Polaris: Usable Virus Protection for Windows

Ka-Ping Yee
University of California, Berkeley
Berkeley, CA 94720
ping@zesty.ca

Marc D. Stiegler, Alan H. Karp, Tyler Close
Hewlett-Packard Labs
1501 Page Mill Road
Palo Alto, CA 94304
{marc.d.stiegler, alan.karp, tyler.close}@hp.com

Mark S. Miller
Johns Hopkins University
Baltimore, MD 21218
markm@caplet.com

ABSTRACT

We are developing Polaris, an environment for running unmodified Microsoft Windows applications that protects users from viruses and spyware while keeping the user experience as smooth and unchanged as possible. The design philosophy underlying Polaris is the *principle of least authority*, but it is built on Microsoft Windows, an operating system that provides little or no support for least authority operation. We describe the Polaris user experience, the way Polaris works, and the user interface design challenges we faced in developing Polaris.

1. INTRODUCTION

Users of Microsoft Windows suffer frequent attacks from e-mail viruses, macro viruses, and spyware [2]. All three can be characterized as software that causes damage if run with too much authority.

- E-mail viruses (and other dangerous programs) arrive in executable attachments to e-mail messages. When a user opens such an attachment, it launches as a running program with the user's authority and abuses that authority to e-mail itself to other victims, install itself on the system, or damage the user's files.
- Macro viruses arrive embedded in Microsoft Office documents with programmable behaviour (such as Word or Excel files). When a user opens such a document, the macro abuses the user's authority to propagate itself to other documents.
- Spyware can arrive included with other software that the user intends to download, or it can arrive in an involuntary download triggered by arriving at a website or by clicking on an apparently unrelated link. When the spyware runs, it abuses the user's authority to install itself on the system, monitor the user's activities, add toolbars and buttons, and display advertisements.

The predominant attempts to address these problems can be classified into three categories: scanners, filters, and prompts.

- Anti-virus scanners and spyware scanners may take up to an hour to finish scanning the entire contents of a hard disk, and must be run frequently, imposing severe costs in system usability. A scanner cannot defend against a new virus until the anti-virus vendor has added the virus to its database and the scanner has downloaded the update. Even if the virus is registered and the scanner is run daily, an infected computer still has up to a day to attack other computers.
- Filters for e-mail and other network traffic can only catch viruses they recognize, so their accuracy is limited by their databases of known viruses. Their heuristics for detecting viruses can sometimes be falsely triggered by harmless traffic, resulting in lost e-mail or broken application behaviour.
- Prompts are a nuisance, and they often ask users to make security decisions based on insufficient information. A good example is the warning that Word displays when opening a document with macros, shown in Figure 1. Another common type of prompt asks the user to verify a digital certificate before downloading or installing software. Digital certificates are intended to securely label code from trustworthy sources, but certification authorities have demonstrated that they are not a reliable means of establishing trust [1]. Moreover, the user interface for checking certificates is so poor that certificates are rarely examined.

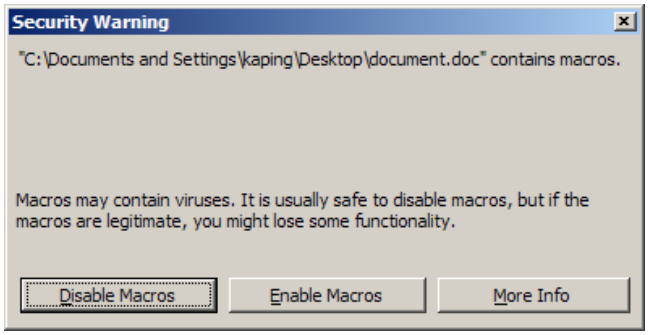

Figure 1. Microsoft Word asks the user to choose between accepting an absurdly large risk and not getting work done.

All of these mechanisms impede usability while failing to fully address the problems[1]. All of these mechanisms also share the assumption that it is feasible to determine whether a program is safe to run merely by inspecting the program. We believe that challenging this assumption can lead to a better solution.

[1] Note that we limit the scope of this discussion to security problems caused by excess authority, not by programming errors in the operating system. (Polaris is not an operating system or an operating system modification, merely a tool that runs on Microsoft Windows.) E-mail viruses, macro viruses, and spyware would continue to be serious problems even if there were no programming errors in Windows, because it is simply not designed to solve these problems.

2. LEAST AUTHORITY DESIGN

Microsoft Windows, MacOS, Linux, and all Unix-based operating systems provide security controls designed primarily to support distinctions between user accounts. In these systems, it is comparatively easy to grant particular permissions to selected users, but it is comparatively difficult to grant permissions only to selected application processes. By default, any program that a user runs is given the authority to do anything that user can do.

The aforementioned problems can be better addressed simply by following the *principle of least authority*[2]: each program should only be given the least authority necessary to perform its intended task. Polaris [4] limits the authorities of application programs by confining each application in a separate user account. Windows permission settings are used to give the confined account very limited access to the disk. The application is only allowed to write in a temporary area, read system libraries in the Windows directory, and read libraries and data files in the application's own installation directory.

3. SECURITY BY DESIGNATION

An application that performs useful tasks will usually, over the course of its execution, need access to information outside of the application itself. For example, most applications need to open and save files on the disk. The key usability design challenge is to provide the necessary extensions of authority while maintaining user control over which authorities are granted.

Our approach to this challenge is to look for the user's acts of designation that correspond to acts of authorization [5]. Usually, normal use includes an act of designation, so we don't have to inconvenience the user with any extra work. For example, when the user double-clicks a file to open it in its associated application, that double-click can be interpreted as an indication that the launched application should be granted access to that one file. When the user issues a "Save" command and selects the desired location and name of the file to save, the application should be granted access to write just that one file.

To handle double-clicking on files, Polaris changes the default file type associations so that files associated with confined applications are launched via Polaris instead. When the user opens such a file, Polaris provides a copy of the file to the confined application. If the confined application makes changes to the file, Polaris copies back these changes to the original file.

To handle requests to open and save files from within the application, Polaris replaces the application's Open and Save dialog boxes with dialog boxes from Polaris. After the user selects a file, Polaris copies and synchronizes the file between the user's account and the confined application's account so that the application effectively has access to only the selected file.

When there is no act of designation indicating a desire to grant authority, we make some compromises. For example, when the user views a web page containing images on the local disk, the user does not select the image files even though the browser needs to read those files. In this case, we grant the browser read-only access to the directory containing the page.

[2] This is also known as the *principle of least privilege* [3]. However, the term *privilege* is commonly used to refer to representations of permission in system data structures, whereas what matters is the actual potential to do harm. We prefer using the term *authority* in order to emphasize the latter.

4. USAGE

Aside from the initial step of setting up the confined accounts, the experience of using Polaris is mostly the same as using a computer without Polaris. Window title bars are adjusted to add an indication of the confinement domain, as in Figure 2. Applications and their file dialog boxes operate normally, though sometimes a brief flash of the application's dialog box is visible before it is replaced with the dialog box from Polaris.

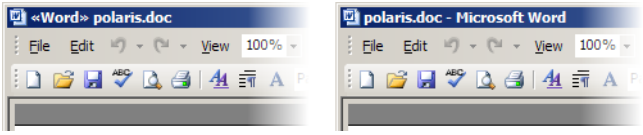

Figure 2. Left: a confined window. Right: a normal window.

5. EVALUATION

A pre-alpha version of Polaris has been used in day-to-day work by about 20 people at HP Labs, some of them for over six months. For the most part, our users aren't aware of its presence and continue to use their computers normally. We have several known cases of viruses that were rendered harmless by Polaris, which we found because anti-virus software detected a virus in a confined account where it could do no damage.

6. CONCLUSIONS

Polaris achieves protection against entire families of attacks while minimizing its impact on usability and functionality by applying the principle of least authority at a per-application level. In particular, Polaris can immediately defend against new viruses found in the wild, unlike anti-virus scanners, which can only defend against viruses that have been caught and inspected.

Our ultimate goal is to free users from having to run virus scanners, run spyware scanners, and answer security prompts, and to let them just use applications normally without fear of viruses. For Microsoft Office and other common applications, Polaris is a big step toward achieving this goal. The current release does not limit network access and does not provide good support for linking and embedding of Office documents, and we are working on solutions to these issues for the beta version.

7. REFERENCES

[1] B. Edelman. How VeriSign Could Stop Drive-By Downloads. http://benedelman.org/news/020305-1.html

[2] S. Granneman. Linux vs. Windows viruses. http://www.securityfocus.com/columnists/188

[3] J. H. Saltzer, M. D. Schroeder. The Protection of Information in Computer Systems. *Proceedings of the IEEE, 63*, 9 (Sep. 1975), 1278–1308.

[4] M. Stiegler, A. H. Karp, K.-P. Yee, M. S. Miller. Polaris: Virus Safe Computing for Windows XP. HP Labs Technical Report HPL-2004-221. http://www.hpl.hp.com/techreports/2004/HPL-2004-221.pdf

[5] K.-P. Yee. Aligning Usability and Security. In *IEEE Security & Privacy Magazine*, Sep. 2004.