UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

Palo Alto Networks, Inc.
Petitioner

v.

Finjan, Inc.
Patent Owner

U.S. Patent No. 8,141,154
Filing Date: June 14, 2010
Issue Date: March 20, 2012

Title: System and Method for Inspecting
Dynamically Generated Executable Code

Inter Partes Review No. 2015-01979

SUPPLEMENTAL DECLARATION OF AVIEL D. RUBIN IN SUPPORT OF PETITIONER'S REPLY

I, Aviel Rubin, declare as follows:

I have personal knowledge of the facts stated in this supplemental declaration, and could and would testify to these facts under oath if called upon to do so.

I have been retained by counsel for Palo Alto Networks, Inc. (Petitioner) in this case as an expert in the relevant art. I am being compensated for my work at the rate of $688 per hour. No part of my compensation is contingent upon the outcome of this petition.

I was asked to study U.S. Patent 8,141,154, its prosecution history, and the prior art, and to render opinions on the obviousness or non-obviousness of the claims of the ’154 patent in light of the teachings of the prior art, as understood by a person of ordinary skill in the art in the 2005 timeframe. I previously executed a declaration in support of Palo Alto Networks’ Petition for *Inter Partes* Review (Ex. 1002). This supplemental declaration addresses positions and testimony raised by Finjan in its Patent Owner Response.

In addition to the material and information I reviewed and considered in connection with my previous declaration and my own expertise in the field, this declaration is based on the following information:

Exhibit No.	Description of Document
1038	Medvidovic deposition transcript, IPR2015-01979, Nov. 21, 2016.
1044	Nebenzahl & Wood, *Install-time Vaccination of Windows Executables to Defend Against Stack Smashing Attacks*, Technical Report EES2003-9, Nov. 4, 2003.

I. BASED ON THE TEACHINGS OF KHAZAN, A PERSON OF ORDINARY SKILL WOULD HAVE KNOWN HOW TO INSTRUMENT APPLICATIONS

1. It is my opinion that, based on the disclosures in Khazan, it would have been obvious to a person of ordinary skill in 2005 how to instrument applications using techniques disclosed in Khazan and techniques that were well known in the art.

2. The Khazan reference broadly describes that its instrumentation techniques can be applied to many different types of programs and code:

> [0114] It should be noted that the foregoing techniques are applied in particular to binary machine executable codes. However, the foregoing techniques may be characterized as extensible and generally applicable for use with any one of a variety of different types of binary and machine-executable programs, as well as script programs, command program, and the like. (Ex. 1003 at 25 [114]; *see also id.* at 26 [0118].)

3. Khazan describes instrumenting libraries and executable applications, such as Win32 EXE files, for example:

> [0042] In this embodiment, prior to executing the application executable 102, an analysis may be performed by the static analyzer 104 to examine and identify calls or invocations made from the application executable 102 to a predetermined set of target functions or routines. An embodiment may also identify additional information

> about these functions, such as, for example, particular locations within the application from which the calls to these functions are made, parameter number and type information for each call, the values that some of these parameters take at run-time, and the like. For example, in one embodiment, it may be determined that the target function calls to be identified are those that are external to the application 102, such as those calls that are made to system functions. (Ex. 1003 at 18 [0042].)

4. In the passage cited above, Khazan discloses, as part of the instrumentation process, scanning an executable file in order to find function calls. The passage also describes the case in which function calls to external libraries are found. The reason that Khazan breaks external function calls out as a special case is because calls to functions inside of an application may be treated differently than calls to a function external to the application. Khazan then describes instrumenting these functions using, in one example, code included in the Microsoft Detours package. (Ex. 1003 at 21[0079], 23 [0091].)

5. Khazan describes that Detours may be used to instrument an application by replacing "the first few instructions of the target function with an unconditional jump to a user provided wrapper or stub function." (Ex. 1003 at 23 [0091].) A person of ordinary skill at the time would have known that Detours could be used to instrument an application. This is shown, for example, in the Detours paper, published in 1999, which describes applying Detours

instrumentation to Win32 portable executable files, which are applications. (Ex. 1012 at 3-4.)

6. Khazan also teaches that: "Win 32 API functions are instrumented for the purpose of being intercepted although an embodiment may monitor or intercept any one or more different functions or routines." (Ex. 1003 at 21 [79].)

7. Khazan gives examples of how these calls may be implemented in assembly language in an application binary:

> [0046] The particular type of target calls and their form may vary in accordance with each embodiment. For example, in one embodiment, the binary representation of the application executable 102 may include a jump instruction, a call instruction, or other types of instructions transferring control from the application as may be the case for various routines being monitored." (Ex. 1003 at 18 [0046]; *see also id.* at 19 [0048].)

8. Thus, Khazan discloses how to recognize any functions that are within the application executable and how to find their locations in the executable.

9. In another example, Khazan also describes how to find such call instructions using the Ida Pro disassembler:

> "[0045] An embodiment may examine the application executable 102 using any one of a variety of different techniques to look for any calls to one or more predetermined functions or routines. The static analyzer 104 may examine the binary code of the application executable 102 to look for predetermined call instructions, or other type of transfer instructions associated with calls to target functions. One embodiment uses the IDA Pro Disassembler by DataRescue (http://www.datarescue.com/idabase/) and Perl scripts in performing the static analysis of the application executable 102 to obtain the list

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.