Virus Threats to Personal Digital Assistants

Simon Foley* and Robert Dumigan,

Department of Computer Science,

University College,

Cork, Ireland

August 24, 1999

Abstract

Like Personal Computers, Personal Digital Assistants are vulnerable to attack by computer viruses. In this article we discuss potential virus threats to 3COM's Palm hand-held computer.

1 Introduction

Personal Digital Assistants (PDAs) are small hand-held computing devices that support a variety of applications, ranging from conventional electronic organizer programs to spreadsheets, electronic mail and web browser clients.

*Correspondence should be directed to S. Foley at the address above.

A PDA is commonly viewed as an extension of a user's workstation (or server); carrying data and programs that often mirror data and programs from the workstation. Synchronization between the workstation and the PDA is performed on a regular basis, ensuring that changes made to data stored on the PDA are reflected on the workstation, and vice-versa. In addition to synchronization, programs and data may also be exchanged between PDAs, either via a workstation, or directly, if wireless communication is supported.

For example, [4] describes an application whereby a chronic disease sufferer uses a Palm hand-held to maintain a regular diary of pain parameters that are tailored to the patient. The data collected becomes part of a larger database when the hand-held is synchronized with a workstation for subsequent analysis by the health-care provider. This illustrates the difference between a PC and a PDA: by virtue of its size, a PDA is instantly available and can be viewed as an integral part of (mobile) person's workspace.

Existing PDA platforms include the (now discontinued) Apple Newton, Psion, 3COM's Palm hand-helds, and Microsoft Windows CE based devices. Our study focused on 3COM's Palm hand-held, specifically the PalmIII. This is a mature platform with over four million units sold worldwide since its introduction in 1997, and with over five thousand applications available [8]. Many of these applications are low-cost shareware and are available for download from centralized world-wide-web repositories such as `http://www.palmcentral.com`, or from CD collections such as [7]. With such a large number of applications available it is interesting to note that (to-date) there have been no reports of Palm viruses 'in the wild' [11].

This article describes observations from our study into the vulnerability of the Palm hand-held to virus attack. Not surprisingly, the Palm hand-held is as vulnerable to virus attack as a conventional PC, although there are some interesting differences in how Palm hand-held viruses are spread and activated.

2 The Palm Hand-held Architecture

The Palm hand-held range is based around the Motorola 68328 DragonBall series processor: a 16MHz micro-controller based on the Motorola 68000 processor with no memory management unit. Its form-factor is small, fitting inside a shirt-pocket, and uses a 160×160 pixel LCD touch-sensitive screen. A stylized form of handwriting (graffiti) is used for input. To facilitate low power consumption, memory is small, typically ranging from 2 to 8Mbytes, and is used to store application programs and data. An RS232 serial port is used primarily for synchronization with a workstation/fileserver. Recent models also provide an infrared irDA port, and the PalmVII provides a form of wireless Internet access.

The Palm operating system (PalmOS) is based on a pre-emptive multi-tasking real-time kernel with (user-interfaced) applications effectively running within a single thread. A memory manager manages that part of memory used to hold applications and data, and is comparable to the file system on a conventional system. This memory is organized as a heap and is structured as a collection of databases, each one storing a program or data. The records of a program (database) provide the code, forms, global variable data and other resources that make up the application.

3 Anatomy of a Palm virus

A computer virus is essentially a piece of code that copies itself onto a host program, modifying the program and often in a malicious way [1]. The virus executes when its host program begins to run. Like a biological virus, a computer virus can replicate itself, infecting other programs. Figure 1 describes the abstract behavior of a simple Palm virus embedded within an application. Its behavior is not unlike a conventional transient file-style virus which is active only when its host program runs [1].

on *launch* of host application:
    infect another application at random;
    if *triggered* then *attack*, possibly based on launch-code;

Figure 1: Anatomy of a Simple Palm virus

3.1 Launching a Virus

While a Palm application may be launched directly by the user, it may also be launched indirectly as a result of certain events, and every Palm application is expected to properly deal with these events. For example, PalmOS provides a text-string global find operation. When invoked, the find operation polls each application on the device (by invoking it). Each application, recognizing the event, searches to determine whether it can match the string within its own databases and any matches are passed back to the find operation.

This indirect invocation of an application is unlike a typical PC-based application: it is not necessary for a user to directly launch an infected Palm program. This can form the basis of some interesting virus attacks. For example, the infra-red irDA link may be used to 'beam' an infected Palm program from one hand-held to another. When a beamed application is received, PalmOS launches the application so that it may initialize itself; which in turn, activates the virus.

A variety of events can lead to the indirect launch (selectively or all) of applications. These include, synchronization, programmed alarms, certain operating system preference changes, or a soft-reset of the unit. Given that it is likely that a typical end-user will be unaware of this indirect invocation, avoiding the invocation of a suspect application is insufficient to prevent further virus spread.

3.2 Virus Spread and Infection

A Palm hand-held becomes infected when an application containing a virus is installed. Traditionally, Palm applications are installed by synchronizing the hand-held unit with a workstation. The Palm hand-held has no hardware memory protection and thus, once a Palm hand-held has become infected, the virus can spread freely, attaching itself to all (RAM-based) applications within the unit.

A typical user will generally acquire new Palm software by downloading the application from an Internet repository and store it on a local workstation/server. While a Palm program is stored as a database by PalmOS, this database is stored as a special serialized/pickled file ('PRC' format [10]) on a conventional file-system, facilitating its distribution and exchange. Synchronization downloads the new PRC files from the user's workstation/server and unpacks them into their correct program database format.

We argue that a Palm virus spread in this way will have limited success: after the initial infection, subsequent spread to other programs is generally limited to the hand-held unit alone. Figure 3.2 outlines the necessary steps for an infected program to spread from one hand-held unit to another. Once

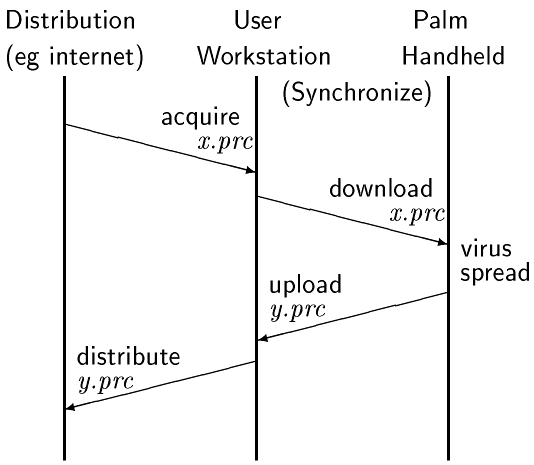

Figure 2: Spread of Infection based on Synchronization

a program is infected (on the hand-held unit) it must be uploaded to the user's workstation/file server and then distributed before it can become a threat to other hand-held units. While users regularly synchronize databases (containing application data) on the hand-held with their workstation, it is not usually necessary to upload programs since the original backup copies are already saved on the user workstation.

This observation can also be applied to any locally developed Palm programs. These are typically developed/coded on the user's local workstation; a cross-compiler and linker generates a PRC program file ready for downloading to a hand-held unit or to an emulator for testing. Thus, an infected Palm program cannot spread to any new programs that are to be distributed while the PRC file is that generated from the compilation and is not uploaded from a hand-held unit. Exceptions to this are Palm-based compilers, such as PocketC, which compile code on the hand-held unit. While such compilers are useful for prototyping simple programs, it is more often the case that programs are developed using the workstation-based tools.

Synchronization can provide a simple barrier: while an infected program can be installed it cannot spread to other programs beyond the hand-held unit[1]. This barrier fails once hand-held units become more connected. Any protocol that facilitates direct communication between Palm hand-helds may facilitate the spread of viruses. For example, most Palm hand-helds can communicate directly using the infra-red based irDA protocol, allowing infected programs to be beamed from one hand-held unit to another.

[1] This observation is not strictly true. In theory, a workstation based virus could attack the workstation synchronization program so that it uploads infected PRC files from the hand-held. However, in this case it would seem more effective for such a virus to search for and infect the PRC files stored on the file-system instead.

3.3 Virus Attacks

Benign and malignant Palm viruses are not unlike those found on a PC, illegally attaching themselves to applications and with attacks that range from unwanted inconvenience to catastrophic failure. Palm handhelds are commonly used as personal organizers, and maintaining the integrity of schedules, address-books, and so forth, is important. The data resources managed by these applications have standard database formats and can be the target of data-diddling attacks which randomly modify fields in these databases.

Because of the nature of its design, a Palm hand-held can be subjected to denial of service attacks that appear to have a greater impact on the Palm hand-held than on a conventional PC environment. Events that poll applications such as 'find' and synchronization can be significantly interfered with and/or slowed down when an infected application is indirectly invoked. The Palm hand-held is designed as a low-power device, operating on just two AAA-sized batteries for up to two months, depending on usage. Power consumption is is minimized by shutting down power to components such as the serial port, processor, screen, and so forth, when not in use. Thus, a virus that opens and leaves the serial port active, can have a significant impact on power consumption.

3.4 Patching Viruses

The functionality of PalmOS can be patched by user extensions that hook into the system-traps of the many PalmOS system calls. For example, the system event handler can be patched so that a virus scanner is first called before installing a new program in memory. These patches are usually called 'hacks', after the 'hackmaster extension' program which facilitates the development, installation and management of multiple PalmOS patches [3].

This PalmOS system-trap scheme is similar to the MS-DOS interrupt facility. Thus, we can have a PalmOS variation of the MS-DOS terminate-and-stay-resident(TSR) style viruses. For example, a virus installs itself as a hack which subsequently runs every time any application is launched.

4 Engineering a Palm Virus

Official documentation on the format of Palm applications is not currently available [5]. Thus, we relied on material such as [10] and byte-level editors and disassemblers to conjecture the general format of a Palm program. The format of a Palm program (executable) is comparable to that of an Apple Macintosh program (M68K series processors), organized as a resource database of code, data and other records. Each resource database (storing a Palm program) contains a code record labeled as `CODE1` to which (under current versions of PalmOS) control is passed when the program is invoked.

For the purposes of our study, we prototyped a simple virus that resides near the start of a program's `CODE1` resource (record). The virus was developed using the GNU C tool-set and it's executable code size is less than 700 bytes. It has a simple benign behavior—infecting other programs without any database corruption. Since PalmOS provides no memory protection, it is relatively straightforward for an (infected) program to open and access the `CODE1` resource record of any other program and infect it with a copy of the virus.

5 Virus Detection and Prevention

We believe that it will be possible to detect simple (non-camouflaged) Palm viruses by using the conventional approach of searching executables for certain bit-string patterns that signify the signatures of known Palm viruses. Each time a new application is downloaded to the handheld it should be scanned for known viruses. In addition to maintaining the signature of every known virus, an anti-viral program should also be capable of removing these known viruses from infected programs. This anti-viral program may run either on the user's workstation, scanning Palm programs before download, or on the handheld device itself.

The task of scanning a Palm program for viruses can be computationally intensive, depending primarily on the number of known signatures maintained in its (potentially large) virus database. Since the Palm handheld is a relatively slow device with limited memory, processor and/or memory intensive tasks should be performed off-line, that is, by the user's workstation. Therefore, it is preferable to scan a palm application at the user's (presumably trusted) workstation *before* downloading it to the handheld device.

This strategy of off-line virus-checking is inadequate if palm-programs are installed directly, for example, beamed directly using the irDA protocol from another palm handheld, or directly downloaded from some untrusted system. In this case the beamed program should be scanned by the receiving device. We have implemented a simple virus scanner that is invoked (via a system trap) whenever a beamed program is received. An infected program can either be rejected or repaired. This approach is reasonable if there are only a small number of known palm viruses. However, we conjecture that a point will come whereby the amount of time taken to check a program against every known virus signature will take longer than is acceptable to a typical user.

For example, we conducted a very simple experiment finding that it takes, on average, approximately 10 milliseconds to scan 1Kbyte of executable code for the signature of our prototype virus. Palm applications are typically small; scanning an application with 10Kbyte of code for 10 such viruses should take approximately 1 second, which is reasonable. However, scanning for more than 100 such viruses (10 seconds) would begin to be unreasonable to the average user[2].

Alternatively, a variety of conventional cryptographic-based schemes could be used to ensure that illegally modified programs cannot be installed on the handheld device. A framework supporting application-signing could be built on top of existing infrastructures for distributing public keys and managing trust, such as PGP and X509. Public keys represent either the original developers/vendors of the software or trusted third parties who have checked the application against known viruses and can certify that the code is free of known viruses.

Implementing such a scheme on a handheld unit would appear to be feasible. [2] gives performance measurements for basic cryptographic primitives run on the Palm handheld. Using these measurements, we estimate

[2]This is a worst-case estimation which assumes that it is necessary to search the entire code resource. In practice, the search can be improved using optimized search algorithms and noting that virus (signatures) are generally found towards the start or end of the application code.

that it would take approximately 740 milliseconds to verify the signature on an application with 10Kbytes of code (assuming 512 bit key based RSA signature of SHA-1 generated hash). Interestingly, it would take over 5 seconds to generate the signature for the same code resource. Therefore, while signature verification can be done on the handheld device, code-signatures are best generated off-line. Needless to say, this is only one line of defense: such a scheme fails when an unknowingly infected program is signed.

6 Discussion and Conclusion

This article considered some of the virus threats to the Palm handheld, many of which are generally applicable to other Personal Digital Assistants. Once an infected program is installed, the virus can spread freely to other programs on the handheld unit. While we know of no Palm viruses in the wild, we argue that such viruses can be generally expected to behave in much the same way as conventional PC viruses such as transient file and TSR viruses.

We argue that, given the way it is typically used, a Palm handheld is probably less vulnerable to virus attack than a conventional PC. The impediment to infection is mainly due to the current practice of software installation by off-line synchronization with a workstation/file-server. However, we speculate that with the current trend to provide more connected units (wireless, irDA, etc.), alternative and possibly more convenient installation practices will emerge, facilitating the spread of viruses.

In the current absence of any known Palm viruses in the wild, we make some general recommendations on safely handling Palm programs.

- *Obtain applications from reliable sources.* Palm applications should be downloaded/registered directly from a trusted repository.

- *Check new software before installing.* It is usually a good idea to first run any new software using POSE—an easy to use Palm OS emulator that runs on the users workstation. This may help to identify potential problems before installation on the handheld unit.

- *Synchronize regularly.* This ensures that up to date 'backups' of application databases are maintained on the user's workstation/file-server. Since synchronization is an integral part of how a handheld is used, we suspect that average users probably have a superior backup regime for their handheld than for their PC.

- *Avoid uploading programs from a handheld.* If an infected program is uploaded to the workstation/server, the possibility of subsequent distribution will spread the virus. The disadvantage of this regime is that program preferences are typically stored as a record within a program's resource database. Thus, under the proposed regime, if an application needs to be re-installed, installation of the original distribution copy will require the user to reconfigure preferences.

- *Use a separate resource database to store application data.* In theory, an application developer could store all application data as additional records within the program's resource database. This would conflict with the previous two recommendations. Therefore, it is preferable (and usual) to manage a Palm program's data in a separate database, facilitating synchronization.

- *Avoid irDA program exchanges.* Beaming a program from one handheld to another is probably the easiest way to spread a virus.

The Palm handheld range provides flexible basis for mobile computing in a convenient form-factor. However, in using a micro-controller unit with no memory management, kernel-level protection mechanisms do not appear to be an issue in the design of PalmOS. While programming-level structures prevent erroneous or accidental access to data [6], every Palm program can have complete access to the data and code of every other program. It may be argued that this is a reasonable approach for a low-cost and efficient single-user device, however, the result is a device that is difficult to secure against malicious code. Furthermore, as handheld devices become more connected, it is probable that features such as executable content and application macros will begin to emerge, greatly facilitating virus spread.

This vulnerability could be lessened if programs could be run within separate protection domains, or at least within a sandbox. For example, KVM [9], the Java Virtual Machine for handhelds. With this environment, it should be possible to build security policies that at least attempt to confine the spread and effect of untrusted or malicious code.

References

[1] F Cohen. *A Short Course on Computer Viruses.* John Wiley, 2 edition, 1994.

[2] N Daswani and D Boch. Experimenting with electronic commerce on the PalmPilot. In *Financial Cryptography*, Anguilla, BWI, February 1999.

[3] E Keyes. Hacking the pilot: Bypassing the PalmOS. *PDA Developers Journal*, 6(4), November 1996.

[4] J Mansfeld. Electronic patient diaries. *Handheld Systems Journal*, 7(1), January 1998.

[5] Palm Computing. Frequently Asked Questions. http://www.palm.com. Development Zone Article ID 1057.

[6] Palm Computing. Palm os memory architecture, 1999. Development Zone Article ID 1145.

[7] D Pogue. *PalmPilot: The Ultimate Guide.* O'Reilly, 2 edition, 1999.

[8] Neil Rhodes and Julie McKeehan. *Palm Programming: The Developer's Guide.* O'Reilly & Associates, 1999.

[9] A Taivalsaari, B Bush, and D Simon. The spotless system: Implementing a JavaVM system for the Palm connected organizer. Technical Report TR-99-73, Sun Labs, February 1999.

[10] Theodore Ts'o. The PRC format, July 1997.

[11] WildList Organization International. http://www.wildlist.org.