REMARKS

The Applicants have carefully considered this application in connection with the Examiner's Office Action and respectfully request reconsideration of this application in view of the foregoing amendments and the following remarks. The Applicants thank the Examiner for the Examiner Interview on August 22, 2011 and have taken into consideration the topics of discussion therein when addressing the objections and rejections to this application.

The Applicants previously submitted Claims 1-73 in the application. While Claims 1, 3, 10 and 15 have been amended, no claims have been cancelled herein. For the Examiner's benefit, the Applicants have provided an Appendix II to clearly show the amendments to the specification and claims from the amendment filed on April 4, 2011. With respect to the specification, the Cross Reference to Multiple Reissue Applications is submitted without markings as no changes have been made from the previous amendment. Also, the Examiner has indicated that Claims 8, 9, 13 and 18 would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims. Accordingly, Claims 1-73 are currently pending in the application.

I. Rejections under 35 U.S.C. §102

The Examiner has rejected Claims 1-5, 7, 10-12, 14-17, 19, 21-26, 29-48, 51-56, 58-67 and 69-73 under 35 U.S.C. §102(b) as being anticipated by U.S. Patent Application Publication No. 2002/0002673 to Narin. As the Examiner is no doubt aware, anticipation requires that each and every limitation of the claimed invention be disclosed in a single prior art reference. The disclosed limitations must either be disclosed expressly or inherently and must be arranged as in the rejected claims.

For the reasons as set forth herein, the Applicants believe that Narin does not disclose a computer system, portable computer, computer program product or related method as recited in ones of independent Claims 1, 10, 15, 21, 32, 44 and 64 of the present application. In particular, the Applicants believe that Narin fails to disclose, among other things, a computer system, portable computer, computer program product or related method configured to execute (or open) first and second browser processes in accordance with at least one electronic data processor, and protect data (or a system file) residing in a first memory space (accessible by the first browser process) from corruption by a malware process executing as part of the second browser process as recited in ones of independent Claims 1, 10, 15, 21, 32, 44 and 64 of the present application.

Narin provides a technique for allowing an open or untrusted application to provide untrusted or open features for a secure application that are not directly implemented within the secure application (or closed application). In accordance therewith, an open or untrusted application is run in a separate, auxiliary process from the closed or protected application. The auxiliary process is creating by running a hosting application that has minimal functionality, just enough to be able to host an application and to communicate with the closed process. The auxiliary process is started by the closed process; the closed process controls the lifetime of the auxiliary process and terminates it when the open features that it provides are no longer necessary. (Paragraph [0006].)

In the following excerpt, Narin teaches away from the closed process being a browser process. If the application is trusted, running a browser in-proc may subvert the security scheme of the trusted application. The browser code may not be secure to the same extent as the trusted application. Even if the browser code itself is secure, the browser provides the capability to import executable code from other sources that may not be trusted. If trust is to be maintained,

executable code from unknown sources cannot be given access to the address space of the trusted application and therefore cannot be run in process. (Paragraph [0004].) As it is well settled, a reference may be said to teach away when a person of ordinary skill, upon reading the reference, would be discouraged from following the path set out in the reference or would be led in a direction divergent from the path that was taken by the applicant. (See, *e.g.*, *Spectralytics v. Cordis Corp*., Nos. 2009-1564, 2010-1004 (Fed. Cir. 2011) (citing *In re Gurley*, 27 F.3d 551, 553 (Fed. Cir. 1994)).)

Narin continues that in a world where computers are increasingly called upon to handle secure or sensitive information, there is a tension between trusted applications and open applications. Trusted applications typically provide a circumscribed set of functions that cannot be extended by a user, which means that such applications can be trusted to handle sensitive content in predictable ways. Open applications, on the other hand, provide a wide range of functionality that is, in some cases, user expandable - some open applications, such as browsers, can execute code that is user-implemented or imported from other sites on the Internet. (Paragraph [0016].)

In the detailed description, Narin discusses how the secure application (the trusted application) makes use of a non-secure software object (the open process or untrusted application), and clearly describes them as being distinct and different from each other. Narin describes a web browser as being an example of such a non-secure software object, meaning that a web browsing program cannot be part of the secure application. Narin clearly says in the first sentence below that the non-secure software object provides a service that is not directly implemented within the secure application. This can only mean that they are separate and distinct from each other.

Secure application 312 uses non-secure software object 322 to perform an action or provide a service that is not directly implemented within application 312. Non-secure software object 322 is non-secure in the sense that its behavior cannot be relied upon; for example, non-secure software object may be a program that imports and runs arbitrary code from a remote, non-authenticatable (possibly nefarious) source. A web browser is an example of such a non-secure software object 322, because it retrieves and executes scripts from remote locations that may or may not be trustworthy. As an example, application 312 may provide some type of web browsing capability to its user, but rather than performing the actual web browsing functions itself, application 312 may call upon a general-purpose browsing program to perform the web browsing. In this exemplary case, non-secure software object 322 is such a web browsing program. (Paragraph [0036].)

Narin continues to draw the clear distinction between a web browser and a secure application, again, clearly teaching away from the secure application ever being a web browser. A web browser is an example of a non-secure object that should not be granted access to an address space where decrypted content or decryption keys may be stored. Although certain commercially-available browsers may be a known quantity that can be trusted not to content subversive code, one feature of a browser is that it can load and run arbitrary code from unknown sources (*e.g.*, in the form of an ActiveX® control, a JAVA script or applet, *etc.*). Thus, if the browser runs in the same process as a secure rendering application, the browser could be used to unwittingly download an ActiveX® control that would locate a buffer used by the rendering application to store decrypted content and, say, store that content to the hard disk. (Paragraph [0047].)

Narin belabors the point by providing that the secure rendering application may instruct the browsing program to render a list of links that the user may visit. If the user clicks on any of the links, the browsing program will retrieve the web page associated with that link and display it to the user. It should be observed that it is the browsing program, and not the secure rendering application, that performs the retrieval of web pages. It should further be observed that the downloading of an arbitrary web page in the browser does not, in and of itself, compromise the security of the secure rendering application; since the browser executes in the second process, it has no access to the address space of the secure rendering application that runs in the first process. (Paragraph [0049].)

Narin makes the clearest distinction below between the browser and the secure application, referring to the web browsing function as being a separate program running in a separate process. Narin here is clearly teaching away from the secure application and the non-secure application both comprising browser processes. Preferably, integration between the secure rendering application and the browsing program is as transparent as possible. That is, when the user invokes the secure rendering application, the user should not be aware (and likely does not care) that some of the application's function (*i.e.*, the web browsing function, in this case) is being provided by a separate program running in a separate process. (Paragraph [0050].)

Thus, it is quite clear from the excerpts of the reference reproduced above that Narin fails to disclose, among other things, a computer system, portable computer, computer program product or related method configured to execute (or open) first and second browser processes in accordance with at least one electronic data processor, and protect data (or a system file) residing in a first memory space (accessible by the first browser process) from corruption by a malware process executing as part of the second browser process as recited in ones of independent Claims

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.