UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

PALO ALTO NETWORKS, INC.,
Petitioner

v.

FINJAN, INC.,
Patent Owner

Patent No. 8,141,154

Inter Partes Review No. IPR2016-00151

DECLARATION OF DR. AVIEL D. RUBIN IN SUPPORT OF PETITIONER'S REPLY TO PATENT OWNER RESPONSE

I, Aviel D. Rubin, Ph.D., submit the following declaration in connection with the proceeding identified above.

I. INTRODUCTION

1. I have been 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 Network’ Petition for *Inter Partes Review*. (Ex. 1002.) This supplemental declaration addresses positions and testimony raised by the Patent Owner Finjan in its Patent Owner Response.

II. MATERIALS CONSIDERED

2. In preparing this declaration, I have reviewed, among other things, the following materials: (a) The Patent Owner Response and supporting exhibits, (b) The declaration of Dr. Nenand Medvidovic Ph.D and supporting exhibits, and (c) the Petition for *Inter Partes* Review of the ’154 patent to which my declaration relates.

III. BASED ON THE TEACHINGS OF ROSS, A PERSON OF ORDINARY SKILL IN THE ART WOULD HAVE KNOWN THAT THE HOOK SCRIPTS COULD INCLUDE A CALL TO A FIRST FUNCTION

3. It remains my opinion that based on the disclosures in Ross, it would have been obvious to a person of ordinary skill in the art, in 2005, to ensure that the act of having a hook function supersede a call to an original function could be achieved via a call to a hook function within the hook script.

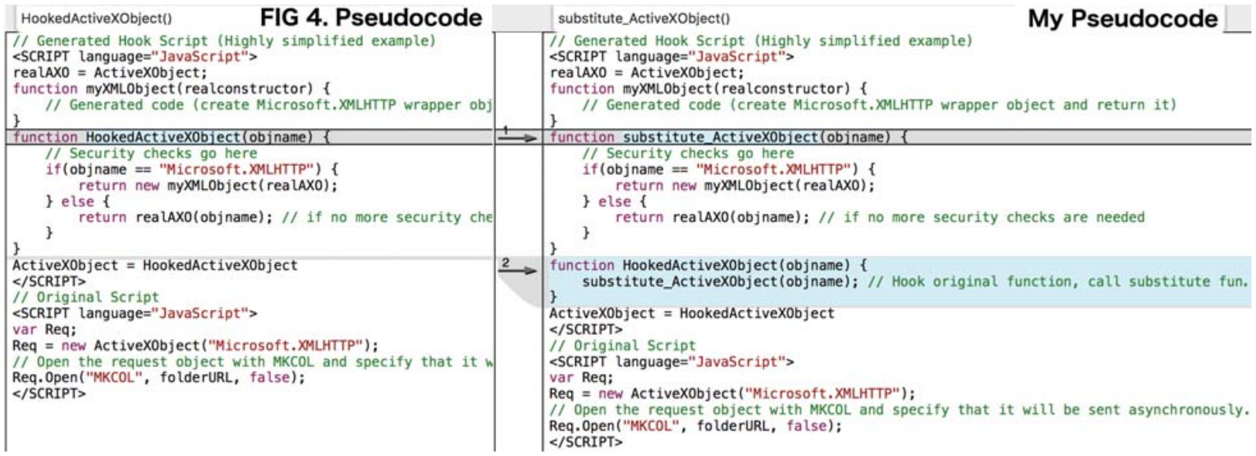
4. Ross states that the example high-level pseudocode provided in FIG. 4 is meant only as an example, and that the functionality of the code provided in FIG. 4 could be achieved through other means. "FIG. 4 shows an example of a combined script 402 including a generated hook script 404 and original script code 302 shown in FIG. 3, according to an embodiment of the present invention. Although shown as a single, combined script 402, generated hook script 404 and original script code 302 may be introduced, or injected, into script processing engine 618 individually by any means as long as a hook script function corresponding to an original script function is processed first." (Ex. 1003 ¶ 31 (emphasis added).)

5. The pseudocode provided in FIG. 4 illustrates one example of a way to ensure that a call to a hook function supersedes a call to its corresponding

original function. The pseudocode of FIG. 4 appears to be high-level pseudocode written in the JavaScript programming language.

6. It would have been obvious to a POSITA, at the time of the filing of the ’154 patent, that one way to ensure that the hook script function corresponding to an original script is processed first would have been to include a call to the hook function within the hook script.

7. Below, I provide pseudocode (also written in high-level JavaScript pseudocode) that illustrates a method for ensuring that a hook function is called first, before the original function, that utilizes a call to the hook function within the hook script itself. I also include Ross’ FIG. 4 pseudocode on the left of the illustration to show how few edits are necessary to achieve this method.



```
// Generated Hook Script (Highly simplified example)
<SCRIPT language="JavaScript">
realAXO = ActiveXObject;
function myXMLObject(realconstructor) {
    // Generated code (create Microsoft.XMLHTTP wrapper obj
}
function HookedActiveXObject(objname) {
    // Security checks go here
    if(objname == "Microsoft.XMLHTTP") {
        return new myXMLObject(realAXO);
    } else {
        return realAXO(objname); // if no more security che
    }
}
ActiveXObject = HookedActiveXObject
</SCRIPT>
// Original Script
<SCRIPT language="JavaScript">
var Req;
Req = new ActiveXObject("Microsoft.XMLHTTP");
// Open the request object with MKCOL and specify that it w
Req.Open("MKCOL", folderURL, false);
</SCRIPT>
```

```
// Generated Hook Script (Highly simplified example)
<SCRIPT language="JavaScript">
realAXO = ActiveXObject;
function myXMLObject(realconstructor) {
    // Generated code (create Microsoft.XMLHTTP wrapper object and return it)
}
function substitute_ActiveXObject(objname) {
    // Security checks go here
    if(objname == "Microsoft.XMLHTTP") {
        return new myXMLObject(realAXO);
    } else {
        return realAXO(objname); // if no more security checks are needed
    }
}
function HookedActiveXObject(objname) {
    substitute_ActiveXObject(objname); // Hook original function, call substitute fun.
}
ActiveXObject = HookedActiveXObject
</SCRIPT>
// Original Script
<SCRIPT language="JavaScript">
var Req;
Req = new ActiveXObject("Microsoft.XMLHTTP");
// Open the request object with MKCOL and specify that it will be sent asynchronously.
Req.Open("MKCOL", folderURL, false);
</SCRIPT>
```

8. In my pseudocode, I show a combined script that includes the original script code, a generated hook script, and a hook function. A call to the

original function ActiveXObject invokes the hook script HookedActiveXObject, this script then calls the hook function substituteActiveXObject, and lastly some associated security checks are called. I name the hook function with the familiar "substitute_" prefix to emphasize the similarity between my pseudocode and that of Table III in the '154 patent. My hook function is equivalent to a call to the original function with a corresponding call to a substitute function as described in the '154 patent.

9. A person of skill in the art would have known that invoking the hook script in the manner described above, and having the hook script include a call to a hook function was an available method to ensure that the hook script function corresponding to an original script is processed first. A person of skill in the art would have readily generated similar pseudocode provided above to effect the functionality described in Ross.

10. In addition, the pseudocode provided in FIG. 4 (reproduced below, with annotations) itself suggests a call to a function within the hook script.

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.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase®
Smarter legal research.