CHAPTER

# Android forensic techniques 6

## INFORMATION IN THIS CHAPTER

- Procedures for handling an Android device
- Imaging Android USB mass storage devices
- Logical techniques
- Physical techniques

## INTRODUCTION

Before we dive into the actual Android forensic techniques, there are a number of considerations that influence which technique forensic analysts should use. In this section, we will discuss the different types of investigations, the differences between logical and physical techniques, and how to limit or avoid modifications to the device.

### Types of Investigations

There are a variety of situations that might benefit from the results of an Android forensic investigation. While the application of forensics is a commonality in all the situations, each one may require different procedures, documentation, and overall focus.

The first situation that people think of in general is investigations that will likely be adjudicated in a criminal or civil court of law. In these situations, there are a number of important considerations:

- Chain of custody
- Detailed contemporaneous notes and final reporting
- Possible validation of results using different tools or investigators
- Fact or opinion based testimony

Another common scenario is internal investigations in corporations. These investigations may end up litigated in court, but often they are used to determine the root cause of an issue (whether that is a system, external attack, or internal employee) and may result in disciplinary action against an employee.

Internal corporate investigations can cover many areas but the most common include:

- Intellectual property or data theft
- Inappropriate use of company resources
- Attempted or successful attack against computer systems
- Employment-related investigations including discrimination, sexual harassment, etc.
- Security audit (random or targeted)

There is also a need for forensics in cases involving family matters. The most common cases involve:

- Divorce
- Child custody
- Estate disputes

One final area where forensic investigation can yield significant value is for the security and operation of a government. Governments are usually the largest employer in a country and the United States is a good example. According to the US Census Bureau, data from the 2009 Annual Survey of Public Employment and Payroll revealed that the Federal government across all functions had over 3 million employees, while state and local governments has 16.6 million full-time equivalent employees (Government Employment & Payroll, n.d.).

Beyond employment-related matters, countries are also the potential target of attacks and foreign government intelligence gathering. Forensics can play a key role in thwarting attacks against a country, investigating successful attacks, counter intelligence scenarios, and in providing valuable intelligence needed for the governing of the country.

## Difference Between Logical and Physical Techniques

Android forensic techniques are either logical or physical in nature. A logical technique extracts allocated data and is typically achieved by accessing the file system. Allocated data simply means that the data are not deleted and are accessible on the file system. One exception to this definition is that some files, such as an SQLite database, can be allocated and still contain deleted records in the database. While recovery of the deleted data requires special tools and techniques, it is possible to recover deleted data from a logical acquisition.

Physical techniques, on the other hand, target the physical storage medium directly and do not rely on the file system itself to access the data. There are advantages to this approach; the most significant is that physical techniques likely provide access to significant amounts of deleted data. As discussed in Chapter 4, file systems often only mark data as deleted or obsolete, and do not actually erase the storage medium unless needed. As physical forensic techniques provide direct access to the storage medium, it is possible to recover both the allocated and the unallocated (deleted or obsolete) data.

Of course, the analysis of an Android physical acquisition is generally far more difficult and time consuming. Also, the physical techniques are more difficult to execute and missteps could leave the device inaccessible.

In Android forensics, the most common logical technique does not provide direct access to the file system and operates at a more abstract and less-effective level than the traditional logical techniques, which can acquire all nondeleted data directly from the file system. This technique, which relies on the Content Providers built into the Android platform and software development kit (SDK), is effective in producing some important forensic data, but only a fraction of the data that are available on the system.

## Modification of the Target Device

One of the guiding principles of any forensic investigation is to avoid modification of the target device in any manner. In many cases, this is achievable. For example, let's assume you are handed a desktop computer that is not powered on. You are informed it was seized from a suspect and that you need to launch a forensic investigation. The device is fairly easy to investigate without material changes to the data after you take custody. A typical investigation would fully document the computer, remove the hard drive, and connect it to a physical write blocker and acquire a bit-by-bit forensically sound image of the hard drive. The investigation would then take place on copies of the forensic image and the original device would remain unchanged.

As the power and functionality of computers has increased, this ideal situation has become more and more difficult to achieve. First, let's assume you are called to the scene of an investigation and there is a desktop computer, but this time the computer is in operation. Any interaction with the computer, whether you simply move it or even physically unplug the device, will modify the device in some way. While many examiners advocate simply unplugging the computer, unplugging the computer still changes the computer as the contents of RAM, open network connections, and more (all of which can be quite valuable in an investigation) are permanently lost.

If you instead decide to examine the device while it is running, all interactions change the device. To further complicate an investigation, it is possible that the computer is leveraging encryption and, while the device is running, that data may be accessible. However, if the device is powered off and you don't have the encryption keys, then you may permanently lose the ability to recover that data.

Another complicating factor can be the existence of servers that have special hardware, complex setups, or that simply cannot be powered down without significant impact to other systems or people. Some examples of complex service setups include complicated RAID setup, setups that rely on network-based storage area networks (SAN), and unsupported hardware. In such cases, the examiner must interact directly with the device while it is running even though those actions change the device.

Of course, mobile devices, and Android devices in particular, are nearly impossible to forensically analyze without any impact to the device. Unlike desktops, notebooks, and servers, there are portions of storage on an Android device that cannot be easily removed. And if the device is powered on, a shutdown of the device or pulling the battery again changes the device.

When mobile phones were first showing up in investigations, there was very little data stored in them that could be extracted from the device. Many investigations used traditional approaches, such as a search warrant on the wireless carrier to obtain call detail records. It was also possible to remove the subscriber identity module (SIM) card on GSM devices and extract some data. As phones began to store more data, there developed a deep divide between examiners who advocated the older methods (which had little impact on the device and subsequently retrieved only nominal data) and those who advocated exploiting the device more fully. The techniques used to exploit the devices did modify the device, leading to the ensuing debate.

As of 2011, much of the debate has subsided because the amount of data mobile devices now hold necessitates the more intrusive techniques. The Association of Chief Police Officers in the United Kingdom produces guidelines that address this issue quite clearly. The guide, *Good Practice Guide for Computer-Based Electronic Evidence* (ACPO Good Practice Guide, n.d.), establishes four principles of computer-based electronic evidence:

1. No action taken by law enforcement agencies or their agents should change data held on a computer or storage media, which may subsequently be relied upon in court.
2. In circumstances where a person finds it necessary to access original data held on a computer or on storage media, that person must be competent to do so and be able to give evidence explaining the relevance and the implications of their actions.
3. An audit trail or other record of all processes applied to computer-based electronic evidence should be created and preserved. An independent third party should be able to examine those processes and achieve the same result.
4. The person in charge of the investigation (the case officer) has overall responsibility for ensuring that the law and these principles are adhered to.

As mobile devices clearly present a circumstance where it is necessary to access the original device directly, then it is permissible provided the examiner is sufficiently trained, provides valid reasons for their approach and keeps a clear audit trail so that their actions are repeatable by a third party. This is certainly good advice and helps provide a solid framework for the forensic investigation of mobile devices.

## PROCEDURES FOR HANDLING AN ANDROID DEVICE

One major challenge for forensic analysts is to devise a protocol for handling the device prior to the analyst taking direct custody. And this is certainly not a new issue for analysts as others involved in the investigation may also handle other digital devices such as computers or laptops. However, mobile devices are still relatively new and are often not handled properly by first responders. There is a tendency to immediately examine the device, which almost inevitably results in data modification and potential loss of access to the device.

## Securing the Device

Many agencies and first responders have established a protocol for securing evidence. The following sections are meant to complement the existing procedures, not replace them. Of course, these represent special procedures, and educating first responders who have many other responsibilities can be quite challenging.

### *Pass Code Procedures*

Pass code locked devices are becoming more common as a result of heightened security awareness in consumers and corporations. In the next section, we cover some specific techniques to circumvent pass codes. However, it is not always possible. The first consideration when obtaining information from a device is whether an opportunity exists to immediately disable or otherwise circumvent the pass code.

If you encounter an Android device and the screen is active, strong consideration should be given to checking and potentially changing its settings. For devices that have pass codes, there is a short period of time (from less than a minute up to about 15 min) where full access to the device is possible without re-entering the pass code. If a device is in this state, there are several steps to consider:

1. Increase the screen timeout to prevent or postpone the screen locking. The location for this setting is not consistent between Android versions and devices. For example, on a G1 running Android 1.5, the timeout can be set by pressing Menu (from the home screen), then Settings, Sound & display, Screen timeout, and then select "Never timeout." On an HTC Incredible running Android 2.2, press Menu (from the home screen), then Settings, Security, Lock phone after, and then finally "15 minutes." As long as the device has some nominal activity in the allotted timeout setting, it will remain accessible.
2. Enable USB debugging and "Stay awake" settings. The location for this setting has remained consistent in devices and can be accessed by pressing Menu (from the home screen), then Settings, Applications and Development. From there, you can check USB debugging and Stay awake. If you select the "Stay awake" setting and then connect it to a charge, the device will never go to sleep, which is effective in preventing the screen lock. By enabling USB debugging, the device can be accessed over USB enabling data extraction.

Of course, these steps are making changes to the device and should be thoroughly logged in the case notes describing the state of the device, the rationale for the attempted changes, and the outcome of each change. This will not only assist in future report writing but will likely be an important factor if your decision to change the device is challenged in court.

To make matters more difficult, it is also important to minimize touching the screen in case the screen lock becomes active. As we will discuss shortly, it is sometimes possible to determine the pattern lock of a device by enhancing photographs of the device's screen. The lesser the interaction a first responder has with the screen, the higher the success rate of this technique.

## Network Isolation

As many examiners likely know, it is important to isolate the device from the network as soon as possible. In the worst-case scenario, a remote wipe could be initiated on the device which, if successful, would prevent the recovery of any data. While most remote wipes are done over the data network, some can be triggered over SMS, and hence ensure the device is fully isolated to prevent remote wipes. In other circumstances, additional messages on the device could be received or even removed by triggers outside your control. As the goal of a forensic image is to preserve the state of the device for additional analysis, any changes should be avoided.

There are a number of ways to isolate a device from the network and each of these methods have advantages and disadvantages. Table 6.1 summarizes the advantages and disadvantages of each technique.

As you can tell, isolating an Android device from the network is not an easy task and each option has advantages and disadvantages. While each examiner or their organization should determine the appropriate steps to undertake, the best option is probably placing the device in Airplane mode. This varies slightly between Android devices and versions but the general approach is the same:

1. Press and hold the Power off button and select Airplane mode.
2. Press Menu (from the home screen), then Settings, then the Wireless option which is generally near the top. Some examples are "Wireless controls" or "Wireless and networks." The next menu should present the Airplane mode option.

Fig. 6.1 is a screenshot from the Power off button approach. Fig. 6.2 shows the option via the Wireless settings.

Regardless of which technique you ultimately choose, the main goal should be to isolate the device from the network as soon as possible.

### *Power and Data Cables*

While most forensic labs will have the cables necessary to charge and connect the device, it is always prudent to seize the cables directly from the scene. It's possible that a newer device is in use and the forensic toolkits do not yet have an appropriate cable. For example, a new specification for connecting media devices was developed called portable digital media interface (PDMI) and is integrated into two Android tablet devices, the Dell Streak and the Samsung Galaxy Tab. The PDMI interface

Table 6.1 Techniques for Device Isolation

| Technique | Advantages | Disadvantages |
|---|---|---|
| Put the device in Airplane mode. This requires that you have full access to the Settings menu. | The device continues running and temporal data remains intact. Disables cellular data network as well as Wi-Fi.com. | You are modifying the device setting further. Only works if you have full access to the device. |
| If the phone is a GSM phone, remove the SIM card. | Easy to remove, effective in disabling all cellular voice, SMS, and data transmissions. | Does not disable Wi-Fi.com or other networks. Does not work on non-GSM phones including CDMA and iDEN phones. |
| Suspend account with wireless carrier. | Effective in disabling all cellular voice, SMS, and data transmissions for any phone. | Process may take some time and require a court order. Does not disable Wi-Fi.com or other networks. |
| Place device in a shielded bag, box, tent, or room. | Faraday shields prevent various types of network transmissions and can be an effective approach if you cannot utilize any of the previous options. | There is some debate about the effectiveness of portable Faraday shields, notably Faraday bags. Also, while the transmissions are blocked, the device attempts to contact the cellular network repeatedly thus draining the battery quickly. Cords cannot be inserted into the enclosure as they will transmit signals. A shielded room dedicated for mobile examinations is ideal. However, they are quite expensive to build and maintain. |
| Turn the device off. | Completely effective in preventing all network transmissions. | The device state is modified and temporal data is lost. Pass code on reboot could be enabled, thus restricting access to the device. |

provides not only power and high-resolution video output, but also offers USB 3.0 support. Whereas the actual examination of one of these devices could be delayed while an appropriate cable is acquired, if it needed charging and you do not have the appropriate cable, the loss of power will result in the loss of temporal data.

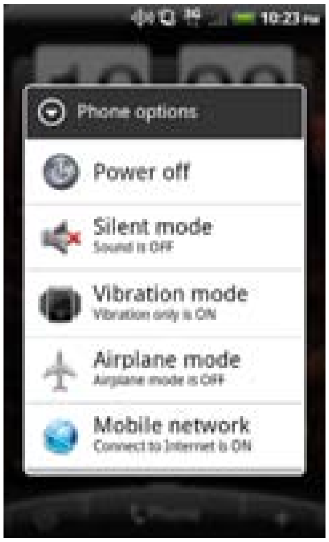


**FIGURE 6.1**

Airplane mode via the Power off button.

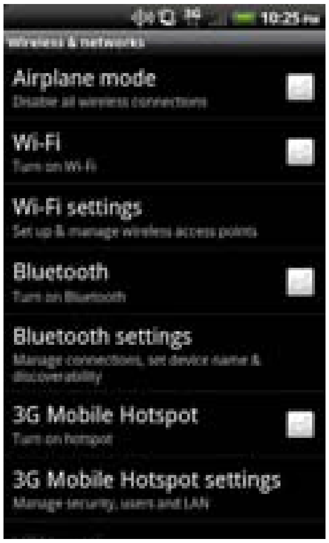

10:25
Wireless & networks
Airplane mode
Disable all wireless connections
Wi-Fi
Turn on Wi-Fi
Wi-Fi settings
Set up & manage wireless access points
Bluetooth
Turn on Bluetooth
Bluetooth settings
Manage connections, set device name & discoverability
3G Mobile Hotspot
Turn on hotspot
3G Mobile Hotspot settings
Manage security, users and LAN

**FIGURE 6.2**

Airplane mode via the Wireless and networks settings.

### *Powered-off Devices*

If a device is already powered off when you encounter it, the best option is to boot it into recovery mode to test for connectivity and root access. The owner may have already enabled USB debugging or have rooted the device, so you may have access to the data without booting into normal operational mode.

This approach is similar to performing forensics on a standard computer hard drive. The last thing any trained forensic analyst would do is boot the computer to determine what operating system is installed. Instead, the hard drive is removed and connected to a write blocker for imaging to prevent any changes to the evidence. Similarly, if a mobile device does not have to boot into normal mode, there is no need to do so as this may make changes to the device. Specific information on how to test a device in recovery mode for sufficient privileges is discussed later in this chapter.

## How to Circumvent the Pass Code

The ability to circumvent the pass code on an Android device is becoming more important as they are utilized frequently and, in most cases, do not allow data extraction. While there is no guaranteed method, there are a number of techniques which have worked in certain situations.

As previously discussed, there are three types of pass codes Android devices currently support. The first is a pattern lock. This was the default on the initial Android devices. To access the device, the user draws a pattern on the locked phone and, if drawn properly, the device is unlocked. An example of a pattern lock on an HTC Incredible is shown in Fig. 6.3.

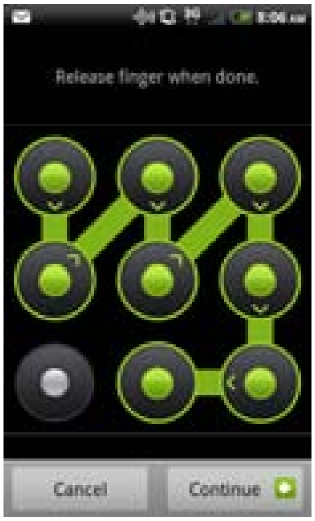


**FIGURE 6.3**

Android pattern lock.

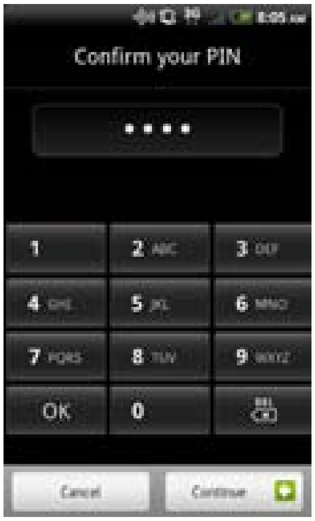


FIGURE 6.4

Android PIN lock.

The second type of pass code is the simple personal identification number (PIN) which is commonly found on other mobile devices. Fig. 6.4 is an example of a PIN-enabled HTC Incredible.

The final type of pass code currently found on Android devices is a full, alphanumeric code, as shown in Fig. 6.5.

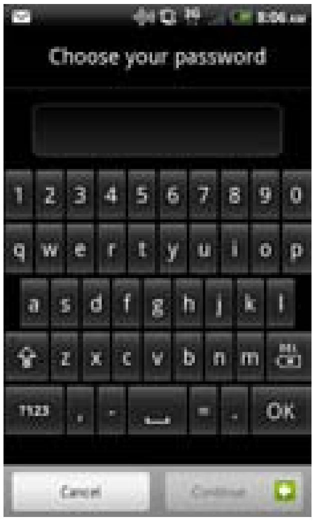


FIGURE 6.5

Android alphanumeric lock.

As discussed in Chapter 4, not all pass codes were created equal. The most effective pass code is one that allows or requires an alphanumeric password, as these are far more difficult to circumvent.

### *Utilize ADB if USB Debugging is Enabled*

The first technique you should attempt, provided the phone is powered on, is to connect with the Android Debug Bridge (ADB) over USB, which was covered extensively in Chapter 3. Whereas only a fraction of Android devices will allow an ADB connection through the USB debugging setting, it is certainly worth trying as it easily provides sufficient access for data extraction. The most common reasons for which users enable USB debugging include:

- App development and testing
- Certain apps require this setting, such as PDAnet, which allows the device to provide Internet access to a tethered device over USB
- Custom ROMs
- Developer phones such as Google's Android developer phone (ADP1)
- Device hacking

It is quite simple to determine if USB debugging is enabled, provided you are using the Ubuntu virtual machine (VM) or have a forensic workstation with a properly installed and configured Software Development Kit. With the phone running in normal mode, plug it into the Ubuntu VM. From the command prompt type "adb devices." If USB debugging is enabled, the ADB daemon will return the device serial number along with the mode that the phone is presently in.

```
ahoog@ubuntu:~$ adb devices
List of devices attached
HT08XHJ00657    device
```

If it is disabled, it will not return anything when the "adb devices" command is entered.

```
ahoog@ubuntu:~$ adb devices
List of devices attached

ahoog@ubuntu:~$
```

Remember to pass the device through to your VM if you are running the command inside a virtual workstation. If the VM can't see the device, you will get the same result as if the USB debugging were not enabled. Once you verify that the USB connection is passed through to the Ubuntu VM, you can execute the lsusb command to verify that the operating system is aware of the connection:

```
ahoog@ubuntu:~$ sudo lsusb -v
[sudo] password for ahoog:

Bus 001 Device 005: ID 0bb4:0c9e High Tech Computer Corp.
Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB               2.00
  bDeviceClass            0 (Defined at Interface level)
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0        64
  idVendor           0x0bb4 High Tech Computer Corp.
  idProduct          0x0c9e
  bcdDevice            2.26
  iManufacturer           1 HTC
  iProduct                2 Android Phone
  iSerial                 3 HT08XHJ00657
  bNumConfigurations      1
<snip>
    Interface Descriptor:
      bLength                 9
      bDescriptorType         4
      bInterfaceNumber        1
      bAlternateSetting       0
      bNumEndpoints           2
      bInterfaceClass       255 Vendor Specific Class
      bInterfaceSubClass     66
      bInterfaceProtocol      1
      iInterface              4 ADB
<snip>
```

In this example, emphasis was placed on several areas that clearly show the Android device is connected and, in such cases, we can see an ADB interface is exposed. If the device is connected but you cannot connect via ADB, you should also kill your local ADB daemon and then start it again. This is easily accomplished as follows:

```
ahoog@ubuntu:~$ adb kill-server
ahoog@ubuntu:~$ adb devices
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
List of devices attached
HT08XHJ00657    device
```

If the USB debugging is enabled, a forensic analyst can use the interface to gain access and perform a logical recovery of the device, which is covered in detail later in this chapter.

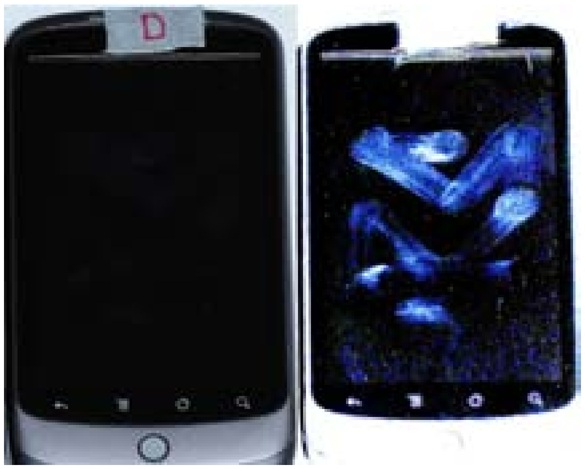


**FIGURE 6.6**

Enhanced photo showing smudge attack.

### *Smudge Attack*

Initially, Android devices used the pattern lock for pass code protection instead of a numeric or alphanumeric code. A recent paper entitled “Smudge Attacks on Smartphone Touch Screens” by the University of Pennsylvania Department of Computer and Information Science demonstrated a technique for accessing pattern locked Android devices by enhancing photographs of the screen (Aviv, Gibson, Mossop, Blaze, Smith, n.d.). The paper’s summary states:

> *Our photographic experiments suggest that a clean touch screen surface is primarily, but not entirely, reflective, while a smudge is primarily, but not entirely, diffuse. We found that virtually any directional lighting source that is not positioned exactly at a complementary angle to the camera will render a recoverable image of the smudge. Very little photo adjustment is required to view the pattern, but images generally rendered best when the photo capture was overexposed by two to three f-stops (4 to 8 times “correct” exposure).*

If care was taken by the first responders to minimize contact with the device’s screen, this recovery technique may be viable. As an example of what is possible, Fig. 6.6 shows photos of the same Android device displayed side by side. The same original photo was used for both images but the image on the right was enhanced as part of the smudge attack process to highlight the contact points.

### *Recovery Mode*

Some users install a custom ROM which usually enables root access to the device through a modified recovery mode. Most custom ROMs install a modified recovery partition which simplifies the process used to install the custom ROM. There are

**Table 6.2** Key Combinations to Boot into Recovery Mode

| Device | Key Combination |
|---|---|
| HTC G1 | Hold home button and press power button. Use volume down to select RECOVERY and press power key. |
| Nexus One | Hold volume down and press power button. |
| Motorola Droid | Hold X key and press power button. |
| HTC Incredible | Hold volume down and press power button. Use volume down to select RECOVERY and press power key. |

several popular recovery partitions that are primarily used with custom ROMs and both offer shell access with root privileges from within the recovery console itself. As the phone is not booted into normal mode, the pass code is circumvented and the user data partitions can be mounted read-only, thus preventing changes to that area.

Forensic analysts should attempt to boot into recovery mode if the device is powered off, when they take custody. If, instead, the device is running and a pass code is present, you should first attempt to connect via ADB and consider smudge attack. If neither of these is successful, you should then try to reboot into recovery mode. Like many other techniques, recovery mode is accessed in different ways depending on the device manufacturer and model. Table 6.2 covers the key combinations to access recovery mode on the phones referenced throughout this book. Each assumes the device is powered off already.

Once in recovery mode, you can connect the device to your Ubuntu workstation and attempt to connect using ADB. If the device is running a nonmodified recovery mode, the connection will fail. The screen generally shows a triangle with an exclamation point inside it and often a small Android device next to it. On other devices, you will be presented with the somewhat famous three Androids on a skateboard. Finally, other recovery modules clearly show they are in modified recovery code and provide a wide range of device options.

### *Flash a New Recovery Partition*

There are a number of protocols, utilities, and devices that allow a skilled examiner to flash the recovery partition of a device with a modified image.

The first available protocol supporting this approach was fastboot. Fastboot is a NAND flash update protocol executed over USB while in bootloader mode. Most devices ship with bootloader protection enabled, which prohibits the use of this protocol. However, it is possible that the protection has been disabled. To determine if bootloader protection is enabled, you must access the bootloader and look at the signature information, which will indicate S-ON or S-OFF. The S represents security, and so S-ON, the default production build, has security enabled; S-OFF indicates security is not enabled. Some devices ship with S-OFF, such as the Google Nexus One, as it is preloaded with Google's Engineering SPL/Bootloader.

Other rooting techniques also disable this protection, so checking this on a pass code protected device may yield results. You can access the main bootloader using the first part of the key combinations in Table 6.2 in the previous section.

Fastboot does not require USB debugging to access the device. Hence, like recovery mode, it can be used to gain access to the device's data. Once the new recovery partition is available, the device should be rebooted into recovery mode and forensic imaging can take place.

Other techniques exist which allow the recovery partition to be flashed with a new image. Some examples include:

- Motorola's RSD Lite
- sbf_flash
- Samsung's Odin Multiloader

While these utilities and protocols may ultimately provide the privileges that a forensic analyst requires, there is considerable effort required to not only locate and test the techniques but to understand them sufficiently to use them in a forensic investigation.

### *Screen Lock Bypass App*

Security researcher Thomas Cannon recently developed a technique that allows a screen lock bypass by installing an app through the new web-based Android Market (Cannon, T., n.d.). Cannon's technique utilizes a new feature in the web-based Android Market that allows apps to be installed directly from the web site. As such, you must have access to the Android Market using the primary Gmail user name and password for the device, which may be accessible from the primary computer of the user. Alternatively, you could access the Android Market if you knew the user name and password and had sufficient authority. Changing the user's Gmail password would not work in this instance.

Cannon explains the technique on this web site as in the following section (Cannon, T., n.d.).

### *How it Works*

The procedure is quite simple really. Android sends out a number of broadcast messages which an application can receive, such as SMS received or Wi-Fi.com disconnected. An application has to register its receiver to receive broadcast messages and this can be done at run time, or for some messages, at install time. When a relevant message comes in, it is sent to the application and if the application is not running it will be started automatically.

After testing out various broadcast messages the best one I found for the purpose of this utility was android.intent.action.PACKAGE_ADDED. This exists in all APIs as version 1 and is triggered when an application is installed. Hence, to get the application to execute remotely, we first deploy it from the Android Market, then deploy any other application that will cause the first one to launch.

Once launched it is just a matter of calling the disableKeyguard() method in KeyguardManager. This is a legitimate API to enable applications to disable the screen lock when, say, an incoming phone call is detected. After finishing the call the app ought to enable the screen lock again, but we just keep it disabled.

This technique is certainly worth consideration if you have proper access to the Android Market.

### Use Gmail User/Pass

On most Android phones, you can circumvent the pass code if you know the primary Gmail user name and password registered with the device. After a number of failed attempts (ten attempts on the G1), you will be presented with a screen that asks if you forgot your pass code. From there, you can enter the Gmail user name and password and you will then be prompted to reset the pass code. This technique does not require the phone to be online as it uses credential information cached on the phone.

If you do not have the current Gmail user name and password, but have sufficient authority (i.e., court order) to reset the password, you could attempt to compel Google to reset the account password. You would then have to connect the Android device to the network and gain access. This issue presents many challenges, including the need to place the device online, putting it at risk for remote wipe in addition to making changes to the device. Reports on various law enforcement mailing lists indicate this technique does not always work.

If this approach is attempted, additional research is warranted. In particular, it would be prudent to control the Internet connection the device uses, most likely a Wi-Fi.com access point. You could then limit the network access to only those which the Google server needed for authentication. In addition, a detailed network capture of test devices should be analyzed as well as the actual changes made to the device.

### JTAG and Chip-off

At this time, most Android devices do not encrypt the contents of the NAND flash, which makes directly accessing and decoding the memory chips a potential workaround if a pass code is enabled. There are two primary techniques, which provide direct access to the chips. Both are technically challenging. The two techniques are:

- Joint test action group (JTAG)
- Physical extraction (chip-off)

Both techniques are not only technically challenging and require partial to full disassembly of the device, but they require substantial post-extraction analysis to reassemble the file system. For these reasons, JTAG and chip-off would likely be the very last choices to circumvent a locked device.

With JTAG, you connect directly to the device's CPU by soldering leads to certain JTAG pads on the printed circuit board (PCB). Then JTAG software can be used to perform a complete binary memory dump of the NAND flash, modify certain partitions to allow root access, or eliminate the pass code altogether.

In the chip-off procedure, the NAND flash chips are physically extracted from the PCB using heat and air. The chip, usually a small ball grid array (BGA) package, then needs to have the BGA connections regenerated and inserted into special hardware that connects to the chip and reads the NAND flash.

The advantages to these techniques are that they will work in any situation where the NAND flash is not encrypted. However, extensive research, development, testing, and practice are required to execute these techniques.

## IMAGING ANDROID USB MASS STORAGE DEVICES

Every Android device to date has either an external Secure Digital (SD) card or an Embedded MultiMediaCard (eMMC) that provides the large storage space required by many users. These storage devices exist because the user's app data, typically stored in /data/data, is isolated for security and privacy reasons. However, users want to copy songs, pictures, videos, or other files between their Android device and a computer, and these large capacity FAT file system partitions solve that issue. The sensitive user data remains protected, yet the larger and more portable files are accessible to the user.

Initially, the approach to imaging the external storage was to simply remove it from the Android device and image using a USB write blocker. However, a number of challenges arose over time, including:

- Moving to eMMC storage meant that the mass storage was no longer removable.
- Apps can now run from the SD card and in this scenario, the .apk files are encrypted. If capturing an unencrypted copy of the app is critical to an investigation (for example, a case involving malware analysis or a Trojan horse defense), the SD card must remain in the Android device.
- Newer devices are using RAM disks (tmpfs) more frequently to store user data that might be helpful in an investigation. Often, removing the SD card requires the device to be shut down and the battery removed, thus losing the ability to recover the temporal data.

For these reasons, the recommended approach for imaging the USB Mass Storage (UMS) devices on Android no longer involves removing the SD card but instead imaging it via the UMS interface.

### SD Card Versus eMMC

An SD card and eMMC are not all that different. The primary difference, of course, is that the SD cards are portable, easily moving from one device to the next. They use NAND flash, are based on the MultiMediaCard (MMC) specification, and have embedded storage controllers, so that systems MTD is not needed for guest operating systems to read them.

To date, Android devices accept microSD cards generally ranging from 2 GB up to 16 GB. However, larger cards are possible. Depending on the Android device, the SD card may be easily accessed and removed from a running device. However, many require that the device is shut down so that the battery can be removed.

For storage embedded on the device, several manufacturers have begun using eMMC, which consists of embedded storage with an MMC interface integrating directly onto the device's PCB. This standard simplifies accessing NAND flash with the standardized eMMC protocol and is capable of supporting file systems that are not NAND flash aware. This does not necessarily mean the file systems preserve the life of the NAND flash at the same level and sophistication that a NAND flash-aware file system like YAFFS2 does. However, the general lifespan of Android devices is certainly decreasing and is likely not an issue for most users.

## How to Forensically Image the SD Card/eMMC

There are two primary methods to forensically acquire the SD card and eMMC without removing it from the device. The first method, covered here, exposes the UMS device interface to your forensic workstation and allows you to acquire the image with your forensic tool of choice. The second method does not expose the UMS to your forensic workstation and instead uses dd on the Android device. This requires adb port forwarding, which will be covered in the section on physical techniques later in this chapter.

Even though our Ubuntu VM has dd built in, we are going to download, compile, and install an updated version of dd maintained by the Department of Defense's Cyber Crime Center. The program, dc3dd, is a patched version of GNU dd and includes a number of features useful for computer forensics (dc3dd, n.d.), such as:

- Piecewise and overall hashing with multiple algorithms—Supports MD5, SHA-1, SHA-256, and SHA-512.
- Progress meter with automatic input/output file-size probing.
- Combined log for hashes and errors.
- Error grouping—Produces one error message for identical sequential errors.
- Verify mode—Able to hash output files and compare hashes to the acquisition hash.
- Ability to split the output into chunks with numerical or alphabetic extensions.
- Ability to write multiple output files simultaneously.

The program is open source software licensed under the GNU Public license version 3 (GPLv3) and is distributed online through SourceForge and was updated to version 7.0 in August 2010 (dc3dd, n.d.). At this point in the book, you should be fairly comfortable compiling programs and have all the tools needed, so here are the abbreviated steps.

```
mkdir -p ~/src
cd ~/src
curl http://cdnetworks-us-2.dl.sourceforge.net/project/dc3dd/
dc3dd/7.0.0/dc3dd-7.0.0.tar.gz
> dc3dd-7.0.0.tar.gz
tar xzf dc3dd-7.0.0.tar.gz
cd dc3dd-7.0.0/
./configure
make
sudo make install
```

At this point, you could proceed with imaging. However, typing out the entire dc3dd command each time is not only tedious but can result in typos that could cause irreparable damage. So create a shell script, which not only acquires the device but also records various system characteristics, date/time stamps, and creates log files, which can be helpful as you write your report at a later time.

We will place the acquire script in /usr/local/bin so you can easily run the script from any directory as /usr/local/bin is in your execution path by default:

```
ahoog@ubuntu:~$ sudo nano -w /usr/local/bin/acquire-disk.sh
```

Next, copy the following into the script, save by pressing Ctrl-O, and exit with Ctrl-X:

```
#!/bin/bash

CLIENT="${1}"
CASE="${2}"
TAG="${3}"
SERIALNO="${4}"
SOURCEDEV="${5}"
DESTPATH="${6}"

OUTPUTPATH=$DESTPATH/$CLIENT/$CASE/$TAG-$SERIALNO
LOGFILE=$OUTPUTPATH/log/$TAG-$SERIALNO.log
STDERRLOG=$OUTPUTPATH/log/$TAG-$SERIALNO.stderr.log
SEPERATOR="--------------------------------------------\r"

if [ "$#" != 6 ]; then
        echo "Usage: acquire_disk.sh CLIENT CASE TAG SERIALNO SOURCEDEV
DESTPATH"
        exit 2
fi

# check directories, created if needed
if [ ! -d "$DESTPATH" ]; then
        echo "Destination path [$DESTPATH] does not exist, exiting"
        exit 1
fi

if [ -d "$DESTPATH/$CLIENT/$CASE/$TAG-$SERIALNO" ]; then
        echo "$DESTPATH/$CLIENT/$CASE/$TAG-$SERIALNO already exists, can't
overwrite evidence"
        exit 1
fi
```

```
GOTROOT=`whoami`

if [  "$GOTROOT" != "root" ]; then
       echo "must be root to execute"
       exit 1
fi

mkdir -p $OUTPUTPATH/log

echo -e "Start date/time" >> $LOGFILE
echo -e "$SEPERATOR" >> $LOGFILE
echo -e "`/bin/date`\n" >> $LOGFILE 2>> $STDERRLOG

echo -e "uname -a" >> $LOGFILE
echo -e "$SEPERATOR" >> $LOGFILE
echo -e "`uname -a`\n" >> $LOGFILE 2>> $STDERRLOG

echo -e "dmesg | tail -50" >> $LOGFILE
echo -e "$SEPERATOR" >> $LOGFILE
echo -e "`dmesg | tail -50`\n" >> $LOGFILE 2>> $STDERRLOG

echo -e "lshw" >> $LOGFILE
echo -e "$SEPERATOR" >> $LOGFILE
echo -e "`lshw`\n" >> $LOGFILE 2>> $STDERRLOG

VERSION=`fdisk -v`
echo -e "fdisk -l $SOURCEDEV [$VERSION]" >> $LOGFILE
echo -e "$SEPERATOR" >> $LOGFILE
echo -e "`fdisk -l $SOURCEDEV`\n" >> $LOGFILE 2>> $STDERRLOG

VERSION=`mmls -V`
echo -e "mmls $SOURCEDEV [$VERSION]" >> $LOGFILE
echo -e "$SEPERATOR" >> $LOGFILE
echo -e "`mmls $SOURCEDEV`\n" >> $LOGFILE 2>> $STDERRLOG

VERSION=`fsstat -V`
echo -e "fsstat $SOURCEDEV [$VERSION]" >> $LOGFILE
echo -e "$SEPERATOR" >> $LOGFILE
echo -e "`fsstat $SOURCEDEV`\n" >> $LOGFILE 2>> $STDERRLOG

VERSION=`dc3dd --version 2>&1 | grep dc3dd`
echo -e "dc3dd [$VERSION]" >> $LOGFILE
echo -e "$SEPERATOR" >> $LOGFILE
echo -e "dc3dd if=$SOURCEDEV of=$OUTPUTPATH/$TAG-$SERIALNO.dc3dd verb=on
hash=sha256 hlog=$OUTPUTPATH/log/$TAG-$SERIALNO.hashlog
log=$OUTPUTPATH/log/$TAG-$SERIALNO.log rec=off\n" >> $LOGFILE
dc3dd if=$SOURCEDEV of=$OUTPUTPATH/$TAG-$SERIALNO.dc3dd verb=on hash=sha256
hlog=$OUTPUTPATH/log/$TAG-$SERIALNO.hashlog log=$OUTPUTPATH/log/
$TAG-$SERIALNO.log rec=off

echo -e "ls -lR $DESTPATH/$CLIENT/$CASE/$TAG-$SERIALNO" >> $LOGFILE
echo -e "$SEPERATOR" >> $LOGFILE
echo -e "`ls -lR $DESTPATH/$CLIENT/$CASE/$TAG-$SERIALNO`\n" >> $LOGFILE

echo -e "End date/time" >> $LOGFILE
echo -e "$SEPERATOR" >> $LOGFILE
echo -e "`/bin/date`\n" >> $LOGFILE

#sha256sum all log files
cd $OUTPUTPATH/log/
sha256sum * > $TAG-$SERIALNO.sha256.log
```

Next, you have to change permissions, so that you can run the script and then run it without parameters to see the usage help:

```
ahoog@ubuntu:~$ sudo chmod 755 /usr/local/bin/acquire-disk.sh
ahoog@ubuntu:~$ sudo /usr/local/bin/acquire-disk.sh
Usage: acquire_disk.sh CLIENT CASE TAG SERIALNO SOURCEDEV DESTPATH
```

The great thing about this script, and open source in general, is that you can simply change it as you see fit. If you do not want to track client name, then simply remove it from the script.

Next, we have to mount the UMS device on your Ubuntu workstation. As covered in Chapter 1 in the Ubuntu VM setup, it is critical that you have disabled automount on your workstation. If you did not do this, please review the steps necessary and complete before presenting the UMS devices to the VM.

Additionally, the ideal situation would first connect the Android device to a hardware-based USB write blocker. However, some write blockers seem to have trouble when the connected device exposes more than one device ID. You should experiment with your USB write blocker and, ideally, have this working first.

**NOTE**

**Tableau UltraBlock USB**

The Tableau UltraBlock USB model T8, running the latest firmware from August 9, 2009, was only able to pass through the first USB device found on the reference HTC Incredible, and so we were unable to use it when analyzing a device. Tableau has a new UltraBlock USB device which may work; however, we have not verified this. Examiners should test the various USB write blockers they have for compatibility.

Next, we need to determine to what devices the UMS is mapped. This information is displayed in the kernel logs and can be easily accessed with the “dmesg” command:

```
ahoog@ubuntu:~/$ dmesg
<snip>
[327202.720222] usb 1-1: new high speed USB device using ehci_hcd and address 12
[327203.032759] scsi11 : usb-storage 1-1:1.0
[327204.039549] scsi 11:0:0:0: Direct-Access     HTC      Android Phone    0100
PQ: 0 ANSI: 2
[327204.044572] scsi 11:0:0:1: Direct-Access     HTC      Android Phone    0100
PQ: 0 ANSI: 2
[327204.047208] scsi 11:0:0:2: CD-ROM            HTC      Android Phone    0100
PQ: 0 ANSI: 2
[327204.049854] sd 11:0:0:0: Attached scsi generic sg2 type 0
[327204.052640] sd 11:0:0:1: Attached scsi generic sg3 type 0
[327204.066738] sr1: scsi3-mmc drive: 0x/0x caddy
[327204.066817] sr 11:0:0:2: Attached scsi CD-ROM sr1
[327204.066892] sr 11:0:0:2: Attached scsi generic sg4 type 5
[327204.082001] sd 11:0:0:0: [sdb] Attached SCSI removable disk
[327204.091070] sd 11:0:0:1: [sdc] Attached SCSI removable disk
```

As discussed previously, the HTC Incredible exposes three USB interfaces in addition to ADB:

- CD-ROM for device driver install (sr1)
- eMMC UMS device (sdb)
- SD card UMS device (sdc)

However, the differences between /dev/sdb and /dev/sdc are not easily discernible until the UMS or Disk drive feature is enabled on the Android device. Once enabled, you should then examine the output of dmesg again.

**WARNING**

**Use hardware write blocker**

Although the automount feature on the Ubuntu workstation has been disabled, it is critical that the forensic analyst connects only the Android device to the workstation through a hardware write blocker to ensure no changes are made to the device. All hardware should be thoroughly tested prior to active use in a case.

```
ahoog@ubuntu:~/$ dmesg
<snip>
[327520.269248] sd 11:0:0:1: [sdc] 3911680 512-byte logical blocks:
(2.00 GB/1.86 GiB)
[327520.298549] sd 11:0:0:1: [sdc] Assuming drive cache: write through
[327520.304747] sd 11:0:0:1: [sdc] Assuming drive cache: write through
[327520.304757]  sdc: sdc1
[327522.267959] sd 11:0:0:0: [sdb] 13844464 512-byte logical blocks:
(7.08 GB/6.60 GiB)
[327522.271097] sd 11:0:0:0: [sdb] Assuming drive cache: write through
[327522.277187] sd 11:0:0:0: [sdb] Assuming drive cache: write through
[327522.277202]  sdb:
```

It is now clearer that /dev/sdb is the 7 GB storage device (which is the eMMC) while the 2 GB SD card is mapped to /dev/sdc. We can now acquire the devices using our acquire script or any forensic imaging tool available on your forensic workstation. The script takes the following six parameters:

1. Client —This parameter creates the folder structure, examples might be "sheriffs-office" or a client name such as "viaforensics."
2. Case—This parameter provides a case name, such as af-book.
3. Tag—This parameter is tag number for the evidence you are forensically imaging, item001 in our example.
4. Serialno—This is the serial number of the device, disk, SD card, etc. If you do not have access to a serial number, you can type any text you choose such as unknown-serialno.
5. Sourcedev—This is the device you want to acquire such as /dev/sdb, /dev/sdc, etc. You can determine this using dmesg which is explained next.
6. Destpath—The top-level directory where the folders should be created. It could be your home directory (~) or perhaps a folder called clients (~/clients).

For this example, create a folder in your home directory called sd-emmc and then run the acquire script with sudo permissions.

```
ahoog@ubuntu:~$ sudo acquire-disk.sh viaforensics af-book item001
unknown-serialno /dev/sdc ~/sd-emmc
Cannot determine file system type

dc3dd 7.0.0 started at 2011-02-22 04:36:05 -0600
compiled options:
command line: dc3dd if=/dev/sdc of=/home/ahoog/sd-emmc/viaforensics/
af-book/item001-unknown-serialno/item001-unknown-serialno.dc3dd
verb=on hash=sha256 hlog=/home/ahoog/sd-emmc/viaforensics/af-book/
item001-unknown-serialno/log/item001-unknown-serialno.hashlog
log=/home/ahoog/sd-emmc/viaforensics/af-book/item001-unknown-serialno/
log/item001-unknown-serialno.log rec=off
device size: 3911680 sectors (probed)
sector size: 512 bytes (probed)
2002780160 bytes (1.9 G) copied (100%), 808.727 s, 2.4 M/s

input results for device '/dev/sdc':
   3911680 sectors in
   0 bad sectors replaced by zeros
   fc8f3d6dc7e659c3124a4113d2d0ebe87466b497038aedf9f7a1b89c44eda8b9 (sha256)

output results for file '/home/ahoog/sd-emmc/viaforensics/af-book/
item001-unknown-serialno/item001-unknown-serialno.dc3dd':
   3911680 sectors out

dc3dd completed at 2011-02-22 04:49:34 -0600
```

You can then use the same general command, but change the parameters to image the eMMC which, for the device, is located at /dev/sdb. After these commands complete, the forensic images and log files are in ~/sd-emmc and are structured as follows:

```
hoog@ubuntu:~$ tree -h sd-emmc/
sd-emmc/
└── [4.0K]  viaforensics
    └── [4.0K]  af-book
        ├── [4.0K]  item001-emmc-unknown-serialno
        │   ├── [6.6G]  item001-emmc-unknown-serialno.dc3dd
        │   └── [4.0K]  log
        │       ├── [ 686]  item001-emmc-unknown-serialno.hashlog
        │       ├── [ 39K]  item001-emmc-unknown-serialno.log
        │       ├── [ 374]  item001-emmc-unknown-serialno.sha256.log
        │       └── [   0]  item001-emmc-unknown-serialno.stderr.log
        └── [4.0K]  item001-sd-unknown-serialno
            ├── [1.9G]  item001-sd-unknown-serialno.dc3dd
            └── [4.0K]  log
                ├── [ 726]  item001-sd-unknown-serialno.hashlog
                ├── [ 39K]  item001-sd-unknown-serialno.log
                ├── [ 296]  item001-sd-unknown-serialno.sha256.log
                └── [   0]  item001-sd-unknown-serialno.stderr.log

6 directories, 10 files
```

For each UMS device forensically imaged, we not only have the verified image but also a hashlog for the dd image, log file with date, time, system info and commands run, an error log, and finally a listing of each log file and its sha256 hash. This ensures sufficient details are known about the imaging process.

**TIP**

**Encrypted apps on the SD card**

If apps are installed on the SD card, they are encrypted and thus, if the files are examined from the SD card image, they will be unreadable. However, when the SD card is not mounted on your forensic workstation, the unencrypted .apk files are mounted in /mnt/asec. If an investigation relies on .apk app analysis, ensure you acquire a copy of the unencrypted files too.

## LOGICAL TECHNIQUES

As discussed at the start of this chapter, logical forensic techniques extract data that is allocated. This is typically achieved by accessing the file system. Logical techniques are often the first type of examination a forensic analyst will run because they are not only easier to execute but often provide sufficient data for the case. Android forensics physical techniques can provide far more data. However, they are more difficult to successfully execute and take considerably more effort to analyze.

Logical techniques also have the advantage of working in far more scenarios as the only requirement is that USB debugging is enabled. In other words, Android forensics logical techniques do not require root access.

In this section, we first cover techniques that are freely available (although AFLogical is only free for active law enforcement and government agencies) followed by a review of available commercial software.

### ADB Pull

In Chapter 4, the recursive adb pull command was demonstrated several times as various parts of the file system were copied to the Ubuntu workstation for further analysis. Unless an Android device has root access or is running a custom ROM, the adb daemon running on the device that proxies the recursive copy only runs with shell permissions. As such, some of the more forensically relevant files are not accessible. However, there are still files which can be accessed.

If you attempt to access files that the shell user does not have permissions to, it simply does not copy the files:

```
ahoog@ubuntu:~$ adb pull /data adbpull
pull: building file list...
0 files pulled. 0 files skipped.
```

However, if you have sufficient privileges (root in the next example), then this method is very simple and effective:

```
ahoog@ubuntu:~$ adb pull /data adbpull/
pull: building file list...
<snip>
pull: /data/miscrild_nitz_long_name_31026 -> data/misc/rild_nitz_long_name_31026
pull: /data/misc/akmd_set.txt -> data/misc/akmd_set.txt

712 files pulled. 0 files skipped.
963 KB/s (208943249 bytes in 211.671s)
```

As you can see from the output above, the entire "/data" partition was copied to a local directory in just over three and a half minutes. The directory structure is maintained during the copy so you can then simply browse or otherwise analyze the files of interest from the workstation.

As most phones will not have root access (at least by default), this technique may appear to be of little value. However, it is a powerful utility to understand and there are several scenarios ideal for this approach. These scenarios include:

- On nonrooted devices, an adb pull can still access useful files such as unencrypted apps, most of the tmpfs file systems that can include user data such as browser history, and system information found in "/proc," "/sys," and other readable directories.
- On rooted devices, a pull of nearly all directories is quite simple and certain files and directories from "/data" would be of interest.
- When utilizing the physical technique, it is not always possible to mount some acquired file systems such as YAFFS2. If adbd is running with root permissions, you can quickly extract a logical copy of the file system with adb pull.

As adb is not only a free utility in the Android SDK but also very versatile, it should be one of the primary logical tools used on a device.

**WARNING**

**adb Pull issues**

Some recursive pulls using adb can fail in the middle of the data transfer due to permission or other issues. You should closely monitor the results of the command to determine if any issues were encountered. Breaking the recursive pull of large directories into smaller data pulls may yield better results.

## Backup Analysis

When Android was first released, it did not provide a mechanism for users to backup their personal data. As a result, a number of backup applications were developed and distributed on the Android Market. For users running custom ROMs, there was an even more powerful backup utility developed called nandroid.

Many of the backup utilities have a "Save to SD Card" option (which users found extremely convenient) as well as several options to save to "the cloud." Either way, users could take a backup of their devices, and if needed they could restore required data. This is not only a great way for users to protect themselves from data loss, but it can be a great source of information for forensic analysts.

One of the more popular backup apps is RerWare's My Backup Pro which can take a backup of device data using Content Provider and even the entire "/data/data" files if the device has root access. The user can choose between saving to the SD card and saving to RerWare's server. The app supports (RerWare, LLC, n.d.) the following:

- Application install files (if phone has root access, this includes APK + Data and Market Links)
- Contacts
- Call log
- Browser bookmarks
- SMS (text messages)
- MMS (attachments in messages)
- System settings
- Home screens (including HTC Sense UI)
- Alarms
- Dictionary
- Calendars
- Music playlists
- Integrated third-party applications

The last bullet, "integrated third-party applications," refers to companies who provide RerWare hooks for data backup. At least initially, RerWare would pay developers to include RerWare backup support in their apps.

Interestingly, the app runs not only on Android but also on Windows Mobile, Blackberry, and soon Symbian OS. The user can take a backup on one platform and restore on a completely different supported OS. RerWare saves a single SQLite file to the SD card when the device backup is stored locally.

In the more recent releases of Android, a new backup API is now available. Developers can simply integrate these APIs into their apps and the rest of the backup is handled by Android and Google. This provides the users with secure, cloud-based backups with consistency across apps, and will likely become the de facto standard. Unfortunately, current research has not yet discovered useful artifacts from the new backup APIs left on an Android device.

Regardless of the backup app, forensic analysts should determine if one was installed and, if so, where the backup data is stored. The SD card should be examined as well as other devices such as a computer or laptop. The data saved in a backup is obviously of significant value in an examination.

## AFLogical

AFLogical is an Android forensics logical technique which is distributed free to law enforcement and government agencies. The app, developed by viaForensics, extracts data using Content Providers, which are a key feature of the Android platform. This is the same technique that commercial forensics tools use for logical forensics.

Recall that Android's security model is effective in limiting access to app data except in a few circumstances. Here is a quick recap of the key components of Android's security model:

- Each application is assigned a unique Linux user and group id.
- Apps execute using their specific user ID in a dedicated process and Dalvik VM.
- Each app has dedicated storage, generally in "/data/data," that only the app can access.

However, the Android framework does provide a mechanism by which apps can share data. An app developer can include support for Content Providers within their application, which allows them to share data with other apps. The developer controls what data is exposed to other apps. During the install of an app, the user controls whether or not an app should gain access to the requested Content Providers.

Some examples of Content Providers are:

- SMS/MMS
- Contacts
- Calendar
- Facebook
- Gmail

And there are many more.

The AFLogical app takes advantage of the Content Provider architecture to gain access to data stored on the device. Similar to commercial Android logical tools, USB debugging must be enabled on the device for AFLogical to extract the data. The current version, 1.5.1, extracts data from 41 Content Providers and provides the output information to the SD card in CSV format and as an info.xml file, which provides details about the device and installed apps. AFLogical supports devices running Android 1.5 and later, and has been specifically updated to support extraction of large data sets such as an SMS database with over 35,000 messages. The currently supported Content Providers are:

1. Browser Bookmarks
2. Browser Searches
3. Calendars
4. Calendar Attendees
5. Calendar Events
6. Calendar Extended Properties
7. Calendar Reminders
8. Call Log Calls
9. Contacts Contact Methods
10. Contacts Extensions
11. Contacts Groups
12. Contacts Organizations
13. Contacts Phones

14. Contacts Settings
15. External Media
16. External Image Media
17. External Image Thumb Media
18. External Videos
19. IM Account
20. IM Accounts
21. IM Chats
22. IM Contacts Provider (IM Contacts)
23. IM Invitations
24. IM Messages
25. IM Providers
26. IM Provider Settings
27. Internal Image Media
28. Internal Image Thumb Media
29. Internal Videos
30. Maps-Friends
31. Maps-Friends extra
32. Maps-Friends contacts
33. MMS
34. Mms Parts Provider (MMSParts)
35. Notes
36. People
37. People Deleted
38. Phone Storage (HTC Incredible)
39. Search History
40. SMS
41. Social Contracts Activities

Let's walk through the steps for running AFLogical on a device. First, ensure you have downloaded AFLogical, which requires registration and approval from via-Forensics. (You can access the AFLogical page at http://viaforensics.com/products/tools/aflogical/.) Next, you need to replace the user's SD card with an SD card you control and ensure that USB debugging is enabled on the device. Then connect the Android device to your Ubuntu workstation and make sure you pass the USB connection through to the VM.

**WARNING**

**Replace user's SD card**

This version of AFLogical writes content directly to the SD card and it is important that the user's SD card is removed and replaced with the examiner's SD card. Failure to do this will either write data to the user's SD card or AFLogical will fail as it cannot write to the SD card. The commercial version of this app will eventually replace the writing to the SD card in favor of port forwarding over adb.

From a terminal session, verify you can see the device:

```
$adb devices
List of devices attached
0403555511112222F           device
```

Assuming you saved the AFLogical app in your home directory, you can install it with the following command:

```
ahoog@ubuntu:~$ adb install ~/AndroidForensics.apk
523 KB/s (31558 bytes in 0.058s)
        pkg: /data/local/tmp/AndroidForensics.apk
Success
```

Note: If AFLogical is already installed on the device, an error will display and you must uninstall the existing app before you can install the new version. To uninstall, run the following command:

```
ahoog@ubuntu:~$ adb install /opt/via/AFLogical/AndroidForensics.apk
824 KB/s (31558 bytes in 0.037s)
        pkg: /data/local/tmp/AndroidForensics.apk
Failure [INSTALL_FAILED_ALREADY_EXISTS]

ahoog@ubuntu:~$ adb uninstall com.viaforensics.android
Success
```

After the application is successfully installed, you can run the program from either the Android device directly or via command line. If you run the app from command line, you can simply start the app and then complete using the device or have it run automated. To run the app and extraction automatically, execute the following:

```
ahoog@ubuntu:~$ adb shell am start -n
com.viaforensics.android/com.viaforensics.android.ExtractAllData
Starting: Intent { cmp=com.viaforensics.android/.ExtractAllData }
```

The program immediately starts and begins to extract data from all supported Content Providers. If you are viewing the screen, you would see an image similar to Fig. 6.7.

Or you can simply start the app with the following:

```
ahoog@ubuntu:~$ adb shell am start -n
com.viaforensics.android/com.viaforensics.android.ForensicsActivity
Starting: Intent { cmp=com.viaforensics.android/.ForensicsActivity }
```

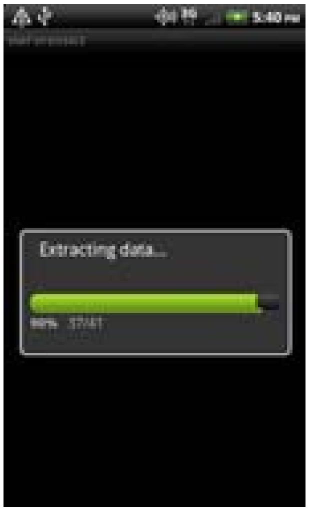
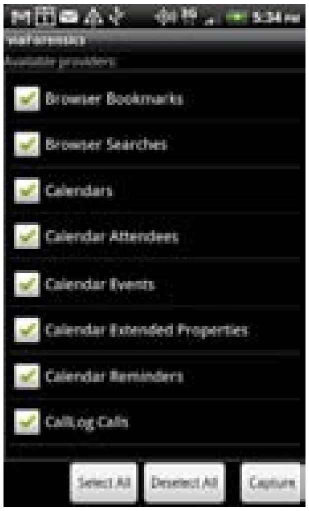


**FIGURE 6.7**

AFLogical, extract all from command line.

And then complete the acquisition using the screen presented on the device as shown in Fig. 6.8.

Otherwise, you can simply run the app directly from the All Apps screen on the device as shown in Fig. 6.9. First, access the Android app menu,



✓ Browser Bookmarks
✓ Browser Searches
✓ Calendars
✓ Calendar Attendees
✓ Calendar Events
✓ Calendar Extended Properties
✓ Calendar Reminders
✓ CallLog Calls

**FIGURE 6.8**

AFLogical, run from command line.

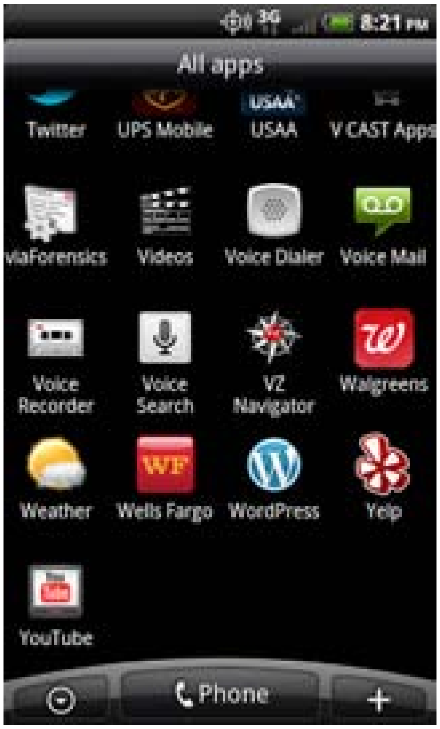


**FIGURE 6.9**

AFLogical in All Apps list.

look for a program called viaForensics and click on the icon to launch the app.

You will then be presented with the AFLogical data extraction screen. You can select or deselect individual Content Providers or leave all of them selected. Next, you hit Capture which will start the data collection process as illustrated in Fig. 6.10.

Once the data collection is complete, you will receive the corresponding message shown in Fig. 6.11.

The extracted data are saved to the SD card of the device in a directory called forensics and a subdirectory named after the date in YYYYMMDD.HHMM format. For this example, we moved the files from the SD card to an AFLogical directory on the local file system using adb pull. If you examine that folder, you see:

```
ahoog@ubuntu:~$ ls AFLogical/
20110221.1708
```

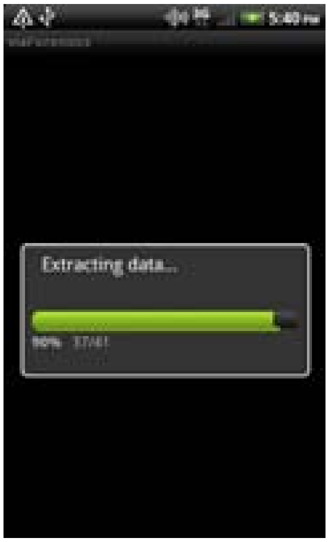


**FIGURE 6.10**

AFLogical capturing data.

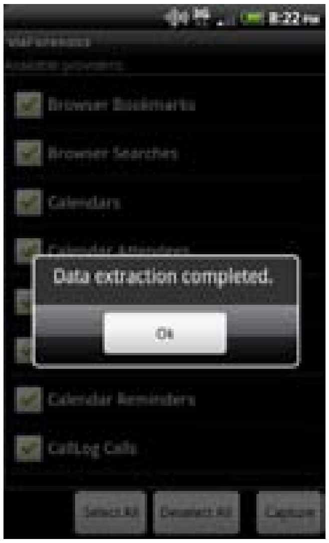


**FIGURE 6.11**

AFLogical, data extraction complete.

This then contains the extracted data:

```
ahoog@ubuntu:~$ ls AFLogical/20110221.1708/
Browser Bookmarks.csv           IM Providers.csv
Browser Searches.csv            IM ProviderSettings.csv
CallLog Calls.csv               info.xml
Contacts ContactMethods.csv     Internal Image Media.csv
Contacts Extensions.csv         Internal Image Thumb Media.csv
Contacts Groups.csv             Internal Videos.csv
Contacts Organizations.csv      Maps-Friends contacts.csv
Contacts Phones.csv             Maps-Friends.csv
Contacts Settings.csv           Maps-Friends extra .csv
External Image Media.csv        MMS.csv
External Image Thumb Media.csv  MMSParts.csv
External Media.csv              People.csv
External Videos.csv             PhoneStorage (HTC Incredible).csv
IM Account.csv                  sanitize.sh
IM Accounts.csv                 Search History.csv
IM Chats.csv                    SMS.csv
IM Contacts.csv                 Social Contracts Activities.csv
IM Invitations.csv
```

The CSV files can be viewed using any editor or spreadsheet. There is also a file in the directory called info.xml, which contains information about the device including the IMSI, IMEI, Android version, network provider, and more, as well as the list of all installed apps.

```
ahoog@ubuntu:~$ less info.xml
<android-forensics>
<date-time>20110221.1708</date-time>
<IMSI>removed</IMSI>
<IMEI>removed</IMEI>
<build>
        <version.release>2.2</version.release>
        <version.sdk>8</version.sdk>
        <version.incremental>264707</version.incremental>
        <board>inc</board>
        <brand>verizon_wwe</brand>
        <device>inc</device>
        <display>FRF91</display>
        <fingerprint>verizon_wwe/inc/inc/inc:2.2/FRF91/264707:user/release-
keys</fingerprint>
        <host>HPA003</host>
        <id>FRF91</id>
        <model>ADR6300</model>
        <product>inc</product>
        <tags>release-keys</tags>
        <time>1285855309000</time>
        <type>user</type>
        <user>root</user>
</build>
<applications>
        <app>
                <label>Network Location</label>
                <className>null</className>
                <dataDir>/data/data/com.google.android.location</dataDir>
                <descriptionRes>0</descriptionRes>
                <flags>48709</flags>
                <manageSpaceActivityName>null</manageSpaceActivityName>
                         <name>null</name>
                         <packageName>com.google.android.location</packageName>
                         <permission>null</permission>
                         <processName>system</processName>
                         <publicSourceDir>/system/app/NetworkLocation.apk
                         </publicSourceDir>
                         <sourceDir>/system/app/NetworkLocation.apk</sourceDir>
                         <taskAffinity>com.google.android.location</taskAffinity>
                         <uid>1000</uid>
                         <enabled>true</enabled>
                         <description>null</description>
                         <packageinfo>                   <versionCode>8</versionCode>
                                 <versionName>2.2</versionName>
                         </packageinfo>  </app>
                 <app>
                         <label>IMDb</label>
                         <className>null</className>
                         <dataDir>/data/data/com.imdb.mobile</dataDir>
         <snip>
```

The data can now be analyzed by the forensic examiner and easily shared with others.

**WARNING**

**Uninstall AFLogical**

Do not forget to uninstall AFLogical. Failure to uninstall the app would mean that the Android device could potentially be returned to the owner with the forensic agent still accessible. To uninstall AFLogical, key in the following command:

```
adb uninstall com.viaforensics.android
```

This should return Success. Alternately, you can go to the home screen, press Menu, select Applications, Manage Applications, viaForensics, and finally Uninstall.

## Commercial Providers

Many of the commercial mobile forensic software vendors now support Android. To date, the forensic software only supports a logical examination of an Android device using the same Content Provider technique used by AFLogical. It can be helpful for a forensic examiner to understand how each of the forensic software vendors implement Android support.

Each software company provided an evaluation copy of their software as well as an overview of their platform, which is included at the beginning of each section. A Motorola Droid running Android 2.2 was used for the examination. This section is not intended to evaluate each platform, but rather provide a helpful overview. The following forensic software packages were provided (reviewed in alphabetical order):

- Cellebrite UFED
- Compelson MOBILedit!
- EnCase Neutrino

- Micro Systemation XRY
- Paraben Device Seizure
- viaForensics' viaExtract

Two additional forensic software packages were tested. However, issues were encountered which prevented their inclusion at this time. These vendors do provide a forensic solution for Android and, if interested, you should review their offerings independently. The software packages omitted were Oxygen Forensic Suite 2010 and Logicube's CellDEK.

The challenge with any such overview is that the forensic software is updated frequently enough that a newer version likely already exists. Forensic examiners and security engineers interested in a particular software package should check the vendor's web site or contact them directly.

### *Cellebrite UFED*

The following overview of Cellebrite was provided by the vendor:

The Cellebrite UFED Forensic system is a stand-alone device capable of acquiring data from approximately 1600 mobile devices and storing the information on a USB drive, SD card or PC. UFED also has a built-in SIM card reader and cloner. The ability to clone a SIM card is a powerful feature as you can create and insert a clone of the original SIM and the phone will function normally. However, it will not register on the mobile carrier's network, eliminating the need for Faraday bags and the possibility that the data on the phone will be updated (or erased). The UFED package ships with about 70 cables for connecting to most mobile devices available today. Connection protocols supported include serial, USB, infrared, and Bluetooth.

Cellebrite also distributes the UFED Report Manager, which provides an intuitive reporting interface and allows the user to export data/reports into Excel, MS Outlook, Outlook Express, and CSV or to simply print the report.

The UFED device fully supports Unicode and thus, can process phones with any language enabled. Also, the following data types are extracted:

- Phone Book
- Text Messages
- Call History (Received, Dialed, Missed)
- SIM ID Cloning
- Deleted Text Messages off SIM/USIM
- Audio Recordings
- Videos
- Pictures
- Phone Details (IMEI/ESN phone number)

#### Installation

The UFED system is a stand-alone unit and is packaged in a soft case containing the UFED device, user manual, software CD-ROM, USB Bluetooth radio (Cambridge

FIGURE 6.12

UFED instructions for Android device.

Silicon Radio Ltd), 250 MB USB drive, and roughly 72 cables for connecting to supported devices.

The UFED system provides several mechanisms by which the firmware and software can be updated. After setting the date and time, an examiner can simply connect the UFED system to the network via an Ethernet cable, provided DHCP and Internet access are available. Next, select Services, Upgrade, Upgrade Application Now, and select HTTP Server as the source. For this test, the latest Application software, version 1.1.0.5, was located and installed. As the UFED system is a stand-alone solution, no additional installs are necessary.

## Acquisition

The acquisition of the Motorola Droid was quite fast and simple on the UFED system. After powering the device on, select Extract Phone Data, Motorola CDMA, Moto. A855 Droid (Android), USB disk drive (destination), and the desired Content types. The following instructions were then displayed by the UFED system (see Fig. 6.12):

```
Moto. A855 Droid (Android):
Before starting the transaction, prepare phonebook for transfer, as follows:
1. Make sure SD card is inserted into the phone.
2. Go to "Contacts".
3. Press on "Menu" Key on the phone.
4. Select "Import/Export".
5. Select "Export to SD card" and "OK".
6. Wait for completing of Export to .vcf file.

To enable phone USB Connectivity, set Connection settings as follows:
Menu -> Settings -> Applications -> Development -> Select the checkboxes: "USB
debugging" and "Stay awake".
```

It should be noted that the contacts list will be saved to the SD card if the suggested steps are followed. After performing these steps, you hit Continue and the acquisition proceeds. The UFED system next prompts:

```
Set USB to Mass Storage (Memory Card) mode on the SOURCE phone
```

And after this step is completed, the acquisition proceeds. You may be prompted to set UMS again before the acquisition is complete. The acquisition process took just over three minutes and provided the following prompt:

```
Moto. A855 Droid (Android):
Please return the Connection settings back: Menu -> Settings -> Applications ->
Development -> unmark the checkboxes: "USB debugging" & "Stay awake"
```

The results from the acquisition were stored on the flash drive that was plugged into the UFED. A 25 MB folder was created on this drive with folders for videos, audio, and images. There were also three files of interest created: PhoneBook 2010_11_23 (001).htm, SMSMessages 2010_11_23 (001).htm, and Report.htm. All these can be viewed in a web browser. The file Report.htm contains the entire report of the extraction. This contains sections for Phone Examination Report Properties, Phone Examination Report Index, Phone Contacts, Phone SMS—Text Messages, Phone Incoming Calls List, Phone Outgoing Calls List, Phone Missed Calls List, Images, Ringtones, Audio, and Video.

The entire acquisition process took approximately ten minutes. After the acquisition was completed, a quick examination of the SD card revealed a file named 00001.vcf that contained the contact information from the export process.

Phone information was well laid out and was quite accurate.

### Data presentation and analysis

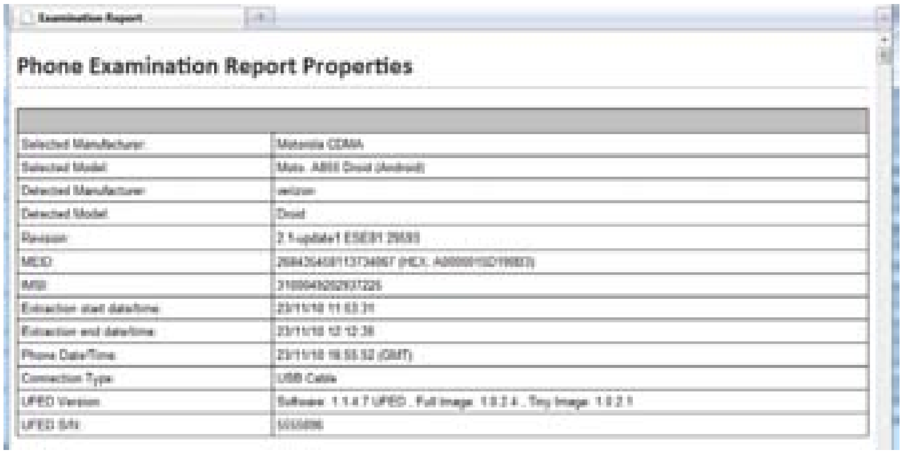
The acquisition data was stored on a flash drive connected to the UFED system and contained a folder that stored videos, audio files, and images as well as three HTML files, which contained the report data:

1. PhoneBook 2010_11_23 (001).htm
2. SMSMessages 2010_11_23 (001).htm
3. Report.htm

These files can be viewed in a web browser and samples from the report are displayed in Fig. 6.13, in which thorough phone information was captured.

Fig. 6.14 shows how the Phone Contacts are laid out.

Fig. 6.15 shows text messages displayed chronologically with detailed information on whether the message was sent or received.



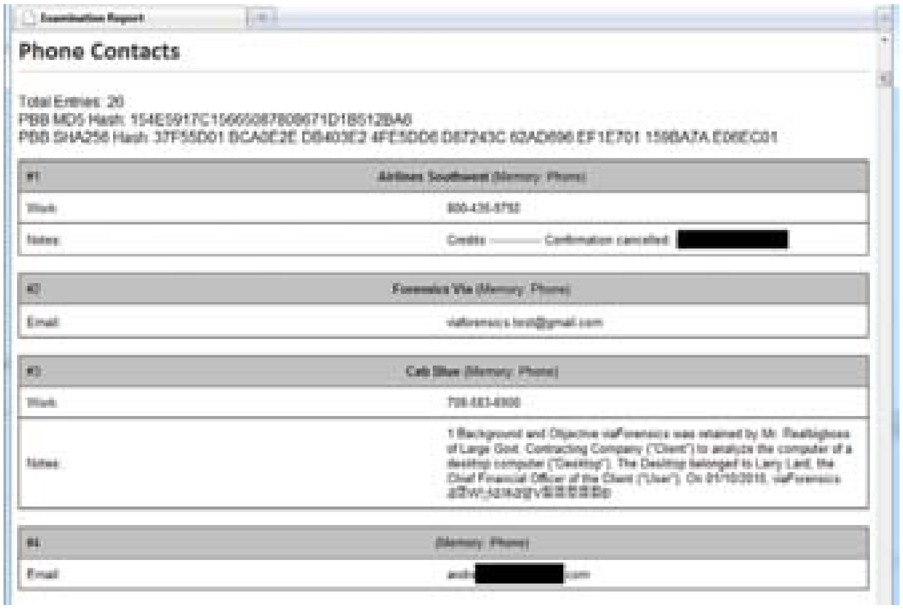
Examination Report

Phone Examination Report Properties

| | |
|---|---|
| Selected Manufacturer | Motorola CDMA |
| Selected Model | Moto. A855 Droid (Android) |
| Detected Manufacturer | verizon |
| Detected Model | Droid |
| Revision | 2.1-update1 ESE81 29593 |
| MEID | 268435458113734867 (HEX: A000001SD19887) |
| IMSI | 310004202937225 |
| Extraction start date/time | 23/11/10 11:03:31 |
| Extraction end date/time | 23/11/10 12:12:38 |
| Phone Date/Time | 23/11/10 18:53:32 (GMT) |
| Connection Type | USB Cable |
| UFED Version | Software: 1.1.4.7 UFED , Full image: 1.0.2.4 , Tiny image: 1.0.2.1 |
| UFED S/N | 5555896 |

**FIGURE 6.13**

UFED phone information reporting.



Examination Report

Phone Contacts

Total Entries: 20
PBB MD5 Hash: 154E5917C1566508780B671D1B512BA6
PBB SHA256 Hash: 37F55D01 BCA0E2E DB403E2 4FE5DD6 D87243C 62AD696 EF1E701 159BA7A E06EC01

| #1 | Airlines Southwest (Memory: Phone) |
|---|---|
| Work | 800-435-9792 |
| Notes | Credits ———— Confirmation cancelled: [redacted] |

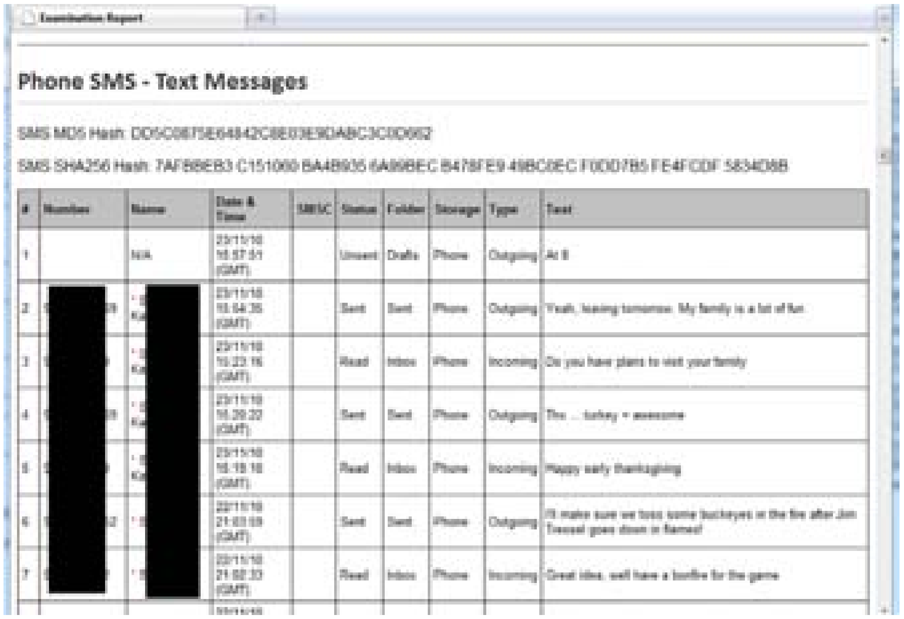
| #2 | Forensics Via (Memory: Phone) |
|---|---|
| Email | viaforensics.test@gmail.com |

| #3 | Cab Blue (Memory: Phone) |
|---|---|
| Work | 708-583-6300 |
| Notes | 1 Background and Objective viaForensics was retained by Mr. Realbigboss of Large Govt. Contracting Company ("Client") to analyze the computer of a desktop computer ("Desktop"). The Desktop belonged to Larry Lard, the Chief Financial Officer of the Client ("User"). On 01/10/2010, viaForensics [illegible] |

| #4 | (Memory: Phone) |
|---|---|
| Email | andr[redacted].com |

**FIGURE 6.14**

UFED phone contacts reporting.



Examination Report

Phone SMS - Text Messages

SMS MD5 Hash: DD5C0875E64842C8E03E9DABC3C0D992

SMS SHA256 Hash: 7AF88EB3 C151060 BA4B935 6A89BEC 8478FE9 498C0EC F0DD7B5 FE4FCDF 5834D8B

| # | Number | Name | Date & Time | SMSC | Status | Folder | Storage | Type | Text |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | N/A | 23/11/10 15:57:51 (GMT) | | Unsent | Drafts | Phone | Outgoing | At 8 |
| 2 | [redacted] | [redacted] | 23/11/10 15:54:35 (GMT) | | Sent | Sent | Phone | Outgoing | Yeah, leaving tomorrow. My family is a lot of fun. |
| 3 | [redacted] | [redacted] | 23/11/10 15:23:16 (GMT) | | Read | Inbox | Phone | Incoming | Do you have plans to visit your family |
| 4 | [redacted] | [redacted] | 23/11/10 15:20:22 (GMT) | | Sent | Sent | Phone | Outgoing | Thx ... turkey + awesome |
| 5 | [redacted] | [redacted] | 23/11/10 15:18:18 (GMT) | | Read | Inbox | Phone | Incoming | Happy early thanksgiving |
| 6 | [redacted] | [redacted] | 22/11/10 21:03:19 (GMT) | | Sent | Sent | Phone | Outgoing | I'll make sure we toss some buckeyes in the fire after Jim Tressel goes down in flames! |
| 7 | [redacted] | [redacted] | 22/11/10 21:02:33 (GMT) | | Read | Inbox | Phone | Incoming | Great idea, we'll have a bonfire for the game |

**FIGURE 6.15**

UFED SMS reporting.

However, deleted text messages are not displayed, nor are MMS messages.

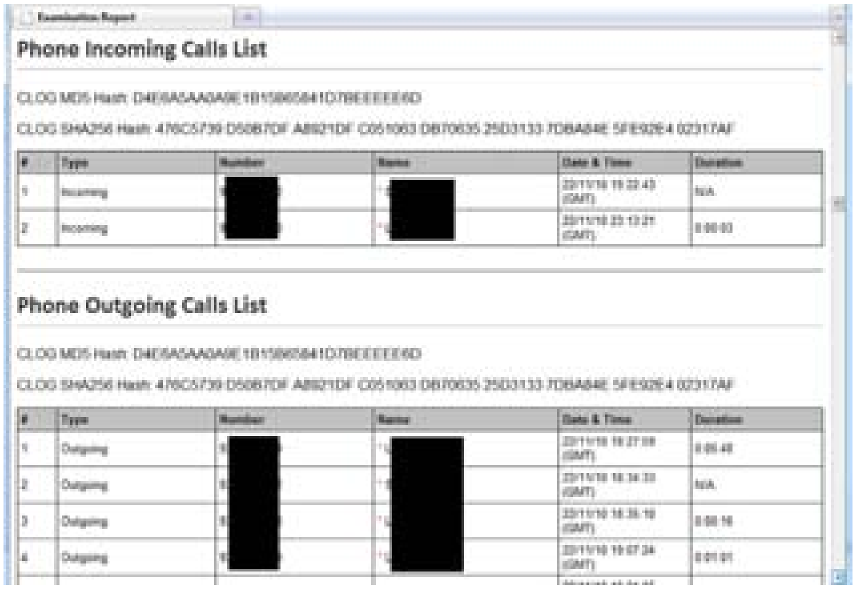
Call logs are displayed chronologically as shown in Fig. 6.16, and include the length of the call. They are categorized into Incoming, Outgoing, and Missed sections.

Several calls were deleted from the call logs; however, UFED was able to extract and display the details.

All of the images found on the phone are reported, along with a thumbnail of the image and details, including file name, size, date and time created, and resolution, as shown in Fig. 6.17.

Deleted images did not appear, and it seemed as though a duplicate of each image was created. Both audio and video files are reported. The report includes file name, file size, date and time created, and a link to view or listen to the media, as shown in Fig. 6.18.

No deleted videos were returned and songs uploaded to the device did not appear in the report. The audio files that were reported were returned from Google Maps Navigation.



Examination Report

Phone Incoming Calls List

CLOG MD5 Hash: D4E6A5AA0A9E 1B15B65841D7BEEEEE6D

CLOG SHA256 Hash: 476C5739 D50B7DF A8921DF C051063 DB70635 25D3133 7DBA84E 5FE92E4 02317AF

| # | Type | Number | Name | Date & Time | Duration |
|---|---|---|---|---|---|
| 1 | Incoming | [redacted] | [redacted] | 22/11/10 19:22:43 (GMT) | N/A |
| 2 | Incoming | [redacted] | [redacted] | 22/11/10 23:13:21 (GMT) | 0:00:03 |

Phone Outgoing Calls List

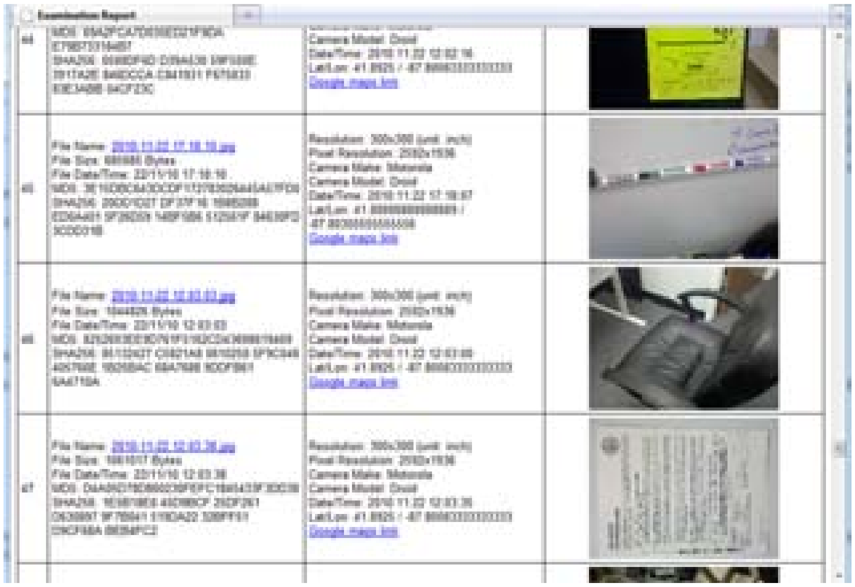
CLOG MD5 Hash: D4E6A5AA0A9E 1B15B65841D7BEEEEE6D

CLOG SHA256 Hash: 476C5739 D50B7DF A8921DF C051063 DB70635 25D3133 7DBA84E 5FE92E4 02317AF

| # | Type | Number | Name | Date & Time | Duration |
|---|---|---|---|---|---|
| 1 | Outgoing | [redacted] | [redacted] | 22/11/10 19:27:09 (GMT) | 0:05:48 |
| 2 | Outgoing | [redacted] | [redacted] | 22/11/10 18:34:33 (GMT) | N/A |
| 3 | Outgoing | [redacted] | [redacted] | 22/11/10 18:35:10 (GMT) | 0:00:16 |
| 4 | Outgoing | [redacted] | [redacted] | 22/11/10 19:07:24 (GMT) | 0:01:01 |

**FIGURE 6.16**

UFED phone calls reporting.

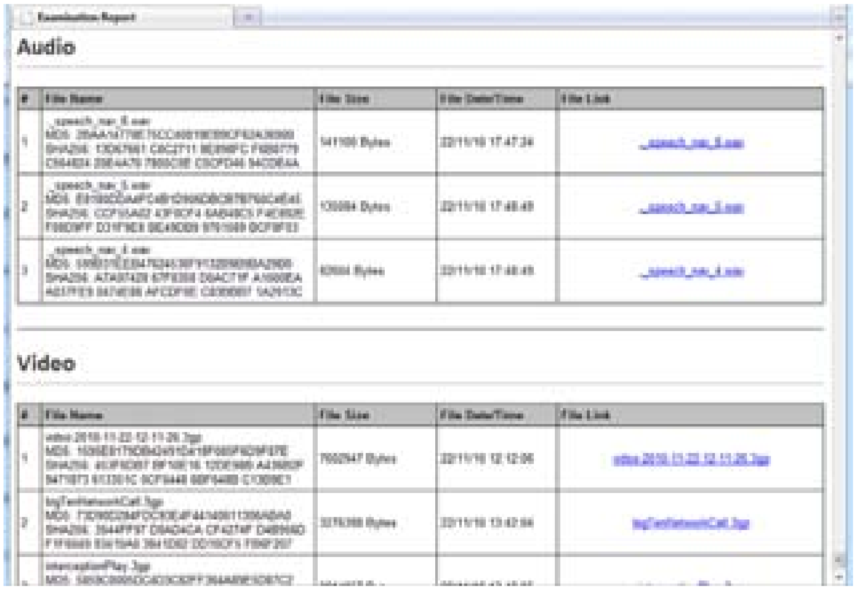


Examination Report

| | | | |
|---|---|---|---|
| 44 | MD5: E9A2FCA7D[illegible] E79B7[illegible] SHA256: [illegible] | Camera Model: Droid Date/Time: 2010.11.22 12:02:16 Lat/Lon: 41.8925 / -87.8666333333333 Google maps link | |
| 45 | File Name: 2010.11.22 17.18.18.jpg File Size: 681685 Bytes File Date/Time: 22/11/10 17:18:18 MD5: [illegible] SHA256: [illegible] | Resolution: 300x300 (unit: inch) Pixel Resolution: 2592x1936 Camera Make: Motorola Camera Model: Droid Date/Time: 2010.11.22 17:18:07 Lat/Lon: 41.8888888888889 / -87.8638888888888 Google maps link | |
| 46 | File Name: 2010.11.22 12.03.03.jpg File Size: 1044826 Bytes File Date/Time: 22/11/10 12:03:03 MD5: [illegible] SHA256: [illegible] | Resolution: 300x300 (unit: inch) Pixel Resolution: 2592x1936 Camera Make: Motorola Camera Model: Droid Date/Time: 2010.11.22 12:03:00 Lat/Lon: 41.8925 / -87.8666333333333 Google maps link | |
| 47 | File Name: 2010.11.22 12.03.36.jpg File Size: 1061017 Bytes File Date/Time: 22/11/10 12:03:36 MD5: [illegible] SHA256: [illegible] | Resolution: 300x300 (unit: inch) Pixel Resolution: 2592x1936 Camera Make: Motorola Camera Model: Droid Date/Time: 2010.11.22 12:03:35 Lat/Lon: 41.8925 / -87.8666333333333 Google maps link | |

**FIGURE 6.17**

UFED images reporting.



Audio

| # | File Name | File Size | File Date/Time | File Link |
|---|---|---|---|---|
| 1 | _speech_nar_6.ear MD5: [illegible] SHA256: [illegible] | 141100 Bytes | 22/11/10 17:47:24 | _speech_nar_6.ear |
| 2 | _speech_nar_5.ear MD5: [illegible] SHA256: [illegible] | 135084 Bytes | 22/11/10 17:48:48 | _speech_nar_5.ear |
| 3 | _speech_nar_4.ear MD5: [illegible] SHA256: [illegible] | 83004 Bytes | 22/11/10 17:48:48 | _speech_nar_4.ear |

Video

| # | File Name | File Size | File Date/Time | File Link |
|---|---|---|---|---|
| 1 | video 2010-11-22-12-11-26.3gp MD5: [illegible] SHA256: [illegible] | 7002547 Bytes | 22/11/10 12:12:06 | video 2010-11-22-12-11-26.3gp |
| 2 | bigTenNetworkCall.3gp MD5: [illegible] SHA256: [illegible] | 3276388 Bytes | 22/11/10 13:42:04 | bigTenNetworkCall.3gp |
| [illegible] | interceptionPlay.3gp MD5: [illegible] | [illegible] | [illegible] | [illegible] |

**FIGURE 6.18**

UFED audio and video reporting.

### *Compelson MOBILedit!*

The following overview of MOBILedit! was provided by the vendor:

With just a single click, MOBILedit! Forensic collects all possible data from mobile phones and generates extensive reports onto a PC that can be stored or printed. It is the most universal mobile phone solution with software supporting most GSM phones and open architecture allowing the support of any phone. The system allows you to customize the output making it completely adaptable to the needs of your judicial system.

MOBILedit! Forensic does a complete analysis of the phone including its phone book, last dialed numbers, missed calls, received calls, MMS messages, SMS messages, photos, videos, files, phone details, calendar, notes, tasks, and much more.

MOBILedit! Forensic caters to the entire world with reports that can be generated in any language. You are able to prepare creative templates according to your specific needs. You construct all the text that you would like to see appear in every final report. It also allows for XML export, so that you can connect the application with other systems. The XSL module exports and nicely formats all data in the package to an Internet browser. You can burn, send, and share the report as needed.

MOBILedit! Forensic reports can be created without the touch of a human hand. While there is no need to import or export stubs of data from SIMs or phones, it is possible in manual investigation mode in MOBILedit! Forensic. It is read-only and hence, it prevents changes in the device, which could mean the disappearance of evidence. All items are also protected against later modifications by MD5 hash codes used in digital signatures. It helps you to quickly locate the possible place of modification.

MOBILedit! Forensic also has frequent updates and upgrades so that you can be sure you are using the absolute latest in technology. Its detailed reports and user-friendly design make it a pleasure to work with.

### Installation

The MOBILedit!4 Forensic application was downloaded from www.mobiledit.com and the install only took a few minutes. After the installation is completed and the application is run for the first time, you are presented with a prompt to check for updates.

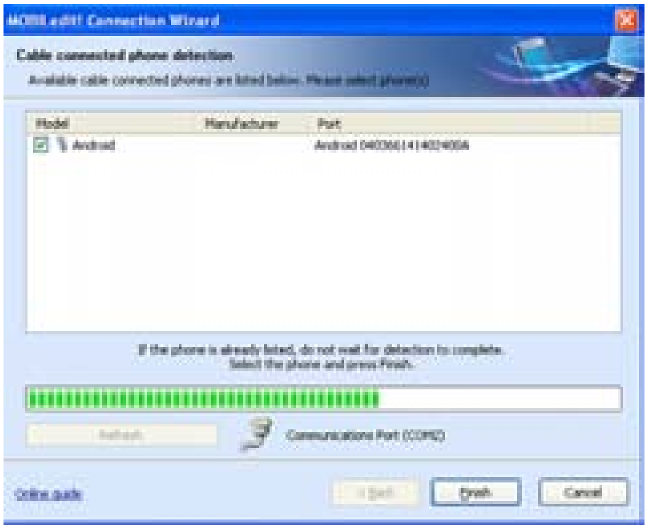
To activate the software, Compelson sends an e-mail with an "activation card" attachment. This PDF file includes installation instructions as well as an activation key that worked without any issues.

### Acquisition

To begin the acquisition, the examiner must first connect the Android device to the forensic workstation using USB and ensure USB debugging is enabled. MOBILedit! attempts to detect the device as shown in Fig. 6.19.

After clicking "Finish," there was a notification prompting the installation of the "Connector" app on the device, shown in Fig. 6.20.

Following the quick installation, you create a name for the investigation and select the type of data you want to extract. In the example shown in Fig. 6.21, the


MOBILedit! Connection Wizard
Cable connected phone detection
Refresh
Communications Port (COM2)
Online guide
< Back
Finish
Cancel

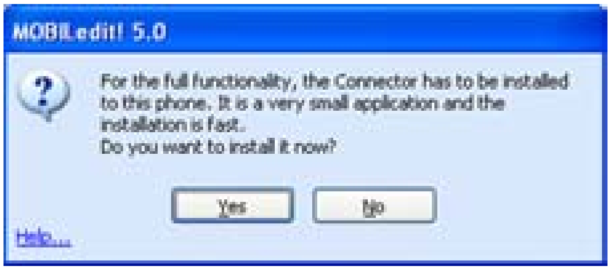
Available cable connected phones are listed below. Please select phone(s)

| Model | Manufacturer | Port |
|---|---|---|
| ✓ Android | | Android 040366141402400A |

If the phone is already listed, do not wait for detection to complete.
Select the phone and press Finish.

FIGURE 6.19

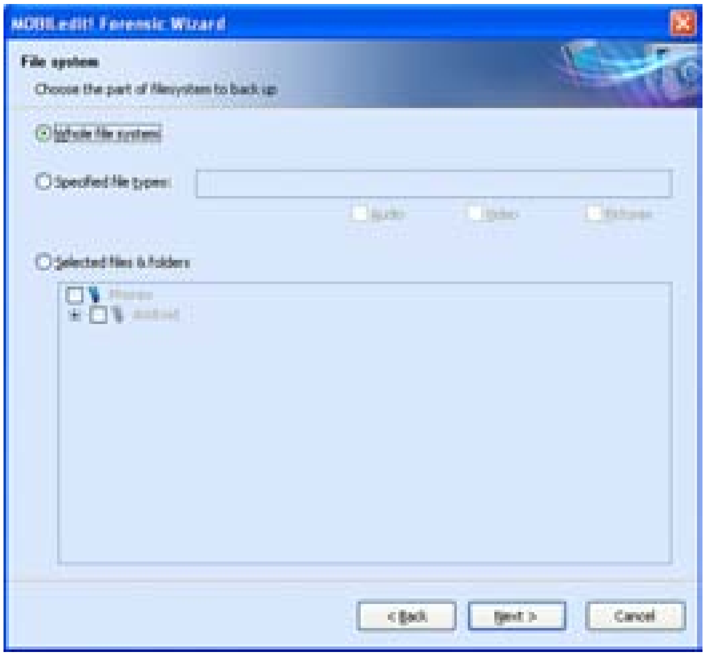
detect Android device.


MOBILedit! 5.0

For the full functionality, the Connector has to be installed to this phone. It is a very small application and the installation is fast.
Do you want to install it now?

**FIGURE 6.20**

Install the connector.


MOBILedit! Forensic Wizard

**File system**

Choose the part of filesystem to back up

[✓ Whole file system]

Specified file types:

Audio Video Pictures

Selected files & folders

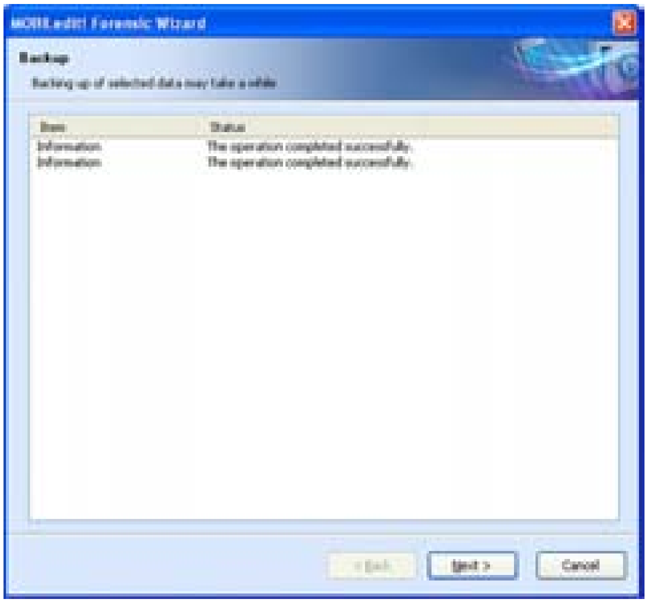
Phone
Android

**FIGURE 6.21**

Take backup of the whole file system.

option to take a backup of the "Whole file system" was selected, which then executed without error and presented a success status as illustrated in Fig. 6.22.

You can then decide if you want to add this to an already existing case or create a new one. For this example, shown in Fig. 6.23, a new case was created and a data export format option of XLS was selected.



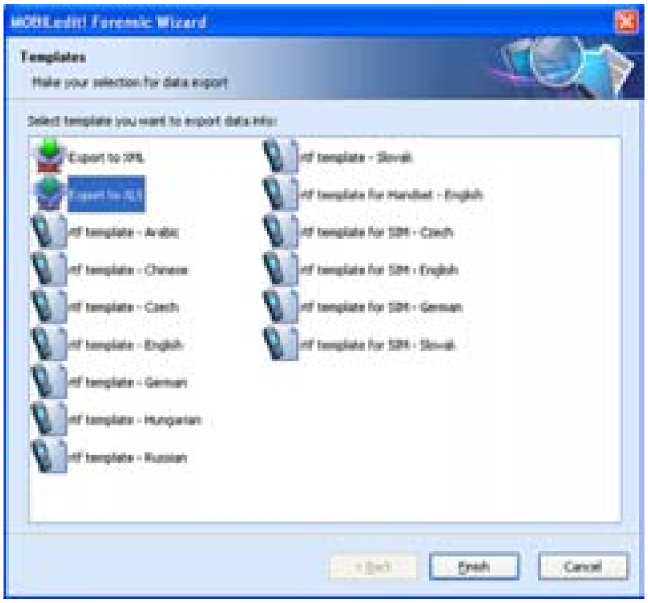
Backup

Backing up of selected data may take a while

| Item | Status |
|---|---|
| Information | The operation completed successfully. |
| Information | The operation completed successfully. |

**FIGURE 6.22**

Operation completed successfully.



Templates

Make your selection for data export

Select template you want to export data into:

- Export to XML
- Export to XLS
- rtf template - Arabic
- rtf template - Chinese
- rtf template - Czech
- rtf template - English
- rtf template - German
- rtf template - Hungarian
- rtf template - Russian
- rtf template - Slovak
- rtf template for Handset - English
- rtf template for SIM - Czech
- rtf template for SIM - English
- rtf template for SIM - German
- rtf template for SIM - Slovak

**FIGURE 6.23**

Data export format.

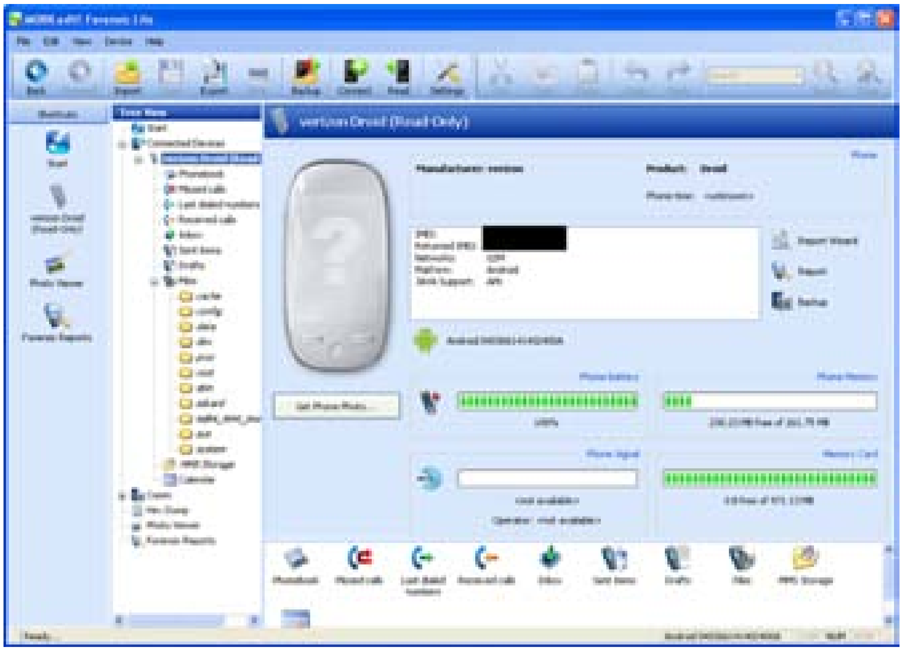


**FIGURE 6.24**

Main screen.

## Data presentation and analysis

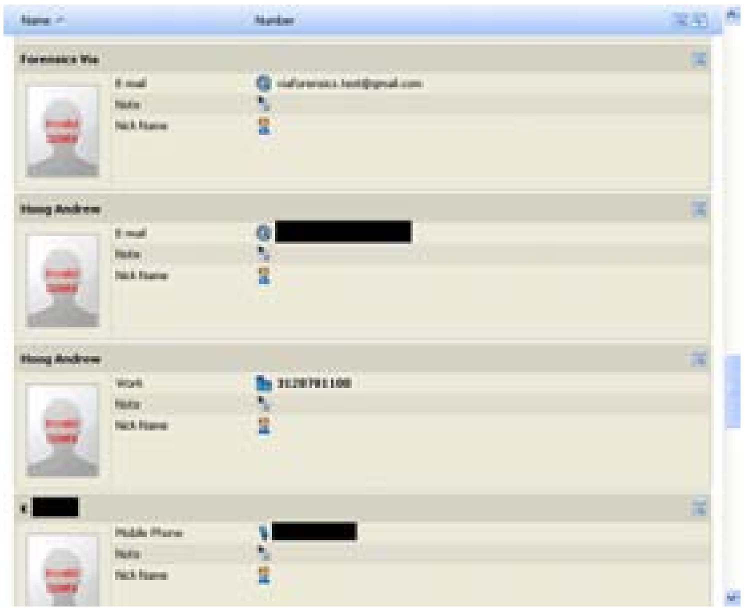
Immediately following the acquisition of the device, MOBILedit displays statistics on the devices that were acquired, as well as a view of the application data available for analysis. Fig. 6.24 shows the main screen where the examiner can see specific device information including the IMEI number, serial number, and details on the amount of Phone memory, Battery signal, Network signal, and Memory card space available on the device.

The next option in the Tree View is the Phonebook where the examiner can view all contacts stored within the Phonebook including e-mail address, phone numbers, nicknames, and any notes entered regarding the contact, as shown in Fig. 6.25.

Call logs are next and are separated into Missed calls, Last dialed numbers, and Received calls as shown in Fig. 6.26.

SMS messages are similarly separated into categories including Inbox, Sent items, and Drafts. Each section contains the date and time the message was received (or sent), the message content, and who the message was from. Contact names are linked to the Phonebook, so both name and phone numbers are displayed. Fig. 6.27 shows the SMS message inbox.

Any MMS messages are displayed within the "MMS Storage" folder, shown in Fig. 6.28. On the left-hand side, information about the message is displayed,


Name Number

Forensics Wu
E-mail viaforensics.test@gmail.com
Note
Nick Name

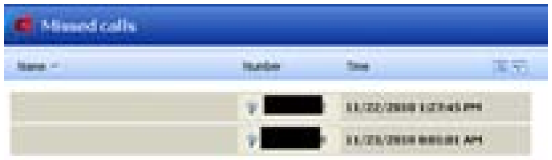
Hoog Andrew
E-mail [redacted]
Note
Nick Name

Hoog Andrew
Work 3128781100
Note
Nick Name

[redacted]
Mobile Phone [redacted]
Note
Nick Name

**FIGURE 6.25**

Phonebook.


Missed calls

| Name | Number | Time |
|---|---|---|
| | [redacted] | 11/22/2010 1:27:45 PM |
| | [redacted] | 11/21/2010 8:03:01 AM |

**FIGURE 6.26**

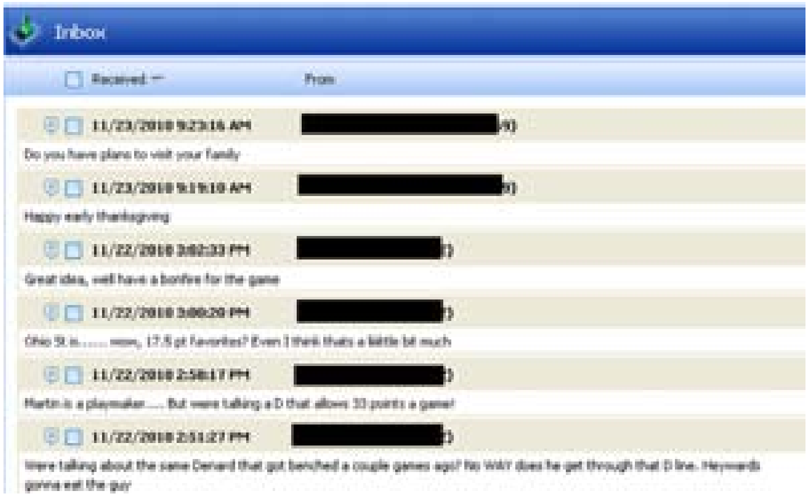
Call logs—Missed calls.

including the subject, number it was sent from, number it was sent to, and date and time. On the right-hand side is a preview of the actual image.

Selecting the Calendar option will literally pull up a calendar within the reporting tool as shown in Fig. 6.29.

Additional data extracted from the device or SD card is shown within the “files” directory. This directory contains a listing of the file system on the device. While some of these folders are empty (such as cache, config, and data), there are also some folders which contain raw files acquired from the device. For example, within the SD card folder, the subfolder “secret stuff” contained two files shown in Fig. 6.30.

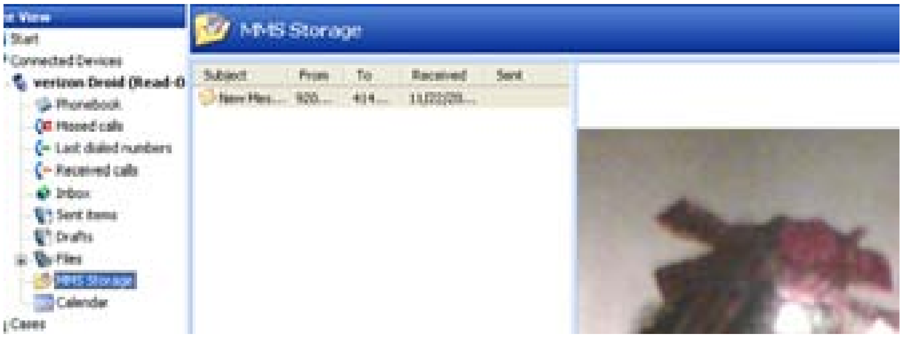
Finally, the tool also provides a hex dump capability for specified files. After selecting “Hex Dump,” and then a file (in this example, a .jpg file was selected),



| Received | From |
|---|---|
| 11/23/2010 9:23:16 AM | [redacted](9) |
| Do you have plans to visit your family | |
| 11/23/2010 9:19:10 AM | [redacted]9) |
| Happy early thanksgiving | |
| 11/22/2010 3:02:33 PM | [redacted]) |
| Great idea, well have a bonfire for the game | |
| 11/22/2010 3:00:20 PM | [redacted]) |
| Ohio St is....... wow, 17.5 pt favorites? Even I think thats a little bit much | |
| 11/22/2010 2:58:17 PM | [redacted]) |
| Martin is a playmaker..... But were talking a D that allows 33 points a game! | |
| 11/22/2010 2:51:27 PM | [redacted]) |
| Were talking about the same Denard that got benched a couple games ago? No WAY does he get through that D line. Heywards gonna eat the guy | |

**FIGURE 6.27**

SMS messages—Inbox.



| Subject | From | To | Received | Sent |
|---|---|---|---|---|
| New Mes... | 920... | 414... | 11/22/20... | |

**FIGURE 6.28**

MMS storage.

the hex dump is viewed on the right-hand side using a hex editor as shown in Fig. 6.31.

Most of the raw user data files on the Android can be found within the “data” folder and, when MOBILedit created an entry in the Tree View for this folder (under the “Files” directory), it did not contain any files.

One thing to note is that when the acquisition and analysis was complete, the “MOBILedit! Connector” application was still installed on the device. Examiners should strongly consider manually uninstalling the software from the device after the investigation is complete.

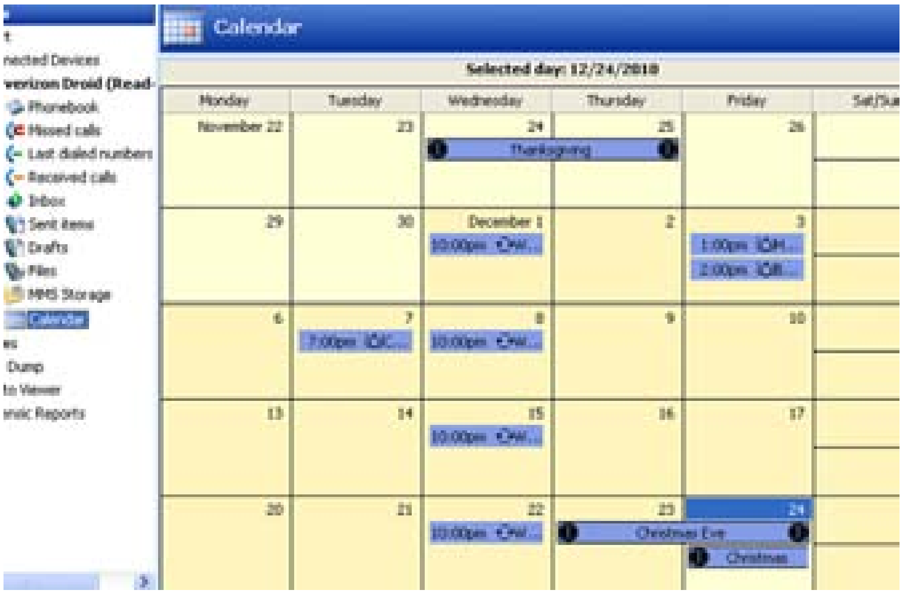


FIGURE 6.29

Calendar.

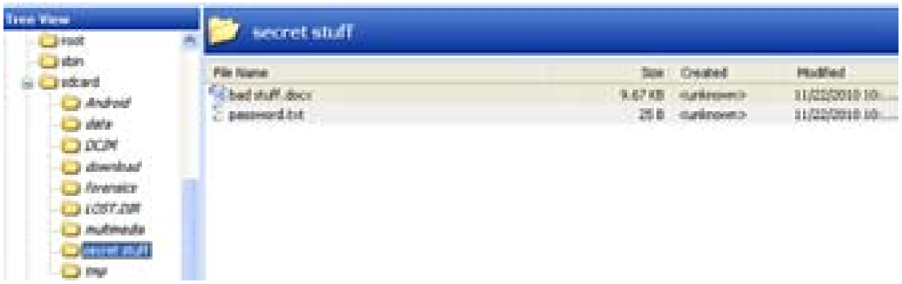


FIGURE 6.30

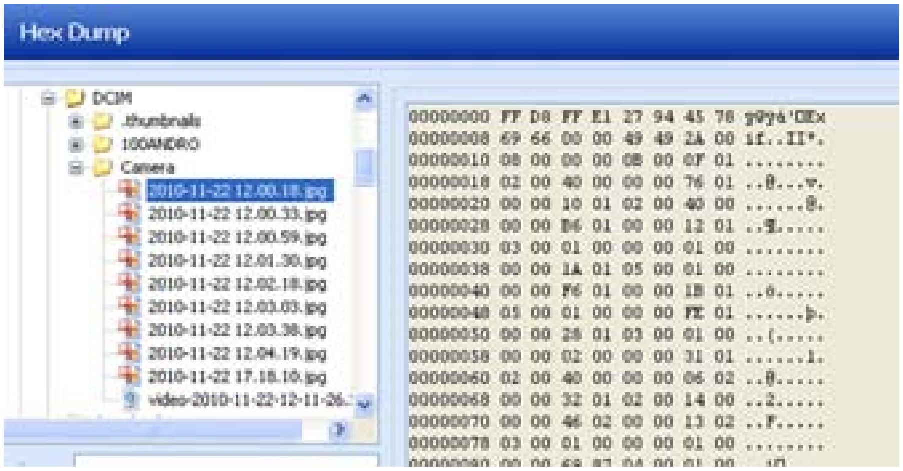
SD card files.

## *EnCase Neutrino*

The following overview of Neutrino was provided by the vendor:

EnCase® Neutrino® is designed for law enforcement, security analysts, and eDiscovery specialists who need to forensically collect and review data from mobile devices. Investigators can process and analyze mobile device data alongside other types of digital evidence within any EnCase® product.

The solution features hardware support and parsing capabilities for the most common mobile devices and Smartphone operating systems including iPhone, Palm,



```
00000000 FF D8 FF E1 27 94 45 78 ÿØÿá'”Ex
00000008 69 66 00 00 49 49 2A 00 if..II*.
00000010 08 00 00 00 0B 00 0F 01 ........
00000018 02 00 40 00 00 00 76 01 ..@...v.
00000020 00 00 10 01 02 00 40 00 ......@.
00000028 00 00 B6 01 00 00 12 01 ..¶.....
00000030 03 00 01 00 00 00 01 00 ........
00000038 00 00 1A 01 05 00 01 00 ........
00000040 00 00 F6 01 00 00 1B 01 ..ö.....
00000048 05 00 01 00 00 00 FE 01 ......þ.
00000050 00 00 28 01 03 00 01 00 ..(.....
00000058 00 00 02 00 00 00 31 01 ......1.
00000060 02 00 40 00 00 00 06 02 ..@.....
00000068 00 00 32 01 02 00 14 00 ..2.....
00000070 00 00 46 02 00 00 13 02 ..F.....
00000078 03 00 01 00 00 00 01 00 ........
```

**FIGURE 6.31**

Hex dump.

BlackBerry®, Android, Windows Mobile, Motorola, Nokia, Samsung, and many more. Investigators can collect, analyze, and preserve all potentially relevant data including:

- Device Settings
- Contacts
- Call logs
- E-mail
- Images
- SMS/MMS
- Calendars
- Other files stored on the device

With EnCase Neutrino an investigator can:

- Collect data from a wide variety of devices, following an easy to use wizard
- Correlate data from multiple devices and computer media
- Seamlessly integrate collected data into EnCase Forensic or EnCase Enterprise for analysis
- Parse data quickly to improve speed of investigation process
- Access more data on selected devices in comparison to similar products

## Installation

The installation of EnCase Neutrino first required installing EnCase, then Neutrino. The software installation proceeded without issue when following the on-screen instructions. To use the software, you must have a hardware USB dongle provided by EnCase.

### Acquisition

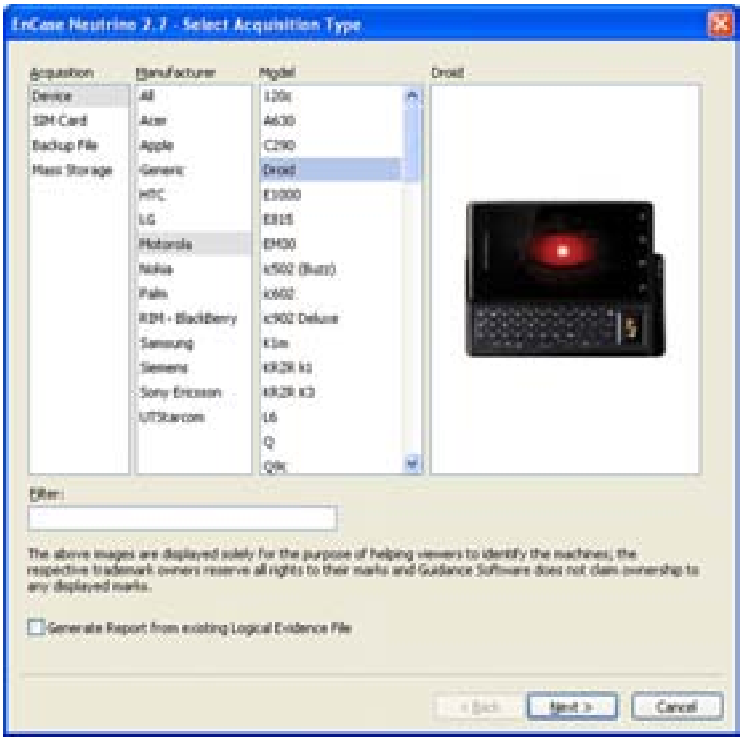
The acquisition of an Android device by Neutrino is handled in a single screen. First, you select the device, manufacturer, and model. Next, you enter basic information about the case and the device. Finally, you connect the device to your forensic workstation and click "Acquire Current Item." Fig. 6.32 shows the acquisition screen.

The acquisition took less than one minute. Once completed, click Generate Report to see the results of the acquisition.

### Data presentation and analysis

Neutrino reports have a short and a detailed report. The short report, shown in Fig. 6.33, shows all of the entries of the detailed report but with fewer details.

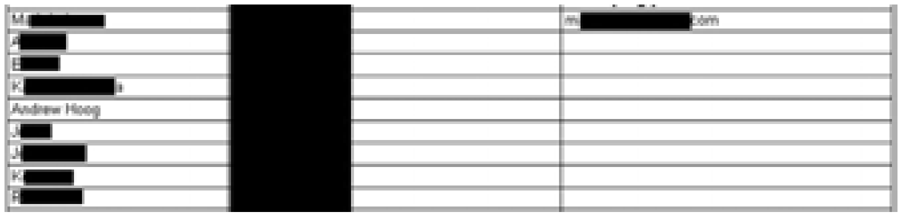
And the detailed view provides much greater detail about the contacts as shown in Fig. 6.34.



The above images are displayed solely for the purpose of helping viewers to identify the machines; the respective trademark owners reserve all rights to their marks and Guidance Software does not claim ownership to any displayed marks.

**FIGURE 6.32**

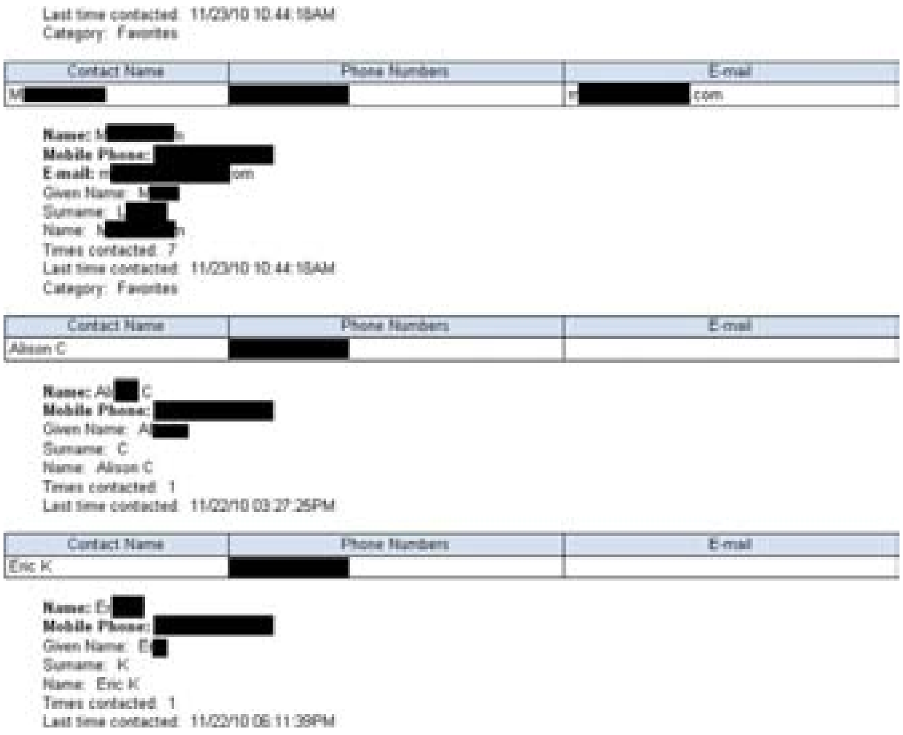
Neutrino acquisition screen.



| | | | |
|---|---|---|---|
| M[redacted] | [redacted] | | m[redacted]com |
| A[redacted] | [redacted] | | |
| B[redacted] | [redacted] | | |
| K[redacted]a | [redacted] | | |
| Andrew Hoog | [redacted] | | |
| J[redacted] | [redacted] | | |
| J[redacted] | [redacted] | | |
| K[redacted] | [redacted] | | |
| R[redacted] | [redacted] | | |

**FIGURE 6.33**

Neutrino contacts in short report.



Last time contacted: 11/23/10 10:44:18AM
Category: Favorites

| Contact Name | Phone Numbers | E-mail |
|---|---|---|
| M[redacted] | [redacted] | m[redacted]com |

**Name:** M[redacted]n
**Mobile Phone:** [redacted]
**E-mail:** m[redacted]om
Given Name: M[redacted]
Surname: L[redacted]
Name: N[redacted]n
Times contacted: 7
Last time contacted: 11/23/10 10:44:18AM
Category: Favorites

| Contact Name | Phone Numbers | E-mail |
|---|---|---|
| Alison C | [redacted] | |

**Name:** Ali[redacted] C
**Mobile Phone:** [redacted]
Given Name: Al[redacted]
Surname: C
Name: Alison C
Times contacted: 1
Last time contacted: 11/22/10 03:27:26PM

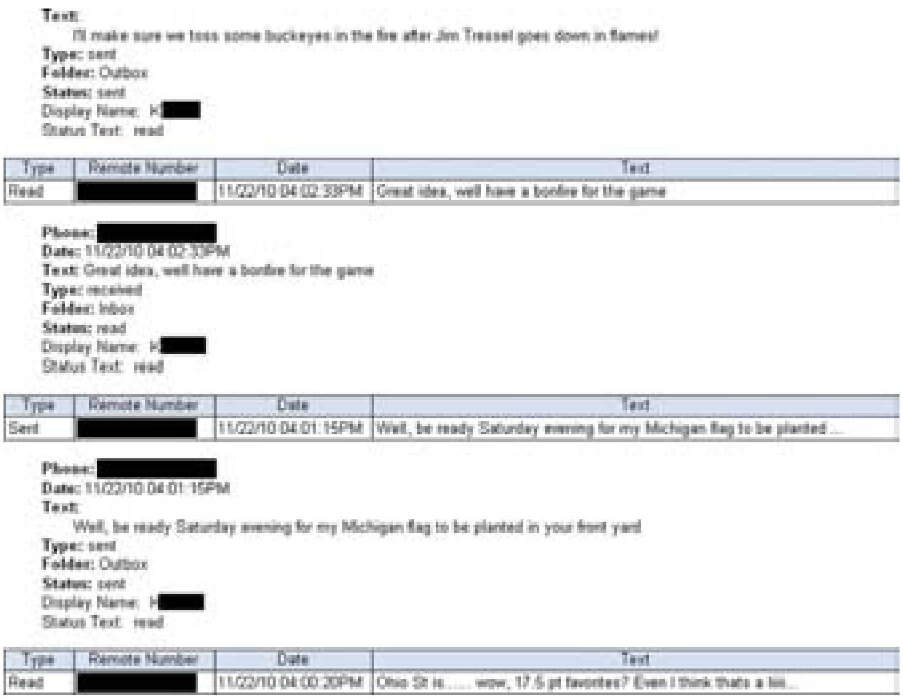
| Contact Name | Phone Numbers | E-mail |
|---|---|---|
| Eric K | [redacted] | |

**Name:** Er[redacted]
**Mobile Phone:** [redacted]
Given Name: Er[redacted]
Surname: K
Name: Eric K
Times contacted: 1
Last time contacted: 11/22/10 06:11:39PM

**FIGURE 6.34**

Neutrino contacts in detailed report.

The SMS section also has a short and detailed view. The short view only shows the other phone number in the conversation, the date and time of the message, the message direction (sent or received), and the message contents. The detailed view, shown in Fig. 6.35, includes information such as the name associated with the phone number involved with the message and the status of the message.



**Text:**
I'll make sure we toss some buckeyes in the fire after Jim Tressel goes down in flames!
**Type:** sent
**Folder:** Outbox
**Status:** sent
Display Name: K[redacted]
Status Text: read

| Type | Remote Number | Date | Text |
|---|---|---|---|
| Read | [redacted] | 11/22/10 04:02:33PM | Great idea, well have a bonfire for the game |

**Phone:** [redacted]
**Date:** 11/22/10 04:02:33PM
**Text:** Great idea, well have a bonfire for the game
**Type:** received
**Folder:** Inbox
**Status:** read
Display Name: K[redacted]
Status Text: read

| Type | Remote Number | Date | Text |
|---|---|---|---|
| Sent | [redacted] | 11/22/10 04:01:15PM | Well, be ready Saturday evening for my Michigan flag to be planted ... |

**Phone:** [redacted]
**Date:** 11/22/10 04:01:15PM
**Text:**
Well, be ready Saturday evening for my Michigan flag to be planted in your front yard
**Type:** sent
**Folder:** Outbox
**Status:** sent
Display Name: K[redacted]
Status Text: read

| Type | Remote Number | Date | Text |
|---|---|---|---|
| Read | [redacted] | 11/22/10 04:00:20PM | Ohio St is...... wow, 17.5 pt favorites? Even I think thats a litt... |

**FIGURE 6.35**

Neutrino SMS in detailed report.

MMS messages only appear in the detailed view, shown in Fig. 6.36.

No deleted SMS messages were recovered. The detailed view of the user's web history, shown in Fig. 6.37, provides considerable details.

The only photos recovered from the device were the images sent as MMS, not the photos or videos saved on the device's SD card.

The report can be exported in other formats, such as to HTML, so that the entire report can be viewed in one page, as illustrated in Fig. 6.38

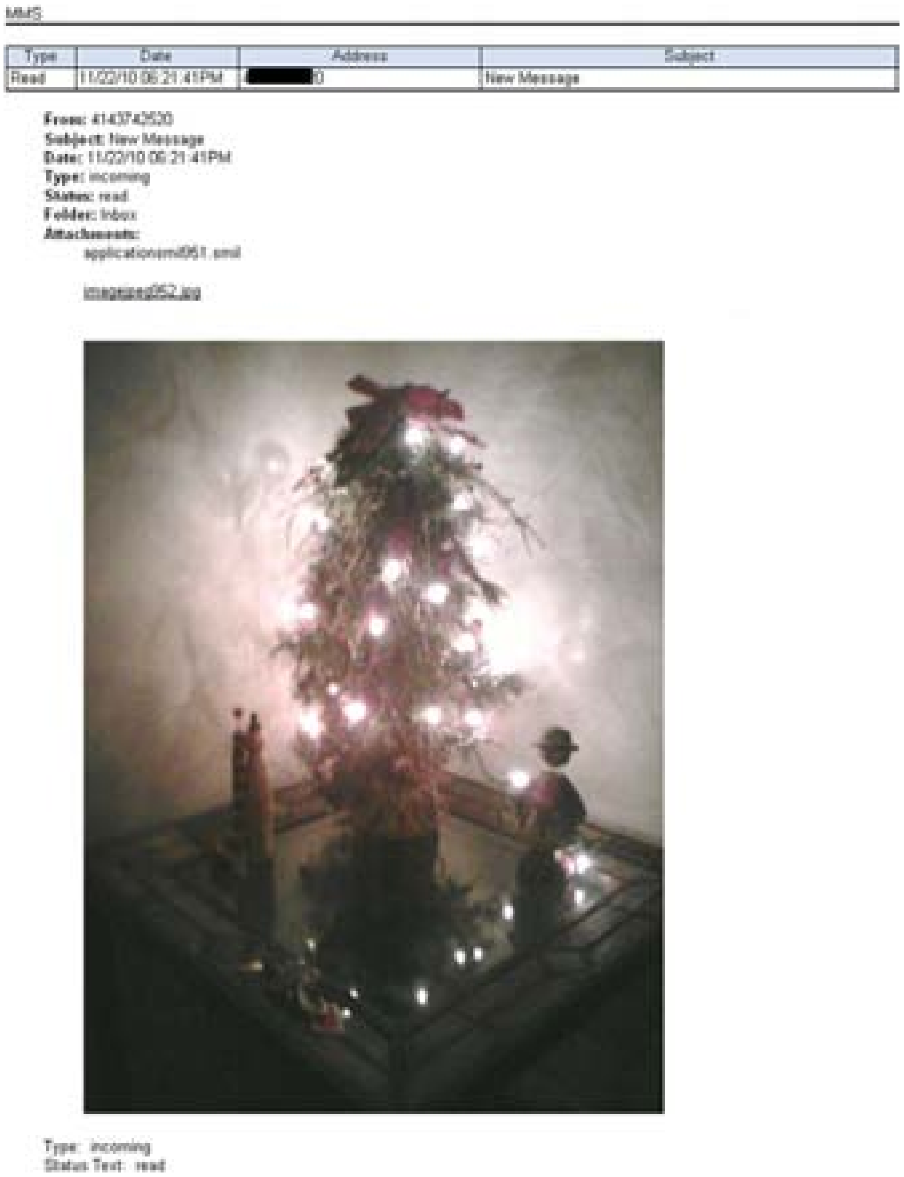
## *Micro Systemation XRY*

The following overview of XRY was provided by the vendor:

XRY is a dedicated mobile device forensic tool developed by Micro Systemation (MSAB) based in Stockholm.

XRY has been available since 2002 and "XRY Complete" is a package containing the software and hardware to allow logical and physical analysis of mobile devices. The product comes shipped in a handy portable case with bespoke interior and all the necessary hardware included, which are as follows:

- XRY Forensic Pack Software License Key
- Communication hub for USB, Bluetooth, and Infrared connectivity



MMS

| Type | Date | Address | Subject |
|---|---|---|---|
| Read | 11/22/10 06:21:41PM | 4[redacted]0 | New Message |

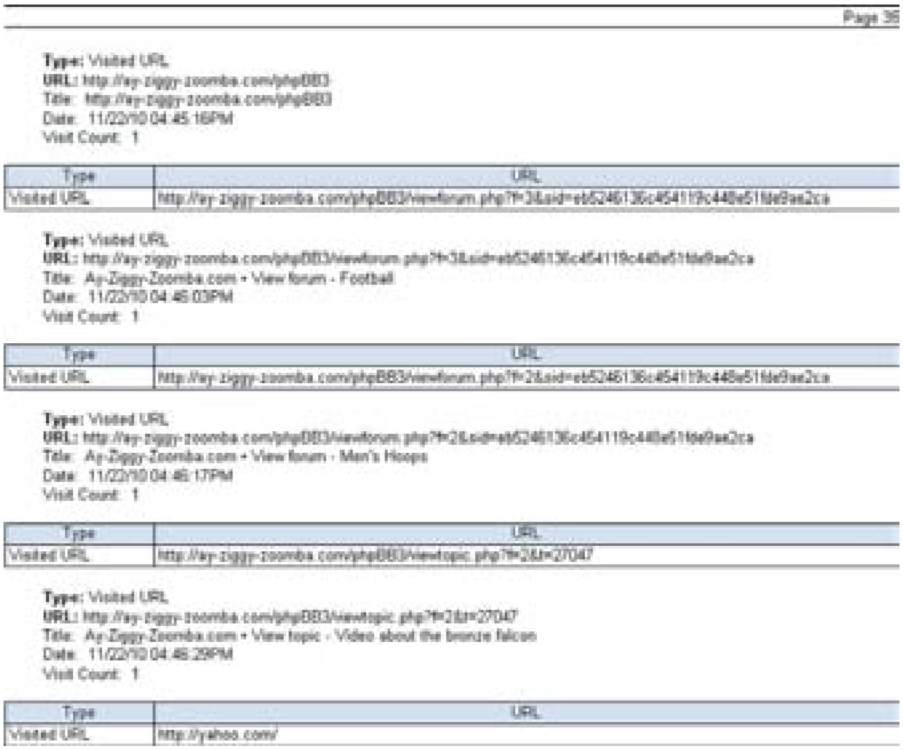
**From:** 4143742520
**Subject:** New Message
**Date:** 11/22/10 06:21:41PM
**Type:** incoming
**Status:** read
**Folder:** Inbox
**Attachments:**
applicationsmil061.smil

imagejpeg062.jpg

Type: incoming
Status Text: read

**FIGURE 6.36**

Neutrino MMS in detailed report.



Page 36

**Type:** Visited URL
**URL:** http://ay-ziggy-zoomba.com/phpBB3
Title: http://ay-ziggy-zoomba.com/phpBB3
Date: 11/22/10 04:45:16PM
Visit Count: 1

| Type | URL |
|---|---|
| Visited URL | http://ay-ziggy-zoomba.com/phpBB3/viewforum.php?f=3&sid=eb5246136c454119c448e51fde9ae2ca |

**Type:** Visited URL
**URL:** http://ay-ziggy-zoomba.com/phpBB3/viewforum.php?f=3&sid=eb5246136c454119c448e51fde9ae2ca
Title: Ay-Ziggy-Zoomba.com • View forum - Football
Date: 11/22/10 04:46:03PM
Visit Count: 1

| Type | URL |
|---|---|
| Visited URL | http://ay-ziggy-zoomba.com/phpBB3/viewforum.php?f=2&sid=eb5246136c454119c448e51fde9ae2ca |

**Type:** Visited URL
**URL:** http://ay-ziggy-zoomba.com/phpBB3/viewforum.php?f=2&sid=eb5246136c454119c448e51fde9ae2ca
Title: Ay-Ziggy-Zoomba.com • View forum - Men's Hoops
Date: 11/22/10 04:46:17PM
Visit Count: 1

| Type | URL |
|---|---|
| Visited URL | http://ay-ziggy-zoomba.com/phpBB3/viewtopic.php?f=2&t=27047 |

**Type:** Visited URL
**URL:** http://ay-ziggy-zoomba.com/phpBB3/viewtopic.php?f=2&t=27047
Title: Ay-Ziggy-Zoomba.com • View topic - Video about the bronze falcon
Date: 11/22/10 04:46:29PM
Visit Count: 1

| Type | URL |
|---|---|
| Visited URL | http://yahoo.com/ |

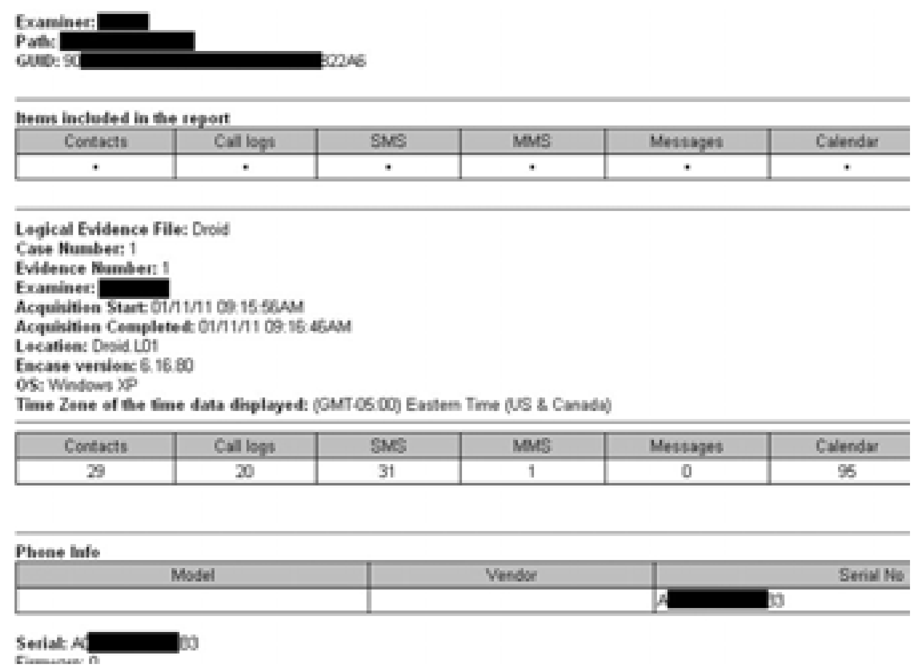
**FIGURE 6.37**

Neutrino web history in detailed report.

- SIM ID cloner device
- Pack of SIM clone cards
- Write-protected universal memory card reader
- Complete set of cables for logical and physical acquisition
- XACT Hex Viewer software application
- XRY Reader Tool for distribution to third parties

XRY was designed and refined with the input of forensic investigators and a wizard guides you through the entire process to assist the examination. The new unified Logical/Physical extraction Wizard and the resulting reports help to show the examiner the full contents of the device in a neat, clean, and professional manner.

One of the unique features of XRY is the Device Manual with a complete and detailed list of available support for each device; identifying what data can be retrieved, and also what cannot be recovered which is sometimes just as relevant to investigators.



**Examiner:** [redacted]
**Path:** [redacted]
**GUID:** 90[redacted]B22A6

**Items included in the report**

| Contacts | Call logs | SMS | MMS | Messages | Calendar |
|---|---|---|---|---|---|
| • | • | • | • | • | • |

**Logical Evidence File:** Droid
**Case Number:** 1
**Evidence Number:** 1
**Examiner:** [redacted]
**Acquisition Start:** 01/11/11 09:15:56AM
**Acquisition Completed:** 01/11/11 09:16:46AM
**Location:** Droid.L01
**Encase version:** 6.16.80
**OS:** Windows XP
**Time Zone of the time data displayed:** (GMT-05:00) Eastern Time (US & Canada)

| Contacts | Call logs | SMS | MMS | Messages | Calendar |
|---|---|---|---|---|---|
| 29 | 20 | 31 | 1 | 0 | 95 |

**Phone Info**

| Model | Vendor | Serial No |
|---|---|---|
| | | A[redacted]03 |

**Serial:** A0[redacted]03

**FIGURE 6.38**

Neutrino report exported as HTML.

All extractions, logical or physical, are saved in an XRY file which remains—for forensic security purposes. From the XRY file, you can create reports as required in Word, Excel, Open Office, or PDF. You can include case data, and references, choose what data is included in the report or not and then distribute it to other parties involved in the investigation; lawyers, prosecutors or other investigators. MSAB offer a free XRY reader and you can provide this to third parties to allow them to make notes on the report—while still maintaining the original forensic integrity of the data.

Within the package is the XACT Hex Viewer application to undertake more detailed examination of the raw data recovered and assist with searching and manual decoding to supplement the automatic decoding available in XRY Physical.

Version 5.1 of the XRY Forensic Pack was released on June 28, 2010 with additional support for the Apple iPad.

## Installation

XRY is a Windows application that you install from a single setup program provided by the vendor. The setup includes an installation wizard, checks for software updates online, and takes approximately 15 minutes. The software requires the use of a hardware dongle to operate.

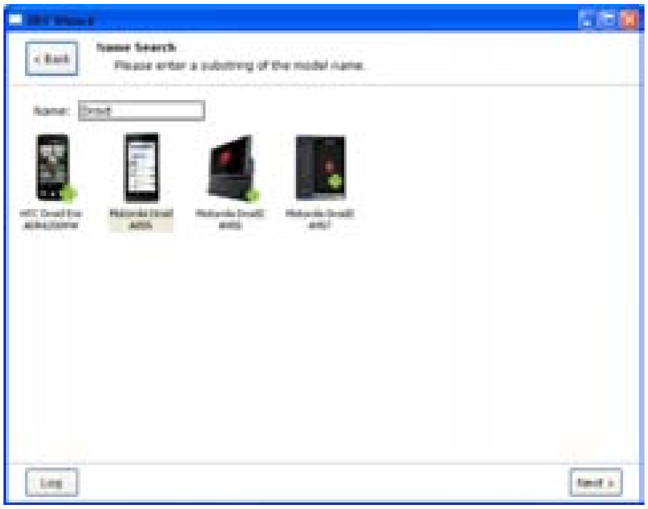

Name Search
Please enter a substring of the model name.
Name: Droid

**FIGURE 6.39**

XRY—Search device type.

## Acquisition

After the installation is complete, run the software, select Extract Data, and then choose to extract data from a phone. After that, you must identify the device type, which can be done in several ways. For this example, a search was executed by selecting Name Search, search for Droid, and select Motorola Droid A855, and then select Next. This is illustrated in Fig. 6.39.

Although the MSAB forensic suite supports physical extractions of some devices, only logical extraction is available for Android. After selecting logical acquisition, the software displays the data available for extraction as shown in Fig. 6.40.

Select Next and then the option to extract data by cable and a full read and finally click Next to start data extraction. When the extraction is complete, you see the extraction complete screen shown in Fig. 6.41.

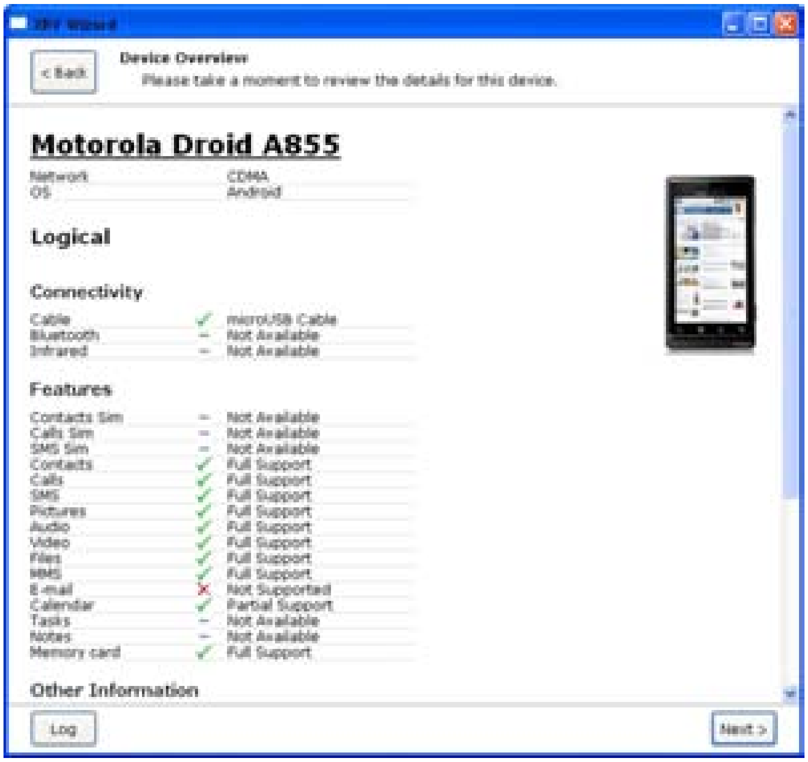
## Data presentation and analysis

After extraction, the application displays the data within the application and is easy to navigate. The contact list includes not only the details of a contact but where the contact was stored, as shown in Fig. 6.42.

The call logs provide the type of call (dialed or received), name, number, time, duration, and storage location, as shown in Fig. 6.43.

Fig. 6.44 shows the SMS messages including the number, name, message, time, status, storage, index, and the folder it is located in.

Fig. 6.45 shows that the only image extracted from the device was from an MMS.



Device Overview
Please take a moment to review the details for this device.

**Motorola Droid A855**

| | |
|---|---|
| Network | CDMA |
| OS | Android |

**Logical**

**Connectivity**

| | | |
|---|---|---|
| Cable | ✓ | microUSB Cable |
| Bluetooth | – | Not Available |
| Infrared | – | Not Available |

**Features**

| | | |
|---|---|---|
| Contacts Sim | – | Not Available |
| Calls Sim | – | Not Available |
| SMS Sim | – | Not Available |
| Contacts | ✓ | Full Support |
| Calls | ✓ | Full Support |
| SMS | ✓ | Full Support |
| Pictures | ✓ | Full Support |
| Audio | ✓ | Full Support |
| Video | ✓ | Full Support |
| Files | ✓ | Full Support |
| MMS | ✓ | Full Support |
| E-mail | ✗ | Not Supported |
| Calendar | ✓ | Partial Support |
| Tasks | – | Not Available |
| Notes | – | Not Available |
| Memory card | ✓ | Full Support |

**Other Information**

**FIGURE 6.40**

XRY data types available for Droid extraction.

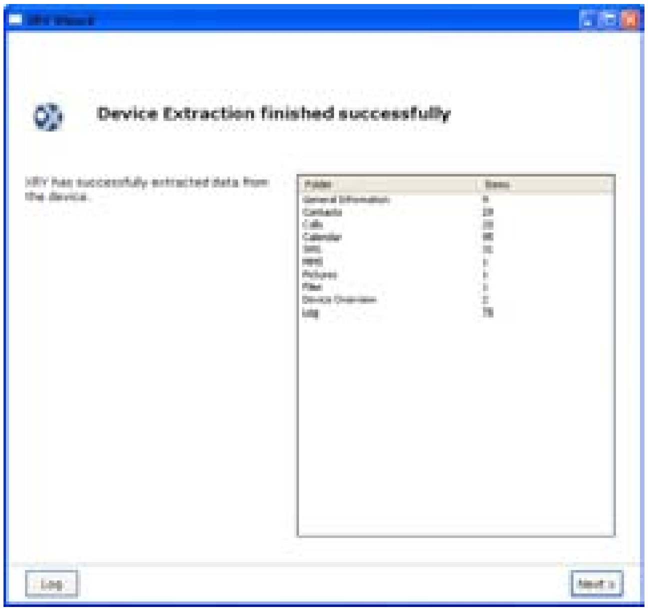
Other data was extracted. However, it is not displayed in a dedicated report section. For example, you can review the web browser history, web site bookmarks, and Google search history in the Log section of the report.

MSAB also has a tool called XACT, which provides a hex view of specific entries. For example, Fig. 6.46 shows the contents of an SMS message.

## *Paraben Device Seizure*

The following overview of Device Seizure was provided by the vendor:

Paraben's Device Seizure (DS) is a handheld forensics tool that enables the investigator to perform logical and physical data acquisitions, deleted data recovery, and full data dumps, on approximately 2400 models of cell phones, PDAs/smart-phones, and portable GPS units. Physical data acquisition—often where deleted data is found—is possible from approximately two-thirds of the supported models. Furthermore, DS has been documented and verified as being 100% forensically



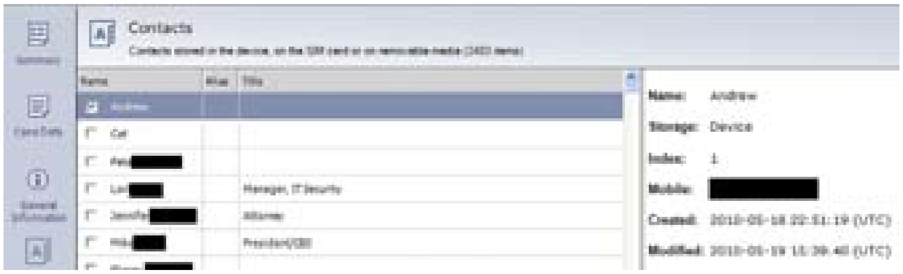
**Device Extraction finished successfully**

XRY has successfully extracted data from the device.

| Folder | Items |
|---|---|
| General Information | 6 |
| Contacts | 29 |
| Calls | 23 |
| Calendar | 48 |
| SMS | 31 |
| MMS | 1 |
| Pictures | 1 |
| Files | 1 |
| Device Overview | 2 |
| Log | 78 |

FIGURE 6.41

XRY extraction complete.



Contacts

Contacts stored in the device, on the SIM card or on removable media ([illegible] items)

| Name | Alias | Title |
|---|---|---|
| Andrew | | |
| Cal | | |
| Pete[redacted] | | |
| Lar[redacted] | | Manager, IT Security |
| Jennifer[redacted] | | Attorney |
| Mike[redacted] | | President/CEO |
| [illegible][redacted] | | |

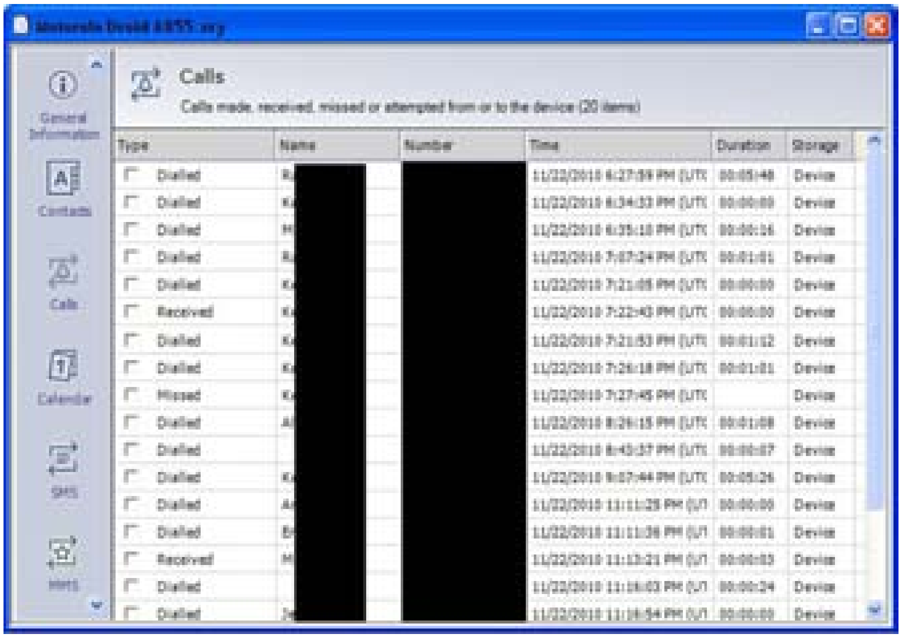
Name: Andrew
Storage: Device
Index: 1
Mobile: [redacted]
Created: 2010-05-18 22:11:19 (UTC)
Modified: 2010-05-19 15:39:40 (UTC)

FIGURE 6.42

XRY contacts list.

sound, meaning the digital evidence is never altered in any way. These functions are all possible through a standard USB data cable connection with any PC.

Over the past two years Google's Android operating system for mobile devices has had a significant impact on the industry. Paraben focuses on staying in step with the latest innovations and has added support for the Android OS to DS. With the release of DS version 4.0, an investigator has the capabilities of acquiring the most commonly sought-after data such as call logs, address book, and SMS messages.



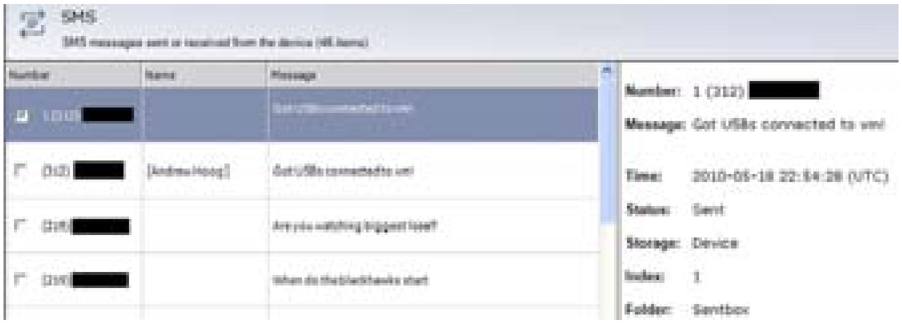
Calls

Calls made, received, missed or attempted from or to the device (20 items)

| Type | Name | Number | Time | Duration | Storage |
|---|---|---|---|---|---|
| Dialed | R[redacted] | [redacted] | 11/22/2010 6:27:59 PM (UT | 00:05:48 | Device |
| Dialed | K[redacted] | [redacted] | 11/22/2010 6:34:33 PM (UT | 00:00:00 | Device |
| Dialed | M[redacted] | [redacted] | 11/22/2010 6:35:10 PM (UT | 00:00:16 | Device |
| Dialed | R[redacted] | [redacted] | 11/22/2010 7:07:24 PM (UT | 00:01:01 | Device |
| Dialed | K[redacted] | [redacted] | 11/22/2010 7:21:05 PM (UT | 00:00:00 | Device |
| Received | K[redacted] | [redacted] | 11/22/2010 7:22:43 PM (UT | 00:00:00 | Device |
| Dialed | K[redacted] | [redacted] | 11/22/2010 7:21:53 PM (UT | 00:01:12 | Device |
| Dialed | K[redacted] | [redacted] | 11/22/2010 7:26:18 PM (UT | 00:01:01 | Device |
| Missed | K[redacted] | [redacted] | 11/22/2010 7:27:45 PM (UT |  | Device |
| Dialed | A[redacted] | [redacted] | 11/22/2010 8:26:15 PM (UT | 00:01:08 | Device |
| Dialed | [redacted] | [redacted] | 11/22/2010 8:43:37 PM (UT | 00:00:07 | Device |
| Dialed | K[redacted] | [redacted] | 11/22/2010 9:07:44 PM (UT | 00:05:26 | Device |
| Dialed | A[redacted] | [redacted] | 11/22/2010 11:11:25 PM (U | 00:00:00 | Device |
| Dialed | E[redacted] | [redacted] | 11/22/2010 11:11:36 PM (U | 00:00:01 | Device |
| Received | M[redacted] | [redacted] | 11/22/2010 11:13:21 PM (U | 00:00:03 | Device |
| Dialed | [redacted] | [redacted] | 11/22/2010 11:16:03 PM (U | 00:00:24 | Device |
| Dialed | J[redacted] | [redacted] | 11/22/2010 11:16:54 PM (U | 00:00:00 | Device |

**FIGURE 6.43**

XRY call log.



SMS

SMS messages sent or received from the device (48 items)

| Number | Name | Message |
|---|---|---|
| 1 (312) [redacted] |  | Got USBs connected to vm! |
| (312) [redacted] | [Andrew Hoog] | Got USBs connected to vm! |
| (219) [redacted] |  | Are you watching biggest loser? |
| (219) [redacted] |  | When do the blackhawks start |

**Number:** 1 (312) [redacted]

**Message:** Got USBs connected to vm!

**Time:** 2010-05-18 22:54:28 (UTC)

**Status:** Sent

**Storage:** Device

**Index:** 1

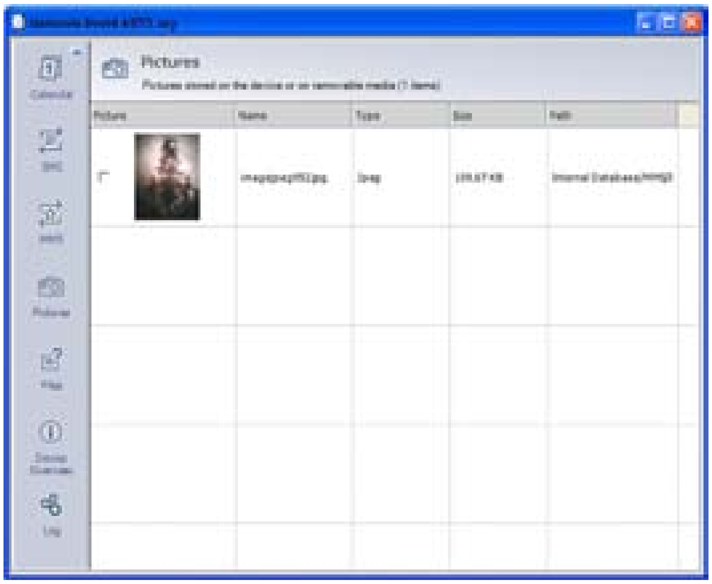
**Folder:** Sentbox

**FIGURE 6.44**

XRY SMS.

Beyond these data types, DS will also acquire multimedia files—MMS messages, images, video, and audio files. The full list of data types that can be acquired from Android models are as follows:

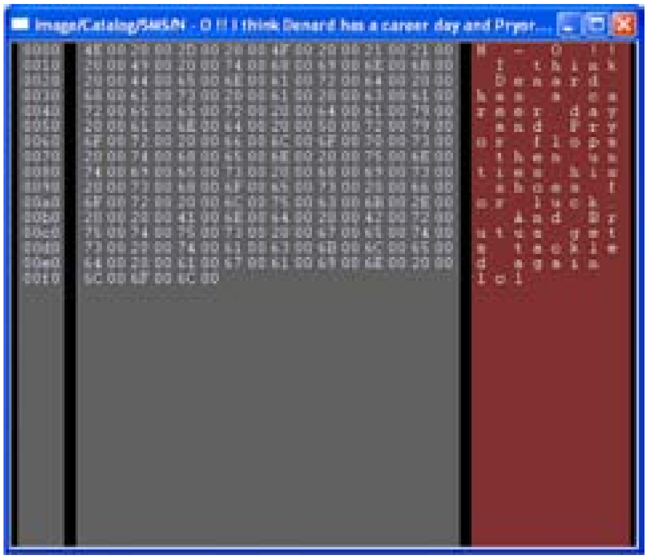
- Address Book including contacts groups, organizations, and address book settings, along with the standard name, phone number, and address
- SMS messages


Pictures
Pictures stored on the device or on removable media (1 items)
Calendar
SMS
MMS
Pictures
Files
Device Overview
Log

| Picture | Name | Type | Size | Path |
|---|---|---|---|---|
| | [illegible].jpg | Jpeg | [illegible] KB | Internal Database/[illegible] |

**FIGURE 6.45**

XRY images.



Image/Catalog/SMS/N - O !! I think Denard has a career day and Pryor...

```
[illegible]
```

**FIGURE 6.46**

XACT—SMS message.

- MMS messages
- Call history
- Contact methods
- Browser history
- External image media (metadata)
- External image thumbnail media (metadata)
- External media, audio, and miscellaneous data (metadata)
- External videos (metadata)
- List of all applications installed and their version

### Installation

Paraben DS version 4.1.3971.37683 was installed on the Windows forensic workstation. The setup process required the installation of many required drivers that took a considerable amount of time. The software must be registered prior to use, which is achieved either using a hardware dongle or through a registration key file provided by Paraben. To install the registration key file, you simply copy the file into the DS install directory, which is likely C:\Program Files\Paraben Corporation\Device Seizure.

### Acquisition

To start the acquisition of a new Android device, you first open a new case and complete the required case information section. You then choose "Data Acquisition" and select Android at which point the following directions are provided:

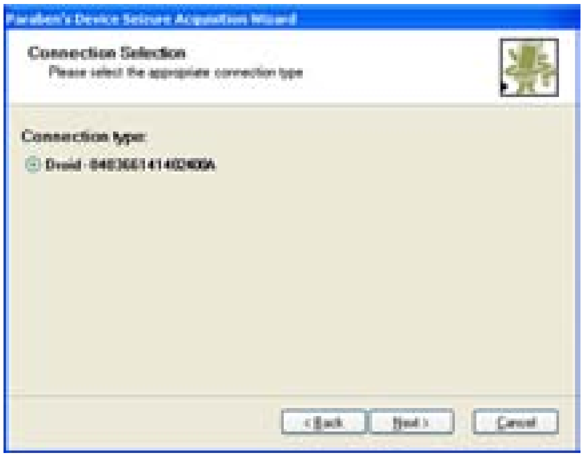
```
Android based cell phones must be placed into "debugging" mode*. Follow the
instructions below:
1. On the cell phone, navigate to Settings>Application Settings and select the
Unknown Sources option.
2. On the cell phone, navigate to Settings>Application Settings>Development and
select the USB debugging option.
3. Install drivers for your cell phone. These drivers are provided in the
Device Seizure Drivers Pack that can be downloaded on www.paraben.com.
4. Connect the cell phone to the USB port on your computer.
* Please note that AT&T and Motorola have taken the "Unknown Sources", found on
step 1, out of Android devices. Device Seizure does not support these models.
```

According to these instructions, Motorola phones are not supported. However, the acquisition of the Motorola Droid was successful.

Follow the instructions, then click "Next" at which point DS attempts to identify the phone. Ensure the identified device information is accurate and click "Next" as shown in Fig. 6.47.

The next screen provides a list of supported data types that DS can extract from the device. All were selected, which includes acquiring the part of the file system it can read and the SD card, so the acquisition process is slow. Fig. 6.48 shows the DS acquisition timing.

The acquisition completed approximately two hours later and the prompt to sort files was accepted. The process of sorting the files took considerable time, but was



Paraben's Device Seizure Acquisition Wizard

Connection Selection

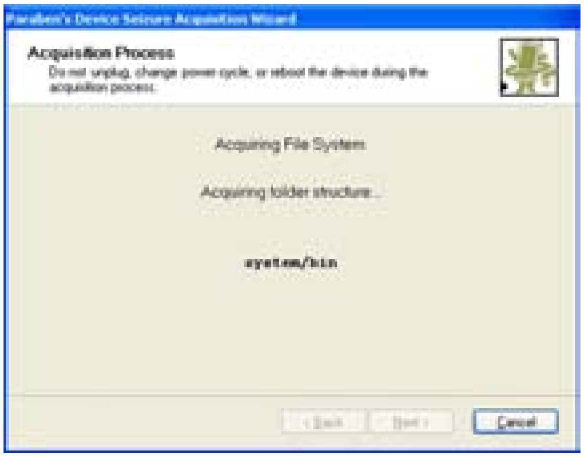
Please select the appropriate connection type

Connection type:

○ Droid - 040366141402400A

**FIGURE 6.47**

Device Seizure device identification.



Paraben's Device Seizure Acquisition Wizard

Acquisition Process

Do not unplug, change power cycle, or reboot the device during the acquisition process.

Acquiring File System

Acquiring folder structure...

system/bin

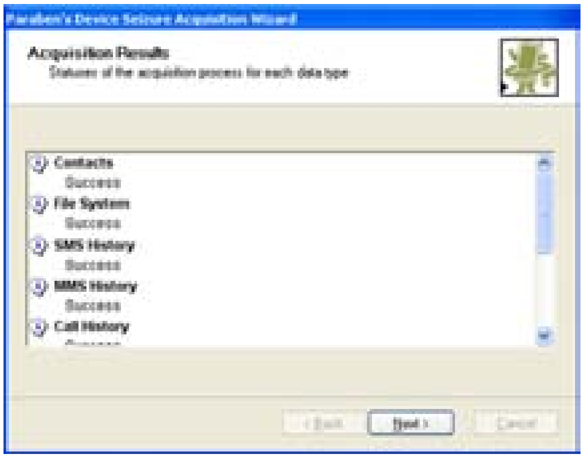
**FIGURE 6.48**

Device Seizure acquisition timing.

less than two hours. At that point, the acquisition process was complete. Fig. 6.49 shows the DS acquisition complete output.

## Data presentation and analysis

Device Seizure displays the acquired data with the application in an easy to browse and navigate structure. The acquired directory structures are shown in Fig. 6.50.



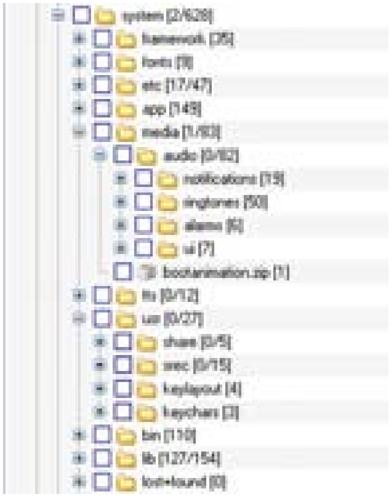
Paraben's Device Seizure Acquisition Wizard

Acquisition Results
Statuses of the acquisition process for each data type

```
Contacts
    Success
File System
    Success
SMS History
    Success
MMS History
    Success
Call History
```

FIGURE 6.49

Device Seizure acquisition complete.



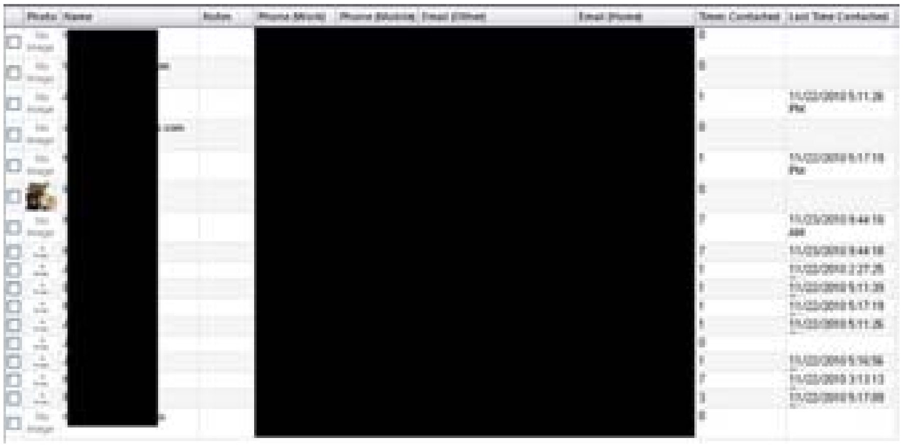
```
system [2/628]
    framework [35]
    fonts [9]
    etc [17/47]
    app [149]
    media [1/83]
        audio [0/82]
            notifications [19]
            ringtones [50]
            alarms [6]
            ui [7]
        bootanimation.zip [1]
    tts [0/12]
    usr [0/27]
        share [0/5]
        srec [0/15]
        keylayout [4]
        keychars [3]
    bin [110]
    lib [127/154]
    lost+found [0]
```

FIGURE 6.50

Device Seizure Droid directory structure.

Contacts provide not only the name, notes, phone numbers, and e-mail, but also helpful fields, such as number of times contacted, last time contacted, and a photo, if available, as shown in Fig. 6.51.

The SMS reporting provides the expected fields, but deleted messages were not included. The report does not cross-reference the contact data with the phone



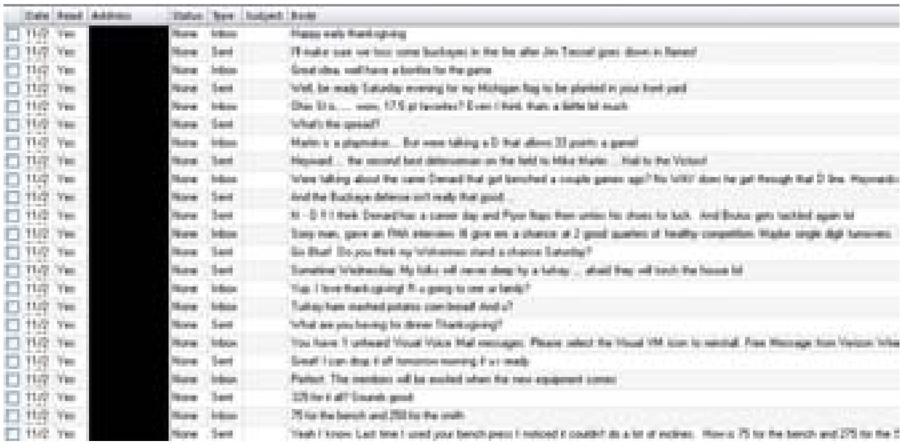
| | Photo | Name | Notes | Phone (Work) | Phone (Mobile) | Email (Other) | Email (Home) | Times Contacted | Last Time Contacted |
|---|---|---|---|---|---|---|---|---|---|
| ☐ | No Image | [redacted] | | [redacted] | [redacted] | [redacted] | [redacted] | 0 | |
| ☐ | No Image | [redacted] | | [redacted] | [redacted] | [redacted] | [redacted] | 0 | |
| ☐ | No Image | [redacted] | | [redacted] | [redacted] | [redacted] | [redacted] | 1 | 11/22/2010 5:11:26 PM |
| ☐ | No Image | [redacted] | | [redacted] | [redacted] | [redacted] | [redacted] | 0 | |
| ☐ | No Image | [redacted] | | [redacted] | [redacted] | [redacted] | [redacted] | 1 | 11/22/2010 5:17:19 PM |
| ☐ | | [redacted] | | [redacted] | [redacted] | [redacted] | [redacted] | 0 | |
| ☐ | No Image | [redacted] | | [redacted] | [redacted] | [redacted] | [redacted] | 7 | 11/23/2010 9:44:18 AM |
| ☐ | [illegible] | [redacted] | | [redacted] | [redacted] | [redacted] | [redacted] | 7 | 11/23/2010 9:44:18 |
| ☐ | [illegible] | [redacted] | | [redacted] | [redacted] | [redacted] | [redacted] | 1 | 11/22/2010 2:27:25 |
| ☐ | [illegible] | [redacted] | | [redacted] | [redacted] | [redacted] | [redacted] | 1 | 11/22/2010 5:11:26 |
| ☐ | [illegible] | [redacted] | | [redacted] | [redacted] | [redacted] | [redacted] | 1 | 11/22/2010 5:17:19 |
| ☐ | [illegible] | [redacted] | | [redacted] | [redacted] | [redacted] | [redacted] | 1 | 11/22/2010 5:11:26 |
| ☐ | [illegible] | [redacted] | | [redacted] | [redacted] | [redacted] | [redacted] | 0 | |
| ☐ | [illegible] | [redacted] | | [redacted] | [redacted] | [redacted] | [redacted] | 1 | 11/22/2010 5:16:56 |
| ☐ | [illegible] | [redacted] | | [redacted] | [redacted] | [redacted] | [redacted] | 7 | 11/22/2010 3:13:13 |
| ☐ | [illegible] | [redacted] | | [redacted] | [redacted] | [redacted] | [redacted] | 3 | 11/22/2010 5:17:09 |
| ☐ | No Image | [redacted] | | [redacted] | [redacted] | [redacted] | [redacted] | 0 | |

**FIGURE 6.51**

Device Seizure contacts.

number, so the examiner must either know the phone number or handle the cross-referencing themselves. Fig. 6.52 shows the DS SMS.

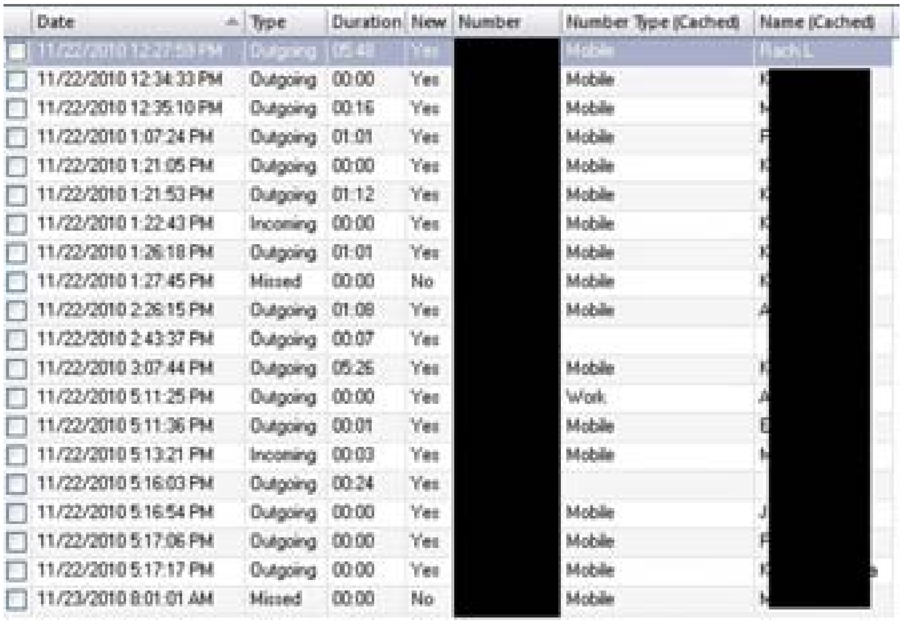
However, the call logs do perform the cross-reference and display the date, message type, duration, number, number type, name, or whether the call was a new call (presumably the first time that number appeared in the call logs). Fig. 6.53 shows the call logs.



| | Date | Read | Address | Status | Type | Subject | Body |
|---|---|---|---|---|---|---|---|
| ☐ | 11/2 | Yes | [redacted] | None | Inbox | | Happy early thanksgiving |
| ☐ | 11/2 | Yes | [redacted] | None | Sent | | I'll make sure we toss some buckeyes in the fire after Jim Tressel goes down in flames! |
| ☐ | 11/2 | Yes | [redacted] | None | Inbox | | Great idea, we'll have a bonfire for the game |
| ☐ | 11/2 | Yes | [redacted] | None | Sent | | Well, be ready Saturday evening for my Michigan flag to be planted in your front yard |
| ☐ | 11/2 | Yes | [redacted] | None | Inbox | | Ohio St is.... wow, 17.5 pt favorites? Even I think that's a little bit much |
| ☐ | 11/2 | Yes | [redacted] | None | Sent | | What's the spread? |
| ☐ | 11/2 | Yes | [redacted] | None | Inbox | | Martin is a playmaker... But were talking a D that allows 33 points a game! |
| ☐ | 11/2 | Yes | [redacted] | None | Sent | | Hayward.... the second best defenseman on the field to Mike Martin ... Hail to the Victors! |
| ☐ | 11/2 | Yes | [redacted] | None | Inbox | | Were talking about the same Denard that got benched a couple games ago? No WAY does he get through that D line. Hayward |
| ☐ | 11/2 | Yes | [redacted] | None | Sent | | And the Buckeye defense isn't really that good... |
| ☐ | 11/2 | Yes | [redacted] | None | Sent | | [illegible] I think Denard has a career day and Pryor flaps them unties his shoes for luck. And Boom gets tackled again lol |
| ☐ | 11/2 | Yes | [redacted] | None | Inbox | | Sorry man, gave an RNA extension. ill give you a chance at 2 good quarters of healthy competition. Maybe single digit turnovers |
| ☐ | 11/2 | Yes | [redacted] | None | Sent | | Go Blue! Do you think my Wolverines stand a chance Saturday? |
| ☐ | 11/2 | Yes | [redacted] | None | Sent | | Sometime Wednesday. My folks will never sleep by a turkey ... afraid they will torch the house lol |
| ☐ | 11/2 | Yes | [redacted] | None | Inbox | | Yup. I love thanksgiving! R u going to see ur family? |
| ☐ | 11/2 | Yes | [redacted] | None | Inbox | | Turkey ham mashed potatos corn bread! And u? |
| ☐ | 11/2 | Yes | [redacted] | None | Sent | | What are you having for dinner Thanksgiving? |
| ☐ | 11/2 | Yes | [redacted] | None | Inbox | | You have 1 unheard Visual Voice Mail messages. Please select the Visual VM icon to reinstall. Free Message from Verizon Wire |
| ☐ | 11/2 | Yes | [redacted] | None | Sent | | Great! I can drop it off tomorrow morning if u r ready |
| ☐ | 11/2 | Yes | [redacted] | None | Inbox | | Perfect. The members will be excited when the new equipment comes |
| ☐ | 11/2 | Yes | [redacted] | None | Sent | | 325 for it all? Sounds good |
| ☐ | 11/2 | Yes | [redacted] | None | Inbox | | 75 for the bench and 250 for the smith |
| ☐ | 11/2 | Yes | [redacted] | None | Sent | | Yeah I know. Last time I used your bench press I noticed it couldn't do a lot of inclines. How is 75 for the bench and 275 for the |

**FIGURE 6.52**

Device Seizure SMS.



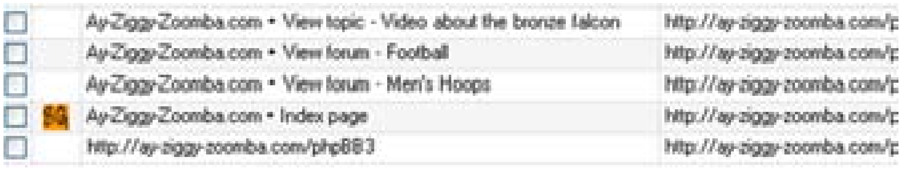
| | Date | Type | Duration | New | Number | Number Type (Cached) | Name (Cached) |
|---|---|---|---|---|---|---|---|
| ☐ | 11/22/2010 12:27:59 PM | Outgoing | 05:48 | Yes | [redacted] | Mobile | Rach, L |
| ☐ | 11/22/2010 12:34:33 PM | Outgoing | 00:00 | Yes | [redacted] | Mobile | K[redacted] |
| ☐ | 11/22/2010 12:35:10 PM | Outgoing | 00:16 | Yes | [redacted] | Mobile | M[redacted] |
| ☐ | 11/22/2010 1:07:24 PM | Outgoing | 01:01 | Yes | [redacted] | Mobile | F[redacted] |
| ☐ | 11/22/2010 1:21:05 PM | Outgoing | 00:00 | Yes | [redacted] | Mobile | K[redacted] |
| ☐ | 11/22/2010 1:21:53 PM | Outgoing | 01:12 | Yes | [redacted] | Mobile | K[redacted] |
| ☐ | 11/22/2010 1:22:43 PM | Incoming | 00:00 | Yes | [redacted] | Mobile | K[redacted] |
| ☐ | 11/22/2010 1:26:18 PM | Outgoing | 01:01 | Yes | [redacted] | Mobile | K[redacted] |
| ☐ | 11/22/2010 1:27:45 PM | Missed | 00:00 | No | [redacted] | Mobile | K[redacted] |
| ☐ | 11/22/2010 2:26:15 PM | Outgoing | 01:08 | Yes | [redacted] | Mobile | A[redacted] |
| ☐ | 11/22/2010 2:43:37 PM | Outgoing | 00:07 | Yes | [redacted] | | [redacted] |
| ☐ | 11/22/2010 3:07:44 PM | Outgoing | 05:26 | Yes | [redacted] | Mobile | K[redacted] |
| ☐ | 11/22/2010 5:11:25 PM | Outgoing | 00:00 | Yes | [redacted] | Work | A[redacted] |
| ☐ | 11/22/2010 5:11:36 PM | Outgoing | 00:01 | Yes | [redacted] | Mobile | E[redacted] |
| ☐ | 11/22/2010 5:13:21 PM | Incoming | 00:03 | Yes | [redacted] | Mobile | M[redacted] |
| ☐ | 11/22/2010 5:16:03 PM | Outgoing | 00:24 | Yes | [redacted] | | [redacted] |
| ☐ | 11/22/2010 5:16:54 PM | Outgoing | 00:00 | Yes | [redacted] | Mobile | J[redacted] |
| ☐ | 11/22/2010 5:17:06 PM | Outgoing | 00:00 | Yes | [redacted] | Mobile | F[redacted] |
| ☐ | 11/22/2010 5:17:17 PM | Outgoing | 00:00 | Yes | [redacted] | Mobile | K[redacted] |
| ☐ | 11/23/2010 8:01:01 AM | Missed | 00:00 | No | [redacted] | Mobile | M[redacted] |

**FIGURE 6.53**

Device Seizure call logs.

A complete web history is available and parsed, including visit count and bookmarks. However, the data view is quite long and only the beginning information is displayed in Fig. 6.54.

Device Seizure allows the examiner to select any file and extract it to the forensic workstation for additional analysis. This is helpful for viewing or analyzing file types not supported natively in the DS environment. As the file sorting option was chosen during the acquisition stage, each extracted file was identified and grouped by type allowing quick access to files of interest. This is shown in Fig. 6.55.



| | | | |
|---|---|---|---|
| ☐ | | Ay-Ziggy-Zoomba.com • View topic - Video about the bronze falcon | http://ay-ziggy-zoomba.com/p |
| ☐ | | Ay-Ziggy-Zoomba.com • View forum - Football | http://ay-ziggy-zoomba.com/p |
| ☐ | | Ay-Ziggy-Zoomba.com • View forum - Men's Hoops | http://ay-ziggy-zoomba.com/p |
| ☐ | ▨ | Ay-Ziggy-Zoomba.com • Index page | http://ay-ziggy-zoomba.com/p |
| ☐ | | http://ay-ziggy-zoomba.com/phpBB3 | http://ay-ziggy-zoomba.com/p |

**FIGURE 6.54**

Device Seizure web history.

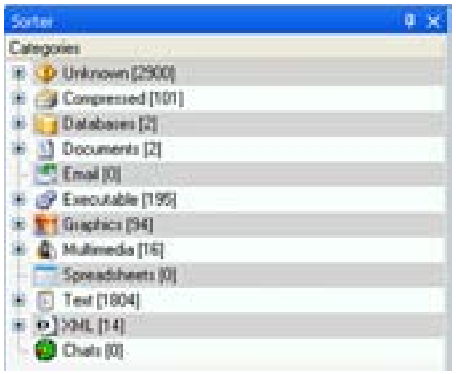


**FIGURE 6.55**

Device Seizure's file sorter.

### *viaForensics' ViaExtract*

The following overview of viaExtract was provided by the vendor:

viaExtract is the latest Android forensic solution from viaForensics, a leader and innovator in the field. In addition to their mobile forensics white papers and book, viaForensics' provides a free Android forensics solution for law enforcement and government agencies called AFLogical.

Building on this experience, viaForensics developed viaExtract, which extracts, analyzes, and reports on data in Android devices. viaExtract is a modular solution and will next offer an Android forensic physical technique based on viaForensics' research and development. Up to date information on viaExtract is available online at http://viaforensics.com/products/viaextract/ including support for Android forensics physical techniques, additional supported mobile platforms, and advanced forensic recovery techniques.

viaForensics is a forensics and security firm that actively investigates mobile devices and traditional computers. Their direct experience as examiners has led to the development of a tool specifically tailored to forensic examiners. The tool was designed for frequent updates as the mobile forensic discipline is changing rapidly. A unique debug and reporting system integrated into viaExtract simplifies the process of sending debug and sanitized data to viaForensics to assist with the design and improvement of viaExtract operating on the diverse Android ecosystem.

#### Installation

The viaExtract software is distributed as a virtual machine, so it runs on Microsoft Windows, Apple OS X, Linux, or other operating systems that run supported virtualization software. The software is fully configured, as are necessary drivers and supporting libraries, which greatly simplifies the installation. There are several supported virtualization packages, which are free, including:

- Oracle's VirtualBox
- VMWare Player

A number of commercial packages are also available. The virtual machine is downloaded from viaForensics' web site and then imported into the supported software. Examiners can use features built into virtualization software, such as taking snapshots to restore the software to a pristine state after each case, or integrate it directly into their host operating system by sharing data storage and other valuable features.

## Acquisition

After viaExtract is imported into the host system's virtualization software and is running, the forensic examiner logs into the Ubuntu virtual machine and runs viaExtract as shown in Fig. 6.56.

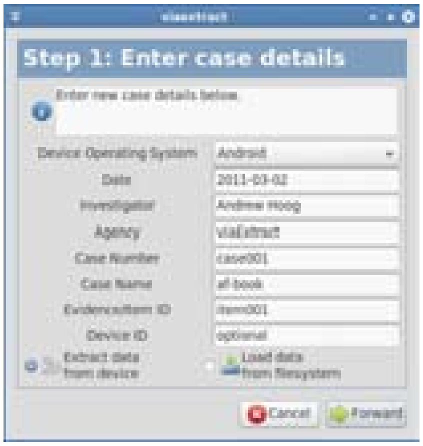
The examiner can then start a new case or open a previous one, as illustrated in Fig. 6.57.

After entering the case details, the examiner can then choose to extract data directly from an Android device or to load from a previous data extraction located on the file system. The latter feature is useful for cases where the examiner used via-Forensics' free AFLogical software to extract data from Android devices. It also allows the examiner to generate a new forensic report from a previous device's extracted data, which is quite useful as new reporting features are added.

For this example, we will extract data from an Android device by clicking Forward. We are then presented with the Load data screen, which provides

**FIGURE 6.56**

viaExtract software.



viaextract

Step 1: Enter case details

Enter new case details below.

| | |
|---|---|
| Device Operating System | Android |
| Date | 2011-03-02 |
| Investigator | Andrew Hoog |
| Agency | viaExtract |
| Case Number | case001 |
| Case Name | af-book |
| Evidence/Item ID | item001 |
| Device ID | optional |

Extract data from device / Load data from filesystem

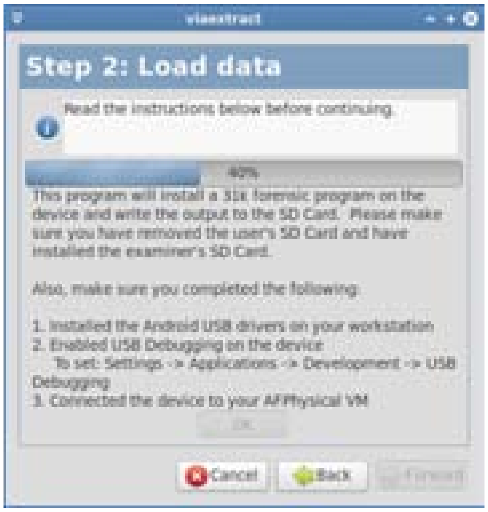
**FIGURE 6.57**

viaExtract—New case.

directions for enabling USB debugging. After you click OK the data extraction begins, as shown in Fig. 6.58.

After the data extraction is complete, the examiner is presented with a list of data extracts and has the ability to select what they want to include in the forensic analysis and report, as shown in Fig. 6.59.

After the selections are completed and the examiner clicks Apply, the report logic is executed and the data extraction is complete.



viaextract

Step 2: Load data

Read the instructions below before continuing.

40%

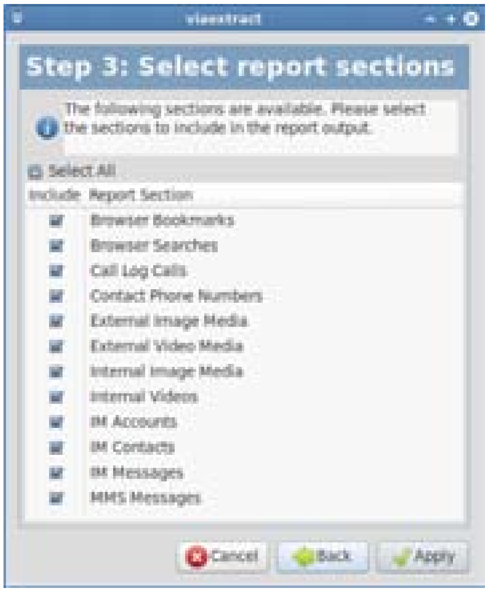
This program will install a 31k forensic program on the device and write the output to the SD Card. Please make sure you have removed the user's SD Card and have installed the examiner's SD Card.

Also, make sure you completed the following:

1. Installed the Android USB drivers on your workstation
2. Enabled USB Debugging on the device
   To set: Settings -> Applications -> Development -> USB Debugging
3. Connected the device to your AFPhysical VM

**FIGURE 6.58**

viaExtract—Data extraction.



viaextract

Step 3: Select report sections

The following sections are available. Please select the sections to include in the report output.

Select All

| Include | Report Section |
|---|---|
| ✓ | Browser Bookmarks |
| ✓ | Browser Searches |
| ✓ | Call Log Calls |
| ✓ | Contact Phone Numbers |
| ✓ | External Image Media |
| ✓ | External Video Media |
| ✓ | Internal Image Media |
| ✓ | Internal Videos |
| ✓ | IM Accounts |
| ✓ | IM Contacts |
| ✓ | IM Messages |
| ✓ | MMS Messages |

**FIGURE 6.59**

viaExtract—Forensic analysis and report.

## Data presentation and analysis

After the data extraction is complete, viaExtract presents the analyzed data to the user. By navigating the selections on the left side of the application, the examiner can view different sections of the report. For example, the first section presented is the Device Information section as shown in Fig. 6.60.

Next, Fig. 6.61 shows browser history and bookmarks that are available in the report.

In this example, you can see several features of the report view, including:

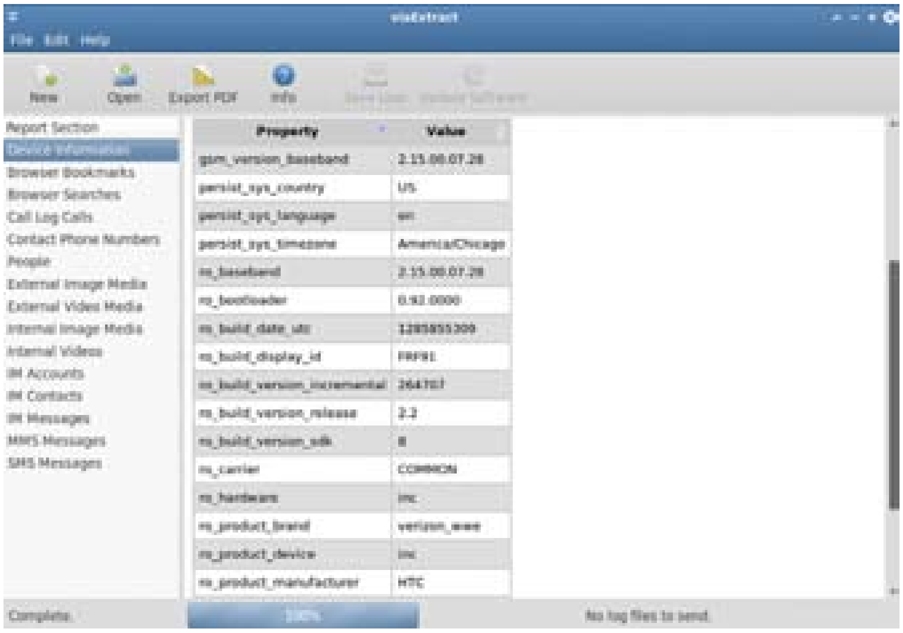
- Ability to filter, on the fly, any section of the report
- Ability to sort ascending or descending on any column

In the next example, a filter of viaforensics.com was applied against 29 people records and 2 remained. However, many of the fields were displayed to the right of the screenshot as shown in Fig. 6.62.

In total, viaExtract currently supports just over 41 Content Providers. However, in the next release, roughly 100 Content Providers will be actively queried. If the device responds to the Content Provider, the extraction and subsequent reporting will succeed. Fig. 6.63 is an example of the Call Logs.

Fig. 6.64 shows the video media metadata.

Reports can also be exported to PDF format as shown in Fig. 6.65.

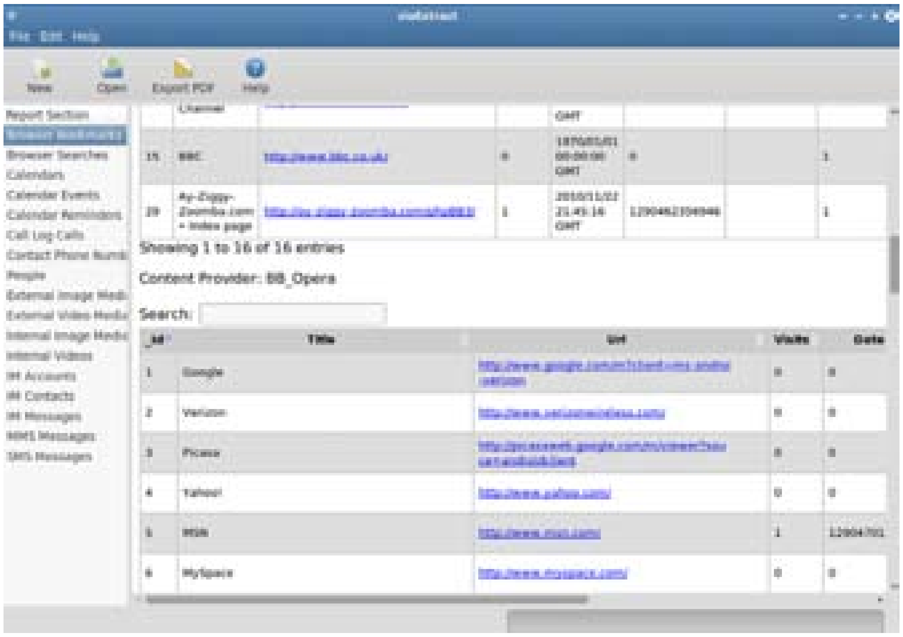


| Property | Value |
|---|---|
| gsm_version_baseband | 2.15.00.07.28 |
| persist_sys_country | US |
| persist_sys_language | en |
| persist_sys_timezone | America/Chicago |
| ro_baseband | 2.15.00.07.28 |
| ro_bootloader | 0.92.0000 |
| ro_build_date_utc | 1285855300 |
| ro_build_display_id | FRF91 |
| ro_build_version_incremental | 264707 |
| ro_build_version_release | 2.2 |
| ro_build_version_sdk | 8 |
| ro_carrier | COMMON |
| ro_hardware | inc |
| ro_product_brand | verizon_wwe |
| ro_product_device | inc |
| ro_product_manufacturer | HTC |

Complete. 100% No log files to send.

**FIGURE 6.60**

viaExtract—Device info.



|  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|
|  | Channel |  |  | GMT |  |  |  |
| 15 | BBC | http://www.bbc.co.uk/ | 0 | 1970/01/01 00:00:00 GMT | 0 |  | 1 |
| 29 | Ay-Ziggy-Zoomba.com - Index page | [illegible] | 1 | 2010/11/02 21:45:16 GMT | 1290462356946 |  | 1 |

Showing 1 to 16 of 16 entries

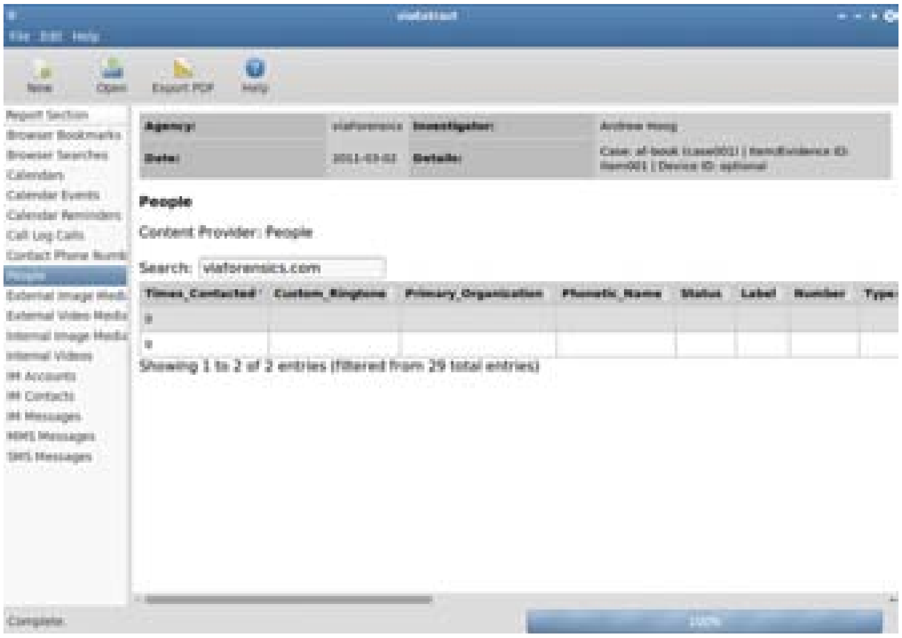
Content Provider: BB_Opera

Search:

| _id | Title | Url | Visits | Date |
|---|---|---|---|---|
| 1 | Google | [illegible] | 0 | 0 |
| 2 | Verizon | http://www.verizonwireless.com/ | 0 | 0 |
| 3 | Picasa | [illegible] | 0 | 0 |
| 4 | Yahoo! | http://www.yahoo.com/ | 0 | 0 |
| 5 | MSN | http://www.msn.com/ | 1 | 12904701 |
| 6 | MySpace | http://www.myspace.com/ | 0 | 0 |

**FIGURE 6.61**

viaExtract—Browser history and bookmarks.



| | | | |
|---|---|---|---|
| **Agency:** | viaforensics | **Investigator:** | Andrew Hoog |
| **Date:** | 2011-03-02 | **Details:** | Case: af-book (case001) \| Item/Evidence ID: item001 \| Device ID: optional |

**People**

Content Provider: People

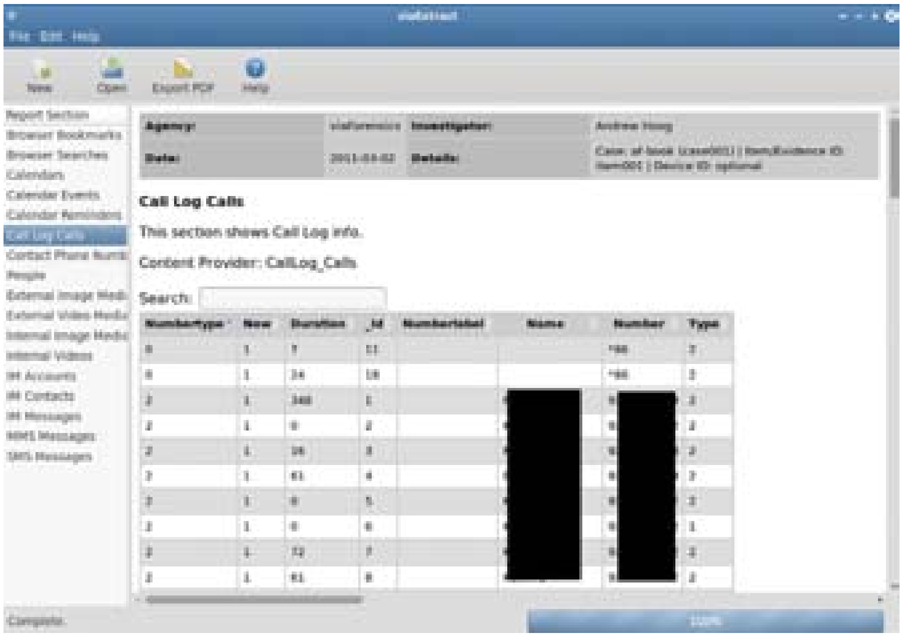
Search: viaforensics.com

| Times_Contacted | Custom_Ringtone | Primary_Organization | Phonetic_Name | Status | Label | Number | Type |
|---|---|---|---|---|---|---|---|
| 0 | | | | | | | |
| 0 | | | | | | | |

Showing 1 to 2 of 2 entries (filtered from 29 total entries)

**FIGURE 6.62**

viaExtract—People records filtered.



| | | | |
|---|---|---|---|
| **Agency:** | viaforensics | **Investigator:** | Andrew Hoog |
| **Date:** | 2011-03-02 | **Details:** | Case: af-book (case001) \| Item/Evidence ID: item001 \| Device ID: optional |

**Call Log Calls**

This section shows Call Log info.

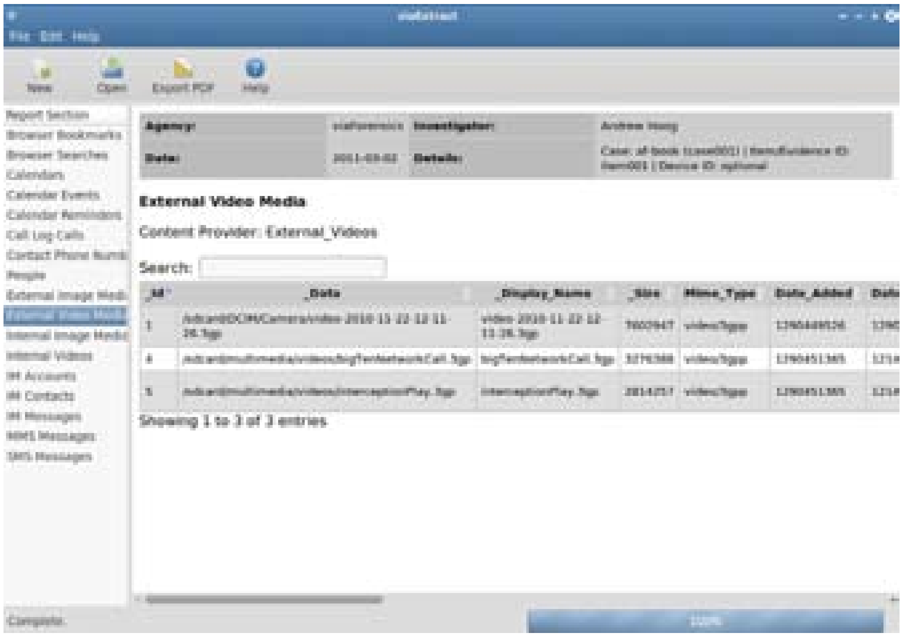
Content Provider: CallLog_Calls

Search:

| Numbertype | New | Duration | _id | Numberlabel | Name | Number | Type |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 7 | 11 | | | *86 | 2 |
| 0 | 1 | 24 | 18 | | | *86 | 2 |
| 2 | 1 | 348 | 1 | | [redacted] | [redacted] | 2 |
| 2 | 1 | 0 | 2 | | [redacted] | [redacted] | 2 |
| 2 | 1 | 16 | 3 | | [redacted] | [redacted] | 2 |
| 2 | 1 | 61 | 4 | | [redacted] | [redacted] | 2 |
| 2 | 1 | 0 | 5 | | [redacted] | [redacted] | 2 |
| 2 | 1 | 0 | 6 | | [redacted] | [redacted] | 1 |
| 2 | 1 | 72 | 7 | | [redacted] | [redacted] | 2 |
| 2 | 1 | 81 | 8 | | [redacted] | [redacted] | 2 |

**FIGURE 6.63**

viaExtract—Call logs.



| | | | |
|---|---|---|---|
| **Agency:** | viaForensics | **Investigator:** | Andrew Hoog |
| **Date:** | 2011-03-02 | **Details:** | Case: af-book (case001) \| Item/Evidence ID: item001 \| Device ID: optional |

**External Video Media**

Content Provider: External_Videos

Search:

| _id | _Data | _Display_Name | _Size | Mime_Type | Date_Added | Date |
|---|---|---|---|---|---|---|
| 1 | /sdcard/DCIM/Camera/video-2010-11-22-12-11-36.3gp | video-2010-11-22-12-11-36.3gp | 7602947 | video/3gpp | 1290449526 | 1290 |
| 4 | /sdcard/multimedia/videos/bigTenNetworkCall.3gp | bigTenNetworkCall.3gp | 3276388 | video/3gpp | 1290451365 | 1214 |
| 5 | /sdcard/multimedia/videos/interceptionPlay.3gp | interceptionPlay.3gp | 2814257 | video/3gpp | 1290451365 | 1214 |

Showing 1 to 3 of 3 entries

**FIGURE 6.64**

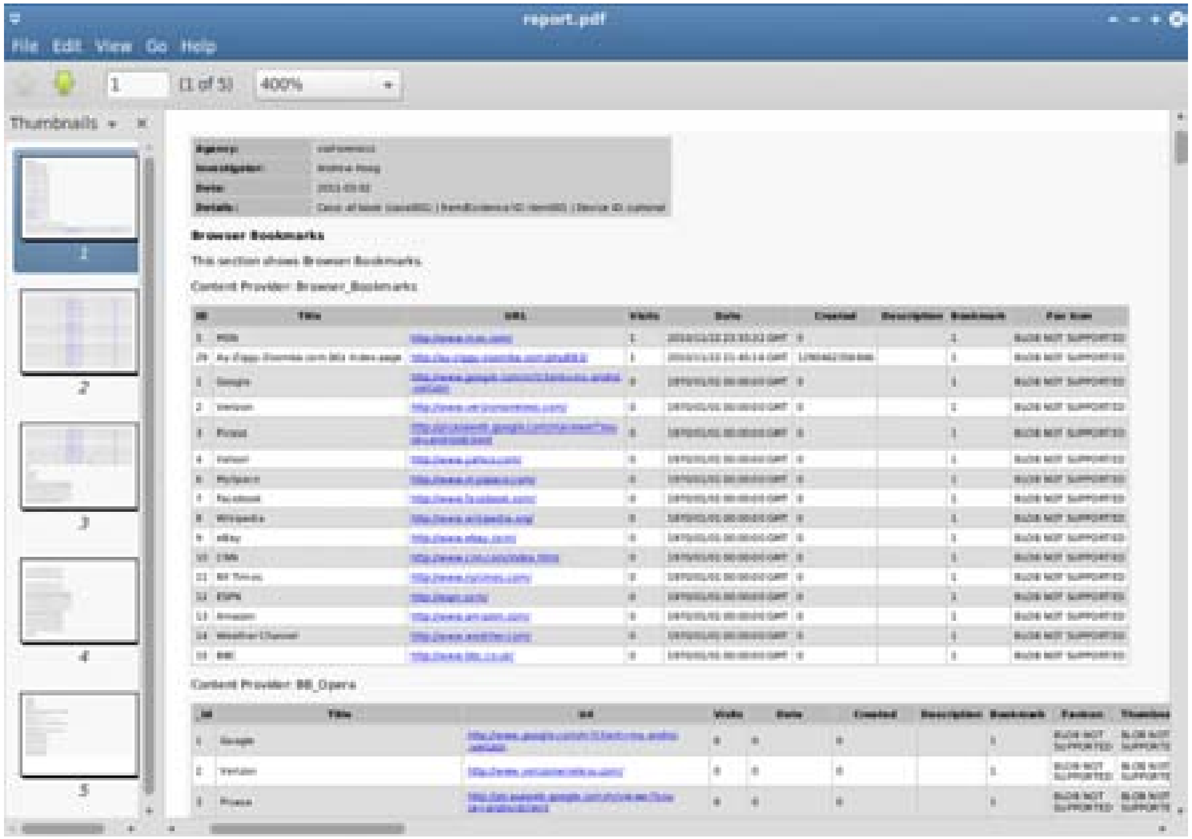
viaExtract—Video media metadata.

## PHYSICAL TECHNIQUES

Forensic techniques that acquire physical images of the targeted data storage typically result in exponentially more data being recovered and often circumvent pass code protection. These techniques provide access to not only deleted data but also data that was simply discarded as the system no longer required it. For example, some systems track the last time a web site was visited and the date field is updated each time the site is accessed again. The previous date and time data was not specifically deleted but was not tracked by the system. On Android devices using YAFFS2, the previous values are recoverable provided garbage collection did not occur. As such, the physical techniques provide access to not only deleted data but also access to obsolete data on the system.

The Android forensics physical techniques fall into two broad categories:

- Hardware: Methods which connect hardware to the device or physically extract device components
- Software: Techniques which run as software on devices with root access and provide a full physical image of the data partitions



| | |
|---|---|
| **Agency:** | [illegible] |
| **Investigator:** | [illegible] |
| **Date:** | [illegible] |
| **Details:** | [illegible] |

**Browser Bookmarks**

This section shows Browser Bookmarks.

Content Provider: Browser_Bookmarks

| ID | Title | URL | Visits | Date | Created | Description | Bookmark | Fav Icon |
|---|---|---|---|---|---|---|---|---|
| 1 | MSN | [illegible] | 1 | [illegible] GMT | 0 | | 1 | BLOB NOT SUPPORTED |
| 29 | [illegible] | [illegible] | 1 | [illegible] GMT | [illegible] | | 1 | BLOB NOT SUPPORTED |
| 1 | Google | [illegible] | 0 | 1970/01/01 00:00:00 GMT | 0 | | 1 | BLOB NOT SUPPORTED |
| 2 | Verizon | [illegible] | 0 | 1970/01/01 00:00:00 GMT | 0 | | 1 | BLOB NOT SUPPORTED |
| 3 | Picasa | [illegible] | 0 | 1970/01/01 00:00:00 GMT | 0 | | 1 | BLOB NOT SUPPORTED |
| 4 | Yahoo! | [illegible] | 0 | 1970/01/01 00:00:00 GMT | 0 | | 1 | BLOB NOT SUPPORTED |
| 6 | MySpace | [illegible] | 0 | 1970/01/01 00:00:00 GMT | 0 | | 1 | BLOB NOT SUPPORTED |
| 7 | Facebook | [illegible] | 0 | 1970/01/01 00:00:00 GMT | 0 | | 1 | BLOB NOT SUPPORTED |
| 8 | Wikipedia | [illegible] | 0 | 1970/01/01 00:00:00 GMT | 0 | | 1 | BLOB NOT SUPPORTED |
| 9 | eBay | [illegible] | 0 | 1970/01/01 00:00:00 GMT | 0 | | 1 | BLOB NOT SUPPORTED |
| 10 | CNN | [illegible] | 0 | 1970/01/01 00:00:00 GMT | 0 | | 1 | BLOB NOT SUPPORTED |
| 11 | NY Times | [illegible] | 0 | 1970/01/01 00:00:00 GMT | 0 | | 1 | BLOB NOT SUPPORTED |
| 12 | ESPN | [illegible] | 0 | 1970/01/01 00:00:00 GMT | 0 | | 1 | BLOB NOT SUPPORTED |
| 13 | Amazon | [illegible] | 0 | 1970/01/01 00:00:00 GMT | 0 | | 1 | BLOB NOT SUPPORTED |
| 14 | Weather Channel | [illegible] | 0 | 1970/01/01 00:00:00 GMT | 0 | | 1 | BLOB NOT SUPPORTED |
| 15 | BBC | [illegible] | 0 | 1970/01/01 00:00:00 GMT | 0 | | 1 | BLOB NOT SUPPORTED |

Content Provider: BB_Opera

| _id | Title | Url | Visits | Date | Created | Description | Bookmark | Favicon | Thumbna |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Google | [illegible] | 0 | 0 | 0 | | 1 | BLOB NOT SUPPORTED | BLOB NOT SUPPORTE |
| 2 | Verizon | [illegible] | 0 | 0 | 0 | | 1 | BLOB NOT SUPPORTED | BLOB NOT SUPPORTE |
| 3 | Picasa | [illegible] | 0 | 0 | 0 | | 1 | BLOB NOT SUPPORTED | BLOB NOT SUPPORTE |

**FIGURE 6.65**

viaExtract—Export to PDF.

The hardware-based methods required specialized and often expensive equipment and training but can be very effective on devices where root access is unattainable. The software-based physical techniques are a more direct path to acquisition and are often the best place to start. Of course, before software-based techniques are possible, you must have root access on the device.

## Hardware-Based Physical Techniques

The two hardware-based physical techniques are JTAG and chip-off and a brief overview is provided in this section.

### JTAG

The JTAG was created in the 1980s to develop a standard for testing the wiring and interconnects on printed circuit boards (PCB). By 1990, the standard was complete and became an Institute of Electrical and Electronics Engineers standard, specifically IEEE 1149.1-1990 (IEEE SA, n.d.), and then a later update in 2001 named IEEE 1149.1-2001. The standard was widely accepted and today most PCBs have JTAG test access ports (TAPs) that facilitate access to the central processing unit (CPU).

A JTAG TAP exposes various signals and most mobile devices include the following:

1. TDI—Test Data In
2. TDO—Test Data Out
3. TCK—Test Clock
4. TMS—Test Mode Select
5. TRST—Test Reset
6. RTCK—Return Test Clock

A major obstacle to JTAG is locating the TAPs on the PCB and tracing them to the CPU to determine which pad is responsible for each test function. This is very difficult to achieve even if the chip manufacturer has published a CPU map. In addition, it can be extremely difficult to trace the JTAG functions from the chip and it may require first removing the CPU from the PCB. Device manufacturers have JTAG schematics, but they are generally considered company confidential and are only released to authorized service centers. Another approach is to determine the functions of each pad by reading the voltage at each pad and, based on the reference voltage, identifying the function. In some instances the JTAG pin-outs are published by flasher box manufacturers or various online groups. Fig. 6.66 is an example of the JTAG pin-outs for a T-Mobile HTC G1. The six pin-outs are indicated by the small white circles and the legend on the right provides the detailed information.

In most cases, your soldered wire leads to the pads on the PCB, and the other side is connected to a special device (flasher box) which, through software, will

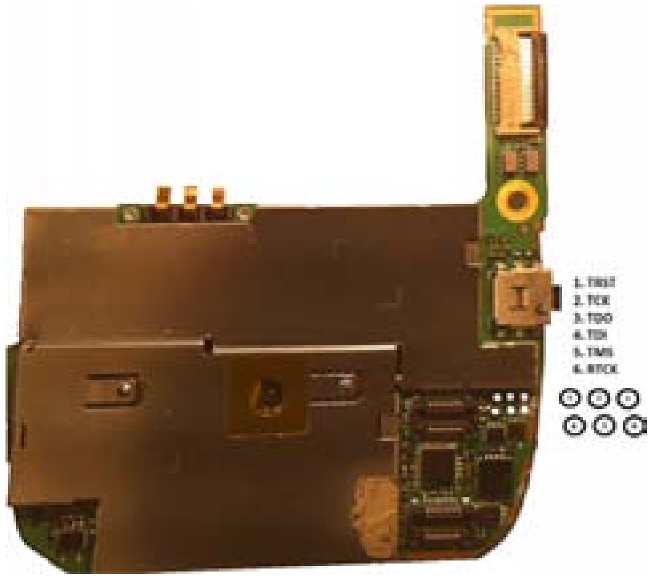


**FIGURE 6.66**

T-Mobile HTC G-1 PCB.

manage the CPU. Some companies make custom connectors which support a specific device and simplify the connection to the pads by placing the PCB between two jig boards with pogo pins. The pogo pins make contact with the JTAG pads on the PCB and can then easily connect to the flasher box. However, experienced engineers may find that soldering the leads directly to the PCB provides a more stable connection.

Once the leads are connected to the appropriate pads, power must be applied to the board to boot the CPU. Each CPU manufacturer publishes the reference voltage for their hardware and this voltage must not be exceeded. Some flasher boxes provide an option for managing the voltage but in general the power should be managed through an external power supply with a built-in digital voltmeter to ensure accuracy. Once the board is powered on, the flasher box software has the ability to perform a full binary memory dump of the NAND flash. However, the connection is serial and takes a considerable amount of time. Despite all of the complexities, if the JTAG technique is executed properly, the phone can be reassembled and will function normally with no data loss.

Though JTAG is an option for extracting data from an Android device's NAND flash, it is very difficult and should only be attempted by qualified personnel with sufficient training and specific experience in soldering small PCB connections. Errors in soldering to the JTAG pads or applying the wrong voltage to the board could not only disable JTAG but can also seriously damage the device. For these reasons, JTAG is not typically the first choice for a physical forensic image of an Android device.

## Chip-off

Chip-off is a technique where the NAND flash chips are physically removed from the device and examined externally. The chip-off technique allows for the recovery of damaged devices and also circumvents pass code-protected devices. This removal process is generally destructive—it is quite difficult to reattach the NAND flash to the PCB and have the device operate.

There are three primary steps in the chip-off technique:

1. The NAND flash chip is physically removed from the device by either de-soldering it, or using special equipment that uses a blast of hot air and a vacuum to remove the chip. There are also techniques that heat the chip to a specified temperature. It is quite easy to damage the NAND flash in this process and specialized hardware, and even controlling software, exists for the extraction.
2. The removal process often damages the connectors on the bottom of the chip, so it must first be cleaned and then repaired. The process of repairing the conductive balls on the bottom of the chip is referred to as reballing.
3. The chip is then inserted into a specialized hardware device, so that it can be read. The devices generally must be programmed for a specific NAND flash chip and support a number of the more popular chips already.

At this point, you now have a physical image of the data stored on the NAND flash chip.

Although the chip-off process is quite effective, it also has a large barrier to entry. The cost of the equipment and tools is prohibitive and an examiner must again have very specialized training and skills. There is always the risk that the NAND flash chip will be damaged with chip-off, generally in its removal from the PCB. Finally, a clean room with protections from static electricity is also desirable. While local or even State law enforcement agencies and forensic firms may find the cost of chip-off too prohibitive, it is certainly a valid techniques that larger agencies would find useful in their suite of forensic techniques.

## Software-Based Physical Techniques and Privileges

Software-based physical techniques have a number of advantages over the hardware-based techniques. Software-based techniques:

- Are easier to execute;
- Often provide direct access to file systems to allow a complete copy of all logical files (simplifies some analysis);
- Provide very little risk of damaging the device or data loss.

To execute the software-based physical techniques, you first must gain root privileges and then run the acquisition programs.

Unfortunately, root privileges on Android devices are not enabled by default. However, it is possible to gain root privileges in certain scenarios, several of which we will cover next. There are some major challenges to obtaining root privileges to keep in mind though:

1. Gaining root privileges changes the device in many situations.
2. The techniques for root privileges differ not only for each manufacturer and device but for each version of Android and even the Linux kernel in use. Just based on the Android devices and versions developed to date, there are literally thousands of possible permutations.
3. Many of the exploits used to gain root privileges are discussed online and often contain inaccurate information.

Given this, gaining root privileges can be quite difficult and is always very frustrating.

There are three primary types of root privileges:

1. Temporary root privileges attained by a root exploit, which does not survive a reboot. Typically the adb daemon is not running as root in this instance.
2. Full root access attained through a custom ROM or persistent root exploit. Custom ROMs often run the adb daemon as root while most of the persistent root exploits do not.
3. Recovery mode root attained by flashing a custom recovery partition or part of a custom ROM. Custom ROMs often run the adb daemon as root as do most of the modified recovery partitions.

Android enthusiasts who want root access are typically only interested in full, sustained root privileges. However, from a forensics standpoint, temporary root privileges or root access via a custom recovery mode are preferred.

If you need to gain access on a new device or Android version, you must have a separate device used for testing to ensure that the process works and no data are lost. Testing, although time consuming, is an important step in this situation.

The following sections cover each step in detail.

### *su*

The first thing a forensic examiner should check is whether the device already has root privileges. This is the easiest of any technique discussed and is certainly worth the short time it takes to test. The device must have USB debugging enabled but even if the device is locked, you should still check. If the device is not pass code locked, make sure USB debugging is enabled, which was covered in Chapter 3.

Next, connect the device to your workstation and attempt to gain root privileges by requesting super user access with the "su" command as follows:

```
ahoog@ubuntu:~$ adb shell su
su: permission denied
```

In this instance, root privileges were not granted. This is the typical result of the command. However, the following was on a device that had root access:

```
ahoog@ubuntu:~$ adb shell su
#
```

Instead of receiving a permission denied error, root privileges were granted. This is indicated by the new # prompt. Sometimes a device will allow root access but require the user to grant the privileges by clicking OK on a prompt displayed on the device. If the device is not pass code protected, you should check to see if the prompt is displayed.

### Researching Root Privilege Exploits

If the device does not already have root privileges, you can research possible techniques online. This process can be very frustrating as there are many inexperienced people who request help on the various discussion boards. However, while there are substantial amounts of inaccurate information, there are also very knowledgeable resources and techniques which do work.

Although there are many sites available that discuss Android root exploits, one truly stands above all others. The site, xda-developers, is an extremely popular and active site and is self-described as "the largest Internet community of smartphone enthusiasts and developers for the Android and Windows Mobile platforms" (Android & Windows Phone, n.d.). Many of the other web sites post various root exploits but generally link back to a discussion thread on xda-developers.

Often the best approach to researching root exploits is to simply search the Internet with your preferred search engine, have a test device, and a lot of patience.

### Recovery Mode

Recovery mode is an operating mode for Android that was designed to apply updates, format the device, and perform other maintenance on the devices. The stock recovery mode on most devices is very basic, only provides a number of limited functions, and certainly does not provide root privileges in a shell.

Custom recovery partitions, on the other hand, nearly always allow root privileges through the shell. These new recovery partitions are typically installed by the user when the device is rooted and provide various functions that simplify the backup and update processes needed from the custom ROMS.

As with researching root exploits, examiners should use extreme caution when installing a custom recovery partition as the process often contains kernel and radio updates that could render the device unusable (often referred to as "bricked") if there are incompatibilities between the device, kernel, and radio firmware. Extensive testing must be performed on a lab device first to ensure no issues occur. And forensic examiners should understand what is being modified on the device during the installation of a custom recovery firmware.

The software that powers recovery mode is stored on a dedicated partition and is quite small. On many devices, you can see details of the recovery partition by examining /proc/mtd:

```
ahoog@ubuntu:~$ adb shell cat /proc/mtd
dev:    size   erasesize  name
mtd0: 00040000 00020000 "misc"
mtd1: 00500000 00020000 "recovery"
mtd2: 00280000 00020000 "boot"
mtd3: 04380000 00020000 "system"
mtd4: 04380000 00020000 "cache"
mtd5: 04ac0000 00020000 "userdata"
```

This list is from a T-Mobile HTC G1 and you can see that the recovery partition has a size of 0x500000 bytes, which is 5 MB (0x500000 = 5,242,880 then divide by 1024 twice to convert to KB and finally MB). Here are the sizes from other phones used throughout this book:

- T-Mobile HTC G1: 5 MB
- HTC Incredible: 4 MB
- Motorola Droid: 4 MB
- Google Nexus One: 4 MB

This is helpful to understand as we explore techniques to replace the small but important recovery partition in the next section.

In the previous section covering techniques for circumventing pass code-protected devices, accessing the recovery mode was one possible solution. In the same fashion, it is advisable to check the recovery partition for root privileges as it will enable the software-based physical techniques. First, boot the device into recovery mode as covered in Table 6.2, or simply search the Internet for the specific key combination for your device. Once the device is in recovery mode, connect it to your Ubuntu VM and run adb as follows:

```
ahoog@ubuntu:~$ adb devices
List of devices attached
0403555551222244F       recovery
```

In this case, adb discovered a device in recovery mode. However, many devices will simply not have adb access enabled in recovery mode, especially on stock devices. In such a case, you can then determine if the shell has root privileges as follows:

```
ahoog@ubuntu:~$ adb shell
#
```

As we discussed previously, if you are presented with a # prompt, this indicates root privileges. If instead you have a $ prompt, you do not have root privileges. However, you should at least try to gain them by running the su command.

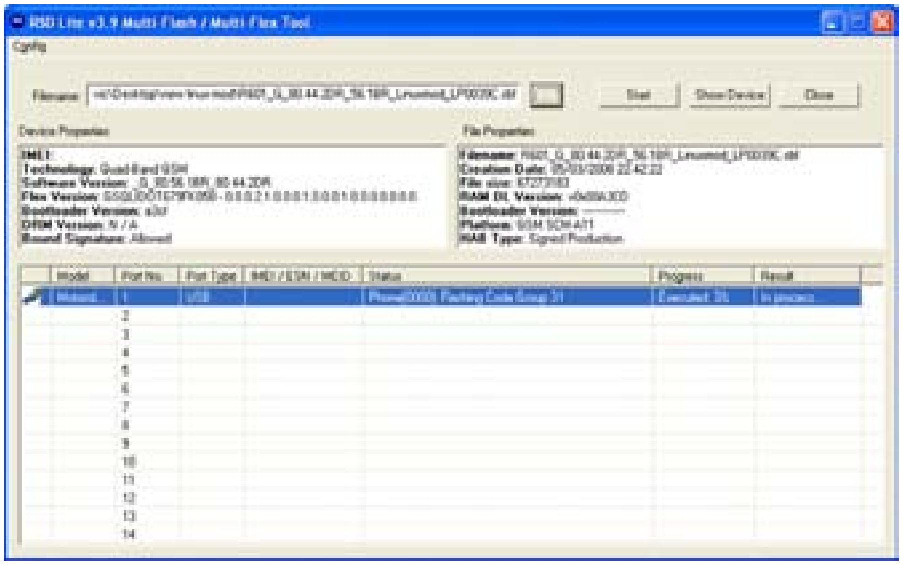
### Boot Loaders

As discussed in Chapter 2, the boot loader is a small program that is executed early in the Android boot process and is responsible for, among other details, selecting and loading the main kernel. On certain devices, special software exists, typically developed by the manufacturer, which can interact with the boot loader. This software is capable of writing new images to the NAND flash of a device. Manufacturers use this software to fix nonfunctional devices and likely in other situations such as development and testing. Forensic examiners can also use the software to flash a utility or exploit to a device's NAND flash, which will provide root privileges. However, the boot loaders of most devices are shipped from the factory in a locked state, which prevents such updates.

One example of software that interacts with Motorola Android devices is a program called RSD Lite developed by Motorola. RSD Lite is proprietary software and appears to only be distributed to Motorola Service Centers for device repair. It is assumed that anyone using this software has full authorization to do so, and this overview is only provided as a example of how some Android devices are flashed.

There are many web sites which discuss RSD Lite and provide guides for using the software. One such site, modmymobile.com, provides an article entitled "[Guide] Flashing Linux Motorola's with RSD Lite Versions," which offers step-by-step instructions for the software ([Guide] Flashing Linux, n.d.).

Provided the device is supported and the boot loader is unlocked, you connect the device to your workstation and then run the software, which detects the phone. You must then provide the appropriate .sbf file and then click Start to flash the device as shown in Fig. 6.67.



Filename: [illegible]

**IMEI:**
**Technology:** Quad Band GSM
**Software Version:** [illegible]
**Flex Version:** [illegible]
**Bootloader Version:** [illegible]
**DRM Version:** N / A
**Bound Signature:** Allowed

**Filename:** [illegible]
**Creation Date:** [illegible]
**File size:** [illegible]
**RAM DL Version:** [illegible]
**Bootloader Version:** --------
**Platform:** GSM SCM-A11
**HAB Type:** Signed Production

| | Model | Port No | Port Type | IMEI / ESN / MEID | Status | Progress | Result |
|---|---|---|---|---|---|---|---|
| | Motorol... | 1 | USB | | Phone[0000]: Flashing Code Group 31 | Executed 31 | In process... |
| | | 2 | | | | | |
| | | 3 | | | | | |
| | | 4 | | | | | |
| | | 5 | | | | | |
| | | 6 | | | | | |
| | | 7 | | | | | |
| | | 8 | | | | | |
| | | 9 | | | | | |
| | | 10 | | | | | |
| | | 11 | | | | | |
| | | 12 | | | | | |
| | | 13 | | | | | |
| | | 14 | | | | | |

**FIGURE 6.67**

RSD Lite.

After the process is completed and the device is restarted, the new recovery partition (and any other areas modified by the .sbf file used) is ready. RSD Lite may provide a good option for forensic analysts who have proper authorization to use RSD Lite.

### *sbf_flash*

Similar to Motorola's RSD Lite is a utility called sbf_flash that does not carry the license and usage restrictions of RSD Lite. The application was developed and posted online by an Android enthusiast and, while distributed in many places online, it is best to retrieve it from the author's blog OPTICALDELUSION, which is updated when new versions are available. This utility was developed on Linux, and now also runs on OS X, and thus, greatly simplifies the flashing of data to the NAND flash via an unlocked boot loader. The latest version of sbf_flash is 1.15 and it supports the following:

```
ahoog@ubuntu:~$ ./sbf_flash -h
SBF FLASH 1.15 (mbm)
http://opticaldelusion.org

Usage: ./sbf_flash <filename>

sbf_flash [options] [sbf file]
    -f               - force; attempt to continue on error
    -v               - verbose output (of CDT)
    -r               - read CDT information from the phone
    -x               - extract sbf file
    -d               - download cg from phone
    --cgname [file]  - upload/download cgname
                       matches any cgname shown with -r
                       optional file arg for contents
```

To use sbf_flash, you must first verify that the device is supported. For this example, we will cover the Motorola Droid; however, other devices are supported. The device must be placed in bootloader mode, which is accomplished by holding the up direction on the D pad while pressing the power button. The boot loader is easily recognized on the Motorola Droid by a solid black screen with the kernel version, USB status, and battery status in white text. Ensure the battery is fully charged before attempting this process as you could easily brick the device if the flashing process is interrupted with a power failure. Also, you must have the SBF file saved to your forensic workstation so you can flash it to the device. This is where extensive testing must occur prior to working on the target device to ensure compatibility and a detailed understanding of the process.

With the Droid in bootloader mode, we can query the device with sbf_flash as follows:

```
ahoog@ubuntu:~$ ./sbf_flash -r
SBF FLASH 1.15 (mbm)
http://opticaldelusion.org
```

```
 >> waiting for phone: Droid found.
 CG63 0xC0000000-0xC001FFFF mbmloader.img
 CG30 0xC0020000-0xC00BFFFF mbm.img
 CG55 0xC00C0000-0xC015FFFF mbmbackup.img
 CG31 0xC0160000-0xC01BFFFF cdt.bin
 CG38 0xD01CE000-0xD0359FFF pds
 CG34 0xC035A000-0xC03BCFFF lbl
 CG57 0xC03BD000-0xC041FFFF lbl_backup
 CG41 0xC0400000-0xC057FFFF sp
 CG42 0xC0580000-0xC061FFFF logo.bin
 CG44 0xC0620000-0xC067FFFF misc
 CG35 0xC0680000-0xC09FFFFF boot
 CG47 0xC0A00000-0xC0E7FFFF recovery
 CG39 0xD0EF4000-0xD9FB6FFF system
 CG40 0xD9FB7000-0xDFF3BFFF cache
 CG37 0xDFF3C000-0xF0D29FFF userdata
 CG53 0xDFD40000-0xDFF3FFFF kpanic
 CG54 0xDFF40000-0xDFFFFFFF rsv

Usage: ./sbf_flash <filename>
```

The sbf_flash utility looks for a device in bootloader mode and immediately flashes the image file to the NAND flash. The status of the update process is displayed on screen and afterwards the Droid is rebooted.

```
ahoog@ubuntu:~$ sbf_flash SPRecovery.sbf
=== SPRecovery.sbf ===
00: RDL03 0x80500000-0x8054CFFF DECE AP
01:  CG47 0xC0A00000-0xC0D5C7FF 02C0 AP

 >> waiting for phone: Droid found.
 >> uploading RDL03
Uploading: 100% OK
 >> verifying ramloader
 -- OK
 >> executing ramloader
Droid found.
 >> sending erase
 >> uploading CG47
Uploading: 100% OK
 >> verifying CG47
 -- OK
 >> rebooting
```

You should be prepared to immediately boot the device into recovery mode as later versions of the Motorola Droid's firmware implemented a routine that checks the hash signatures of the existing recovery partition against the stock recovery partition for that Android version. If there is a disparity, the system will rebuild the stock recovery partition during the boot process and thus overwrite the modified recovery image.

Once the new SBF file has been flashed, and the device is running in the modified recovery mode, you will have root access and can proceed with the software-based physical technique.

### *fastboot*

Fastboot is another utility that flashes images to the NAND flash over USB. The source code for fastboot is contained in the AOSP and thus, the utility is built when you compile the AOSP code. Like sbf_flash, the boot loader must support fastboot, which not only requires a compatible boot loader but also one that has security turned off (S-OFF).

Fastboot was first used on the Google Android developer phone (ADP), which was manufactured by HTC. As such, much of the documentation and references for fastboot refer to the ADP, and HTC has a helpful reference page for the utility (HTC—Developer Center, n.d.). This page contains not only various stock NAND flash image files for the ADP device but also directions on using fastboot and accessing the appropriate mode on the device (HTC—Developer Center, n.d.):

To enter fastboot mode, power up the device (or reboot it) while holding down the BACK key. Hold the BACK key down until the boot loader screen is visible and displays "FASTBOOT." The device is now in fastboot mode and is ready to receive fastboot commands. If you want to exit fastboot mode at this point, you can hold down the keys MENU+SEND+END (on the ADP, SEND is the "Call" key and END is the "End call" key).

Note that the boot loader screen may vary across devices. For ADP devices, the boot loader screen shows an image of skateboarding robots. Other devices may show a different image or color pattern. In all cases, the boot loader screen shows the text "FASTBOOT" when in fastboot mode. The boot loader also shows the radio version.

Once in fastboot mode, verify fastboot detects the device with the following command:

```
ahoog@ubuntu:~$ ./fastboot devices
HT08XHJ00657    fastboot
```

Fastboot provides many options that are detailed when you execute fastboot with the help parameter as follows:

```
ahoog@ubuntu:~$ ./fastboot --help
usage: fastboot [ <option> ] <command>

commands:
  update <filename>                         reflash device from update.zip
  flashall                                  flash boot + recovery + system
  flash <partition> [ <filename> ]          write a file to a flash partition
  erase <partition>                         erase a flash partition
  getvar <variable>                         display a bootloader variable
  boot <kernel> [ <ramdisk> ]               download and boot kernel
  flash:raw boot <kernel> [ <ramdisk> ]     create bootimage and flash it
  devices                                   list all connected devices
  reboot                                    reboot device normally
  reboot-bootloader                         reboot device into bootloader

options:
  -w                                        erase userdata and cache
  -s <serial number>                        specify device serial number
  -p <product>                              specify product name
  -c <cmdline>                              override kernel commandline
  -i <vendor id>                            specify a custom USB vendor id
```

As you can see, once in flashboot mode, it is quite simple to flash the modified recovery partition:

```
ahoog@ubuntu:~$ fastboot flash recovery modified-recovery-image.img
```

After this process completes, you can reboot the phone into recovery mode and proceed with the software-based physical imaging technique.

## AFPhysical Technique

The AFPhysical technique was developed by viaForensics to provide a physical disk image of Android NAND flash partitions. The technique requires root privileges on the device and should support any Android device. The technique, however, is not a simple process and the forensic analyst will have to adapt the technique for the specific device investigated. This is a direct result of the large variations in Android devices not only between manufacturers but between devices running different versions of Android.

The overall process for AFPhysical is quite simple:

1. Acquire root privileges on the target Android device.
2. Identify NAND flash partitions which need to be imaged.
3. Upload forensic binaries to the target Android device.
4. Acquire physical image of NAND flash partitions.
5. Remove forensic binaries if any were stored on nonvolatile storage.

Regardless of the technique, it is assumed you have root privileges on the device. For this example, we will use a Motorola Droid. As we are able to flash a modified recovery partition to a Motorola Droid, this technique will work on a device even if it is pass code locked.

After we have flashed the modified recovery partition and rebooted into recovery mode, connect the device to our Ubuntu VM and verify adb can locate the device by running adb devices.

```
ahoog@ubuntu:~$ adb devices
List of devices attached
040363260C006018        recovery
```

From there, access the shell to ensure you have root privileges:

```
ahoog@ubuntu:~$ adb shell
/ #
```

At this point, we need to understand more about the phone so we can decide what needs to be physically imaged. The first place to start is to examine the mounted file systems, if any:

```
/ # mount
rootfs on / type rootfs (rw)
tmpfs on /dev type tmpfs (rw,mode=755)
devpts on /dev/pts type devpts (rw,mode=600)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
/dev/block/mtdblock7 on /cache type yaffs2 (rw,nodev,noatime,nodiratime)
```

Now, we know that the device uses MTD for NAND flash access as well as YAFFS2. To determine partitions exposed by MTD, we execute the following:

```
/ # cat /proc/mtd
dev:    size   erasesize  name
mtd0: 000a0000 00020000 "mbm"
mtd1: 00060000 00020000 "cdt"
mtd2: 00060000 00020000 "lbl"
mtd3: 00060000 00020000 "misc"
mtd4: 00380000 00020000 "boot"
mtd5: 00480000 00020000 "recovery"
mtd6: 08c60000 00020000 "system"
mtd7: 05ca0000 00020000 "cache"
mtd8: 105c0000 00020000 "userdata"
mtd9: 00200000 00020000 "kpanic"
```

An examiner should choose to image all of the MTD partitions. However, for this example we will focus on mtd8, the user data partition.

As we are now prepared to acquire the device, it may be helpful to refer back to the NAND flash and file system topics in Chapter 4 if some of the terminology or data structures are confusing. There are four Android physical acquisition strategies you can use once you have a device with root access:

1. Full nanddump of all partitions, including data and OOB (preferred)
2. A dd image of partitions, which only acquires the data, not the OOB
3. A logical acquisition of files using tar
4. A logical acquisition of files using adb

In addition, there are two primary ways to save the acquired data from the device:

1. Use adb port forward to create a network between the Ubuntu workstation and Android device over USB
2. Place an SD card into the device, mount, and save locally

There are advantages to both approaches. With adb port forwarding, you do not need to insert your own device and can immediately create the files on your workstation. When you save to the SD card, the acquisition is much faster. Both approaches are valid and will be demonstrated here.

We will start with the full nanddump of the user data partition as this provides the most complete forensic copy of the data. To achieve the nanddump, you must have a version of nanddump compiled for the ARM platform. Cross-compiling nanddump is beyond the scope of this book. However, you can either search for the program on the Internet or follow directions that are also posted online.

**TIP**

**Cross-compiling for ARM**

Cross-compiling source code to run on the ARM platform can be quite difficult and there is sparse support for it online. One possible solution is to use Android's Native Development Kit (NDK) to build compatible binaries. Another option is to use Linux and install a cross-compiler such as Code Sourcery's G++ Lite 2009q3-67 for ARM GNU/Linux from http://www.codesourcery.com/sgpp/lite/arm/portal/release1039. Once a cross-compiler is installed, you must modify the source code's Makefile to indicate the cross-compiling option. Also check this book's web site at http://viaforensics.com/education/android-forensics-mobile-security-book/ for future updates.

To avoid writing any data to the NAND flash, we can again examine the output of the mount command and take note that the "/dev" directory is tmpfs and thus, is stored in RAM. We can therefore push the forensic utilities to "/dev":

```
ahoog@ubuntu:~$ adb push AFPhysical/ /dev/AFPhyiscal
push: AFPhysical/tar -> /dev/AFPhyiscal/tar
push: AFPhysical/md5sum -> /dev/AFPhyiscal/md5sum
push: AFPhysical/nanddump -> /dev/AFPhyiscal/nanddump
push: AFPhysical/nc -> /dev/AFPhyiscal/nc
4 files pushed. 0 files skipped.
1003 KB/s (2803303 bytes in 2.727s)
```

Next we need to make the programs executable on the device. To achieve this, we use the chmod command, which changes the permissions of a file including the execute flag. We will set all files to allow any user to read or execute the program:

```
ahoog@ubuntu:~$ adb shell
/ # cd /dev/AFPhyiscal
/dev/AFPhyiscal # ls -l
-rw-rw-rw-    1 0        0          711168 Jan 24  2011 md5sum
-rw-rw-rw-    1 0        0          669799 Jan 24  2011 nanddump
-rw-rw-rw-    1 0        0          711168 Jan 24  2011 nc
-rw-rw-rw-    1 0        0          711168 Jan 24  2011 tar
/dev/AFPhyiscal # chmod 755 *
/dev/AFPhyiscal # ls -l
-rwxr-xr-x    1 0        0          711168 Jan 24  2011 md5sum
-rwxr-xr-x    1 0        0          669799 Jan 24  2011 nanddump
-rwxr-xr-x    1 0        0          711168 Jan 24  2011 nc
-rwxr-xr-x    1 0        0          711168 Jan 24  2011 tar
```

As you can tell, after we execute the "chmod 755" command on the programs, they each have the execute bit now set, which is represented by the "x" in the file permissions.

If you decided to save the nanddump to the SD card, ensure you place a properly formatted SD card in the device and that it is mounted on the system. Then we can execute nanddump as follows:

```
/ # /dev/AFPhyiscal/nanddump /dev/mtd/mtd8ro > /sdcard/af-book-mtd8.nanddump
ECC failed: 0
ECC corrected: 0
Number of bad blocks: 1
Number of bbt blocks: 0
Block size 131072, page size 2048, OOB size 64
Dumping data starting at 0x00000000 and ending at 0x105c0000...

/ # ls -l /sdcard/af-book-mtd8.nanddump
-rwxrwxrwx    1 0        0       283041792 Jan  1 00:12 /sdcard/
af-book-mtd8.nanddump
```

And ultimately either transfer to your Ubuntu VM using adb pull or remove the SD card and copy via a direct USB connection, which is much faster.

**NOTE**

**MD5 hash**

Although the user data partition was not mounted on the device during acquisition, the md5sum hash signature of "/dev/mtd/mtd8ro" will change even without any writes. This is due to the nature of NAND flash where the operating system and memory are in a nearly constant state of change from wear leveling, bad block management, and other mechanisms which occur despite the lack of changes to the user data. The best approach is to perform an md5sum of the resulting NAND flash file to ensure integrity from that point forward.

The second method for saving the NAND flash file or any other imaged data is to use netcat, which is a utility that allows you to redirect the output of a command to the network. For this approach, you will need two active terminal or ssh sessions. We will refer to them as Session0 and Session1. All of the Session0 commands will run on the Ubuntu VM and thus we will not go into the Android device shell from Session0. The commands which need to execute within the Android device's shell will all take place on Session1.

To begin, we first enable the network connection between the two endpoints using the adb port-forwarding capability:

```
SESSION0
--------------
ahoog@ubuntu:~$ adb forward tcp:31337 tcp:31337
```

This command essentially connects port 31337 on the Android device and the Ubuntu VM. Next, we execute nanddump on the Android device and pipe the output to netcat:

```
SESSION1
--------------
ahoog@ubuntu:~$ adb shell
/ # /dev/AFPhyiscal/nanddump /dev/mtd/mtd8ro | /dev/AFPhyiscal/nc -l -p 31337
ECC failed: 0
ECC corrected: 0
Number of bad blocks: 0
Number of bbt blocks: 0
Block size 131072, page size 2048, OOB size 64
Dumping data starting at 0x00000000 and ending at 0x105c0000...
```

Now that the Android device is sending the nanddump data over netcat, we need to receive it on the Ubuntu VM side:

```
SESSION0
--------------
ahoog@ubuntu:~$ nc 127.0.0.1 31337 > af-book-mtd8.nanddump
```

When nanddump completes, it simply exits without any additional output as does the netcat on the Ubuntu VM. We can verify that the nanddump was received on the workstation with ls:

```
SESSION0
--------------
ahoog@ubuntu:~$ ls -lh af-book-mtd8.nanddump
-rw-r--r-- 1 ahoog ahoog 270M 2011-02-26 20:58 af-book-mtd8.nanddump
```

At this point, you could continue to physically image the MTD partitions needed for the investigation, which should include at least the user data and the cache partitions.

In Chapter 7, we provide a program that will allow you to extract the OOB data from a nanddump to assist with forensic processing such as file carving. As you can generate the dd image in this manner, there is no need to acquire a dd image using the Android device. However, dd is built into Android and so we provide this example which is similar to the use of the nanddump example, except it uses the dd utility, and so does not capture OOB data. This example uses the reference HTC Incredible.

```
SESSION0
--------------
ahoog@ubuntu:~$ adb forward tcp:31337 tcp:31337

SESSION1
--------------
ahoog@ubuntu:~$ adb shell
$ su
# cat /proc/mtd
dev:    size   erasesize  name
mtd0: 000a0000 00020000 "misc"
mtd1: 00480000 00020000 "recovery"
mtd2: 00300000 00020000 "boot"
mtd3: 0f800000 00020000 "system"
mtd4: 000a0000 00020000 "local"
mtd5: 02800000 00020000 "cache"
mtd6: 09500000 00020000 "datadata"
# dd if=/dev/mtd/mtd6 bs=4096 | /dev/AFPhyiscal/nc -l -p 31337
38144+0 records in
38144+0 records out
156237824 bytes transferred in 182.898 secs (854234 bytes/sec)

SESSION0
--------------
ahoog@ubuntu:~$ nc 127.0.0.1 31337 > dd of=htc-datadata.dd bs=4096
```

Due to variations in Android devices, MTD, YAFFS2, and other nuisances, it is not always possible to mount the acquired nanddump image and extract the logical files. As you already have sufficient privileges, it is best to extract the desired logical data. This can be accomplished using a recursive adb pull because the adb daemon running on the device has root privileges. You can also use a utility such as tar to copy the data into a single archive file. In either instance, you must ensure the desired file system is mounted. Some of the modified recovery partitions provide a user interface for mounting the file systems. However, you can also do this on the command line and mount the file system read only. On the Motorola Droid referenced above, do the following:

```
SESSION1
--------------
/ # mount -o ro -t yaffs2 /dev/block/mtdblock8 /data
/ # mount -o ro,remount -t yaffs2 /dev/block/mtdblock7 /cache
/ # mount
rootfs on / type rootfs (rw)
tmpfs on /dev type tmpfs (rw,mode=755)
devpts on /dev/pts type devpts (rw,mode=600)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
/dev/block/mtdblock7 on /cache type yaffs2 (ro)
/dev/block/mmcblk0p1 on /sdcard type vfat
(rw,nodev,noatime,nodiratime,fmask=0000,dmask=0000,allow_utime=0022,
codepage=cp437,iocharset=iso8859-1,errors=remount-ro)
/dev/block/mtdblock8 on /data type yaffs2 (ro)
```

The first command mounts the "/data" partition read only. The second command takes the already mounted "/cache" directory and remounts it read only. You can now perform the adb pull:

```
ahoog@ubuntu:~$ adb pull /data/data/com.android.providers.telephony sms
pull: building file list...
pull: /data/data/com.android.providers.telephony/databases/mmssms.db ->
sms/databases/mmssms.db
pull: /data/data/com.android.providers.telephony/databases/telephony.db ->
sms/databases/telephony.db
2 files pulled. 0 files skipped.
137 KB/s (44032 bytes in 0.311s)
```

The final option is to use the tar utility that places files and directories in a single archive often called a tarball.

```
SESSION1
--------------
ahoog@ubuntu:~$ adb shell
/ # /dev/AFPhyiscal/tar cpv -f - /data/data/com.android.providers.telephony
/cache | /dev/AFPhyiscal/nc -l -p 31337
tar: removing leading '/' from member names
data/data/com.android.providers.telephony/
data/data/com.android.providers.telephony/lib/
data/data/com.android.providers.telephony/databases/
data/data/com.android.providers.telephony/databases/telephony.db
data/data/com.android.providers.telephony/databases/mmssms.db
cache/
cache/recovery/
cache/recovery/log
cache/lost+found/

SESSION0
--------------
ahoog@ubuntu:~$ nc 127.0.0.1 31337 > af-book-droid-files.tar
```

In this example, we passed two directories to tar which we wanted archived: the directory containing SMS/MMS messages in "/data/data" and the "/cache" directory. We sent the archive over the network and received it on the Ubuntu VM. However, you could have also simply saved the archive to the SD card.

Once you have root privileges on an Android device and sufficient understanding of the device's architecture, you can use nanddump, dd, tar, netcat, and adb to create forensic images or simply copies of the data for analysis.

## SUMMARY

There are several techniques that can be used to perform a forensic acquisition of an Android device. If the device is pass code protected, you must circumvent or bypass the protection to extract data. While a number of techniques to circumvent the pass code exist, it is not possible to achieve this in every circumstance. Once the device is accessible, the forensic analyst can choose from a logical acquisition which focuses primarily on undeleted data accessible through Content Providers or the more thorough but technically challenging physical acquisition. While the physical acquisition will produce more data, it generally requires more sophisticated analysis techniques which will be covered in Chapter 7.

## References

*ACPO Good Practice Guide for Computer-Based Electronic Evidence—7Safe Information Security.* (n.d.). Retrieved February 19, 2011, from http://7safe.com/electronic_evidence/index.html#.

*Android & Windows Phone: Tablets, Apps, & ROMs @ xda-developers.* (n.d.). Retrieved February 23, 2011, from http://www.xda-developers.com/.

Aviv, Gibson, Mossop, Blaze, & Smith. (n.d.). Smudge attacks on smartphone touch screens. Retrieved February 21, 2011, from http://www.usenix.org/events/woot10/tech/full_papers/Aviv.pdf.

Cannon, T. (n.d.). Android lock screen bypass. Retrieved February 21, 2011, from http://thomascannon.net/blog/2011/02/android-lock-screen-bypass/.

dc3dd. (n.d.). Retrieved February 22, 2011, from http://dc3dd.sourceforge.net/.

*Government Employment & Payroll.* (n.d.). Retrieved February 19, 2011, from http://www.census.gov/govs/apes/.

[Guide] Flashing Linux Motorola's with RSD Lite Versions. (n.d.). Retrieved February 24, 2011, from modmymobile.com/forums/8-guides-downloads-forum-suggestions/218651-guide-flashing-linux-motorolas-rsd-lite-versions.html.

*HTC—Developer Center.* (n.d.). Retrieved February 28, 2011, from http://developer.htc.com/adp.html.

*IEEE SA—1149.1—1990—IEEE Standard Test Access Port and Boundary-Scan Architecture.* (n.d.). Retrieved February 23, 2011, from http://standards.ieee.org/findstds/standard/1149.1-1990.html.

*RerWare, LLC: Android Backup and BlackBerry Backup—MyBackup Pro.* (n.d.). Retrieved February 22, 2011, from http://www.rerware.com/.