PITO: A Children-Friendly Interface for Security Tools

Luigi Catuogno
Dip. di Informatica ed Applicazioni, Università di Salerno
Via Ponte Don Melillo, 84 084 - Fisciano (SA) - ITALY
luicat@dia.unisa.it

Abstract. Current Operating Systems and user applications provide several security tools that are designed to automatically handle a set of well-known events and threats, whereas they inform the user that something potentially dangerous is happening an ask her to take a decision. Since we believe that this learning process could have a relevant educational value for children, we present, in this paper, an interface for Security tools featured by operating systems, user applications or third party security suites. The interface, that has been designed as "Pito": a digital pet , is purposed to allow children that use computer to be softly introduced in security issues that usually occur when navigating the Internet. Alerts notified by the underlying security tools are mapped as some simple events into the Pito's world. These events change the state of Pito and, according to the current state, the young user is asked to do some action in order to keep Pito's sane and happy, that is to keep safe the computer.

1 Introduction

Current operating systems and applications provide a set of security tools like firewalls, virus scanners, spam filters, diagnostic facilities and so on[1] as well as these critical features are provided by many commercial *security suites*[2][3][4] that have been proposed as an "all inclusive" solution to main security threats.

These security tools (ST), are designed to automatically handle a well-known set of events (in-coming of known virus and intrusion patterns) and, moreover, in presence of some borderline events or user action that may be potentially dangerous, the ST inform the user and possibly ask her if the event (or action) that arisen the event can be approved (confirmed) or not (*e.g.*, execution of an application that have just been downloaded, allowing an application to connect somewhere over the

network, installation of an unsigned device driver and so on). Information produced by STs notifies to the user what is happening in the computer and also educates her about risks and possible consequences of her actions and moreover, it also suggests the proper behavior. ST's notifications are published through different interfaces *e.g.*, message boxes, log file, blinking icons, etc. and are thought to be understood by adult users, that can learn the meaning of messages, that can easily read manuals and are smart enough to reasonably assess the arisen issues.

On the other hand, such some interfaces, are not suitable for young users that often are not able to understand what is happening. Therefore, the usual security infrastructures, even if well configured, are inadequate to protect Children on line. For this reason, the main kind of security measure purposed to young users consists of so-called *Parental Control Systems* (PCS). These products, typically included in the applications used to access the Internet (or even embedded into the operating system)[5], allow parents to monitor or limit what a child can see or do. Parental Control Systems may allow for the blocking of various websites, such as those containing pornography, or other undesired contents. Thus, parents configure the software through a black list, or a through set of rules. Instead of STs, Parental control systems work rather transparently; hence they do neither inform the user nor educate her. In other words, a traditional PCS does not introduce the young user to the problem of security, does not teach him that something of *strange* is happening, that some decision have to be taken and, consequently, which action are suggested to solve the problem.

In this paper we present a "child interface" for security tools that introduces the young users to security issues commonly arisen when somebody uses computers and navigates the internet. Instead of simply rewriting text in ST's message boxes with more simple words, we designed our interface as a Digital Pet[6]. In our project, a small character, named "Pito", lives in his world inside an application window. A configurable subset of security events is mapped to a set of events that happen in the Pito's world and, on the other hand, what Pito does in his world is mapped to a configurable set of security actions in the "real world". Let us to see an example. Suppose that parents agree that their children do not use the computer more than two hours. Thus they configure Pito to look tired when the time is going out. When Pito looks tired, the children simply bring him to bed and, in such a way, they shut down the system. We point out that Pito does not replace a PCS, but integrates it where parents wish to introduce a certain level of *free will* in educating their children to correctly use computers and internet.

In Section 2 we give an overview of the project and describe the architecture of Pito. In Section 3 we discuss some implementation issues of our first *proof-of-concept* prototype. Finally, in Section 4, we outline the work we still have to do.

2 Project Overview

The architecture of Pito is composed of three components: the sensors, the engine and the interface.

A sensor is a stand-alone agent purposed to listen a specific event (*e.g.*, an in-coming connection, the arrival of an e-mail message). Whenever the event is arisen, the sensor forward it to the engine. Note that, sensor are designed to be specific for the underlying operating system, security suite or applications and report the event in a general way to the engine. Thus for example, we can have an anti-spam sensor, plugged in our favorite mail-agent, which reports the arrival of a spam messages (that may carry a fishing attack or simply offending contents). Different mail-agents can be handled by means of their own APIs but all respective sensors forwards events to the engine through the same protocol. Moreover, a sensor could parse system logs and reports some relevant events, warning or errors (*e.g.*, disk full, service failure, etc.) as well as a sensor could be plugged in the firewall in order to handle requests about inbound or outbound connections.

When launched, Pito is in its initial state (*quiet*). The events arisen by the sensors and the actions done by the user may change this state. The engine updates the current state according the in-coming events or actions performed by the user. The way in which the Pito's state changes is defined, through a simple set of rules by means of a configuration panel that allows parents to configure the desired behavior of Pito with respect to a certain set of events and actions. Look at the following scenario as an example. Suppose that Parents configured the anti-spam engine to accept messages with a spam score less than a security threshold and silently discard messages with a score over a higher threshold. Whenever a message with an intermediate score is received, the sensor alerts the engine that a *potentially* spam message has arrived. The engine changes the state of Pito to *scared* showing a mail icon. Now, the user has been perceived that a potentially dangerous message is in the mailbox. Hence she can choice either to inspect it or erase it directly by putting the letter into the Pito's trash can. Moreover, whenever the crash of an OS service occurs so that the system becomes instable, the state of Pito is changed to *sick*. Hence the user has to give medicines to Pito, e.g. all open applications are closed and finally, the system reboots.

The interface maps the Pito's state and occurring events through a small application window on the desktop. Each state is mapped to a given expression of the Pito's face as well as, any configured event is showed as something happening into the Pito's world. The interface also provides some controls that feature allowed actions.

3 Implementing PITO

We are working on a prototype for the Windows operating system. We designed Pito as a WMI[7] based application. We are implementing sensors as WMI providers that present themselves to the engine through a CIM[8] classes and events. This choice makes the integration into the operating system and in many native applications transparent and simple. In fact, handlers of relevant system events, together with newly written providers that handle events and data featured by third party application, lay underneath the same general API, making the Pito engine independent to the current system configuration. Moreover, many commercial

security tools for Windows XP already feature their own WMI provider in order to be compliant to the Windows Security Center, making the implementation and testing of Pito in real-world conditions easy. The outlook of the Pito's window is currently rather poor. A remarkable restyling of the whole interface is currently in progress.

4 Conclusion

In this paper we outline the design of a children-friendly interface to the security tools embedded in applications and operating systems. This interface has been designed as a "Digital Pet". A proof-of-concept implementation of Pito is currently in progress for the Windows Operating system. There are some important aspects of this work that need to be investigated carefully. First, will young users "accept" the presence of Pito and will they use it? Second, will parents agree the approach we suggest? And third, will Pito effectively make conscious the children about security issues? Our first tests with a group of children form the primary school (10 years old) have given encouraging results. We are designing a set of more rigorous experiments with 8-10 years old children (and their parents) in order to validate our approach. These experiments, together with a fully featured prototype will be hopefully discussed in a future work.

References

[1] Microsoft Corporation, 2007, *Windows XP service pack2 (Part 3): The new Security Center*, Technical Article no. 889737, http://msdn.microsoft.com
[2] Symantec Corporation, 2008, *Norton Internet Security*, http://www.symantec.com
[3] Sophos Corporation, 2008, *Sophos Anti-Virus*, http://www.sophos.com
[4] Sophos Corporation, 2008, *Sophos PureMessage*, http://www.sophos.com
[5] Microsoft Corporation, 2006, Microsoft Windows Vista Security Advancements, white paper, http://msdn.microsoft.com
[6] Wikipedia Foundation, 2008, *Digital Pets*, http://en.wikipedia.org/wiki/Digital_Pets
[7] Microsoft Corporation, 2000, *Windows Management Instrumentation*, Technical article number MS-811553, http://msdn.microsoft.com
[8] Distributed Management Task Force, 2007, *CIM schema version 2.15*, http://www.dmtf.org