Triaging modern Android devices (aka android_triage bash script)

By Mattia Epifani - March 30, 2021

When dealing with mobile devices, you need to follow a procedure that includes various stages from identification to reporting. One of the key-point in this process is the **acquisition**, where we extract a copy of the data stored in the mobile device.

While evaluating your acquisition method, you should always consider the old but gold rule of the **order of volatility** (RFC 3227 - "*When collecting evidence you should proceed from the volatile to the less volatile*").

Our general goal is to obtain a **physical dump** of the internal memory, from the very first to the very last bit. On most modern smartphone devices, where file-based encryption (FBE) is in use, a **full file system** is typically our "best evidence".

These extraction methods often rely on vulnerabilities and exploits (either at the hardware or the software level) developed to overcome device security. We use various tools and techniques and sometimes also a physical approach.

In some cases, especially for recent chipsets or operating system versions, there's no public solution to obtain a physical or a full file system dump. In those cases, we have to rely on (and we are limited to) communication options provided by the device.

On Android devices, in particular, we mostly rely on:

1. ADB (Android Debug Bridge)
2. MTP (Media Transfer Protocol)
3. Manufacturer backup tools like Samsung SmartSwitch and Huawei HiSuite

On the ADB side, most of the commercial tools use **four methods**:

1. **ADB Backup**: the tool executes the "**adb backup**" command, with some possible variations among the different tools. This typically doesn't include native and third-party apps data: every single app can specify in the AndroidManifest file (**android:allowBackup** parameter) if its data should be included or not in an ADB

in reading call log and SMS only to the default handler: this required some changes in the flow used by most forensics tools
3. **ADB Screenshot**: the tool leverages the ADB feature that allows taking a screenshot of the device. In this way, also third-party apps can be "screenshotted" and then processed in the tool, by using some image analysis techniques, to parse the content and make it readable and searchable
4. **APK Downgrade**: the tool uninstalls the actual version of a third-party package (app) by preserving user data, and installs a previous version of the same package where the android:allowBackup parameter is set to "true". The tool creates an ADB Backup that includes the specific app and then reinstalls the actual app version.

The Android Debug Bridge is, in reality, more powerful than this, as it allows us to obtain a shell without root privileges. In this shell you can **run commands** (both Linux-native and Android-specific) and **pull files and folders from the device** (where root privileges are not required).

Over the years, various opensource tools and scripts leveraging ADB were published. Among the others, I strongly suggest you to take a look at:

- Andriller
- Android Live Info
- Foroboto
- ADB Export

In this complex scenario, it's difficult to answer the question:

Am I getting the "best evidence" from this mobile device with my tools?

I would say, no.

Just as a simple example, by using the native Android command "**dumpsys usagestats**" you can obtain information about apps usage, including deleted ones (same concept of the "/usagestats/" folder in a full file system acquisition).

None of the four methods used by forensic tools can provide you this information.

Inspired by previous works and after reading a lot of literature on the topic (including the new Android Internals book), I decided to write a **simple bash script, called**

android_triage that leverages ADB at its maximum level.

https://github.com/RealityNet/android_triage

The script was developed with the precious support of my dear friend Giovanni Rattarto, the Tsurugi Project founder. It was tested on Linux and Mac OS X and it only has two requirements: the ADB executable for the specific platform and the dialog command (instructions for Mac OS X available here).

DISCLAIMERS:

- This script is **only for testing purposes**
- It **requires an unlocked device** (not protected with a passcode or with known passcode)
- **ADB must be active on the device** and the device must be paired with the host
- The script **doesn't install or uninstall any APK** and so it's not able to extract, for example, Call Logs, SMS messages, Third-party apps data, and so on

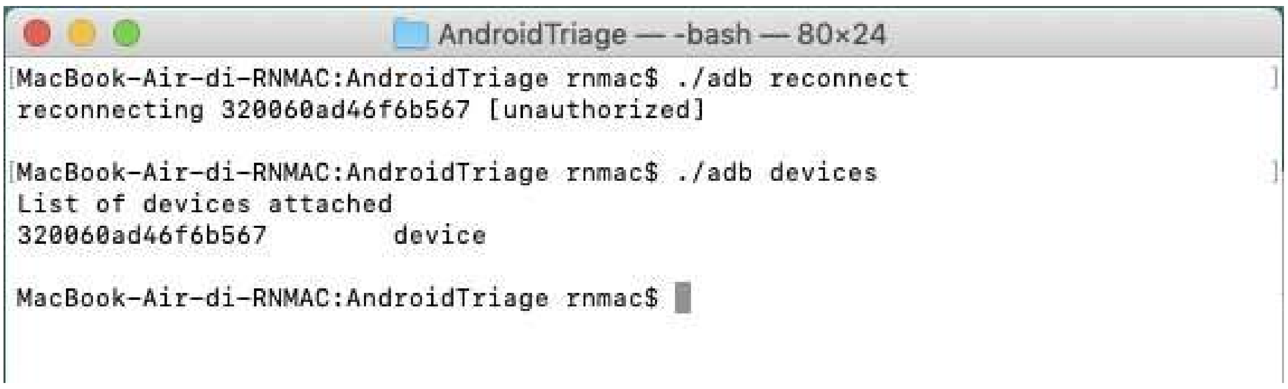
Please test it and provide feedbacks, as we want to improve the script over the time by adding commands and adjusting them based on various Android manufacturers and OS versions.

Once downloaded, you need to make the script executable (**chmod +x android_triage.sh**) and then you need to connect the device.

Before executing the script, you need to **allow ADB communication between the mobile device and your host by pairing them**.

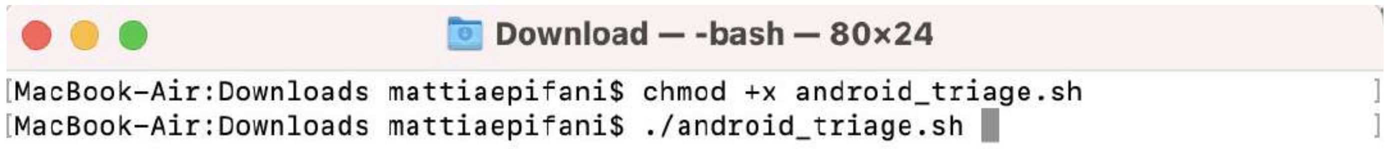
There are various ways to do this: probably the easiest one is to execute **adb reconnect**.

You can always check the pairing status by executing the **adb devices** command: when you see the "**device**" message, it means that the pairing was successful.



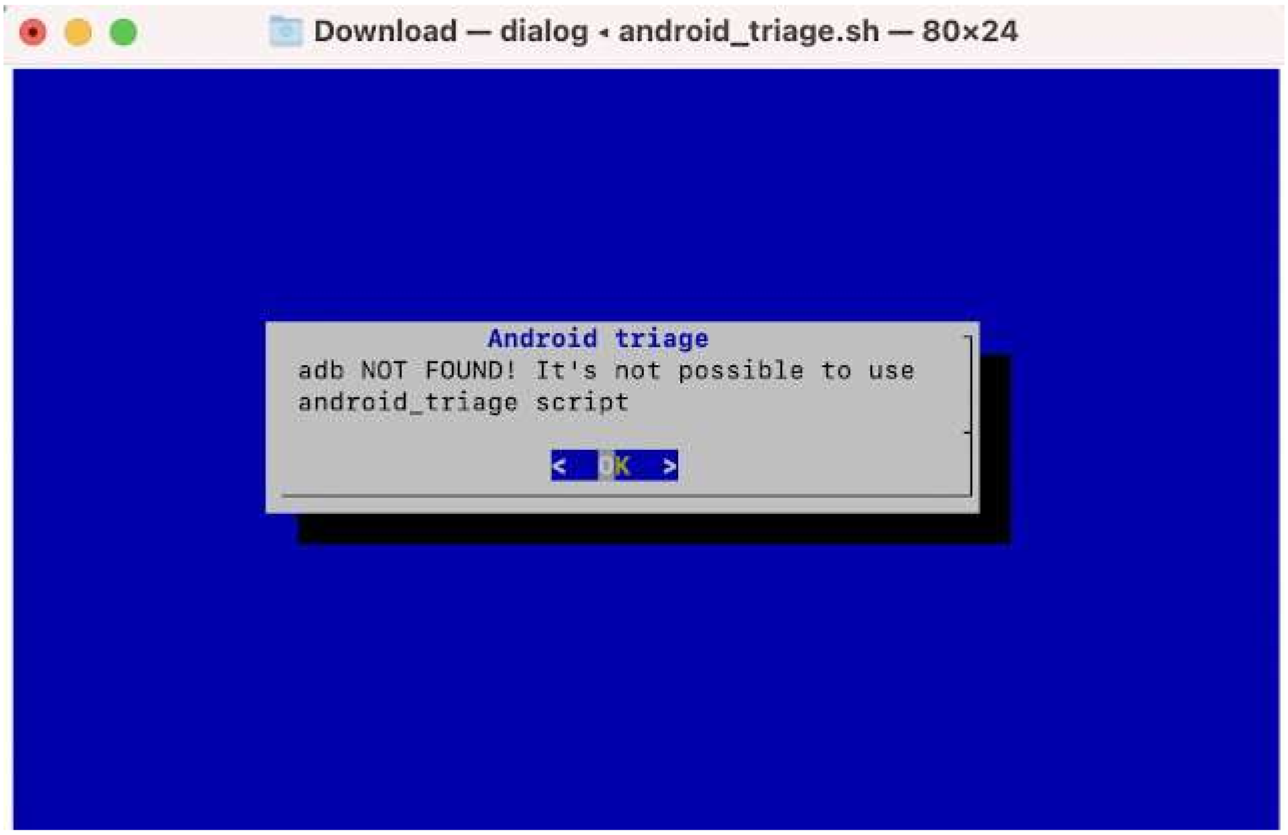
```
MacBook-Air-di-RNMAC:AndroidTriage rnmac$ ./adb reconnect
reconnecting 320060ad46f6b567 [unauthorized]

MacBook-Air-di-RNMAC:AndroidTriage rnmac$ ./adb devices
List of devices attached
320060ad46f6b567        device

MacBook-Air-di-RNMAC:AndroidTriage rnmac$
```

Once the device is properly connected, you can execute the script.



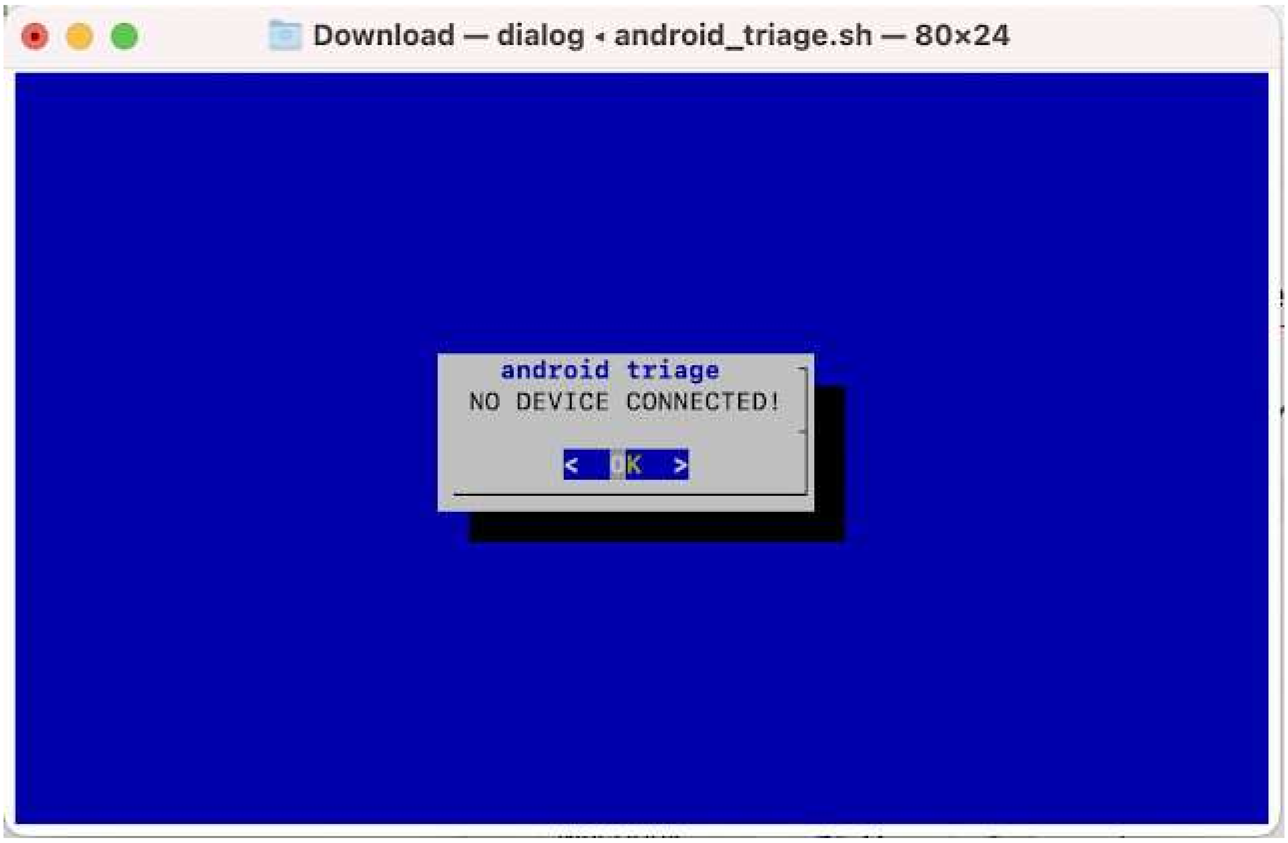
```
[MacBook-Air:Downloads mattiaepifani$ chmod +x android_triage.sh
[MacBook-Air:Downloads mattiaepifani$ ./android_triage.sh
```

If the ADB command is not found in the path or in the script folder, the script will prompt the error message shown in figure.



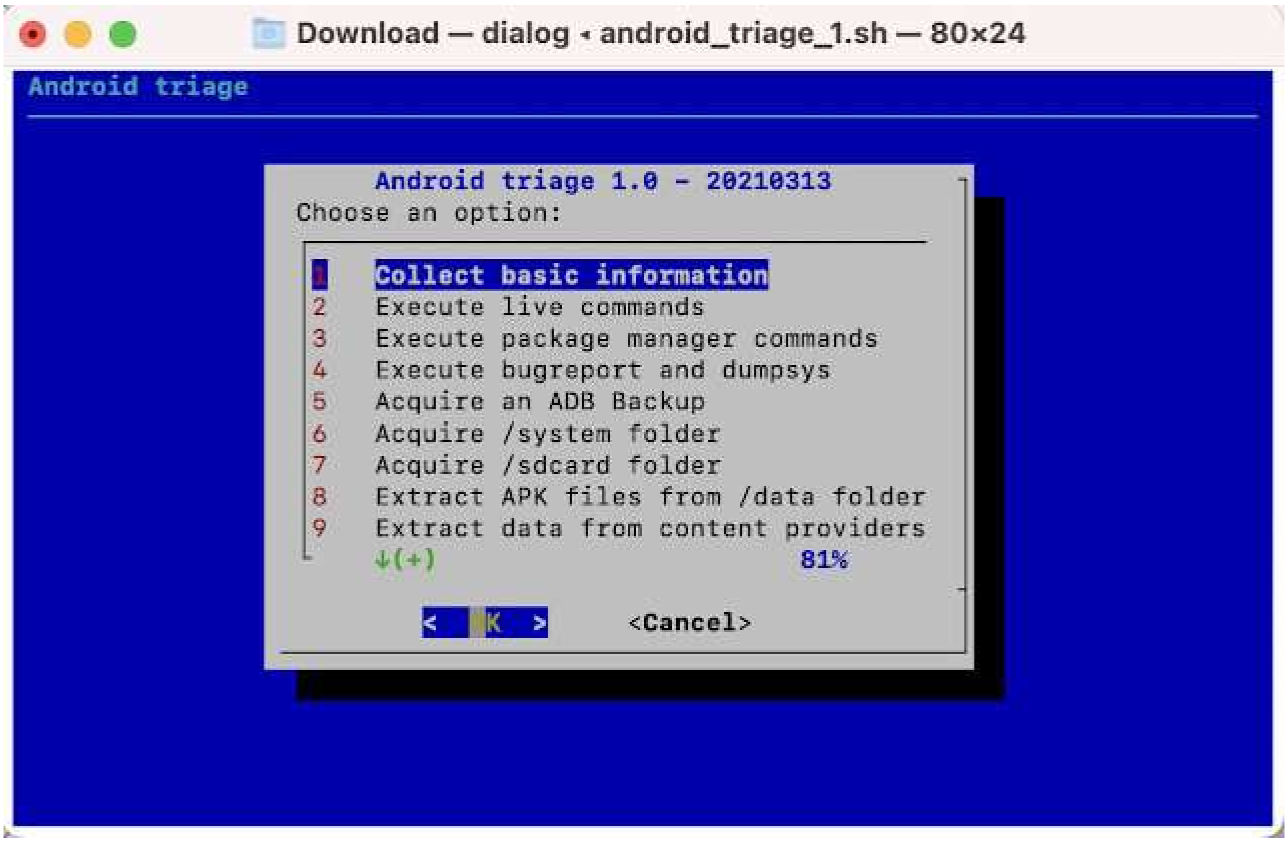
```
adb NOT FOUND! It's not possible to use
android_triage script
```

If the device is not properly connected or paired, the script will prompt the error message shown in figure.



```
Download — dialog ◂ android_triage.sh — 80×24

          android triage
     NO DEVICE CONNECTED!

            <  OK  >
```

Option 1 - Collect Basic Information



```
Download — dialog ◂ android_triage_1.sh — 80×24
Android triage

          Android triage 1.0 - 20210313
     Choose an option:

     1    Collect basic information
     2    Execute live commands
     3    Execute package manager commands
     4    Execute bugreport and dumpsys
     5    Acquire an ADB Backup
     6    Acquire /system folder
     7    Acquire /sdcard folder
     8    Extract APK files from /data folder
     9    Extract data from content providers
          ↓(+)                           81%

          <  OK  >      <Cancel>
```

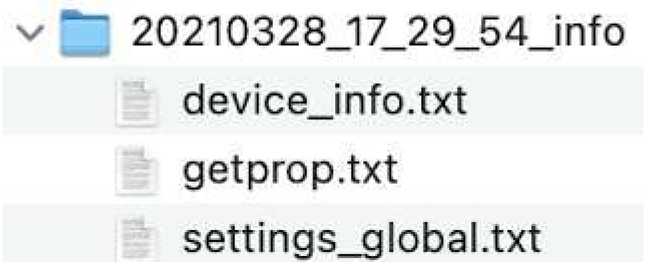
This option collects **basic information about the device** by executing the following commands:

- *adb shell getprop*
- *adb shell settings list system*
- *adb shell settings list secure*
- *adb shell settings list global*

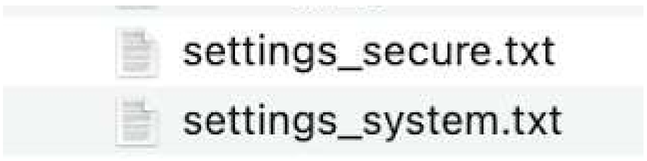
It also generates the "**device_info.txt**" file, containing relevant information about the device:

- Device Manufacturer (*adb shell getprop ro.product.manufacturer*)
- Device Product (*adb shell getprop ro.product.model*)
- Android ID (*adb shell settings get secure android_id*)
- Android Serial Number (*adb shell getprop ril.serialnumber*)
- Product Code (*adb shell getprop ro.product.code*)
- Product Device (*adb shell getprop ro.product.device*)
- Product Name (*adb shell getprop ro.product.name*)
- Chipname (*adb shell getprop ro.chipname*)
- Android Version (*adb shell getprop ro.build.version.release*)
- Android Fingerprint (*adb shell getprop ro.build.fingerprint*)
- Build date (*adb shell getprop ro.build.date*)
- Build ID (*adb shell ro.build.id*)
- Bootloader Version (*adb shell ro.boot.bootloader*)
- Security Patch (*adb shell ro.build.version.security_patch*)
- Bluetooth Address (*adb shell settings get secure bluetooth_address*)
- Bluetooth Name (*adb shell settings get secure bluetooth_name*)
- Timezone (*adb shell getprop persist.sys.timezone*)
- USB Configuration (*adb shell getprop persist.sys.usb.config*)
- Storage Size (*adb shell getprop storage.mmc.size*)
- Notification Sound (*adb shell getprop ro.config.notification_sound*)
- Alerm Alert (*adb shell getprop ro.config.alarm_alert*)
- Ringtone (*adb shell getprop ro.config.ringtone*)
- Media Sound (*adb shell getprop rro.config.media_sound*)
- Device Time (*adb shell date*)

The output is saved in the following path (relative to the script path):
"${ANDROID_ID}/${date +"%Y%m%d_%H_%M_%S"}_info"

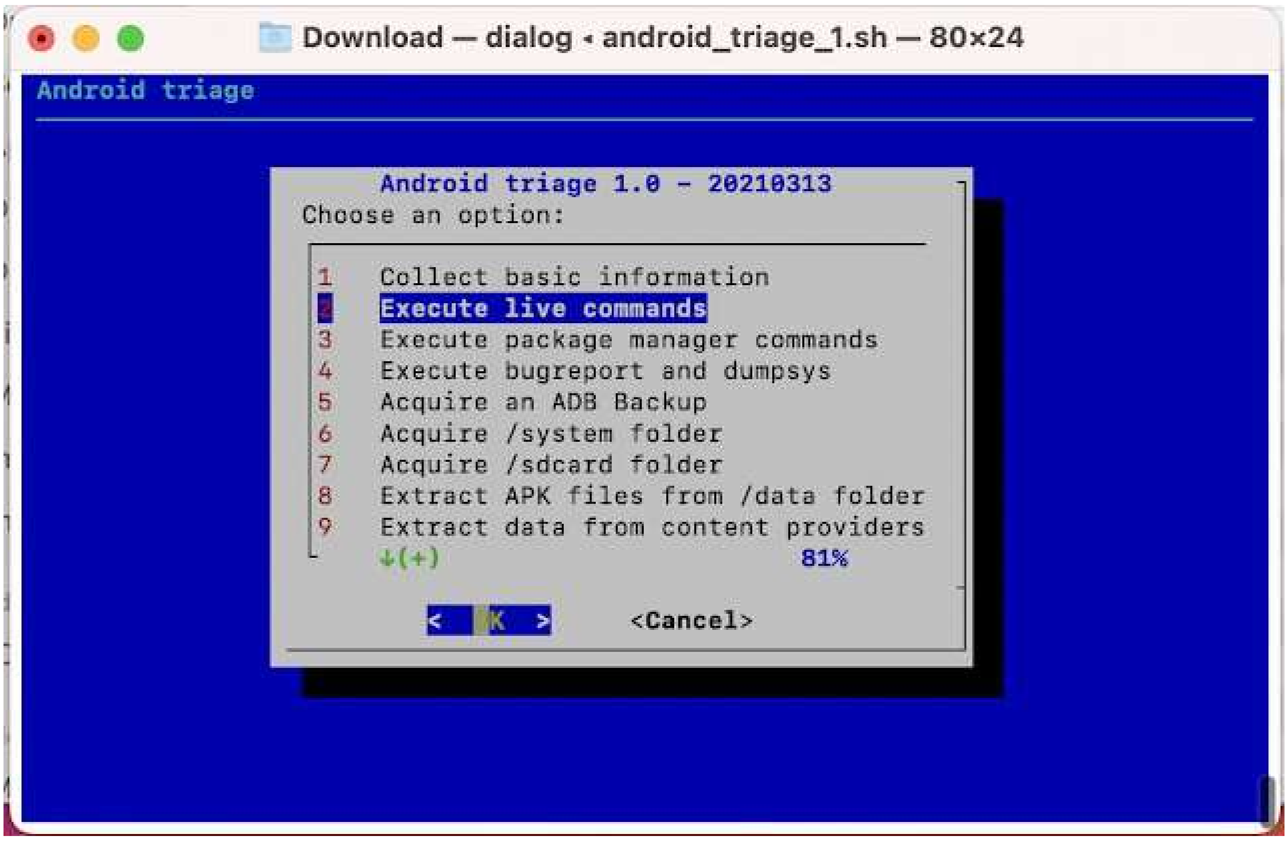


```
20210328_17_29_54_info
    device_info.txt
    getprop.txt
    settings_global.txt
```



settings_secure.txt
settings_system.txt

Option 2 - Execute Live Commands



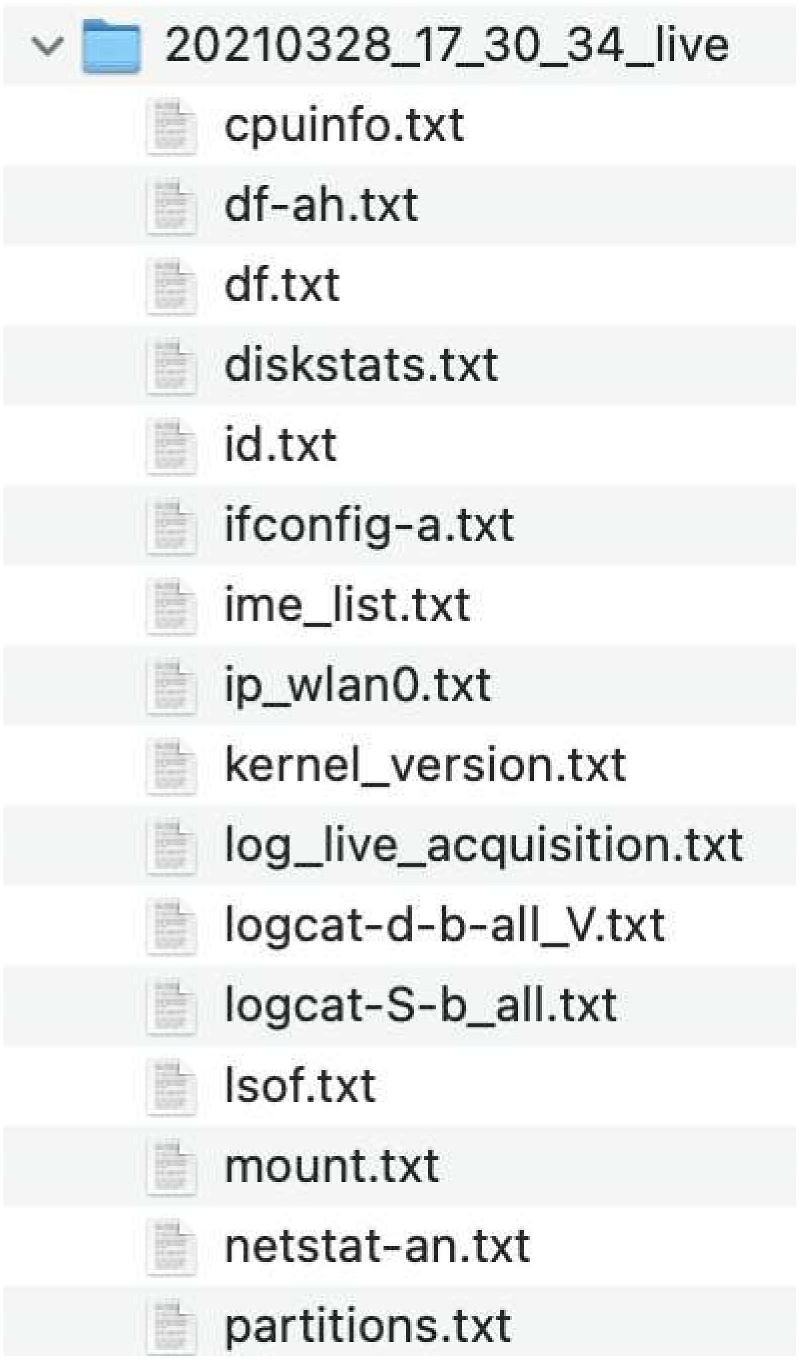
```
Download — dialog ◂ android_triage_1.sh — 80×24
Android triage

            Android triage 1.0 - 20210313
        Choose an option:

        1    Collect basic information
        2    Execute live commands
        3    Execute package manager commands
        4    Execute bugreport and dumpsys
        5    Acquire an ADB Backup
        6    Acquire /system folder
        7    Acquire /sdcard folder
        8    Extract APK files from /data folder
        9    Extract data from content providers
             ↓(+)                            81%

                  < OK >        <Cancel>
```

This option **executes various Linux commands**.

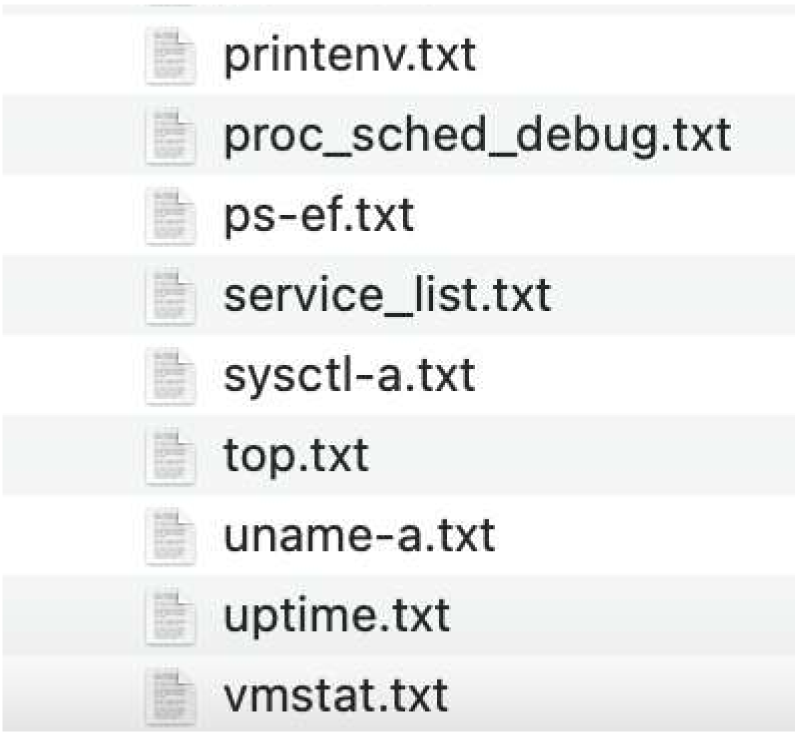
- *adb shell id*
- *adb shell uname -a*
- *adb shell cat /proc/version*
- *adb shell uptime*
- *adb shell printenv*
- *adb shell cat /proc/partitions*
- *adb shell cat /proc/cpuinfo*
- *adb shell cat /proc/diskstats*
- *adb shell df*
- *adb shell df -ah*
- *adb shell mount*
- *adb shell ip address show wlan0*
- *adb shell ifconfig -a*
- *adb shell netstat -an*

- *adb shell lsof*
- *adb shell ps -ef*
- *adb shell top -n 1*
- *adb shell cat /proc/sched_debug*
- *adb shell vmstat*
- *adb shell sysctl -a*
- *adb shell ime list*
- *adb shell service list*
- *adb shell logcat -S -b all*
- *adb shell logcat -d -b all V:**

The output is saved in the following path (relative to the script path): "${ANDROID_ID}/${date +"%Y%m%d_%H_%M_%S"}_live"

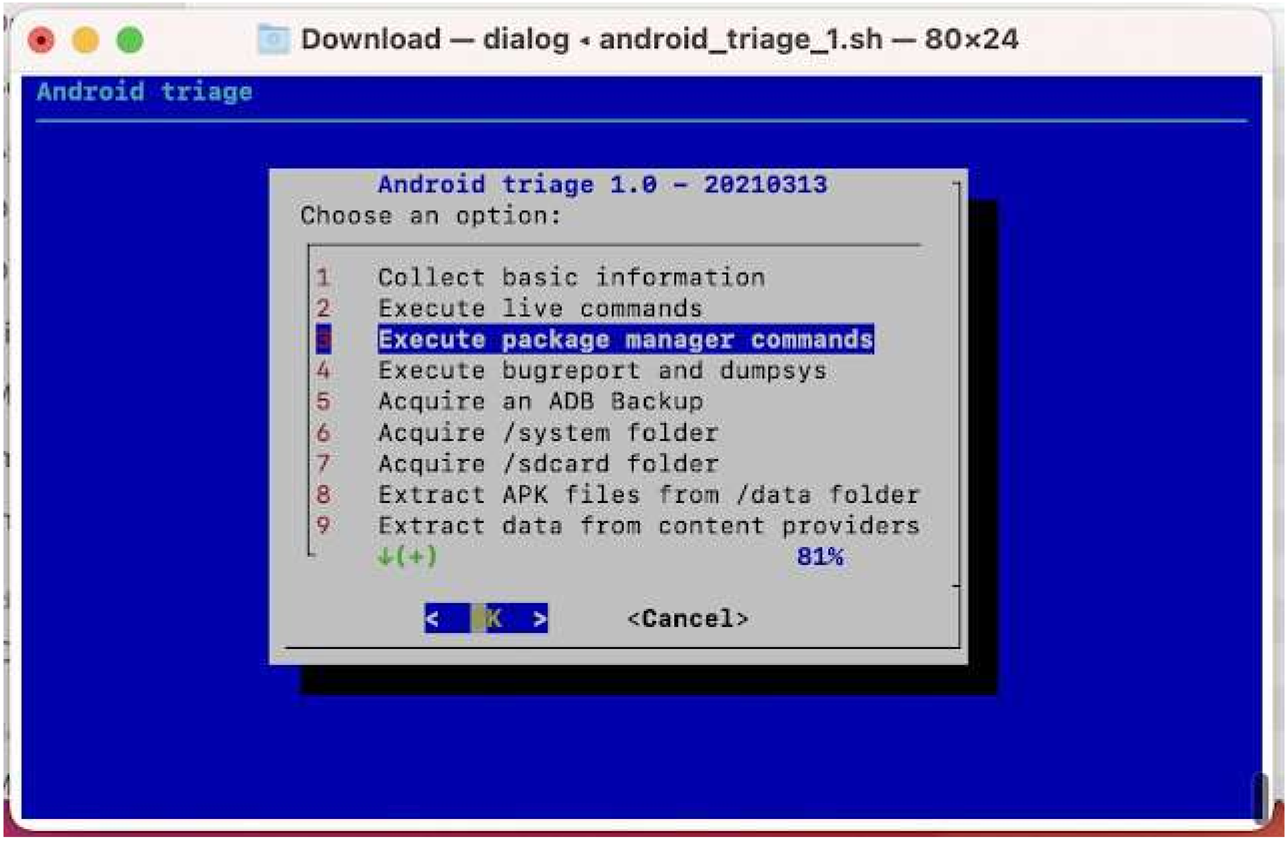


```
20210328_17_30_34_live
    cpuinfo.txt
    df-ah.txt
    df.txt
    diskstats.txt
    id.txt
    ifconfig-a.txt
    ime_list.txt
    ip_wlan0.txt
    kernel_version.txt
    log_live_acquisition.txt
    logcat-d-b-all_V.txt
    logcat-S-b_all.txt
    lsof.txt
    mount.txt
    netstat-an.txt
    partitions.txt
```



```
printenv.txt
proc_sched_debug.txt
ps-ef.txt
service_list.txt
sysctl-a.txt
top.txt
uname-a.txt
uptime.txt
vmstat.txt
```

Option 3 - Execute Package Manager commands



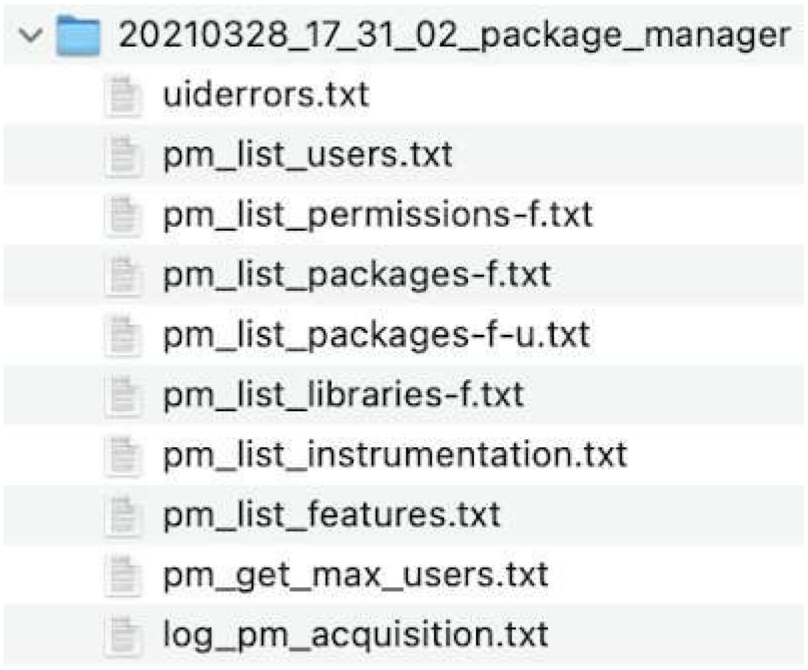
```
Download — dialog ◂ android_triage_1.sh — 80×24
Android triage

        Android triage 1.0 - 20210313
    Choose an option:

    1    Collect basic information
    2    Execute live commands
    3    Execute package manager commands
    4    Execute bugreport and dumpsys
    5    Acquire an ADB Backup
    6    Acquire /system folder
    7    Acquire /sdcard folder
    8    Extract APK files from /data folder
    9    Extract data from content providers
         ↓(+)                              81%

            <  OK  >        <Cancel>
```

This option executes various **Android Package Manager commands**.

- *adb shell pm get-max-users*
- *adb shell pm list users*
- *adb shell pm list features*

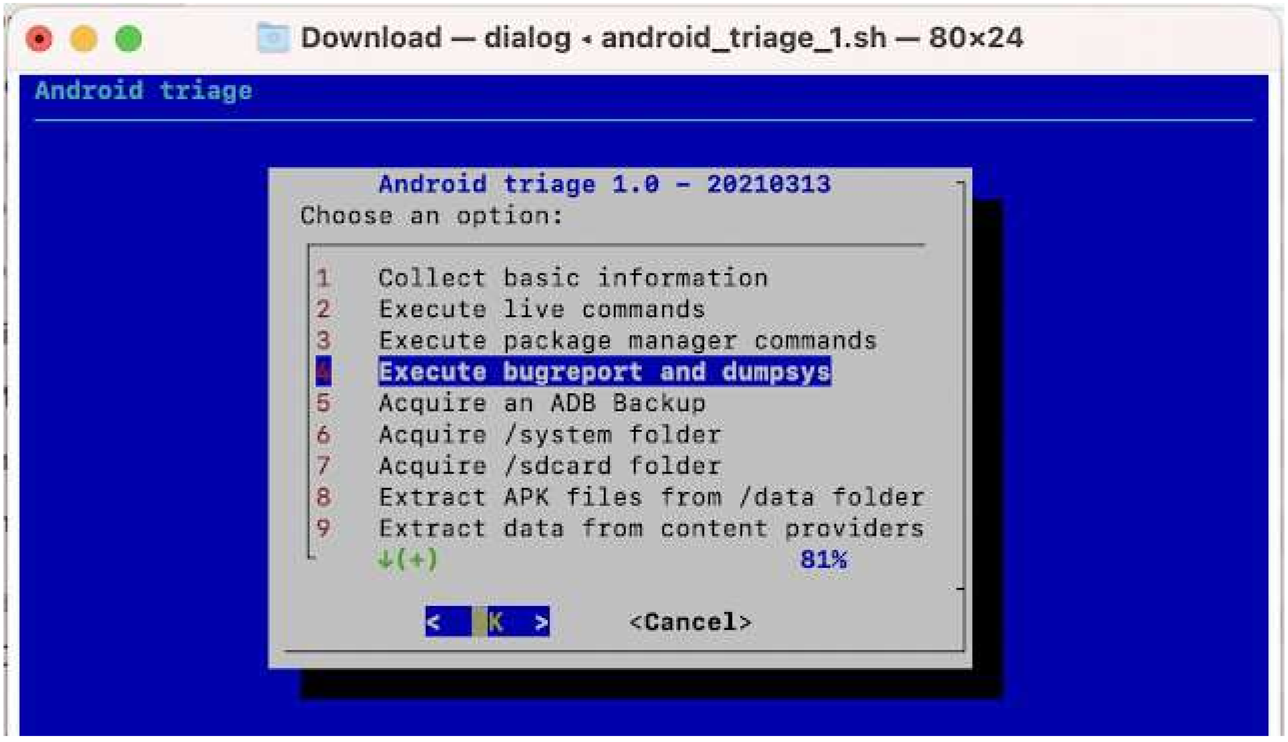
- *adb shell pm list instrumentation*
- *adb shell pm list libraries -f*
- *adb shell pm list packages -f*
- *adb shell pm list packages -f -u*
- *adb shell pm list permissions -f*
- *adb shell cat /data/system/uiderrors.txt*

The output is saved in the following path (relative to the script path): "${ANDROID_ID}/${date +"%Y%m%d_%H_%M_%S"}_package_manager"



```
20210328_17_31_02_package_manager
    uiderrors.txt
    pm_list_users.txt
    pm_list_permissions-f.txt
    pm_list_packages-f.txt
    pm_list_packages-f-u.txt
    pm_list_libraries-f.txt
    pm_list_instrumentation.txt
    pm_list_features.txt
    pm_get_max_users.txt
    log_pm_acquisition.txt
```

Option 4 - Execute Bugreport and Dumpsys



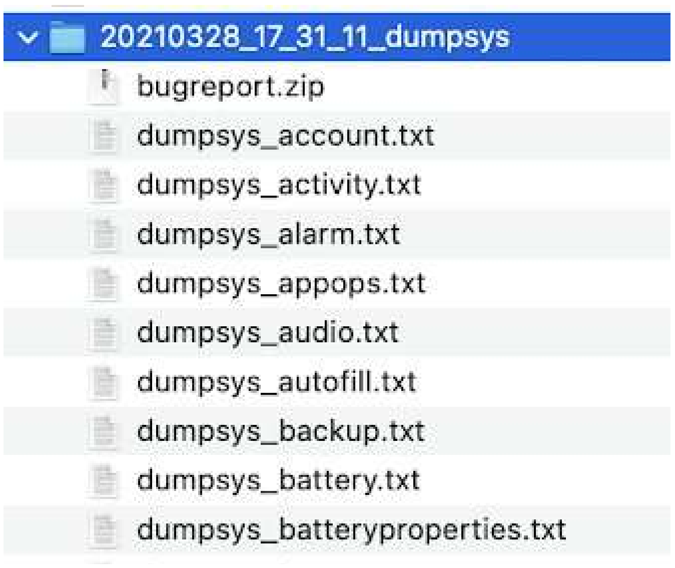
```
Download — dialog ◂ android_triage_1.sh — 80×24
Android triage

        Android triage 1.0 - 20210313
   Choose an option:

   1    Collect basic information
   2    Execute live commands
   3    Execute package manager commands
   4    Execute bugreport and dumpsys
   5    Acquire an ADB Backup
   6    Acquire /system folder
   7    Acquire /sdcard folder
   8    Extract APK files from /data folder
   9    Extract data from content providers
        ↓(+)                            81%

           <  OK  >          <Cancel>
```

This option executes **Android Bugreport and Android Dumpsys commands**.

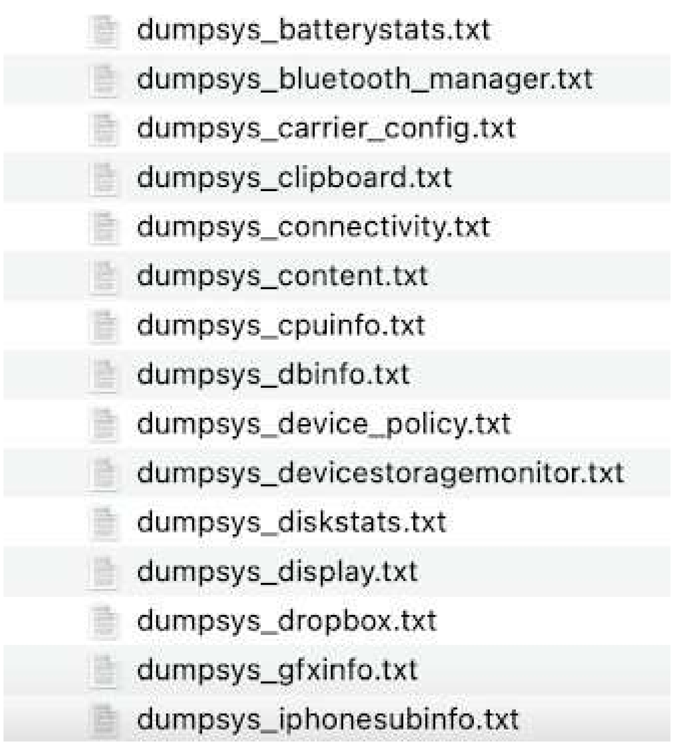
- *adb bugreport*
- *adb shell dumpsys*
- *adb shell dumpsys account*
- *adb shell dumpsys activity*
- *adb shell dumpsys alarm*
- *adb shell dumpsys appops*
- *adb shell dumpsys audio*
- *adb shell dumpsys autofill*
- *adb shell dumpsys backup*
- *adb shell dumpsys battery*
- *adb shell dumpsys batteryproperties*
- *adb shell dumpsys batterystats*
- *adb shell dumpsys bluetooth_manager*
- *adb shell dumpsys carrier_config*
- *adb shell dumpsys clipboard*
- *adb shell dumpsys connectivity*
- *adb shell dumpsys content*
- *adb shell dumpsys cpuinfo*
- *adb shell dumpsys dbinfo*
- *adb shell dumpsys device_policy*
- *adb shell dumpsys devicestoragemonitor*
- *adb shell dumpsys diskstats*
- *adb shell dumpsys display*
- *adb shell dumpsys dropbox*
- *adb shell dumpsys gfxinfo*
- *adb shell dumpsys iphonesubinfo*
- *adb shell dumpsys jobscheduler*
- *adb shell dumpsys location*
- *adb shell dumpsys meminfo -a*
- *adb shell dumpsys mount*
- *adb shell dumpsys netpolicy*
- *adb shell dumpsys netstats*
- *adb shell dumpsys network_management*
- *adb shell dumpsys network_score*
- *adb shell dumpsys notification*

- *adb shell dumpsys package*
- *adb shell dumpsys password_policy*
- *adb shell dumpsys permission*
- *adb shell dumpsys phone*
- *adb shell dumpsys power*
- *adb shell dumpsys procstats --full-details*
- *adb shell dumpsys restriction_policy*
- *adb shell dumpsys sdhms*
- *adb shell dumpsys sec_location*
- *adb shell dumpsys secims*
- *adb shell dumpsys search*
- *adb shell dumpsys sensorservice*
- *adb shell dumpsys settings*
- *adb shell dumpsys shortcut*
- *adb shell dumpsys stats*
- *adb shell dumpsys statusbar*
- *adb shell dumpsys storaged*
- *adb shell dumpsys telecom*
- *adb shell dumpsys usagestats*
- *adb shell dumpsys user*
- *adb shell dumpsys usb*
- *adb shell dumpsys vibrator*
- *adb shell dumpsys wallpaper*
- *adb shell dumpsys wifi*
- *adb shell dumpsys window*

The output is saved in the following path (relative to the script path): "${ANDROID_ID}/${date +"%Y%m%d_%H_%M_%S"}_dumpsys".

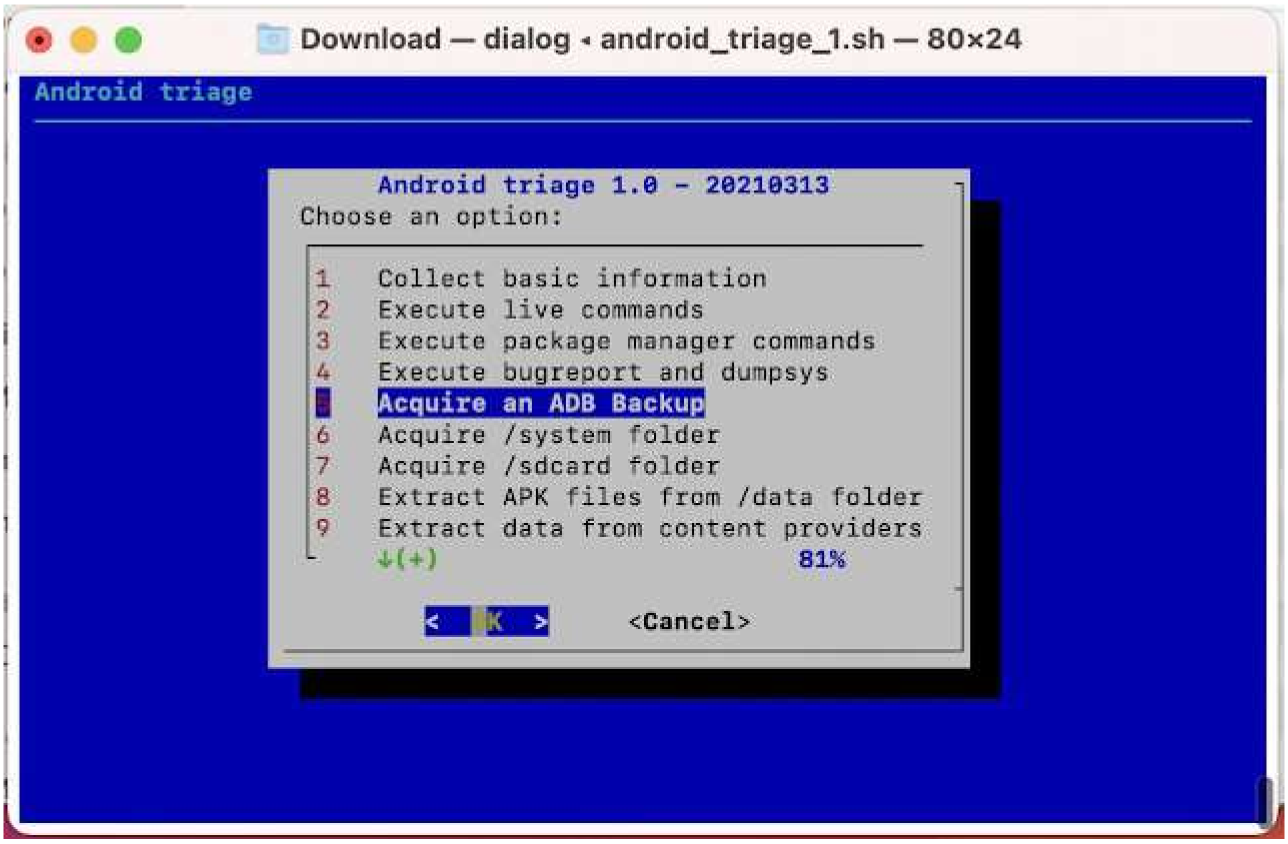


```
20210328_17_31_11_dumpsys
    bugreport.zip
    dumpsys_account.txt
    dumpsys_activity.txt
    dumpsys_alarm.txt
    dumpsys_appops.txt
    dumpsys_audio.txt
    dumpsys_autofill.txt
    dumpsys_backup.txt
    dumpsys_battery.txt
    dumpsys_batteryproperties.txt
```



```
dumpsys_batterystats.txt
dumpsys_bluetooth_manager.txt
dumpsys_carrier_config.txt
dumpsys_clipboard.txt
dumpsys_connectivity.txt
dumpsys_content.txt
dumpsys_cpuinfo.txt
dumpsys_dbinfo.txt
dumpsys_device_policy.txt
dumpsys_devicestoragemonitor.txt
dumpsys_diskstats.txt
dumpsys_display.txt
dumpsys_dropbox.txt
dumpsys_gfxinfo.txt
dumpsys_iphonesubinfo.txt
```

Option 5 - Acquire an ADB Backup

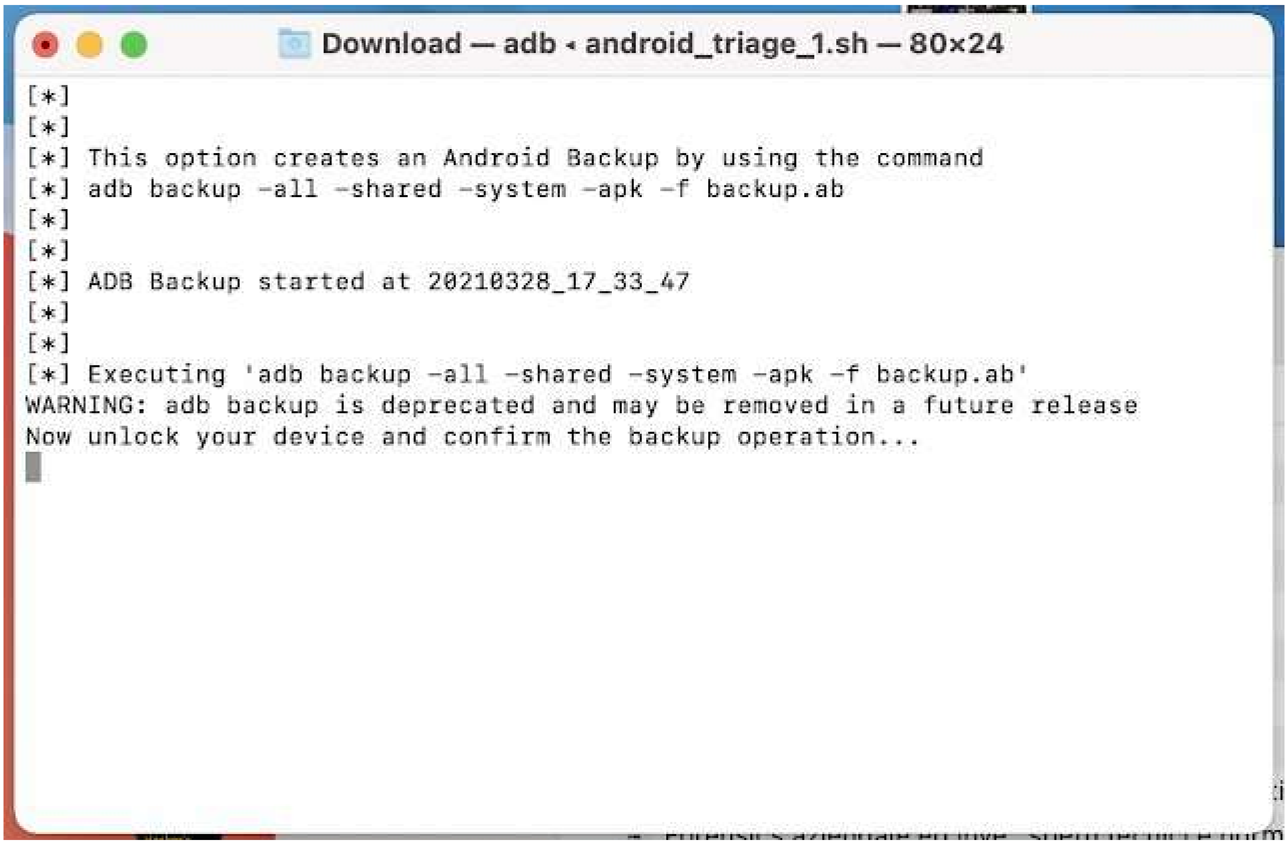


```
Download — dialog ◂ android_triage_1.sh — 80×24
Android triage

        Android triage 1.0 - 20210313
    Choose an option:

     1    Collect basic information
     2    Execute live commands
     3    Execute package manager commands
     4    Execute bugreport and dumpsys
     5    Acquire an ADB Backup
     6    Acquire /system folder
     7    Acquire /sdcard folder
     8    Extract APK files from /data folder
     9    Extract data from content providers
          ↓(+)                            81%

            <  OK  >        <Cancel>
```

This option **creates an ADB Backup** by executing the following command

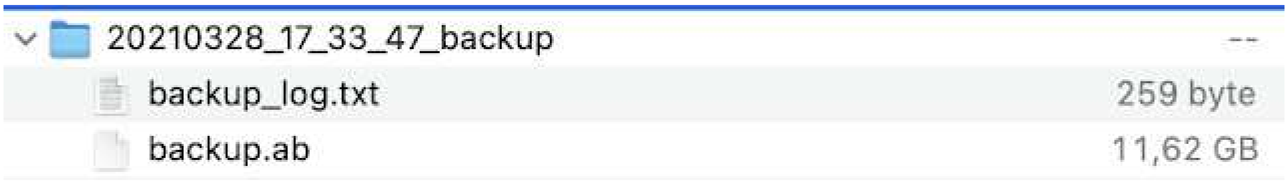
adb backup -all -shared -system -apk -f "$BACKUP_DIR"/backup.ab

The command <u>requires interaction with the device</u>



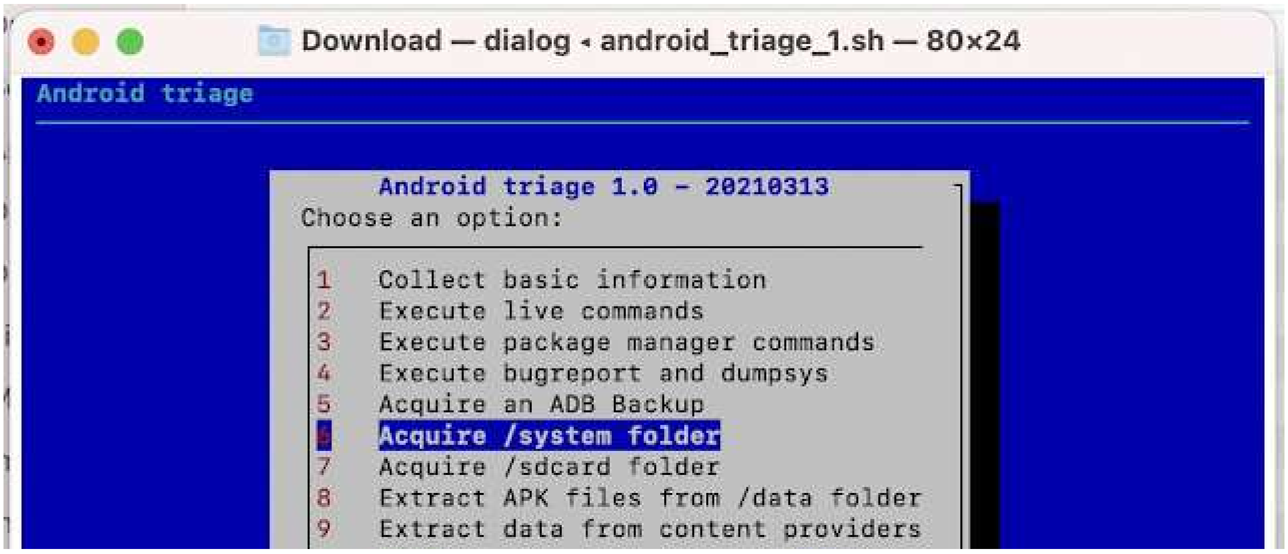
```
Download — adb ◂ android_triage_1.sh — 80×24
[*]
[*]
[*] This option creates an Android Backup by using the command
[*] adb backup -all -shared -system -apk -f backup.ab
[*]
[*]
[*] ADB Backup started at 20210328_17_33_47
[*]
[*]
[*] Executing 'adb backup -all -shared -system -apk -f backup.ab'
WARNING: adb backup is deprecated and may be removed in a future release
Now unlock your device and confirm the backup operation...
```

The output is saved in the following path (relative to the script path): **"${ANDROID_ID}/${date +"%Y%m%d_%H_%M_%S"}_backup"**

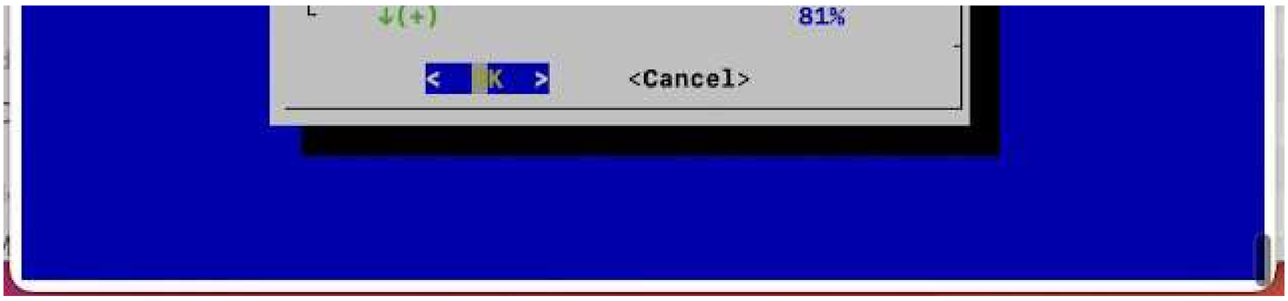


```
20210328_17_33_47_backup      --
    backup_log.txt            259 byte
    backup.ab                 11,62 GB
```

Option 6 - Acquire "/system" folder

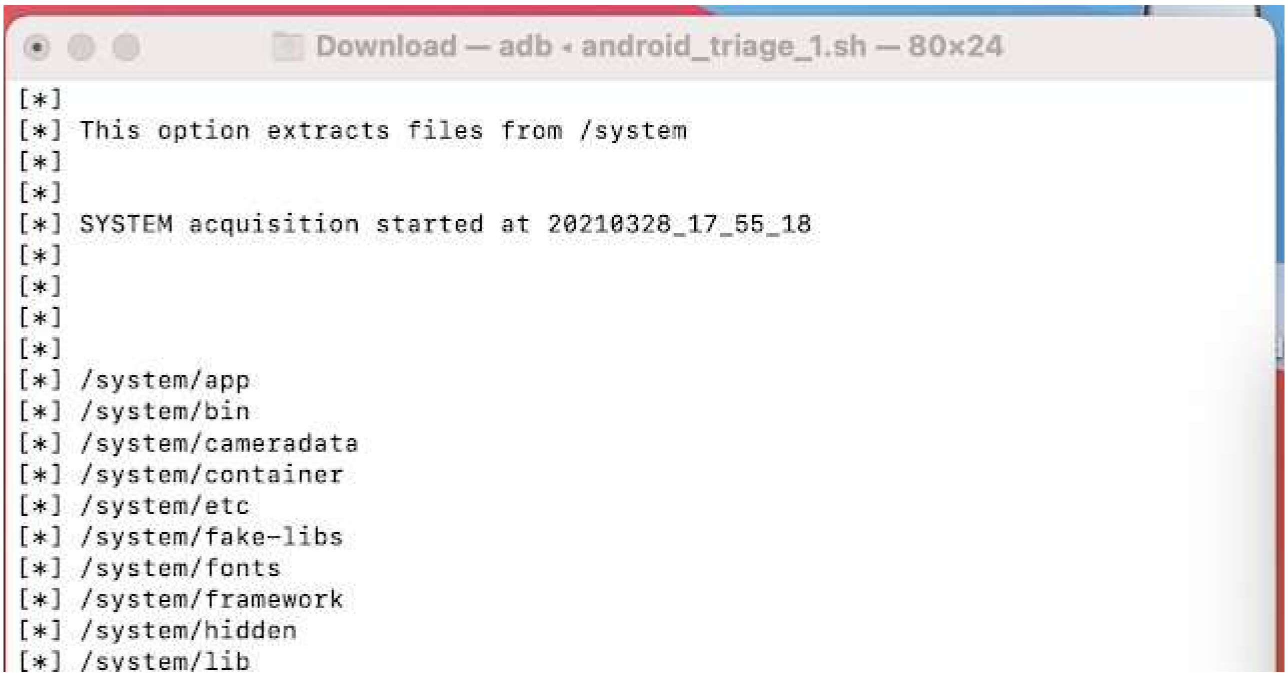


```
Download — dialog ◂ android_triage_1.sh — 80×24
Android triage

        Android triage 1.0 - 20210313
   Choose an option:

   1   Collect basic information
   2   Execute live commands
   3   Execute package manager commands
   4   Execute bugreport and dumpsys
   5   Acquire an ADB Backup
   6   Acquire /system folder
   7   Acquire /sdcard folder
   8   Extract APK files from /data folder
   9   Extract data from content providers
```



This option **pulls the "/system" folder and saves it locally**, also as a tar file.

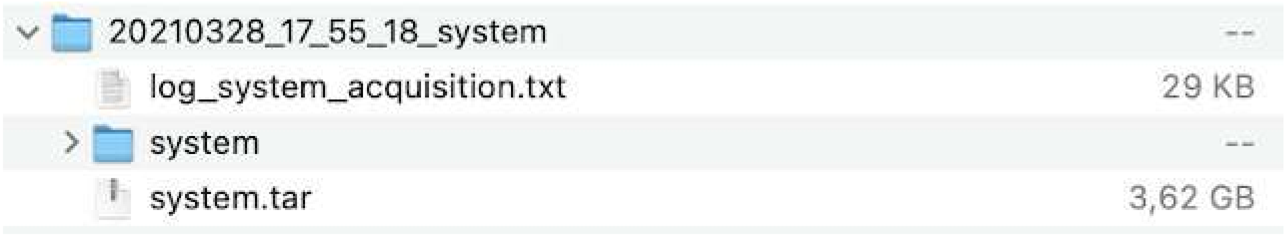
- *adb pull /system/*
- *adb pull /system/app*
- *adb pull /system/camerdata*
- *adb pull /system/container*
- *adb pull /system/etc*
- *adb pull /system/fake-libs*
- *adb pull /system/fonts*
- *adb pull /system/framework*
- *adb pull /system/hidden*
- *adb pull /system/lib*
- *adb pull /system/lib64*
- *adb pull /system/media*
- *adb pull /system/priv-app*
- *adb pull /system/saiv*
- *adb pull /system/tts*
- *adb pull /system/usr*
- *adb pull /system/vendor*
- *adb pull /system/xbin*



```
[*]
[*] This option extracts files from /system
[*]
[*]
[*] SYSTEM acquisition started at 20210328_17_55_18
[*]
[*]
[*]
[*]
[*] /system/app
[*] /system/bin
[*] /system/cameradata
[*] /system/container
[*] /system/etc
[*] /system/fake-libs
[*] /system/fonts
[*] /system/framework
[*] /system/hidden
[*] /system/lib
```

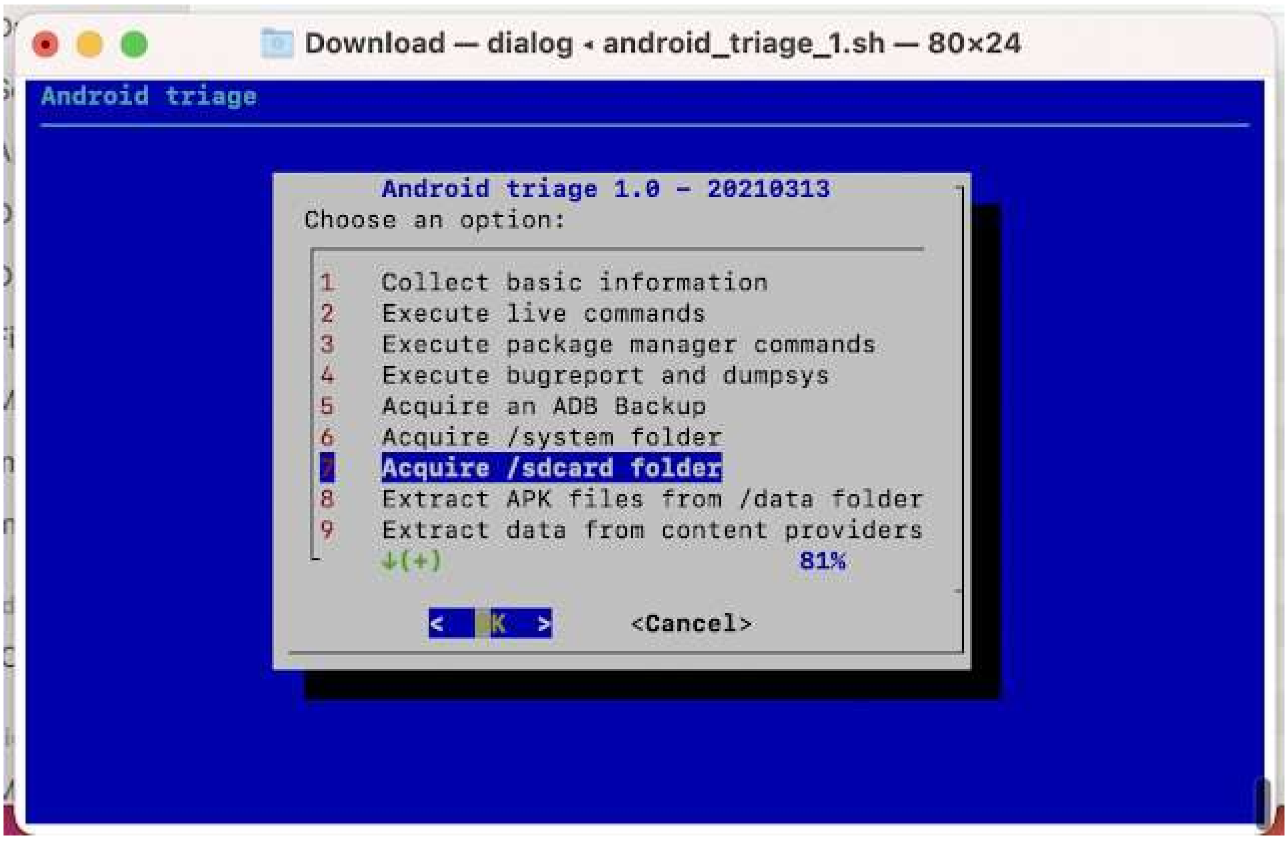
```
[*] /system/lib64
[*] /system/media
[*] /system/priv-app
[*] /system/saiv
```

The output is saved in the following path (relative to the script path): "**${ANDROID_ID}/${date +"%Y%m%d_%H_%M_%S"}_system**"



```
20210328_17_55_18_system           --
    log_system_acquisition.txt     29 KB
  > system                         --
    system.tar                     3,62 GB
```

Option 7 - Acquire "/sdcard" folder



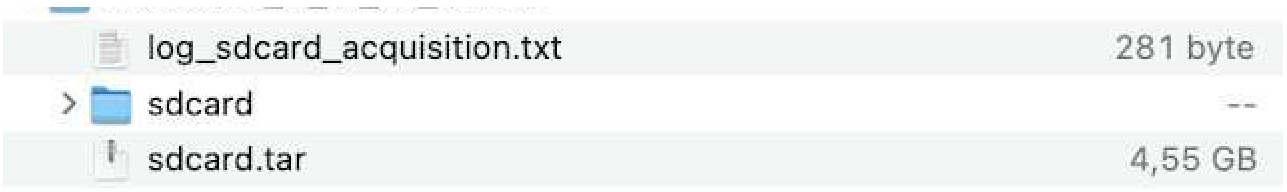
```
Download — dialog ◂ android_triage_1.sh — 80×24
Android triage

        Android triage 1.0 - 20210313
    Choose an option:

     1    Collect basic information
     2    Execute live commands
     3    Execute package manager commands
     4    Execute bugreport and dumpsys
     5    Acquire an ADB Backup
     6    Acquire /system folder
     7    Acquire /sdcard folder
     8    Extract APK files from /data folder
     9    Extract data from content providers
          ↓(+)                            81%

              <  OK  >       <Cancel>
```

This option **pulls the "/sdcard" folder and saves it locally**, also as a tar file.

- *adb pull /scard*

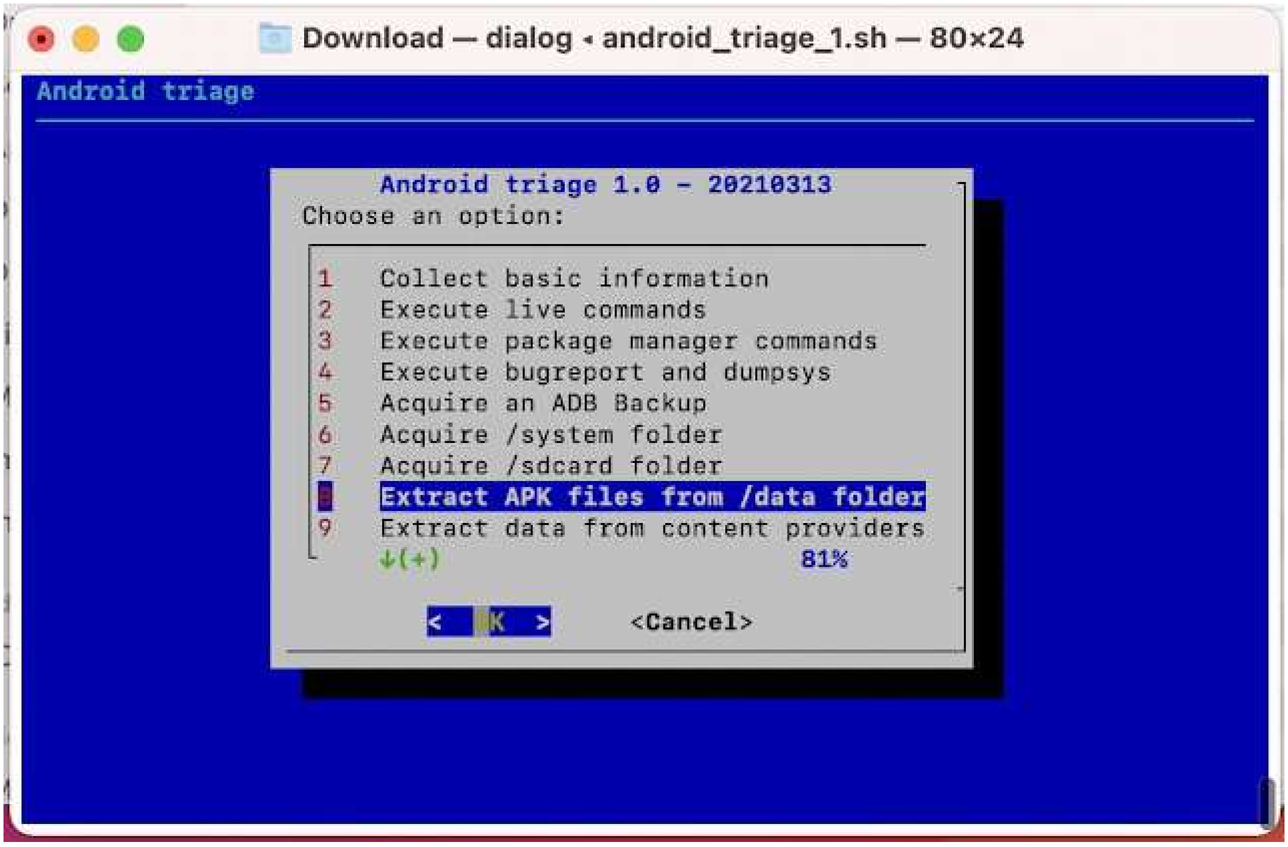
The output is saved in the following path (relative to the script path): "**${ANDROID_ID}/${date +"%Y%m%d_%H_%M_%S"}_sdcard**"

```
20210328_17_57_56_sdcard           --
```



```
log_sdcard_acquisition.txt    281 byte
sdcard                        --
sdcard.tar                    4,55 GB
```

Option 8 - Extract APK files from "/data" folder



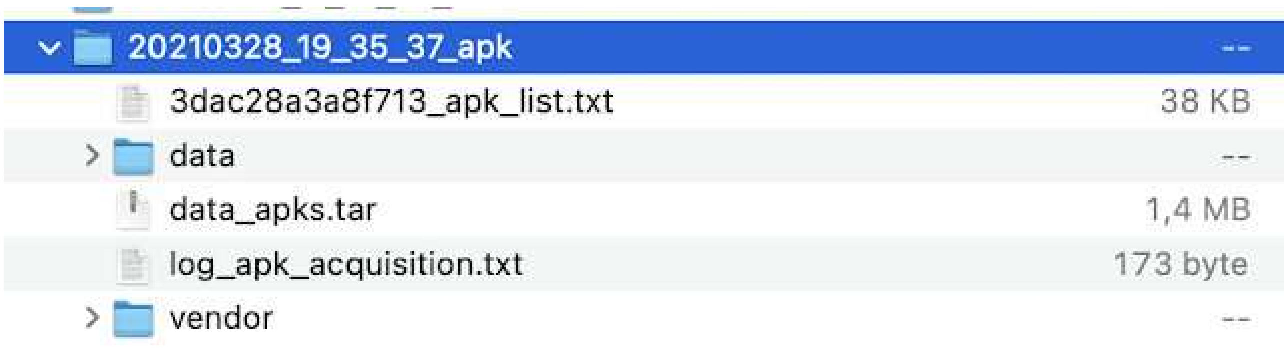
```
Android triage

Android triage 1.0 - 20210313
Choose an option:

1    Collect basic information
2    Execute live commands
3    Execute package manager commands
4    Execute bugreport and dumpsys
5    Acquire an ADB Backup
6    Acquire /system folder
7    Acquire /sdcard folder
8    Extract APK files from /data folder
9    Extract data from content providers
     ↓(+)                             81%

     <  OK  >       <Cancel>
```

This option **pulls APK files from "/data" and "/vendor" folder and saves them locally**, also as a tar file.

```
    $SHELL_COMMAND pm list packages -f -u >
${APK_DIR}/${ANDROID_ID}_apk_list.txt

    SELECTED_FILE=${APK_DIR}/${ANDROID_ID}_apk_list.txt

    echo "[*] Pulling APK files"
    while read -r line
    do
        line=${line#"package:"}
        target_file=${line%%".apk="*}
        target_file=$target_file".apk"
        IFS='/' read -ra tokens <<<"$target_file"
```

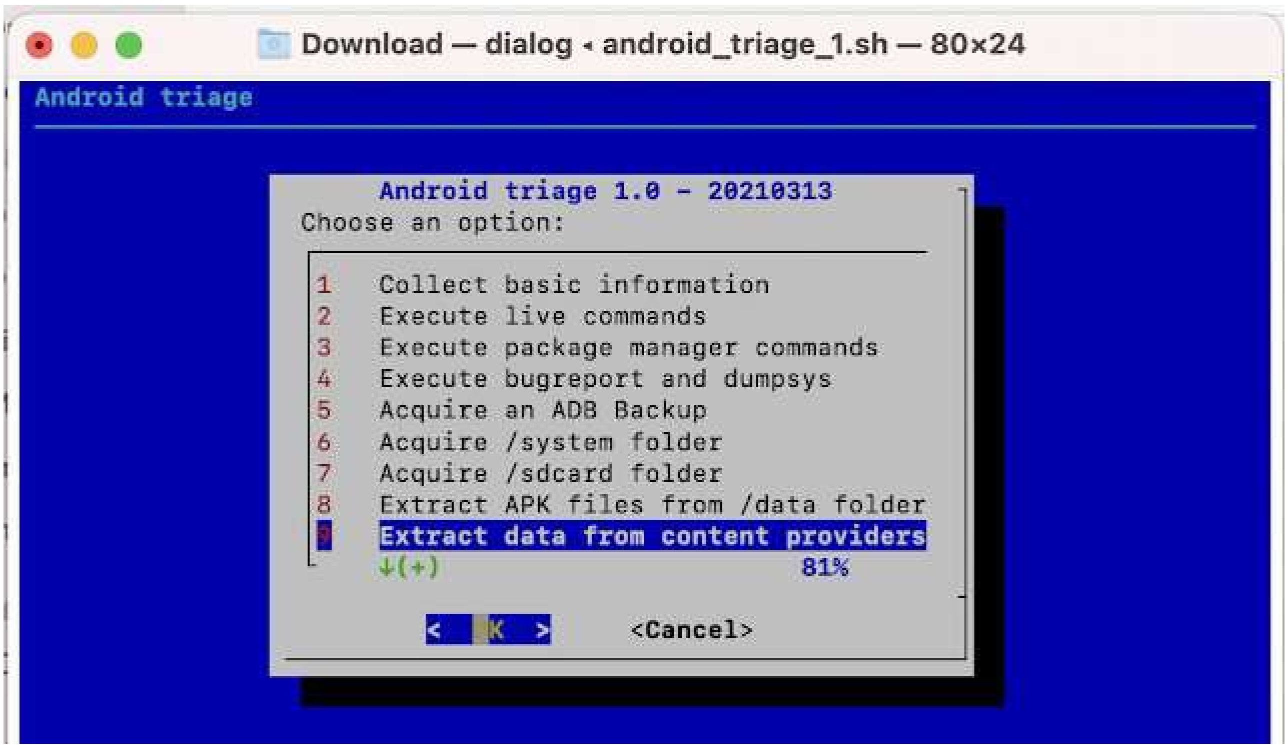
```
            apk_type=${tokens[1]}
            app_folder=${tokens[2]}
            app_path=${tokens[3]}
            apk_name=${tokens[4]}

            if [[ ${apk_type} != "system" ]]; then
               mkdir -p ${APK_DIR}/${apk_type}/${app_folder}/${app_path}
               $PULL_COMMAND $target_file
${APK_DIR}/${apk_type}/${app_folder}/${app_path}/${apk_name}
            fi
        continue
        done < "$SELECTED_FILE"
```

The output is saved in the following path (relative to the script path): "**${ANDROID_ID}/${date +"%Y%m%d_%H_%M_%S"}_apk**"

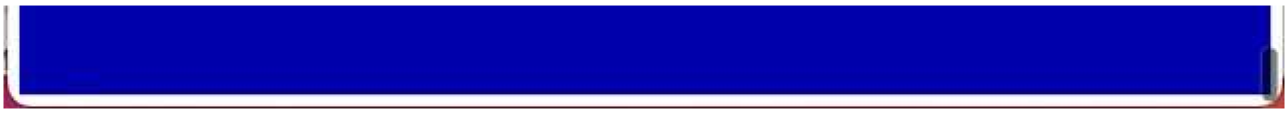


```
20210328_19_35_37_apk                 --
    3dac28a3a8f713_apk_list.txt     38 KB
  > data                              --
    data_apks.tar                  1,4 MB
    log_apk_acquisition.txt      173 byte
  > vendor                            --
```

Option 9 - Extract data from Content Providers



```
Download — dialog ◂ android_triage_1.sh — 80×24

Android triage

          Android triage 1.0 - 20210313
        Choose an option:

        1   Collect basic information
        2   Execute live commands
        3   Execute package manager commands
        4   Execute bugreport and dumpsys
        5   Acquire an ADB Backup
        6   Acquire /system folder
        7   Acquire /sdcard folder
        8   Extract APK files from /data folder
        9   Extract data from content providers
            ↓(+)                            81%

              < OK >          <Cancel>
```



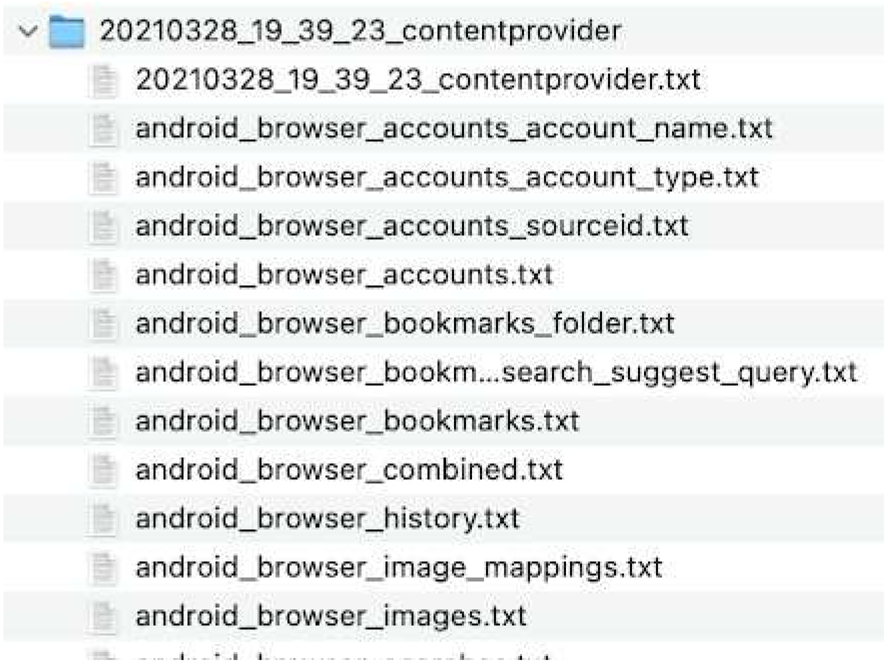
This option **extracts data from Android Content Providers**.

- *adb shell content query --uri content://com.android.calendar/calendar_entities*
- *adb shell content query --uri content://com.android.calendar/calendars*
- *adb shell content query --uri content://com.android.calendar/attendees*
- *adb shell content query --uri content://com.android.calendar/event_entities*
- *adb shell content query --uri content://com.android.calendar/events*
- *adb shell content query --uri content://com.android.calendar/properties*
- *adb shell content query --uri content://com.android.calendar/reminders*
- *adb shell content query --uri content://com.android.calendar/calendar_alerts*
- *adb shell content query --uri content://com.android.calendar/colors*
- *adb shell content query --uri content://com.android.calendar/extendedproperties*
- *adb shell content query --uri content://com.android.calendar/syncstate*
- *adb shell content query --uri content://com.android.contacts/raw_contacts*
- *adb shell content query --uri content://com.android.contacts/directories*
- *adb shell content query --uri content://com.android.contacts/syncstate*
- *adb shell content query --uri content://com.android.contacts/profile/syncstate*
- *adb shell content query --uri content://com.android.contacts/contacts*
- *adb shell content query --uri content://com.android.contacts/profile/raw_contacts*
- *adb shell content query --uri content://com.android.contacts/profile*
- *adb shell content query --uri content://com.android.contacts/profile/as_vcard*
- *adb shell content query --uri content://com.android.contacts/stream_items*
- *adb shell content query --uri content://com.android.contacts/stream_items/photo*
- *adb shell content query --uri content://com.android.contacts/stream_items_limit*
- *adb shell content query --uri content://com.android.contacts/data*
- *adb shell content query --uri content://com.android.contacts/raw_contact_entities*
- *adb shell content query --uri content://com.android.contacts/profile/raw_contact_entities*
- *adb shell content query --uri content://com.android.contacts/status_updates*
- *adb shell content query --uri content://com.android.contacts/data/phones*
- *adb shell content query --uri content://com.android.contacts/data/phones/filter*
- *adb shell content query --uri content://com.android.contacts/data/emails/lookup*
- *adb shell content query --uri content://com.android.contacts/data/emails/filter*
- *adb shell content query --uri content://com.android.contacts/data/emails*
- *adb shell content query --uri content://com.android.contacts/data/postals*
- *adb shell content query --uri content://com.android.contacts/groups*

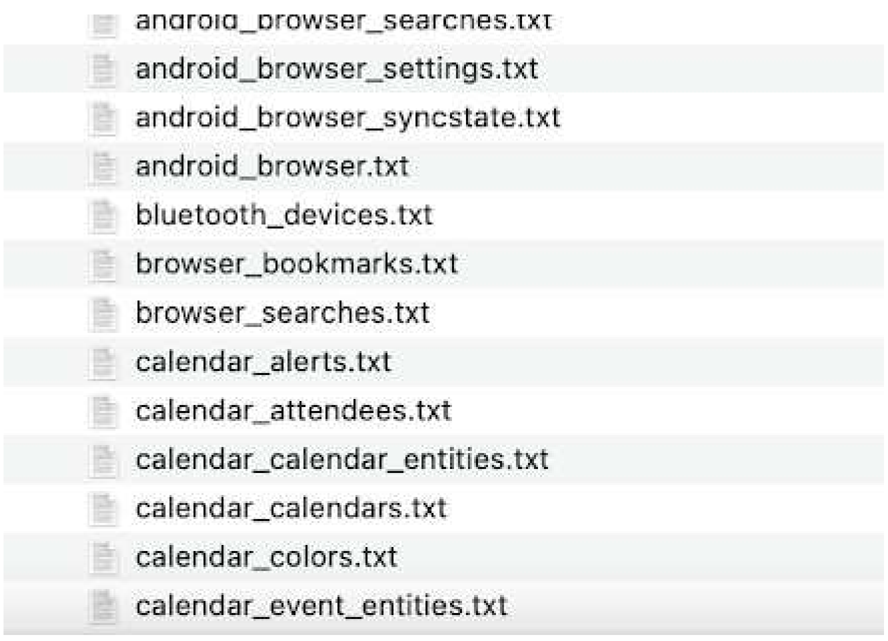
- *adb shell content query --uri content://com.android.contacts/groups_summary*
- *adb shell content query --uri content://com.android.contacts/aggregation_exceptions*
- *adb shell content query --uri content://com.android.contacts/settings*
- *adb shell content query --uri content://com.android.contacts/provider_status*
- *adb shell content query --uri content://com.android.contacts/photo_dimensions*
- *adb shell content query --uri content://com.android.contacts/deleted_contacts*
- *adb shell content query --uri content://downloads/my_downloads*
- *adb shell content query --uri content://downloads/download*
- *adb shell content query --uri content://media/external/file*
- *adb shell content query --uri content://media/external/images/media*
- *adb shell content query --uri content://media/external/images/thumbnails*
- *adb shell content query --uri content://media/external/audio/media*
- *adb shell content query --uri content://media/external/audio/genres*
- *adb shell content query --uri content://media/external/audio/playlists*
- *adb shell content query --uri content://media/external/audio/artists*
- *adb shell content query --uri content://media/external/audio/albums*
- *adb shell content query --uri content://media/external/video/media*
- *adb shell content query --uri content://media/external/video/thumbnails*
- *adb shell content query --uri content://media/internal/file*
- *adb shell content query --uri content://media/internal/images/media*
- *adb shell content query --uri content://media/internal/images/thumbnails*
- *adb shell content query --uri content://media/internal/audio/media*
- *adb shell content query --uri content://media/internal/audio/genres*
- *adb shell content query --uri content://media/internal/audio/playlists*
- *adb shell content query --uri content://media/internal/audio/artists*
- *adb shell content query --uri content://media/internal/audio/albums*
- *adb shell content query --uri content://media/internal/video/media*
- *adb shell content query --uri content://media/internal/video/thumbnails*
- *adb shell content query --uri content://settings/system*
- *adb shell content query --uri content://settings/system/ringtone*
- *adb shell content query --uri content://settings/system/alarm_alert*
- *adb shell content query --uri content://settings/system/notification_sound*
- *adb shell content query --uri content://settings/secure*
- *adb shell content query --uri content://settings/global*
- *adb shell content query --uri content://settings/bookmarks*
- *adb shell content query --uri content://com.google.settings/partner*
- *adb shell content query --uri content://nwkinfo/nwkinfo/carriers*
- *adb shell content query --uri content://com.android.settings.personalvibration.PersonalVibrationProvider/*

- *adb shell content query --uri content://settings/system/bluetooth_devices*
- *adb shell content query --uri content://settings/system/powersavings_appsettings*
- *adb shell content query --uri content://user_dictionary/words*
- *adb shell content query --uri content://browser/bookmarks*
- *adb shell content query --uri content://browser/searches*
- *adb shell content query --uri content://com.android.browser*
- *adb shell content query --uri content://com.android.browser/accounts*
- *adb shell content query --uri content://com.android.browser/accounts/account_name*
- *adb shell content query --uri content://com.android.browser/accounts/account_type*
- *adb shell content query --uri content://com.android.browser/accounts/sourceid*
- *adb shell content query --uri content://com.android.browser/settings*
- *adb shell content query --uri content://com.android.browser/syncstate*
- *adb shell content query --uri content://com.android.browser/images*
- *adb shell content query --uri content://com.android.browser/image_mappings*
- *adb shell content query --uri content://com.android.browser/bookmarks*
- *adb shell content query --uri content://com.android.browser/bookmarks/folder*
- *adb shell content query --uri content://com.android.browser/history*
- *adb shell content query --uri content://com.android.browser/bookmarks/search_suggest_query*
- *adb shell content query --uri content://com.android.browser/searches*
- *adb shell content query --uri content://com.android.browser/combined*

The output is saved in the following path (relative to the script path): "**${ANDROID_ID}/${date +"%Y%m%d_%H_%M_%S"}_contentprovider**"



```
20210328_19_39_23_contentprovider
    20210328_19_39_23_contentprovider.txt
    android_browser_accounts_account_name.txt
    android_browser_accounts_account_type.txt
    android_browser_accounts_sourceid.txt
    android_browser_accounts.txt
    android_browser_bookmarks_folder.txt
    android_browser_bookm...search_suggest_query.txt
    android_browser_bookmarks.txt
    android_browser_combined.txt
    android_browser_history.txt
    android_browser_image_mappings.txt
    android_browser_images.txt
```



```
android_browser_searches.txt
android_browser_settings.txt
android_browser_syncstate.txt
android_browser.txt
bluetooth_devices.txt
browser_bookmarks.txt
browser_searches.txt
calendar_alerts.txt
calendar_attendees.txt
calendar_calendar_entities.txt
calendar_calendars.txt
calendar_colors.txt
calendar_event_entities.txt
```

adb android forensics triage

Enter Comment

Popular posts from this blog

Huawei backup decryptor

By dfirfpi - July 30, 2019

When doing Mobile Forensics the first and usually the hardest step is to get access to user's data . It depends on the case type, but the so called physical acquisition is the analyst object of desire. The reason is simple: iOS and Andro …

KEEP READING

Android Forensics References: a curated list

By Mattia Epifani - September 06, 2022

During the forensic analysis of a mobile device, we often have the need to understand the content of a specific file or folder. This is particularly true when a file or a folder is not parsed by our set of tools. Our approach is, typically, to …

KEEP READING