pdd: Memory Imaging and Forensic Analysis of Palm OS Devices

Joe Grand
`jgrand@mindspring.com`

Abstract

One goal of incident response is to preserve the entire digital crime scene with minimal or no modification of data. This paper introduces `pdd` or "Palm dd", a Windows-based tool for memory imaging and forensic acquisition of data from the Palm operating system (OS) family of Personal Digital Assistants (PDAs). `pdd` will preserve the crime scene by obtaining a bit-for-bit image or "snapshot" of the Palm device's memory contents. Such data can be used by forensic investigators, incident response teams, and criminal and civil prosecutors.

This paper also presents the Palm OS internals (hardware, file system, and debugger functionality), `pdd` details[1] (usage, process, flowchart, and timing), and forensic analysis results (flash memory, record removal and deletion, retrieval of system passwords, and telephony applications).

1 Introduction

PDAs are ubiquitous in the consumer marketplace and it is only natural that they will, as desktop and laptop computers have, become a target for criminal investigations and forensic analysis. `pdd` or other tools that aid in data acquisition and analysis of portable devices should be readily available in any incident response toolkit, as should any tool that maximizes an investigator's ability to collect credible digital evidence.

The Palm OS has been licensed to a number of vendors including Handspring, Sony, IBM, Kyocera, Samsung, QUALCOMM, Franklin Covey, TRG, and Symbol Technologies. Devices running Palm OS own nearly 80 percent of the global handheld computing market[2], equal to approximately 20 million devices, and consist of consumer-based PDAs, telephones integrated with PDA functionality, and barcode and wireless integration for industrial applications. `pdd` has been designed to work with all devices running Palm OS.

[*]Published by the Forum of Incident Response and Security Teams in the *Proceedings of the 14th Annual Computer Security Incident Handling Conference*, Waikoloa, Hawaii, June 24-28, 2002.

[1]The examples and descriptions of `pdd` are for release version 1.1 and may change as the tool is updated.

[2]IDC, December 2000.

In addition to providing a complete, unmodified memory image of the specified memory card, `pdd` retrieves and displays the following device information:

- Card Number
- Manufacturer Name
- Creation Date
- Processor Type
- Free RAM
- ROM Used By OS
- Card Name
- Card Version
- Palm OS Version
- RAM Size
- ROM Size
- Flash ID (if available)

The memory image of the device includes all user applications and databases, which can contain log data, completed 'To Do' items, 'Private' records, passwords, cryptographic components, and other potentially useful information for incident response or forensic analysis purposes. Stray databases that old applications left behind are also retrieved, as are records that have been marked for deletion (which are not removed from the device until the next HotSync operation).

The Palm OS Console Mode Debugger is used to acquire memory card information and to create the image of the selected memory region. Because the operation of `pdd` relies solely on the built-in Palm OS Debugger functionality and does not require any additional software components loaded on the device, the data retrieved is likely to be correct and not tampered with. No third-party applications need to be running on the target system in order for the `pdd` process to work.[3] In some cases, the Palm OS Debugger can be invoked even if the system is password-protected. Other existing Palm acquisition tools make use of the Palm HotSync functionality, which is disabled when the Palm system lock-out is active.

2 Palm OS Details

2.1 Hardware

At the time of this writing, all hardware devices that run the Palm operating system use the Motorola DragonBall MC68328-family of microprocessors which are based on the Motorola 68K core.[4] The original DragonBall processor, released in 1995, has been joined by two subsequent updates, the DragonBall EZ and the DragonBall VZ. Each release has become more streamlined for use in PDAs and other portable devices. Two next-generation processors, the DragonBall Super VZ and the DragonBall MX1, are due for sampling in 2002. The DragonBall MX1 is based on the ARM core architecture. Table 1 shows a selection of Palm OS devices and their core processors.

Palm devices consume battery power even when they are "turned off". If left unattended for an extended period of time, the batteries will deplete and all data stored in RAM will be destroyed.[5] Recharging or replacing the batteries of the device and monitoring the battery levels should be done diligently. For long-term storage of Palm devices containing internal rechargable batteries, it is suggested that each device is constantly connected to a recharging cradle or travel cable to ensure that battery levels are at their maximum.

[3]Since the Palm Debugger uses Remote Procedure Calls (RPCs) to call functions within the Palm OS Application Programming Interface (API), it may be possible for malicious code to trap those API calls to perform other functionality (such as transmitting incorrect data back to `pdd`). [2]

[4]`http://e-www.motorola.com`

[5]The contents of ROM and Flash will not be affected by loss of battery power, as these memory devices are non-volatile and do not require power to retain data.

Device	Processor Type
USRobotics Pilot 1000/5000	DragonBall
USRobotics PalmPilot Personal/Professional	DragonBall
3Com Palm III	DragonBall EZ
3Com Palm Vx	DragonBall EZ
IBM WorkPad c3 (Palm V)	DragonBall EZ
Kyocera QCP 6035 smartphone	DragonBall EZ
Palm Computing m105	DragonBall EZ
Palm Computing m500/m505	DragonBall VZ
Handspring Visor Prism	DragonBall VZ
Handspring Treo 180	DragonBall VZ

Table 1: A sampling of Palm OS devices and their core processors.

2.2 File System

Palm OS does not use a flat file system as do many traditional operating systems. Battery-backed Random Access Memory (RAM) is used for non-volatile storage and is logically divided into *dynamic* and *storage* memory (Figure 1). The Palm OS Memory Architecture documentation [4] provides detail of the Palm OS memory structures and examines the basic building blocks of Palm OS memory: heaps, chunks, and records.

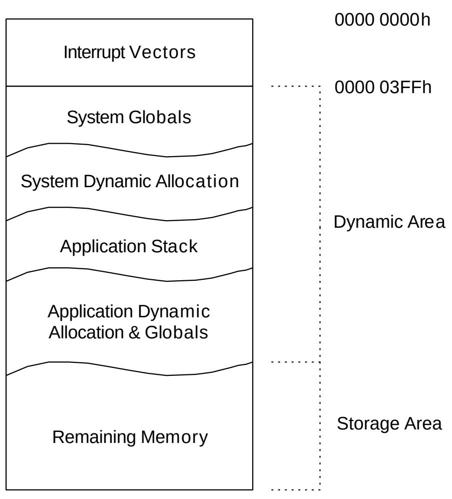

Figure 1: Conceptual view of RAM showing interrupt vector locations and dynamic and storage usage.

Dynamic memory, analogous to RAM installed in a typical desktop system, is used as working space for the program stack and short-term data (e.g., pen strokes, key presses, system events, video memory, global variables, and user interface structures). The size of the dynamic memory region depends on the OS version and on the total memory on the Palm device and changes constantly during device use. The remainder of RAM is used as storage memory, analogous to a disk drive. Regardless of the logical division of RAM, `pdd` images the entire RAM region which contains both dynamic and storage memory areas.

The operating system and other non-transient components are often stored in Read-Only Memory (ROM). Newer devices are moving towards Flash memory as a replacement. This allows the contents of memory to be rewritten which is useful for operating system upgrades.

The memory for each Palm OS device resides on a module known as a *card*. Each card is a logical definition and does not necessarily correspond to a physical card. Palm OS is designed to be modular in that multiple cards (each containing RAM, ROM, or both), can be functional in a given device. This allows for future expansion in which a Palm device might contain an internal card (e.g., RAM and ROM for the operating system) and an external card (e.g., user application and data storage). For example, Palm devices such as the m125, m500, m505, and i705 contain an external expansion slot for SecureDigital/MultiMediaCard (SD/MMC) memory cards. The Sony Clié family of Palm devices supports the Sony Memory Stick. Handspring devices support SpringBoard Modules. The MemPlug series of SpringBoard Modules can read and write to CompactFlash, SmartMedia, SD/MMC, or Sony Memory Stick. Card number '0' is defined by Palm, Inc. as the single internal RAM and ROM module. External memory modules will have a non-'0' card number.

The DragonBall Super VZ and MX1 processors contain embedded support for SD/MMC and Sony MemoryStick, which may lead to more portable devices accepting external memory cards.

2.2.1 Memory Map

The base starting addresses for the RAM and ROM areas of Palm devices vary depending on the processor type, as does the memory range allocated to each memory card. `pdd` will determine the processor type employed in the target device using the `pdd_FtrGet` function and will configure the base addresses (within the DragonBall's possible 4GB of address space) as necessary. Figure 2 shows the memory maps and relevant regions for typical Palm OS devices using DragonBall, DragonBall EZ, or DragonBall VZ processors.[6]

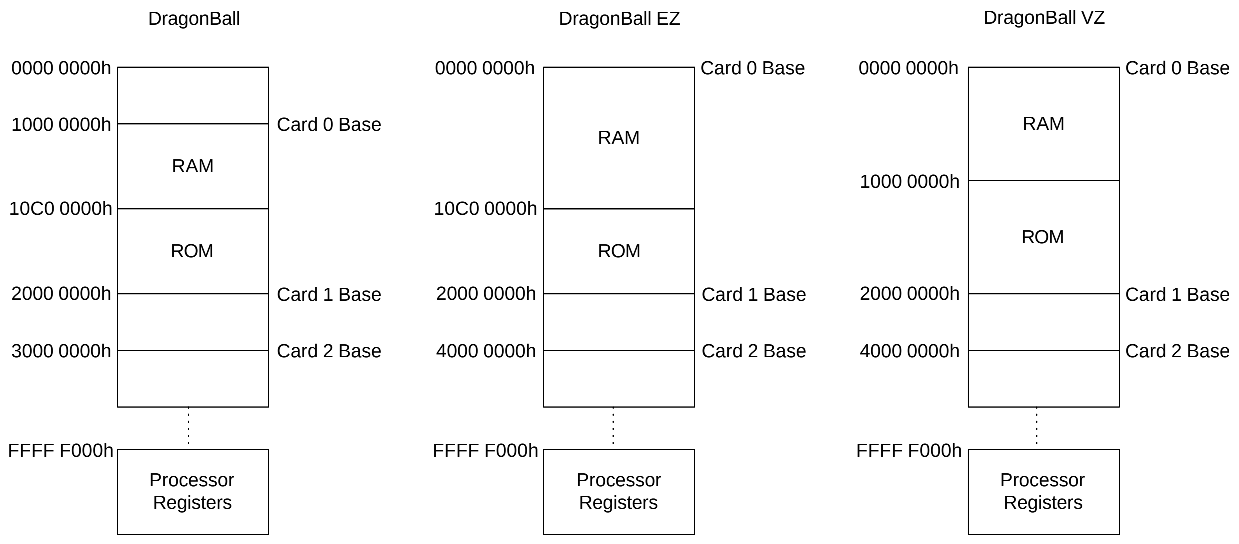

Figure 2: Base addresses of relevant regions depending on processor type.

[6]Independently created, detailed memory maps are available from `http://www.massena.com/darrin/pilot/PilotHackTutorial.htm` (DragonBall) and `http://www.angelfire.com/al/freemb/memmap.html` (DragonBall EZ).

2.2.2 Database Structure

The Palm File Format Specification [3] provides a detailed view into the file formats supported by Palm OS. In general, all data conforms to one of the three defined file types:

• **Palm Database (PDB).** Record database used to store application or user data.

• **Palm Resource (PRC).** Resource database (similar structure to a Palm database). Applications running on Palm OS (also referred to as "programs", "executables", or "binaries") are simply resource databases containing code and user interface resource elements.

• **Palm Query Application (PQA).** Palm database containing world-wide-web content for use with Palm OS wireless devices.

Databases are similar to files on a disk drive; the difference being that they are stored in a sequence of memory chunks called *records* or *resources* instead of in one contiguous area. The database file in device memory contains header information and a sequential list of pointers to records or resources. The records within a database are similarly structured with record header information and record data.

In each database header, there are three timestamps that are helpful in a forensic investigation. Each timestamp is a 4-byte value stored in seconds since 1/1/1904: `creationDate`, the creation date of the database, `modificationDate`, the time of the last modification of the database, and `lastBackupDate`, the time the database was last backed up (if ever). Due to the current lack of memory protection on Palm devices, these timestamps (as with any other data) can be fabricated or modified.

Knowing the structure of the Palm file formats is important if the forensic investigation requires parsing of the raw memory image provided by `pdd` into the individual PDB, PQA, and PRC files that were stored on the handheld device. However, Palm reserves the right to store data in memory using a different format than that used for files stored on a PC: "Currently, the 'on-disk' format, which is the Desktop format, is similar, but not identical, to the handheld format, which is 'in-memory'." [3]

This may pose a problem in the future if Palm OS is licensed to vendors who choose to change the memory structures to fit their particular application. For Palm devices running OS 4.0 or earlier, the in-memory and on-disk formats appear to be similar enough to rely on the Palm OS developer's documentation for parsing and analysis.

PDB Reader[7] is a Windows-based application to view PDB files. Source code is available which could be modified to parse raw memory images, such as provided by `pdd`.

With the announcement and subsequent arrival of Palm OS 5.0[8], which has major changes to the underlying software architecture, care must be taken to ensure that all forensic tools support proper memory acquisition techniques and that the parsing and analysis process conforms to the OS 5.0 data structures.

[7] `http://pws.chartermi.net/~sckienle/palm/pdbreader.zip`
[8] `http://www.palmsource.com/palmos/intro_os5.html`

2.3 Console Mode Debugger

Designed into Palm OS is a serial-based "Palm Debugger," which provides source- and assembly-level debugging of Palm OS executables on the physical device [5]. Entering a short keystroke combination (Figure 3), the Palm device enters "Console Mode," one of the two interfaces provided by the Palm Debugger. First, draw the Shortcut symbol (a cursive 'L' character). Next, tap the stylus twice to generate a dot (a period). Finally, write the number "2". The Palm Debugger, once enabled, listens on the RS232 serial port and Universal Serial Bus (USB) port (if it exists on the device) for communications from the host. A "soft reset" of the Palm device will exit the Palm Debugger. Power consumption is significantly increased when the Palm Console Mode is enabled, so ensure that the Palm device has full battery capacity before beginning the acquisition process.

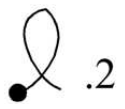

Figure 3: Graffiti stroke required to enable the Palm Console Mode. [5]

The Palm Debugger can be activated even if the Palm OS lockout functionality (using the Palm OS built-in "Security" application) is enabled [2]. This problem is verified to concern Palm OS versions 3.5.2 and earlier. The lockout functionality is assumed by most users to be a sufficient protection feature because a password is required before the device becomes operational. This potential security hole becomes a benefit when performing a forensic analysis. Even if the target device has been locked, a complete memory image of the device can still be obtained (e.g., using the `pdd` tool or by specific commands directly using the Palm Debugger). On devices running Palm OS 4.0 or greater that are placed in the "Turned Off & Locked" state, it appears that the Palm Debugger will not be activated. In this case, the device must be unlocked by entering the user-defined system password before the Palm Debugger can be enabled.

3 pdd Details

3.1 Usage

`pdd` is a Windows-based command line tool that takes the following operands as arguments:

Operand	**Description**	**Default Setting**
if=*value*	Name of serial communications port	COM1
of=*value*	Name of output file	Standard output
card=*n*	Palm card number	0 for built-in card
type=*value*	Palm card memory type; RAM or ROM	RAM
-? or -h	Help	

The `pdd` syntax is similar to that of the Unix-based 'dd' command, in which an input and output are specified along with optional operands (e.g., card number and memory type instead of the block size, data type, and other variables of 'dd').

Since `pdd` does not currently have USB support, the Palm device must be placed in the cradle or attached to a HotSync cable connected to a serial port on the desktop PC. If the arguments were properly specified on the command line, `pdd` will begin execution and display the following to standard error:

```
    Enter console debug mode [<shortcut> .. 2]
```

At this point, `pdd` is waiting for the Palm Console Mode to be enabled on the target device which is done by entering the Graffiti keystroke combination shown in Figure 3. The Palm OS device will enter the Console Mode of the Palm Debugger and will monitor the serial port for communication. No notification of Console Mode activity will be present on the Palm device.

Care should be taken to ensure that the "Auto Off" functionality of the Palm device is disabled during imaging. This can be done through the built-in "Preferences" application or with the Graffiti stroke shown in Figure 3, replacing the '2' with a '3'. If the device powers off during the imaging process, `pdd` may need to be re-started.

If `pdd` can properly communicate with the target Palm device, the imaging process will commence and display the following to standard error:

```
    pdd process beginning.
```

3.2 Process

Initially, `pdd` will retrieve and write the following device information to a file named `pdd.txt`, appending to the file if it already exists:

```
Current Time: Tue Mar 05 16:34:39 2002 UCT
Card Number: 0
Card Name: PalmCard
Manufacturer: Palm Computing
Card Version: 0001
Creation Date: Thu Dec 10 16:10:14 1998 UCT
Palm OS Version: 3.1.0
Processor Type: Motorola DragonBall 68EZ328
RAM Size: 2097152 bytes
Free RAM: 2042784 bytes
ROM Size: 2097152 bytes
ROM Used By OS: 1212412 bytes
Flash ID: 10FF1C795R3G-D
Image Output File: Standard output
Image Memory Type: RAM
Starting Address: $00000000
```

During the entire `pdd` process, critical error information is sent to standard error. After the initial device information has been saved, `pdd` will begin to retrieve the desired memory region from the Palm device, specified on the command line with the `card=` and `type=` operands. The raw memory image is written to standard output unless a filename is specified using the `of=` operand on the command line. `pdd` will continue to retrieve the Palm device memory until the entire region has been completed. Only when the imaging operation is done and successful, `pdd` will automatically soft reset the device in order to exit the debug mode. *<Ctrl>-C* can be used to abort the operation at any time. The device will not be soft reset and the Palm Console Mode will remain enabled.

A soft reset is currently the only method to disable the Palm Console Mode once it has been enabled. This may be a problem in some instances, as a soft reset will re-initialize the dynamic heap and low-memory globals, and clean and compact the storage heap [4]. Potentially, deleted database records (e.g., forensic evidence) could be destroyed (more details in §4.2). Because of this, there may only be one chance to obtain a clean capture, so caution should be taken when performing the initial memory dump as future captures and analysis results may be skewed.

If a cryptographic hash function, such as the MD5 Message-Digest Algorithm (defined in RFC 1321) or SHA-1 (defined in FIPS-180-1), is used on the `pdd` memory images for fingerprinting and integrity checking, the RAM images of subsequent acquisitions of the same device will not match due to the re-initialization of heaps as described above. Additionally, even if the device is not soft reset in between acquisitions, the dynamic heap (which is captured by `pdd`) changes constantly. For a ROM image, the hash will be identical for all subsequent acquisitions (unless data has been modified by the user).

3.3 Flowchart

Figure 4 shows the core operation of `pdd` which occurs in the `pdd()` function. Each routine beginning with '`pdd_`' prepares a Palm Debugger Protocol packet [5] for the desired Palm OS API call or debugger command. Internal details of these routines can be obtained directly from the `pdd` source code.

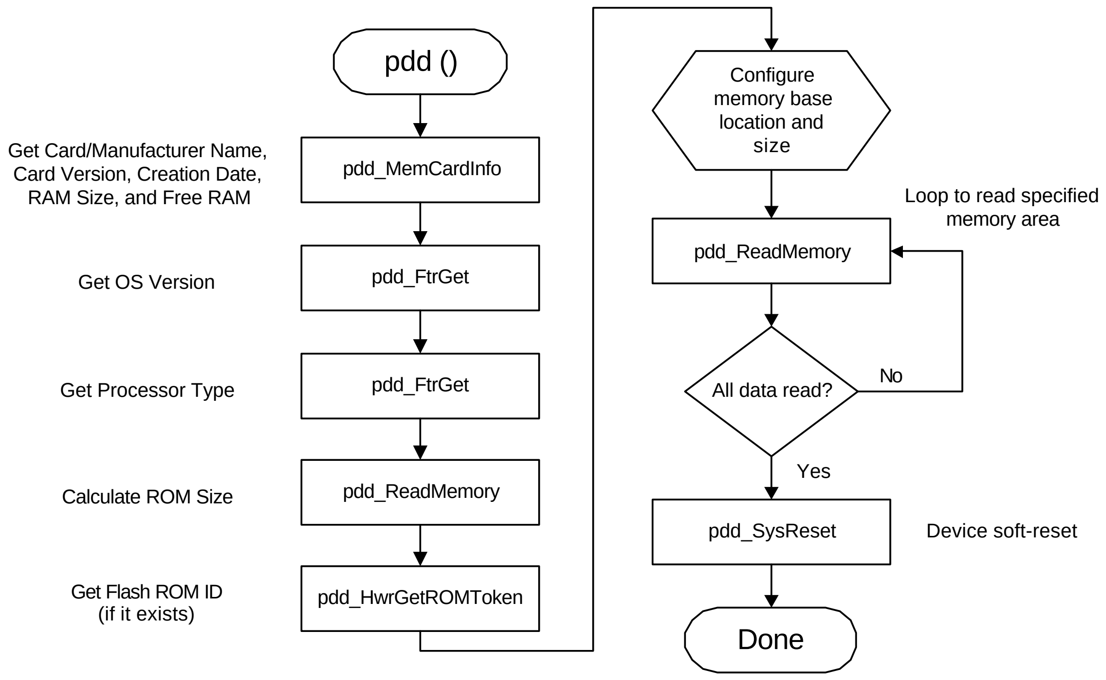

Figure 4: Flowchart of the core `pdd()` functionality.

Each Palm Debugger Protocol packet received from the Palm device contains a cyclic redundancy check (CRC-16) checksum (defined in ISO 3309 and ITU-T V.42) which is checked by `pdd`'s `verify_checksum` function to ensure packet integrity during the data transfer. If the CRC-16 checksum fails at any time during the imaging, `pdd` will exit. The device will not be soft reset and the Palm Console Mode will remain enabled.

`pdd` calculates the actual size of the ROM or Flash device by reading the chip-select registers of the DragonBall processor. This needs to be done because the Palm OS API `MemCardInfo` function used to obtain other card information returns only the size of ROM actually used by Palm OS, not the physical ROM size. It is critical to obtain the entire ROM space for forensic analysis purposes (as discussed in §4.1). It is believed that no other acquisition tool currently exists that retrieves the entire ROM space. The ROM image acquired by `pdd` can be loaded directly into the Palm OS Emulator (detailed in §4.5).

3.4 Timing

The Palm Console Mode defaults to a serial data transfer rate of 57.6kbps. Actual measured data transfer is closer to 29.1kbps. This reduced rate (≈50%) is attributed to the overhead involved in the Palm Debugger Protocol packet generation and parsing and the 256-byte maximum data chunk size per packet. Although the Palm supports serial port rates of up to 230.4kbps, it was determined that the times required to image memory were not greatly reduced at higher data rates and we felt it safer to leave the Palm Console Mode in its default configuration. Table 2 shows the approximate time required to image various memory sizes. These times are within reason if the Palm device is legitimately in possession and does not need to be quickly returned.

Memory Size (kB)	**Time Required** (≈mm:ss)
128	0:36
256	1:12
512	2:24
1024	4:48
2048	9:36
4096	19:09
8192	38:18
16384	76:36

Table 2: Time required for the transfer of various memory sizes using serial port @ 57.6kbps.

4 Forensic Analysis

The data retrieved by `pdd` includes all user applications and databases (along with stray databases that old applications left behind). This provides a significant amount of information, much of which can be extracted and analyzed using `strings`[9] to find printable strings in an object or binary file.

For example, 'To Do' items are often checked as 'completed' and then forgotten about by the user. However, the data still remains on the PDA. Applications that contain cryptographic functionality are also of interest. When encrypted data (e.g., a record or database) is selected to be used or viewed, it is decrypted and the cleartext component is stored in memory for a short length of time[10]. Many times, the data is not re-encrypted until the device is powered off. If `pdd` is run during this time, the cleartext data is still in memory and can be retrieved.

4.1 Flash ROM

Palm OS devices incorporating Flash memory currently use it solely for the storage of the operating system code. Depending on the family of Palm OS device, there remains between 440kB and 1656kB

[9] `http://www.simtel.net/pub/dl/gnuish/strings.zip`

[10] This time period is dependent on how the application implements cryptographic functions.

of unused memory space. Utilities exist, such as Handera's FlashPro[11] or Brayder's JackFlash[12], which take advantage of the unused memory areas of Flash ROM to backup applications and databases. Because of this, it is essential that the ROM of the device be imaged and analyzed during forensic acquisition.

The Flash ID, also known as the ROM Serial Number, is a 12-digit serial number intended to be a unique identifier of the Palm device. This identifier only exists on devices that contain Flash ROM. It is useful to use this number for tracking or documentation purposes, but since no memory protection mechanisms currently exist on Palm OS devices, it is possible to modify the contents of ROM including the area where the Flash ID resides [2]. It is not recommended that this identifier alone be relied upon for device tracking. The serial number can be also be viewed using the built-in Application Launcher application on the Palm device. A checksum digit is calculated during the `pdd_HwrGetROMToken` function to ensure proper transmission from the Palm device.

It should be noted that after a "hard reset" of the Palm device, all data stored in RAM is erased and no evidence of prior data exists. The contents of ROM and Flash are retained.

4.2 Record Removal and Deletion

Records that have been marked for deletion by applications using the Palm API `DmDeleteRecord` function (e.g., from the Address Book, Memo Pad, To Do List, Calendar, etc.), are not actually removed and will remain on the device until a successful HotSync operation to a desktop machine. As noted in the Comments section of the `DmDeleteRecord` function documentation [6], the function "marks the `delete` bit in the database header for the record and disposes of the record's data chunk. It does not remove the record entry from the database header, but simply sets the `localChunkID` of the record entry to `NULL`."

This is a very useful feature for forensic analysis, since if the Palm device is acquired before the HotSync has taken place but after records have been 'deleted', the data can still be recovered. A soft reset of the device will also cause deleted records to be destroyed, unless the `archive` bit is set in the database header.

4.3 Retrieval of System Passwords

The Palm OS system password is set by the user with the built-in "Security" application. The maximum length of the ASCII password is 31 characters. In all versions of Palm OS up to 4.0, it has been confirmed that an obfuscation or hash of the user's system password is stored on the device in the "Unsaved Preferences" database and is also transmitted over the serial cable, airwaves, and networks during a HotSync operation. Depending on the version of Palm OS, the system password is encoded and stored differently.

For Palm OS versions 3.5.2 and earlier, a weak obfuscation method is used to mask the actual ASCII password. Regardless of the length of the ASCII password, the resultant encoded block is always 32 bytes. Two methods are used to encode the ASCII password, depending on its length. For passwords of four characters or fewer, an index is calculated based on the length of the password and the string is XOR'ed (a logical operation) against a 32-byte constant block. For passwords of more than four characters, the string is padded to 32 bytes and run through four rounds of a function that XORs against a 64-byte constant block. By understanding the encoding schema, it is possible to essentially run the routines in reverse to decode the password. The obfuscation process and details are provided in [2].

[11] `http://www.handera.com/products/cat-flashpro.asp`
[12] `http://www.brayder.com/products/jackflash.html`

For Palm OS version 4.0, a 128-byte MD5 hash of the ASCII password is used instead of the weak obfuscation method described above. Even though MD5 is a one-way cryptographic function in which the hash cannot be reversed back into its original input component, it is still possible to perform a dictionary attack against the known MD5 hash. Using a large dictionary file, each word in the dictionary can be hashed and compared to the known hash until a match is found. This attack is viable due to the fact that users of portable devices, especially those that have no keyboard and require character input with a stylus (e.g., Graffiti), often choose short, easily guessable passwords. The chance is high that the user's password is a common word found in a dictionary.

Once the password is determined, one can further investigations by attempting to use this password on other computer or network equipment owned by the person under scrutiny.

4.4 Telephony Applications

For devices that integrate mobile telephones with PDA functionality, additional information is available for acquisition. For example, many phones contain call logs (which keep track of incoming, outgoing, or missed calls) that may be stored in a database on the device. The likelihood is high that PIN numbers, voice mailbox passwords, and other digits dialed and displayed on the device's screen are also be stored on the device.

For example, the Kyocera QCP 6035 smartphone[13] is a tri-mode digital cellular phone that runs Palm OS and supports multiple wireless data technologies. Other examples of Palm-based mobile telephones include the Handspring Treo, Qualcomm pdQ, and Samsung SPH-I300. Using `pdd` to image the Kyocera smartphone's RAM and `strings` to extract ASCII data, an informal investigation yielded the following information:

• **Header** containing the current time and date, HotSync ID, cellular telephone number, time zone, and cellular network provider.

• **Web browser cache** containing HTML documents stored in plaintext.

• **Electronic mail** with full header information and attachments. It is possible to copy the encoded attachments from the `strings` output and convert them back into their original binary form.

• **Scripts and passwords** for network connections and applications.

• **Speed Dial** database (`SpeedDialDB`) containing up to 199 user-configurable, commonly dialed name and phone numbers.

• **Voice Dial** database (`Voice Dial DB`) containing up to 30 user-configurable, commonly dialed names and associated voice tags for hands-free dialing. **Voice Memo** functionality (`Voice Memo DB`) is used to record, store, and play back up to 60 seconds of audio memos. Although the actual voice tags for voice dialing and voice memos are usually stored on the telephone service provider network (e.g., Sprint), some devices may store the actual audio in the databases.

• **Call History** database (`CallHistoryDB`) containing a phone call information log for all incoming, outgoing, and missed phone calls. By default, the most recent 99 calls are stored for up to 7 days. A maximum of 999 phone calls can be stored indefinitely if configured that way by the user. The

[13] `http://www.kyocera-wireless.com/kysmart/kysmart_series.htm`

log includes the call type, time the call was placed, call length, and phone number. Each record is 70 bytes in size and appears to be structured as follows [1]:

```
typedef struct
{
  uchar  CallType;        // 1 byte: 1 = Incoming, 2 = Outgoing, 3 = Missed, 7 = Data
  uchar  Unknown1;        // 1 byte
  uint   TimeCallPlaced;  // 4 bytes: Starting time of call in seconds since 1/1/1904
  uint   CallLength;      // 4 bytes: Length of call in seconds
  uchar  Unknown2[8];     // 8 bytes
  uchar  PhoneNumber[52]; // 52 bytes: Number called or caller ID of incoming call
} CallHistoryDBRecord;
```

Each record in the `CallHistoryDB` has a sequentially increasing unique record ID (`uniqueID` as defined in [3]). If a record is deleted, it will be apparent by the non-sequential order of unique IDs.

• **Recent Calls** database (`RecentCallsDB`) contains similar information to `CallHistoryDB`, and the two databases can be compared against each other for discrepancies (e.g., erased or deliberately modified records).

4.5 Related Work

Palm, Inc.[14] has a number of useful tools available for development and debugging that could also be used to aid in forensic analysis. The Palm Debugger, included with the Metrowerks CodeWarrior for the Palm OS Platform suite, is the official interface to the Console and Debug Modes of the Palm device. The Palm Debugger tool runs on a desktop machine and allows low-level interaction with the Palm device. A complete listing of commands can be found in [5].

The Palm OS Emulator (POSER) is a desktop program that emulates the hardware of the various models of Palm devices and essentially creates a software-based "virtual handheld". Using parsing tools to create a series of PRC, PDB, and PQA files from the raw memory output provided by `pdd`, it would be possible to create an exact model of the target Palm device and operate the model on POSER. As it stands, the ROM image provided by `pdd` can be loaded directly into POSER. Such a system is particularly helpful to ensure the integrity of the target device (e.g., crime scene) while the model is used for investigative purposes.

Shortly after the premiere release of `pdd` in November 2001, two other Palm OS acquisition tools became available: Paraben's PDA Seizure[15] and the Palm acquisition module for Guidance Software's EnCase[16].

PDA Seizure uses the Palm HotSync protocol for some portion of memory imaging. Because of this, the Last HotSync Date is modified on the Palm device. Depending on how the acquisition process is implemented, there may be issues of only retrieving data that is known to the OS as being a database, so the acquisition would miss obfuscated or hidden forms of data.

pilot-link[17] is an open-source toolset for providing synchronization capabilities to Unix-based machines. Included are two utilities, pi-getram and pi-getrom, which use the HotSync conduit to capture full memory images. The Last HotSync Date is not modified on the Palm device when using these tools.

[14] `http://www.palmsource.com/developers`
[15] `http://www.paraben-forensics.com`
[16] `http://www.encase.com`
[17] `http://www.pilot-link.org`

5 Summary

This work presents a new tool for memory imaging and forensic analysis of devices running the Palm operating system. `pdd` retrieves and displays device information and a complete memory image of the Palm device, which includes all user applications and databases. It provides acquisition capabilities for use by forensic investigators, incident response teams, and criminal and civil prosecutors.

6 Availability

`pdd`'s Win32 executable and source code (included for research and legal verification purposes) is currently available for free at:

`http://www.mindspring.com/~jgrand/pdd`

Acknowledgments

The author would like to thank Brian Carrier for his education on incident response and forensic acquisition procedures, and Jesse Kornblum and Eoghan Casey for interesting discussions about Palm forensics.

References

[1] D. Crocker, "callHistoryDB schema," Palm Platform Kyocera Developer Forum e-mail list, `http://www.eScribe.com/computing/kyoceraforum/m63.html`.

[2] Kingpin and Mudge, "Security Analysis of the Palm Operating System and its Weaknesses Against Malicious Code Threats," *10th USENIX Security Symposium*, Washington, D.C., August 2001, `http://www.usenix.org/publications/library/proceedings/sec01/kingpin.html`.

[3] Palm, Inc., *Palm File Format Specification*, DN 3008-003.

[4] Palm, Inc., *Palm OS Memory Architecture*, `http://oasis.palm.com/dev/kb/papers/1145.cfm`.

[5] Palm, Inc., *Palm OS Programming Development Tools Guide*, DN 3011-002.

[6] Palm, Inc., *Palm OS SDK Reference*, DN 3003-003.