid 433 in 5ms  [ 22.160174] c3 sbuf-6-4 not ready to write!  [ 22.160766] c2 init: Service 'surfaceflinger' (pid 433) received signal 9 |

## 低内存导致的定屏

Bug 1357737 - [PSST][Android11][Sharkl3][2G][Go][PTR2Block]进行Monkey4System测试, #0800测试17.49小时出现黑屏 adb可连



此定屏虽然现场有保留, 但现场能够获取的定屏信息比较有限, 需要现场log以及离线Ylog相结合的方式进行分析, 最终定位是由于ZO进程引起的 lmkd pending，lmkd 长时间失效, 导致系统长时间处于低内存状态而引发的定屏。

|  |
| --- |
| 1.定屏现场log: systemserver进程状态异常  s9863a1h10\_go\_32b:/ # ps -A|grep system\_server  system 892 365 0 0 do\_task\_dead 0 Z [system\_server]  1|s9863a1h10\_go\_32b:/ # ps -A|grep " Z "  system 892 365 0 0 do\_task\_dead 0 Z [system\_server]  u0\_a147 974 365 0 0 do\_task\_dead 0 Z [android.youtube]  u0\_a147 2028 365 0 0 do\_task\_dead 0 Z [android.youtube]  u0\_a125 2220 365 0 0 do\_task\_dead 0 Z [id.gms.unstable]  u0\_a125 3002 365 0 0 do\_task\_dead 0 Z [gle.android.gms]  u0\_a147 4093 365 0 0 do\_task\_dead 0 Z [android.youtube]  u0\_a125 5137 365 0 0 do\_task\_dead 0 Z [gle.android.gms]  u0\_a125 6206 365 0 0 do\_task\_dead 0 Z [gle.android.gms]  u0\_a125 7576 365 0 0 do\_task\_dead 0 Z [id.gms.unstable]  u0\_a125 7712 365 0 0 do\_task\_dead 0 Z [.gms.persistent]  u0\_a125 7778 365 0 0 do\_task\_dead 0 Z [gle.android.gms]  u0\_a125 9872 365 0 0 do\_task\_dead 0 Z [id.gms.unstable]  2.Ylog分析:  1).关键线程被Block引发watchdog导致的重启问题, tid=15 的android.display线程等待tid=163持有的0x057590d6锁的释放:  "android.display" prio=5 tid=15 Blocked  | group="main" sCount=1 dsCount=0 flags=1 obj=0x136c41c0 self=0xa593a600  | sysTid=920 nice=-3 cgrp=default sched=0/0 handle=0x7c4251c0  | state=S schedstat=( 9116309523056 1582737754414 6440419 ) utm=674502 stm=237128 core=3 HZ=100  | stack=0x7c322000-0x7c324000 stackSize=1040KB  | held mutexes=  at com.android.server.usage.UsageStatsService.reportEventOrAddToQueue(UsageStatsService.java:765)  - waiting to lock <0x057590d6> (a java.lang.Object) held by thread 163  at com.android.server.usage.UsageStatsService.access$100(UsageStatsService.java:121)  at com.android.server.usage.UsageStatsService$LocalService.reportEvent(UsageStatsService.java:2099)  at com.android.server.am.ActivityManagerService.updateActivityUsageStats(ActivityManagerService.java:3233)  - locked <0x028017c8> (a com.android.server.am.ActivityManagerServiceEx)  at com.android.server.am.ActivityManagerService$LocalService.updateActivityUsageStats(ActivityManagerService.java:19421)  - locked <0x028017c8> (a com.android.server.am.ActivityManagerServiceEx)  at com.android.server.wm.-$$Lambda$UZl9uqUNteVgplGGEK6TMzf-7zk.accept(lambda:-1)  at com.android.internal.util.function.pooled.PooledLambdaImpl.doInvoke(PooledLambdaImpl.java:342)  at com.android.internal.util.function.pooled.PooledLambdaImpl.invoke(PooledLambdaImpl.java:201)  at com.android.internal.util.function.pooled.OmniFunction.run(OmniFunction.java:97)  at android.os.Handler.handleCallback(Handler.java:938)  at android.os.Handler.dispatchMessage(Handler.java:99)  at android.os.Looper.loop(Looper.java:223)  at android.os.HandlerThread.run(HandlerThread.java:67)  at com.android.server.ServiceThread.run(ServiceThread.java:44)  2).发生watchdog后执行退出流程,但在退出过程中又出现异常,导致无法正常退出造成黑屏现象;  3). 从整体上看,系统的可用内存始终在消耗中,并且从 07-02 17:15:20 时间点开始直到发生wtachdog前,其系统内存恶化突然加剧, 存在内存泄漏的现象:  cat /proc/meminfo on 07-02 17:09:14  MemTotal: 1934132 kB  MemFree: 75856 kB  MemAvailable: 468528 kB  Buffers: 228 kB  Cached: 482556 kB  SwapCached: 61200 kB  cat /proc/meminfo on 07-02 17:15:20  MemTotal: 1934132 kB  MemFree: 57688 kB  MemAvailable: 188008 kB  Buffers: 312 kB  Cached: 287192 kB  cat /proc/meminfo on 07-02 17:21:30  MemTotal: 1934132 kB  MemFree: 47916 kB  MemAvailable: 145328 kB  cat /proc/meminfo on 07-02 17:27:44  MemTotal: 1934132 kB  MemFree: 44584 kB  MemAvailable: 78632 kB  Buffers: 664 kB  Cached: 114124 kB  4). 进一步检测ylog发现07-02 17:16时间点出现最后一次 lmkd kill 进程后，从这个时点开始到问题出现点 17：40 只有kswapd 起作用, lmkd 的查杀并不积极. |