Trials@uspto.gov
571-272-7822

Paper 62
Entered: March 15, 2017

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

PALO ALTO NETWORKS, INC. and SYMANTEC CORP.,
Petitioner,

v.

FINJAN, INC.,
Patent Owner.

Case IPR2015-01979[1]
Patent 8,141,154 B2

Before, THOMAS L. GIANNETTI, RICHARD E. RICE, and MIRIAM L. QUINN, *Administrative Patent Judges.*

QUINN, *Administrative Patent Judge.*

FINAL WRITTEN DECISION
35 U.S.C. § 318(a) and 37 C.F.R. § 42.73

[1] This case is joined with IPR2016-00919. Paper 28 ("Decision on Institution of *Inter Partes* Review and Grant of Motion for Joinder," filed by Symantec Corp.).

Palo Alto Networks, Inc. and Symantec Corp. (collectively, "Petitioner") have each filed petitions to institute *inter partes* review of claims 1–8, 10, and 11 of U.S. Patent No. 8,141,154 B2 ("the '154 patent") pursuant to 35 U.S.C. § 311–319. In response to the first petition, filed by Palo Alto Networks, Inc.,[2] Finjan, Inc. ("Patent Owner") filed a Preliminary Response. Paper 6 ("Prelim. Resp."). Upon consideration of the Petition and the Preliminary Response filed by Finjan, we instituted trial as to all the challenged claims. Paper 8 ("Dec.").

Subsequently, Symantec filed a petition seeking review of the same claims of the '154 patent. IPR2016-00919, Paper 3. With this second petition, Symantec filed a motion to join IPR2016-00919 with this proceeding. We granted Symantec's motion, joined the cases, terminated IPR2016-00919, and ordered consolidation of all Petitioner filings in this proceeding. Paper 10, at 5.

During trial, Patent Owner filed a Patent Owner Response;[3] and Petitioner filed a Reply.[4] Patent Owner also filed Motions for Observations of the November 14, 2016 cross- examination of Petitioner's declarant, Dr. Aviel Rubin. Paper 47 ("Mot. for Obs."). Petitioner responded to Patent Owner's Motion for Observations. Paper 49 ("Resp. Obs."). Both parties also filed Motions to Exclude. Paper 46 ("Pet. Mot. to Exclude"); Paper 48 ("PO Mot. to Exclude"). Both parties filed Oppositions and Replies concerning the Motions to Exclude. Papers 50, 51, 53, 55.

[2] Paper 2 ("Petition" or "Pet.").
[3] Paper 22 ("PO Resp.").
[4] Paper 35 ("Reply").

An oral hearing was held on December 15, 2016.[5]

We have jurisdiction under 35 U.S.C. § 6. This Final Written Decision is issued pursuant to 35 U.S.C. § 318(a). For the reasons discussed herein, and in view of the record in this trial, we determine that Petitioner has not shown by a preponderance of the evidence that claims 1–8, 10, and 11 of the '154 patent are unpatentable.

I. BACKGROUND

A. RELATED MATTERS

Petitioner identifies that the '154 patent as the subject of various district court cases filed in the U.S. District Court for the Northern District of California (Case Nos. 3:14-cv-04908, 3:14-cv-02998, 5:15-cv-01353, 5:14-cv-04398, 3:14-cv-01197, and 3:13-cv-05808). Pet. 3. Petitioner also states that petitions for *inter partes* review have been filed regarding other related patents. *Id.* The '154 patent is also the subject of another *inter partes* review: IPR2016-00151 (and IPR2016-01071, joined therewith). In IPR2016-0151, we have issued a Final Written Decision, under 35 U.S.C. § 318(a), concurrently with the instant Final Written Decision.

B. INSTITUTED GROUNDS

We instituted *inter partes* review of claim 1–8, 10, and 11 ("the challenged claims") based on the following specific grounds:

[5] A transcript of the oral hearing is entered in the record as Paper 60 ("Tr.").

Reference[s]	Basis	Claims challenged
Khazan[6] and Sirer[7]	35 U.S.C.§ 103	1–5
Khazan, Sirer, and Ben-Natan[8]	35 U.S.C. § 103	6–8, 10, and 11

Petitioner supports its contentions of unpatentability with declarations from Dr. Aviel Rubin. Ex. 1002 ("Aviel Declaration"); Ex. 1045 ("Supp. Aviel Declaration"). Patent Owner supports its contentions with a declaration from Dr. Nenad Medvidovic. Ex. 2002 ("Medvidovic Declaration"). The cross-examinations of Dr. Rubin and Dr. Medvidovic are entered in the record as Exhibits 2005 and 1038, respectively.

C. THE '154 PATENT (EX. 1001)

The '154 patent relates to computer security and, more particularly, to systems and methods for protecting computers against malicious code such as computer viruses. Ex. 1001, 1:7–9, 8:38–40. The '154 patent identifies the components of one embodiment of the system as follows: a gateway computer, a client computer, and a security computer. *Id.* at 8:45–47. The gateway computer receives content from a network, such as the Internet, over a communication channel. *Id.* at 8:47–48. "Such content may be in the form of HTML pages, XML documents, Java applets and other such web content that is generally rendered by a web browser." *Id.* at 8:48–51. A content modifier modifies original content received by the gateway

[6] Patent Application Pub. No. US 2005/0108562 A1 (Exhibit 1003) ("Khazan").
[7] Sirer et al., *Design and Implementation of a Distributed Virtual machine for Networked Computers* (1999) (Exhibit 1004) ("Sirer").
[8] U.S. Patent No. 7,437,362 B1 (Exhibit 1005) ("Ben-Natan").

computer and produces modified content that includes a layer of protection to combat dynamically generated malicious code. *Id.* at 9:13–16.

D. ILLUSTRATIVE CLAIM

Challenged claims 1, 4, 6, and 10 are independent, and illustrative claim 1 is reproduced below.

> 1. A system for protecting a computer from dynamically generated malicious content, comprising:
>
> a content processor (i) for processing content received over a network, the content including a call to a first function, and the call including an input, and (ii) for invoking a second function with the input, only if a security computer indicates that such invocation is safe;
>
> a transmitter for transmitting the input to the security computer for inspection, when the first function is invoked; and
>
> a receiver for receiving an indicator from the security computer whether it is safe to invoke the second function with the input.

II. ANALYSIS

A. CLAIM INTERPRETATION

In an *inter partes* review, claim terms in an unexpired patent are interpreted according to their broadest reasonable construction in light of the specification of the patent in which they appear. 37 C.F.R. § 42.100(b); *Cuozzo Speed Techs., LLC v. Lee*, 136 S. Ct. 2131, 2142–46 (2016). Consistent with that standard, claim terms also are given their ordinary and customary meaning, as would be understood by one of ordinary skill in the art in the context of the entire disclosure. *See In re Translogic Tech., Inc.*, 504 F.3d 1249, 1257 (Fed. Cir. 2007). There are, however, two exceptions to that rule: "1) when a patentee sets out a definition and acts as his own

lexicographer," and "2) when the patentee disavows the full scope of a claim term either in the specification or during prosecution." *See Thorner v. Sony Computer Entm't Am. LLC*, 669 F.3d 1362, 1365 (Fed. Cir. 2012).

If an inventor acts as his or her own lexicographer, the definition must be set forth in the specification with reasonable clarity, deliberateness, and precision. *Renishaw PLC v. Marposs Societa' per Azioni*, 158 F.3d 1243, 1249 (Fed. Cir. 1998) (citing *In re Paulsen*, 30 F.3d 1475, 1480 (Fed. Cir. 1994)). Although it is improper to read a limitation from the specification into the claims, *In re Van Geuns*, 988 F.2d 1181, 1184 (Fed. Cir. 1993), claims still must be read in view of the specification of which they are a part. *Microsoft Corp. v. Multi-Tech Sys., Inc.*, 357 F.3d 1340, 1347 (Fed. Cir. 2004).

"content"

In our Decision on Institution, we did not construe expressly any claim terms. Dec. 5. During trial, however, Patent Owner proposed a construction of the term "content" as "a data container that can be rendered by a client web browser." PO Resp. 5. Petitioner challenges this construction as unduly narrow in view of the Specification. Reply 6. In particular, Petitioner argues that the Specification does not define the term and provides no "clear disavowal" of claim scope. *Id.* 6–7. According to Petitioner, the Specification and extrinsic evidence support a broader construction of "content" to mean "code." *Id.* at 7–8 (citing Ex. 1001, 12:49–52; Ex. 2005, 80:11–23).

Because they are not consistent with the broadest reasonable interpretation in light of the specification, and as discussed further below, we

do not adopt either of the parties' proposed constructions. Our reasoning follows.

The '154 patent is titled "System and Method for Inspecting Dynamically Generated Executable Code." Ex. 1001, [54]. Although the title refers to "executable code," the term "content" is used elsewhere in the patent when describing the invention. The Abstract further clarifies that a "method for protecting a client computer from dynamically generated malicious *content*, includ[es] receiving at a gateway computer *content* being sent to a client computer for processing, the *content* including a call to an original function[.]" *Id.* Abstract (emphasis added). The gateway computer modifies the "content," which is then transmitted to the client computer for processing there. *Id.*

By way of background, the '154 patent explains that the "ability to run executable code such as scripts within Internet browsers" has caused a new form of viruses "embedded within web pages and other web content, and[, which] begin executing within an Internet browser as soon as they enter a computer." *Id.* at 1:34–40. In particular, the '154 patent describes these new "dynamically generated viruses" as "taking advantage of features of dynamic HTML generation, such as executable code or scripts that are embedded within HTML pages, to generate themselves on the fly at runtime." *Id.* at 3:31–39. Therefore, according to the '154 patent "dynamically generated malicious code cannot be detected by conventional reactive content inspection and conventional gateway level behavioral analysis content inspection, since the malicious JavaScript is not present in the content prior to run-time." *Id.* at 3:65–4:2. The invention, therefore, seeks to protect against "dynamically generated malicious code, in addition

to conventional computer viruses that are statically generated." *Id.* at 4:30–34.

To accomplish this objective, the '154 patent describes the gateway computer receiving "content from a network, such as the Internet, over a communication channel." *Id.* at 8:47–48. The "content may be in the form of HTML pages, XML documents, Java applets and other such web content that is generally rendered by a web browser." *Id.* at 8:48–51; *see also id.* at 13:49–52 ("Such content may be in the form of an HTML web page, an XML document, a Java applet, an EXE file, JavaScript, VBScript, an Active X Control, or any such data container that can be rendered by a client web browser."); 13:49–52. A "content modifier 265" at the gateway modifies "original content received" by the gateway computer and produces modified "content, which includes a layer of protection to combat dynamically generated malicious code." *Id.* at 9:13–16. It does this by scanning the "original content" and identifying certain function calls. *Id.* at 9:16–20. Selected function calls are then replaced with a corresponding substitute function call. *Id.* at 9:21–26.

One example of a function call in the original content is identified as "Document.write ('content that is dynamically generated at run-time')." *Id.* at 11:55–12:2. The original content is modified by replacing the original function call Document.write() with a substitute function call Substitute_document.write(). *Id.* at 10:31–36. The client computer then receives the "content, as modified by the gateway computer." *Id.* at 11:63–64. And it is this modified content that the client computer processes,

by invoking the substitute function call and transmitting the input of that substitute function for inspection. *Id.* at 16:22–29.

From the above descriptions, we understand the '154 patent Specification to refer to three categories of content. First, there is the "original content" that is scanned and modified at the gateway computer. Second, there is the "modified content" transmitted to, and received by, the client computer. Third is the "dynamically generated malicious content" that is generated at runtime and, thus, is undetected by the gateway computer in the "original content."

We also understand that the purpose of the '154 patent is to protect the client computer from this "dynamically generated malicious content," which is sometimes also referred to in the Specification as "dynamically generated malicious code." *See, e.g.*, Ex. 1001, 4:31–33 ("new behavioral analysis technology affords protection against dynamically generated malicious code"); 4:38–40 ("before the client computer invokes a function call that may potentially dynamically generate malicious code"); 8:17–20 ("FIG. 2 is a simplified block diagram of a system for protecting a computer from dynamically generated malicious executable code, in accordance with a preferred embodiment of the present invention"); 8:38–40 ("The present invention concerns systems and methods for protecting computers against dynamically generated malicious code.").

Notwithstanding the variety of content described in the Specification, the term "content" is recited broadly in all challenged claims as "content including a call to a first function." For example, claim 1 recites a content processor for "processing content received over a network, the content

including a call to a first function, and the call including an input." *Id.* at 17:34–36.

The claim language also requires that the processed "content" be received over a network. Because the recited "first function" is the substituted function whose input is verified, the *claimed* "content," in the context of the surrounding claim language, must refer to the modified content received at the client computer. *See id.* at 17:39–40 ("transmitting the input [of the first function call] to the security computer for inspection, when the first function is invoked"). The claimed content cannot refer to the "original content" that is received by the gateway computer and over the Internet because that content, according to the Specification, would be capable of generating the undetected dynamically generated malicious content from which the client computer is to be protected.

Based on this understanding, we do not agree with Patent Owner that the recited "content" is "a data container that can be rendered by a client web browser." *See* PO Resp. 6. Although the Specification states that "content *may be* in the form of an HTML web page, an XML document, a Java applet, an EXE file, JavaScript, VBScript, an ActiveX Control, or any such data container that can be rendered by a client web browser," that passage describes the "original content," not the "modified content." *See* Ex. 1001, 13:49–52. Furthermore, even if that description were applicable to the "modified content," the Specification uses the permissive words "may" and "can," which suggests that the description of the form of the content in the Specification was not intended to set forth a definition for the term "content." *See i4i Ltd. P'ship v. Microsoft Corp.*, 598 F.3d 831, 844

(Fed. Cir. 2010) (declining to limit claim term where the specification used permissive language).

Furthermore, although the Specification addresses embodiments concerning web pages received over the Internet, the Specification does not limit the "content" to web content only, or to content that can be rendered by a web browser. For example, in describing a content processor, the Specification states that it "*may* be a web browser running on client computer 210." Ex. 1001, 10:60–62. This description again uses permissive language that suggests the intent not to limit the content to a data container that can be rendered by a client web browser. We also find it informative that in discussing the communication channels over which the client computer receives the "modified content," the Specification states that "communication channels 220, 225 and 230 [of Figure 2] may each be multiple channels using standard communication protocols such as TCP/IP." Ex. 1001, 8:67–9:2.[9] That is, the network over which the content is received may be any network that delivers data using a standard communication protocol, not just the Internet.

Accordingly, we are not persuaded that the Specification supports a construction of "content" that is limited to the specific embodiment of a data container that can be rendered by a client web browser, as Patent Owner argues. *In re Van Geuns*, 988 F.2d 1181, 1184, (Fed. Cir. 1993) ("Moreover, limitations are not to be read into the claims from the specification.") (internal citations omitted).

[9] TCP/IP is an abbreviation for Transmission Control Protocol over Internet Protocol, and it is the most widely used communication protocol for delivery of data over networks, including the Internet. *TCP/IP*, WILEY ELECTRICAL AND ELECTRONICS ENGINEERING DICTIONARY, 774 (2004) (Ex. 3001).

We are not persuaded, in addition, that Petitioner has made a sufficient showing that a person of ordinary skill in the art would understand the plain meaning of "content" as "code." To support its proposed construction, Petitioner relies on the cross-examination testimony of its own expert, Dr. Aviel Rubin. Ex. 2005, 80:11–23. His testimony, however, is not persuasive because he proffers no reasoning for the conclusion that "content" is "code" under the broadest reasonable interpretation:

> Q What is your understanding of what "content" means?
>
> A In the context of the '154 patent, content would be code.
>
> Q What do you mean by code?
>
> A Code, like an HTML page that has JavaScript in it.
>
> Q When you say code, do you mean any type of code?
>
> A Well, if you just say content, we are going to take the broadest reasonable interpretation of that. It would be any type of code, yes.

Id.[10]

Although it seems reasonable to say that the content includes "code," no persuasive evidence limits the claimed content to only code. As we noted above, the Specification refers to code, sometimes interchangeably with content, but only in the context of dynamically generated code. The dynamically generated code, however, is not generated until runtime and, therefore, is not contained in the "modified content" that the client receives. *See* Ex. 1001, 3:65–4:2 ("dynamically generated code cannot be detected by conventional reactive content inspection and conventional gateway level

[10] We do not give weight to the testimony proffered by Dr. Medvidovic with regard to claim construction of this term given the contradictory positions asserted in this regard. *See* Reply 8.

behavioral analysis content inspection, since the malicious JavaScript is not present in the content prior to run-time."). Furthermore, the Specification describes various *forms* in which the content occurs, such as an HTML web page and Java applets (*id.* at 13:49–52), but does not address sufficiently what is the "content" itself. *But see*, *id.* at 11:50–51 ("suppose the content is an HTML page").

Given the broad disclosure of a network, as discussed above, the reference to a "data container" (*id.* at 13:51–52) and "network content" (*id.* at 4:37–37), the concern over scripts embedded in web pages or "other web content" (*id.* at 1:37–39), we conclude that the Specification of the '154 patent uses the claimed "content" to refer broadly to the data or information, modified for processing, that the client receives from the network, where, in the case of the Internet, it may refer to a web page and its elements. This interpretation is consistent also with the meaning of the term in the art, as evidenced by dictionaries concerning computing and engineering. *See content*, Microsoft Computer Dictionary, 125 (5th ed. 2002) (Ex. 3002) (defining "content" as (1) "the data that appears between the starting and ending tags of an element in an SGML, XML, or HTML document. The content of an element may consist of plain text or other elements," (2) "The message body of a newsgroup article or e-mail message;" and (3) "The 'meat' of a document, as opposed to its format or appearance."); *see also content*, WILEY ELECTRICAL AND ELECTRONICS ENGINEERING DICTIONARY, 142 (2004) (Ex. 3001) ("Information, especially that which is available online, which may be any combination of text, audio, video, files, or the like.").

Accordingly, under the broadest reasonable interpretation in the context of the Specification and the surrounding claim language, we conclude that "content" is data or information, which has been modified and is received over a network.

"call to a first function"

The term "call to a first function" is recited in all challenged claims. The arguments presented regarding this limitation turn on the scope of the word "call." Specifically, Patent Owner attempts to distinguish the claims over Khazan by arguing that a "jump" instruction is not the recited "call" to a function. PO Resp. 25–27. Dr. Medvidovic, Patent Owner's expert, proffers opinions on the issue by relying on a definition of "function call" derived from the Microsoft Press Computer Dictionary. Ex. 2002 ¶ 110 (citing Ex. 2014). That Dictionary provides that a "function call" is "[a] program's request for the services of a particular function." *Id.*; Ex. 2014. It also explains that "[a] function call is coded as the name of the function along with any parameters needed for the function to perform its task." *Id.*

The Specification of the '154 patent does not define the term "call to a first function." The Specification, however, does use the phrase "function call" to state that "before the client computer invokes a *function call* that may potentially dynamically generate malicious code, the client computer passes the input to the function to the security computer for inspection." Ex. 1001, 4:37–43 (emphasis added). The Specification also states that "the present invention operates by replacing original function calls with substitute function calls within the content, at a gateway computer, prior to the content being received at the client computer." *Id.* at 4:57–60. Therefore, we understand the Specification to use the phrase "function call" in the same

sense as the phrase "call to a [] function." That is, a program instruction specifies the function name and its parameters, where execution of the instruction results in the function providing a service. Thus, we find the dictionary definition of the term "function call" applicable here and indicative of the meaning of the term to a person of ordinary skill in the art.

Furthermore, the dictionary definition is consistent with the embodiments described in the Specification. For example, one embodiment of the '154 patent provides for modifying an original function call with "corresponding function calls Substitute_function(input,*)." *Id.* at 9:21–24. That is, the specification describes that the services of the function Substitute_function are being requested by the modified content. Furthermore, the format of the function in this particular embodiment, identifies the name of the function and the parameters "input" and "*". *See also id.*at 9:26–28 (explaining that the "input intended for the original function is also passed to the substitute function, along with possible additional input denoted by '*'"). We note that the "first function" is the substitute function included in the modified content, as discussed above in connection with our analysis of the term "content."

We recognize that the definition of "call to a first function" need not define the particular format of the instruction or further detail regarding its parameters. We reach this determination because the claim language itself requires that either the "call" or the "function" include an input. For example, claim 1 recites the "call including an input," while claim 6 recites "the first function including an input variable."

Accordingly, we determine that a "call to a first function" means an a statement or instruction in the content, the execution of which causes the function to provide a service.

B. PRINCIPLES OF LAW

A claim is unpatentable under 35 U.S.C. § 103(a) if the differences between the claimed subject matter and the prior art are such that the subject matter, as a whole, would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 406 (2007). The question of obviousness is resolved on the basis of underlying factual determinations including: (1) the scope and content of the prior art; (2) any differences between the claimed subject matter and the prior art; (3) the level of ordinary skill in the art; and (4) objective evidence of nonobviousness. *Graham v. John Deere Co.*, 383 U.S. 1, 17–18 (1966).

C. THE LEVEL OF SKILL IN THE ART

In determining the level of ordinary skill in the art at the time of the invention, we note that various factors may be considered, including "type of problems encountered in the art; prior art solutions to those problems; rapidity with which innovations are made; sophistication of the technology; and educational level of active workers in the field." *In re GPAC, Inc.*, 57 F.3d 1573, 1579 (Fed. Cir. 1995) (citing *Custom Accessories, Inc. v. Jeffrey-Allan Indus., Inc.*, 807 F.2d 955, 962 (Fed. Cir. 1986)).

Petitioner asserts, through its expert, Dr. Aviel Rubin, that the "relevant technology field for the '154 patent is security programs, including content scanners for program code." Ex. 1002 ¶ 21. Further, Dr. Rubin

opines that a person of ordinary skill in the art would "hold a bachelor's degree or the equivalent in computer science (or related academic fields) and three to four years of additional experience in the field of computer security, or equivalent work experience." *Id*.

Patent Owner, through its expert, Dr. Nenad Medvidovic, offers a level of ordinary skill that is different from Petitioner's. Ex. 2002 ¶ 35. In Particular, Dr. Medvidovic opines that a person of ordinary skill in the art would have a "bachelor's degree in computer science or related field, and either (1) two or more years of industry experience and/or (2) an advanced degree in computer science or related field." *Id.* In comparison, it appears that the minimum experience under Patent Owner's proffered level of skill is one year less than Petitioner's. Also, Patent Owner proffers an alternative to work experience, namely an advanced degree. There is no specific articulation regarding how the difference of one year experience or the proposed alternative of an advanced degree in lieu of experience tangibly affects our obviousness inquiry. Further, there is no evidence in this record that the differences noted above impact in any meaningful way the level of expertise of a person of ordinary skill in the art. Indeed, we note that Dr. Medvidovic's opinions would not change if he had considered instead the level or ordinary skill in the art proffered by Dr. Rubin. *Id*. ¶ 38.

Accordingly, we determine that in this case no express definition of the level of ordinary skill in the art is necessary and that the level of ordinary skill in the art is reflected by the prior art of record. *See Okajima v. Bourdeau*, 261 F.3d 1350, 1355 (Fed. Cir. 2001); *In re GPAC Inc.*, 57 F.3d 1573, 1579 (Fed. Cir. 1995); *In re Oelrich*, 579 F.2d 86, 91 (CCPA 1978).

D. Obviousness Ground Based on Khazan and Sirer

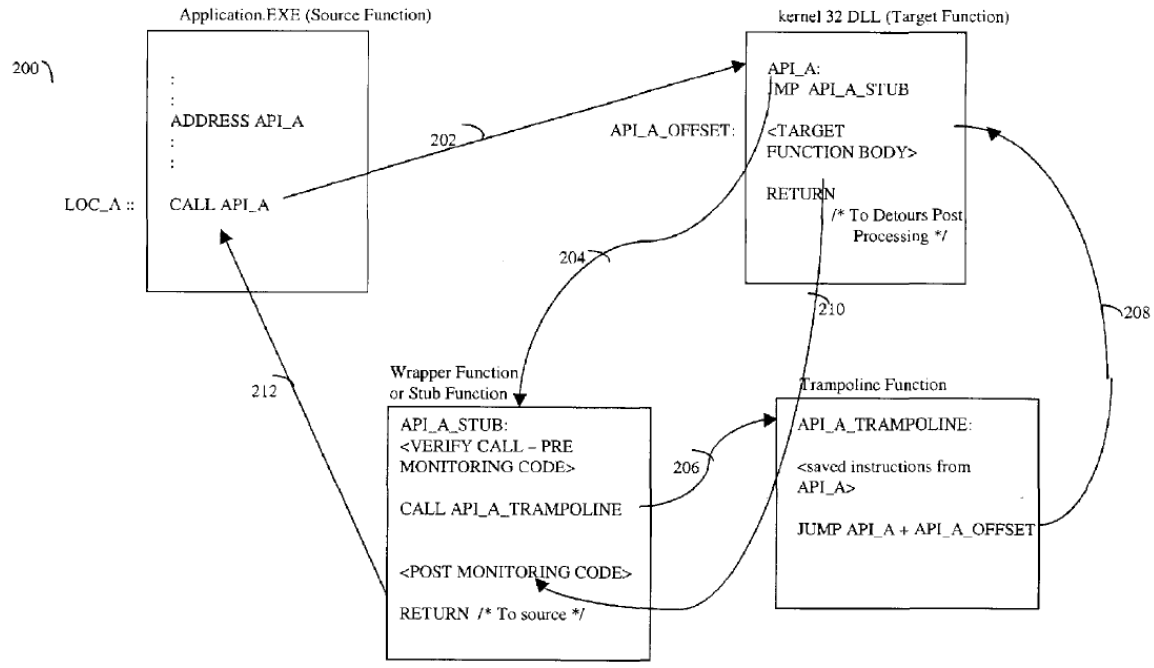
Petitioner asserts that Khazan discloses "every element of the Petitioned Claims except a modified input variable and details of performing dynamic analysis on a remote computer." Pet. 16. In particular, Petitioner relies on a combination of Khazan and Sirer as teaching the "content including a call to a first function," "only if a security computer indicates that such invocation is safe," "transmitter," and "receiver" limitations. Pet. 20–39. Petitioner relies on Khazan alone as disclosing the remaining limitations of independent claims 1 and 4. *Id*. at 19–20.

1. Overview of Khazan (Exhibit 1003)

Khazan is titled "Technique for detecting executable malicious code using a combination of static and dynamic analyses." The Abstract of Khazan states that:

> Described are techniques used for automatic detection of malicious code by verifying that an application executes in accordance with a model defined using calls to a predetermined set of targets, such as external routines. A model is constructed using a static analysis of a binary form of the application, and is comprised of a list of calls to targets, their invocation and target locations, and possibly other call-related information. When the application is executed, dynamic analysis is used to intercept calls to targets and verify them against the model.

Ex. 1003, Abstract. Figure 7, reproduced below, shows in more detail the flow of control between functions at run time to intercept calls to the predetermined functions or routines being monitored as part of dynamic analysis. *Id.* ¶ 25.



The flow in Figure 7 depicts the control flow when a WIN32 API function is invoked at run time from an application using a call instruction. *Id.* ¶ 82. A call is made to the target function API_A. *Id.* ¶ 83. Control transfers (arrow 202) to the target function API_A within the kernel32 DLL. *Id.* The target function API_A includes a transfer or jump instruction to a wrapper function. *Id.* Control, therefore, transfers (arrow 204) to the wrapper function (API_A_STUB). *Id.* The intercepted call is verified. *Id.* ¶ 84. This verification includes using static analysis information, including parameter information. *Id.* ¶ 87. After verification, a trampoline function is invoked (arrow 206) to execute previously saved instructions of API_A, which are the first instructions of the routine API_A that were replaced with a jump instruction to the wrapper function. *Id.* ¶ 88. Control transfers back to the target function to continue execution of the target function body as indicated by arrow 208. *Id.*

2. Overview of Sirer (Ex. 1004)

Sirer is a technical paper from an ACM symposium titled "Design and implementation of a distributed virtual machine for networked computers." Ex. 1004, 1. Sirer describes centralizing service functionality in a distributed virtual machine by portioning static and dynamic components. *Id* at 2. Figure 1, reproduced below, illustrates the organization of those components.

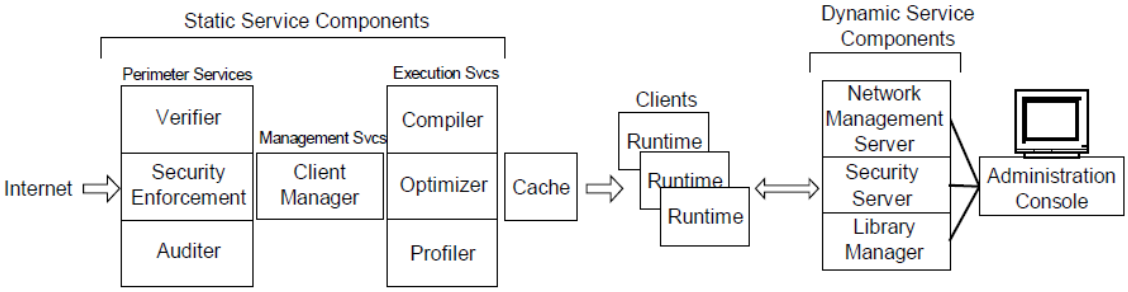


Figure 1. The organization of static and dynamic service components in a distributed virtual machine.

Figure 1 shows static service components, such as security enforcement, running at a network trust boundary. *Id.* at 3. Dynamic service components provide service functionality to clients during run-time as necessary. *Id.* "The code for the dynamic service components resides on the central proxy and is distributed to clients on demand." *Id.* at 4. The security service "forces applications to comply with an organization's security policy by inserting appropriate checks through binary rewriting." *Id.* at 5. "During execution of the rewritten application, the enforcement manager executes the inserted access checks, querying the security service based on the security identifiers and permissions it maintains." *Id.*

3. Whether Sirer is a Printed Publication

Patent Owner contends that Sirer is not prior art under 35 U.S.C. § 102(b) because Petitioner, according to Patent Owner, has failed to

demonstrate that Sirer was publicly accessible. PO Resp. 7–11. In particular, Patent Owner argues that Sirer was not indexed properly and that the location and manner of display of the journal containing it was insufficient to render Sirer publicly accessible. *Id*.

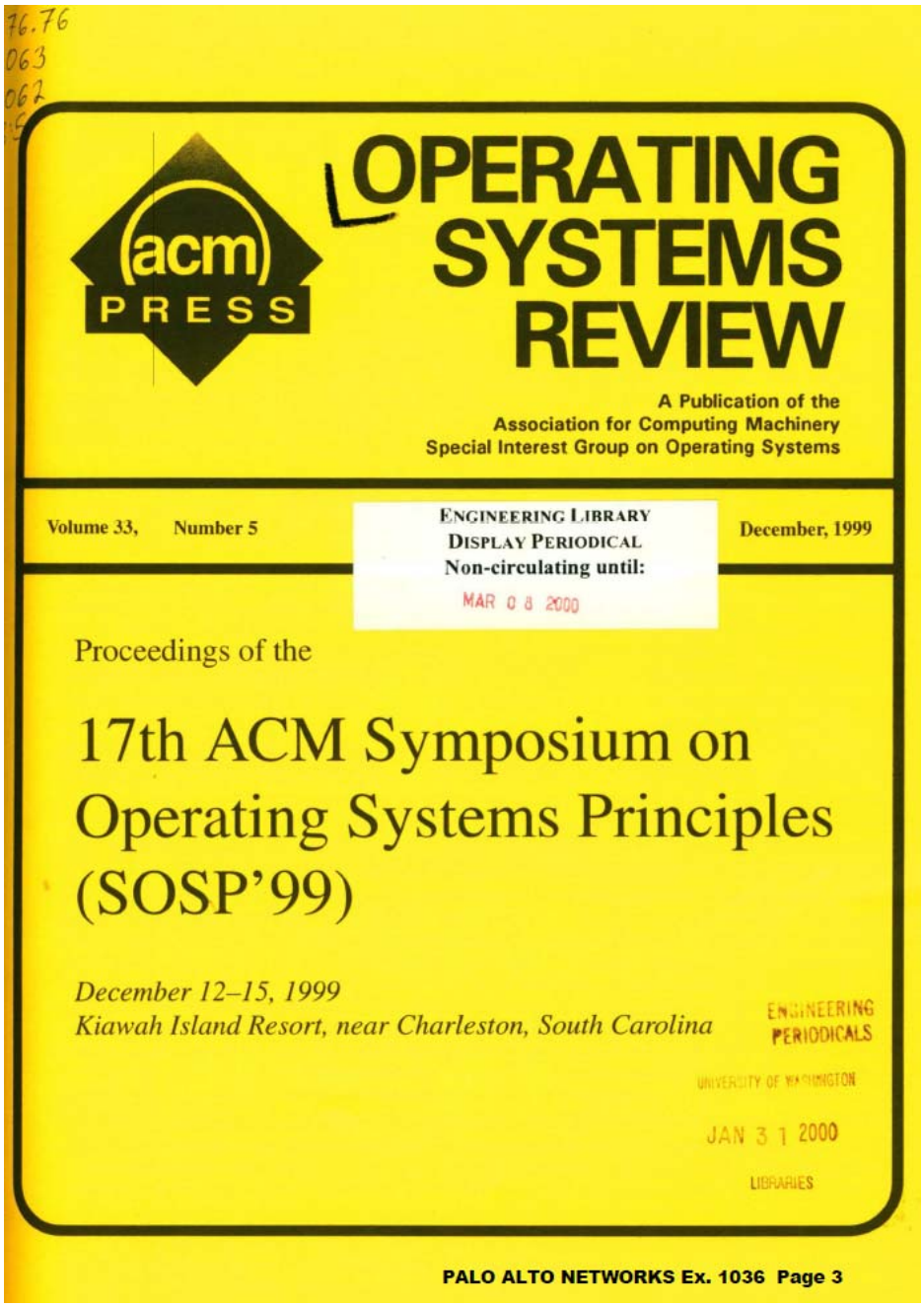
By way of background, Petitioner submitted Sirer as Exhibit 1004, which shows on its face that the reference was included in the Operating Systems Review of the Association of Computing Machinery ("ACM"). *See* Ex. 1004 at 1. For instance, in the upper right corner of the article, a header states that the 17th ACM Symposium on Operating Principles is "*[p]ublished*" as Operating Systems Review 34(5):202–216, December 1999. *Id*. The bottom footer provides a copyright notice dated 1999 by ACM and a statement providing limited rights to copy and to *republish* for a fee or specific permission. *Id*. Petitioner alleges in the Petition that Sirer's publication date is December 1999. Pet. 5.[11] In response to Patent Owner's objections that Sirer's publication date of December 1999 is hearsay and inadmissible evidence of its public accessibility (Paper 10, 2), Petitioner provided supplemental evidence in the form a declaration from a librarian and a library copy of Sirer from an actual Operating Systems Review periodical (Ex. 1036, 3).

[11] The Petition provides as support Mr. Sirer's declaration (Ex. 1008) and a U.S. Patent issued in 2001, which cites Sirer (Ex. 1024). We give no weight to the Sirer Declaration filed as Exhibit 1008. Petitioner failed to produce Mr. Sirer for cross-examination, as our procedures require. *See* PO Mot. to Exclude, (Paper 49) 5–7. As for considering another patent's citation of Sirer, we find that it does not support the assertion that Sirer was published on December 1999. At best, a citation to Sirer in another patent may offer some indicia that the article was available, but the mere citation is not proof of publication or accessibility.

The determination of whether a particular reference qualifies as a prior art printed publication "involves a case-by-case inquiry into the facts and circumstances surrounding the reference's disclosure to members of the public." *In re Klopfenstein*, 380 F.3d 1345, 1350 (Fed. Cir. 2004). The key inquiry is whether the reference was made "sufficiently accessible to the public interested in the art" before the critical date. *In re Cronyn*, 890 F.2d 1158, 1160 (Fed. Cir. 1989). "A reference will be considered publicly accessible if it was 'disseminated or otherwise made available to the extent that persons interested and ordinarily skilled in the subject matter or art exercising reasonable diligence, can locate it.'" *Blue Calypso, LLC v. Groupon, Inc.*, 815 F.3d 1331, 1348 (Fed. Cir. 2016) (quoting *Kyocera Wireless Corp. v. Int'l Trade Comm'n*, 545 F.3d 1340, 1350 (Fed. Cir. 2008)).

Having reviewed the parties' arguments and supporting evidence, we determine that Petitioner has demonstrated sufficiently that Sirer is a printed publication based on the following reasons and factual findings. First, we find that Sirer was published in Volume 33, issue number 5 of the Operating System Review published by ACM. We base our findings on the testimony of Mel DeSart, head librarian of the University of Washington Engineering Library, and the printed material attached as Exhibit A to the declaration of Mel DeSart, filed as Exhibit 1036. We also support our findings based on the totality of the indicia of publication found on Sirer, Exhibit 1004. As noted above, the indicia on the face of Exhibit 1004 in its totality assures us that Sirer is a printed publication. Notwithstanding the copyright date, the first page of the article conveys that the article is published in a volume of the Operating Systems Review, an ACM publication. *See* Ex. 1004, 1. That

indicia is consistent with the printed material provided as Exhibit 1036 and authenticated by Mr. DeSart. *See* Ex. 1036. For example, on page 3 of Exhibit 1036, reproduced below, the cover of the periodical states that Operating Systems Review is "a publication of the Association for Computing Machinery Special Interest Group on Operating Systems." *Id.* at 3.



The cover page reproduced above identifies Volume 33, Number 5, and date December 1999 as containing the "Proceedings of the 17th ACM Symposium on Operating Systems Principles (SOSP'99)." *Id.* This cover page also contains indicia of circulation to the public, such as by its receipt and cataloging at the Engineering Library of the University of Washington.

Id. (displaying a stamp labeled "Engineering Periodicals, University of Washington, Jan 31, 2000"). Additionally, the cover page includes a label stating "Engineering Library Display Periodical Non−circulating until: Mar 08 2000." *Id*. The stamps and labels are described by Mr. Melvin ("Mel") DeSart, Head of the University of Washington Engineering Library, as evidence that the Library's process was to stamp a received periodical and to affix a label when the periodical was chosen for display at the Engineering Library Display Periodicals area. Ex. 1036 ¶ 2. According to DeSart, the stamp and label convey that the article was received (and, therefore, stamped) at the University of Washington Libraries, on January 31, 2000, and was redirected to the Engineering Library, where it was added to the display and made "publicly available" from February 8, 2000 until March 8, 2000. *Id*. ¶ 3.

We credit DeSart's testimony regarding the normal business practices of the Library at which he is employed since March 2000. *Id.* ¶ 1. His opinion is based on personal knowledge of these business practices and his familiarity with the Library's business records. *Id.*; Ex. 2006, 14:5−15:20. The copy of the article, with the first page we discuss above, is a copy of the periodical maintained by the Library in its ordinary course of business, and is authenticated as such. *Id.* ¶ 3.

Further indicia of publication supports our determination that Sirer is a printed publication. The copyright page of the Library copy provides for limited rights to copy and "republish" with permission and/or a fee by contacting the publications department of ACM. *Id*., 5. That page also includes an ACM ISBN number (1-58113-140-2) and instructions on how to order additional copies, information which is also included in the footer of

Exhibit 1004, indicating that copies of the periodical were available from ACM without restriction. *Id*. Therefore, based on the foregoing, we conclude that Sirer is a printed publication. Moreover, considering the dates on the face of the article, the circumstances surrounding the receipt of the periodical at the library, and the business practice of circulating periodicals at the Engineering Library, we determine that Sirer's date of publication is evident as of February 8, 2000, when the Library would have displayed the periodical, and as early as January 2000, when a subscriber to the periodical, such as the Library of Washington, would have received the periodical.[12] *See* Ex. 2006, 17:6–15 (DeSart testifying that journals published by ACM were received directly from the publisher under a subscription); 21:22–22:8.

We also find that skilled artisans exercising reasonable diligence would have been able to locate Sirer. In addition to the accessibility of the article on the library display area and on shelves, DeSart testified that the periodical containing the Sirer, "Operating Systems Review," was catalogued in the library's electronic catalog. Ex. 2006, 10:13–11:23. The periodical could be searched by the title of the periodical and its keywords. *Id*.; 30:14–31:9.

[12] We also note that the periodical appears to be a compendium of articles presented in a symposium during December 12–15, according to the information presented in the cover page. Therefore, December 1999 may not represent accurately the date the article became a printed publication, but merely the date on which the subject matter of the articles may have been presented. Accordingly, the dates corroborated by Mr. DeSart concerning receipt of the periodical at the library and circulation within the library system reasonably confirm that the printed article was *published* after the symposium dates, but no later than the date on which a periodical would have been disseminated to the libraries and its patrons.

Furthermore, Mr. DeSart testified that in 2000 there were a number of science, technology and engineering computer science databases that index content by subject areas. *Id.* at 12:2–18. From this testimony we understand that the article itself would have been indexed by subject matter, for example in a database called "Inspec," which indexes computer science materials and ACM publications, such as the one at issue here. *Id*. That is, a person of ordinary skill in the art with interests in computer operating systems and virtual machines, exercising reasonable diligence, would have been able to locate the Operating Systems Review journal and the Sirer article using a library catalogue or a database.

We note that notwithstanding the evidence of indexing discussed above, the issue of indexing the reference and in what manner is not entirely dispositive because it is not a "necessary condition for a reference to be publicly accessible." *In re Lister*, 583 F.3d 1307, 1312 (Fed. Cir. 2009). In this case, the testimony and the evidence presented support the determination that the periodical containing the Sirer article was sufficiently catalogued at the Engineering Library of the University of Washington to provide meaningful assurance that one of ordinary skill in the art, exercising reasonable diligence, would have been able to locate this particular periodical and the Sirer article itself.

Furthermore, we are persuaded that this case involves an article in a periodical that is unquestionably published and accessible not only directly from the publisher, as discussed above, but via a library. This case is distinguishable from other cases addressing concerns about dissertations, theses, or other research papers housed in a library. *See Cronyn*, 890 F.2d at 1160 (concluding three undergraduate theses housed in a library were not

publicly accessible because the references lacked a subject index); *In re Bayer*, 568 F.2d 1357 (CCPA 1978) (concluding a thesis housed, but not shelved nor catalogued, within a university library was not publicly accessible); *cf. In re Hall*, 781 F.2d 897, 899 (Fed. Cir. 1986) (concluding a dissertation shelved and indexed in a card catalog at a German university was publicly accessible). Rather, the Sirer article, published in a journal or periodical produced by ACM and distributed to subscribers is more akin to the publication addressed in *Voter Verified, Inc. v. Premier Election Solutions, Inc.*, 698 F.3d 1374, 1380 (Fed. Cir. 2012). In *Voter Verified*, a particular article available only through an on-line publication was deemed publicly accessible because the publication was well known to the community interested in the subject matter of the reference, submissions were treated as public disclosures, users could freely and easily copy the content of the on-line publication, and the on-line publication was accessible by a keyword-based search tool. As stated above, the periodical is an ACM publication, directed to computing technology topics, and was available to subscribers, including libraries. In this particular case, the Engineering Library received and circulated the volume containing the Sirer article by displaying it in a periodicals area and making it publicly available from February 8, 2000 to March 8, 2000.

Given the above-described evidence showing accessibility, we are not persuaded by Patent Owner's argument that the lack of evidence of anyone actually accessing Sirer weighs against a finding of public accessibility. PO Resp. 10. Once accessibility is proved, as the evidence shows, "there is no requirement to show that particular members of the public actually received the information." *See Constant v. Adv. Micro-Devices, Inc.*, 848 F.2d 1560,

1569; *see also SRI Int'l, Inc. v. Internet Security Sys., Inc.*, 511 F.3d 1186, 1197 ("[A]ctual retrieval of a publication is not a requirement for public accessibility. . . .").

Accordingly, based on the facts and circumstances of this case, we conclude that the Sirer article was a printed publication that was publicly accessible before the invention date of the '154 patent (i.e., December 12, 2005), and is, therefore, prior art to the challenged claims.

4. Discussion of Claims 1–5

Independent claim1 is directed to a system, while claim 4 is directed to stored program code including functions performed by a computer device, where those functions track the functions recited in claim 1. Similar limitations are analyzed together where appropriate.

a. Content Processor

Claim 1 recites a "content processor." Petitioner points out that Khazan discloses each host having one or more processors that execute the application executable. Pet 19 (citing Ex. 1003 ¶ 40), 47. We agree that Khazan discloses the recited content processor. As Khazan explains, the components that may reside and be executed at the host include application executable 102, one or more libraries, a malicious code detection system, list of target and invocation locations, list of target functions to be identified by static analysis, and a list of target functions whose invocations are to be monitored by dynamic analysis. Ex. 1003 ¶ 40. The processor of the host executes the instructions of the application executable. *Id*. Consistent with this broad disclosure of a processor, Khazan further describes that with embodiments of executable code or programs, the processor is a program

processor, which may be a virtual machine, a script processor or command processor, depending on the type of program. *Id.* ¶ 114.

With regard to claim 4, the claim is directed to program code for causing a computer device to "process content." Pet. 42. Petitioner contends that Khazan discloses hosts that each have a memory (for storing program code) and that the disclosures offered as support for "content processor" are equally applicable to claim 4. *Id.* We agree and determine that based on the disclosures of Khazan discussed above, Khazan discloses a memory storing program code for processing content.

b. Content Received Over a Network

Claims 1 and 4 recite "content received over a network." We find that Khazan teaches or suggests processing "content received over a network" based on the reasons stated below. First, by way of background, Khazan performs two types of analysis, static and dynamic. The static analysis, also referred to in Khazan as part of pre-processing, scans an application or program to identify functions that may be of interest as potentially malicious code. The static analysis produces a list of functions for dynamic analysis, which is performed at run time. In this manner, a function that from static analysis is expected to perform in a certain manner (access certain address space, for example) will be deemed malicious code if at run time, i.e., during dynamic analysis, the function deviates from the expected behavior (accesses a different address space, for example). Ex. 1003 ¶ 115. During pre-processing, or either before or after static analysis, instrumentation (or wrapping the target function) is performed to monitor the operation of that function at run time. *Id.* ¶ 75. The question of where in Khazan this

instrumented code is received and processed is of particular interest because that code must be *received* over a network.

The Petition points out that Khazan's "application executable" is the recited content. *See* Pet. 15 ("static analyzer reviews the downloaded content (called an application executable)"); 19 ("Khazan discloses 'content' such as an instrumented 'application executable')"); *see also* Ex. 1003 ¶ 73 ("At step 128, the instrumented application and associated libraries are executed."). The Petition, however, also points out that an associated library is obtained over a network. Pet. 20. In particular, Petitioner identifies Khazan's claim 35 as supporting its contention that Khazan discloses content received over a network. *Id*. Claim 35 refers to an instrumented binary form of a library. *See* Ex. 1003, p. 14 ("[W]herein said instrumented version of said binary form [of a library] obtained from at least one of: a data storage system and a host other than a host on which said application is executed, and said instrumented version is stored on a storage device."). The Petition also states that Khazan expressly teaches performing instrumentation or wrapping on a separate host and that a person of ordinary skill "would recognize that there is no functional difference between wrapping a function prior to delivery to the client computer and performing the wrapping process at the client computer." Pet. 15 (citing Ex. 1001 at 4:55–60; Ex. 1002 ¶ 71; Ex. 1003 ¶ 75, claims 31–33, 35, 68–70, 72).

In our Decision on Institution, we noted that we understood the Petition to allege that the "content" is disclosed in Khazan via its description of instrumented applications *and* libraries. Dec. 9 ("Petitioner has asserted that Khazan teaches instrumentation of both when it refers to 'instrumented application and libraries.'"); *see also* Dec. Req. for Reh'g (Paper 12) 3 ("we

do not agree with Patent Owner that we overlooked any 'agreement' or misapprehended that the evidence and argument presented regarding the 'content' limitation is limited by the Petition to Khazan's instrumented application executable.").

Patent Owner argues that Petitioner has failed to show that Khazan teaches "content received over a network" based on three contentions. First, Patent Owner contends that Khazan does not disclose an instrumented application executable or instrumented executable. PO Resp. 15–19. Second, Patent Owner contends that Khazan's application executable is not received over a network. *Id*. at 19–21. Finally, Patent Owner argues that Khazan's instrumented library is not "content received over a network." *Id*. at 21–23. We find these arguments unpersuasive in light of our analysis below.

Instrumented Applications

First, we address Patent Owner's argument that Khazan does not disclose instrumented applications. As stated above, Khazan expressly discloses instrumentation (and therefore modifying) of applications and libraries. For instance, Khazan describes that "the instrumentation technique . . . modifies the memory loaded copy of the application and associated libraries to execute additional monitoring code." Ex. 1003 ¶ 75 (cited in Pet. 15); *see also* Ex. 1003, Fig. 4B ("Execute the instrumented application and associated libraries."); ¶ 79 ("Any one of a wide variety of different techniques may be used in connection with instrumenting the application 102 and any necessary libraries."). With regard to applications, Khazan expressly claims performing static analysis and instrumenting an application by reciting, for example, "performing static analysis of an *application*,"

"determining an invocation location *within said application*," and "*instrumenting* one of: a processor of said application and *said application*." *Id*. at p. 13-14 (claims 1, 4, and 28) (emphasis added). With regard to libraries, it is undisputed that Khazan discloses analysis and instrumentation of libraries, and receiving those over a network. *Id*. ¶ 90 (referring to Fig. 8 "the steps described herein may be used in connection with instrumenting the binary form of the libraries that may be sued by the application 102, all operating system libraries or DLLs, or any other set of libraries"); PO Resp. 20–21 ("At most, however, Khazan discusses instrumented ***libraries*** being sent from one host to another.") (emphasis in original); Reply 9 ("Finjan does not dispute that Khazan's instrumented libraries can be received over a network.").

It may be the case that the embodiments illustrated in Khazan's figures specifically address instrumentation of libraries and the run time analysis of those libraries. PO Resp. 15 ("Khazan includes numerous figures and description of how to instrument libraries, but does not include any description of how to instrument an application."). Those embodiments, however, do not negate the descriptions, identified above, of applications and programs (bytecode) analyzed and instrumented using the same techniques as disclosed with respect to the libraries. Reply 10. For example, Khazan describes applying the same instrumentation techniques described with respect to dynamic link libraries or "DLLs" to "binary and machine-executable programs, as well as script programs, command program[s], and the like." Ex. 1003 ¶ 114. In particular, Khazan states that the "foregoing techniques may be used and applied in connection with detecting and analyzing calls to target functions or services made by

[malicious code] from programs in which control is transferred from one point to another." *Id*. Furthermore, we understand Khazan to provide reference to analysis tools, such as Detours and IDA Pro Disassembler, that are applicable to binary code and not limited to instrumentation of libraries. *See* Ex. 1003 ¶ 79 ("the Detours package as provided by Microsoft Research may be used in connection with instrumenting *Win32 functions* for use on Intelx86 machines.") (emphasis added); ¶ 45 ("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"); Reply 10. Accordingly, we find that Khazan discloses *instrumented* applications.

Received Over a Network

The remainder of Patent Owner's arguments are directed to whether Khazan discloses either instrumented applications or libraries "received over a network." PO Resp. 19–21. In particular, Patent Owner contends that Khazan addresses applications resident or already running in client computers when they become infected. *Id*. at 20. From this contention we understand Patent Owner to allege that Khazan would have no need for sending and receiving an instrumented application at a client because that application is being analyzed at the client computer. With regard to the instrumented libraries, although Patent Owner agrees that such libraries are sent from one host to another, those libraries are also already resident before the library can be executed. *Id*. 22.

We find that Khazan teaches or suggests that both applications and libraries are received over a network. In particular, we note that Khazan

addresses a computer system connected to a multitude of hosts via a network, as shown in Figure 1, reproduced below. Ex. 1003, Fig. 1.

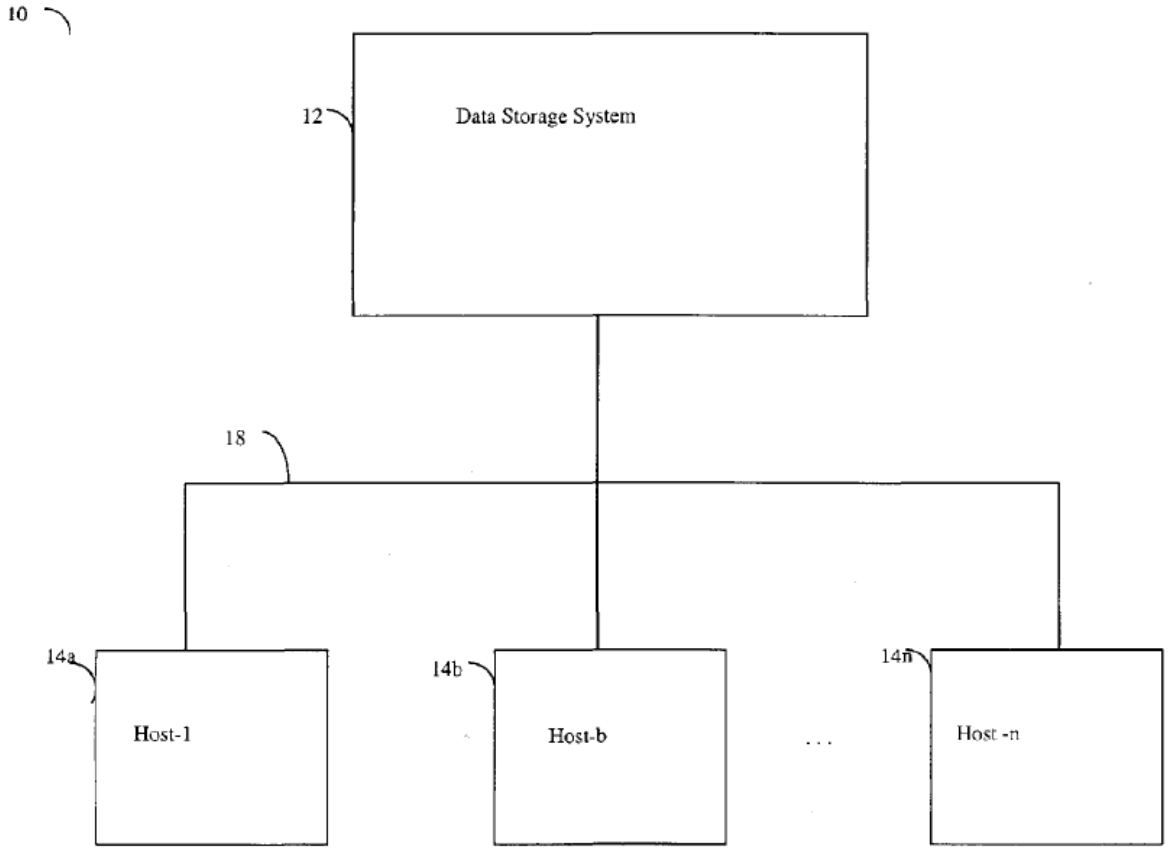


FIGURE 1

Figure 1 illustrates host system 14a (Host -1), 14b (Host-b), and 14n (Host-n) coupled to communication medium 18, which "may be the Internet, an intranet, network or other connection(s) by which host systems 14*a*–14*n* may access and communicate with the data storage system 12, and may also communicate with others included in the computer system 10." *Id.* ¶ 29. The Petition presents the contention that the broad disclosure of Khazan hosts and the various components communicating over a network warrants a

finding that Khazan teaches that its techniques may be performed on a single host or distributed among several hosts. Pet. 14 (citing Ex. 1002 ¶ 71). We agree with this contention. As explained by Dr. Rubin this conclusion "is also evident from [Khazan's] descriptions of embodiments in which the instrumentation is performed in a pre-processing step in which the resulting instrumented code is stored on, e.g., disk for use later." Ex. 1002 ¶ 71. In particular, we find persuasive that Khazan discloses that the instrumentation (or wrapping of a function) occurs on a host that is different from the host that executes the wrapped function. *Id*. (relying on Ex. 1003, claims 31–33, 35). Dr. Rubin further opines that "the end result of the wrapping . . . is the same regardless of where the system performs the wrapping." *Id*. (cited in the Petition at 15).

Patent Owner's expert Dr. Medvidovic disagrees with Dr. Rubin's opinion that instrumentation can occur in many hosts. Ex. 2002 ¶ 71. His testimony is unconvincing, however. Dr. Medvidovic does not address Dr. Rubin's assessment that Khazan teaches instrumentation on a host that is different from the host that executes the wrapped function. Instead, Dr. Medvidovic asserts that Khazan does not disclose applications received through a network. *Id*. Further, Dr. Medvidovic opines that Khazan addresses viruses that infect applications resident within a computer's file rather than in content received over a network. *Id*. We find that Dr. Medvidovic's statements do not address Petitioner's contention and ignore relevant teachings of Khazan. For instance, Khazan teaches that each host accesses information stored in data storage devices using a network (communication medium). Ex. 1003 ¶ 33. And any of Khazan's components—e.g., static analyzer, dynamic analyzer, libraries, application

executable, etc.—may be stored in the data storage system. *See id*. ¶ 72 (describing Figure 4A, which also lists the various lists 106, 111, 112). Therefore, we do not agree with Patent Owner's narrow assessment of Khazan, which would limit application of Khazan's techniques exclusively to a file resident in the host, rather than on content received over a network. Indeed, Khazan expressly discloses an embodiment in which instrumentation is performed "before invocation of the application" allowing for the instrumented library (or application) to be stored on a storage device. Ex. 1003 ¶ 75. That storage device, as discussed above, is accessed via a network. Accordingly, it is reasonable to conclude, and we find that, Khazan teaches or suggests that any host may receive over the network (communication medium 18) instrumented applications or libraries for processing at the host.

As to Dr. Medvidovic's assertion that Khazan concerns viruses at the client device and not in content received over a network, we find the assertion unsupported. Khazan broadly discloses malicious code as "a computer virus, a work, a Trojan application, and the like," and defines it as "machine instructions which, when executed, perform an unauthorized function or task that may be destructive, disruptive, or otherwise cause problems within the computer system upon which it is executed." Ex. 1003 ¶ 5. The concern for malicious code in Khazan does not exclude viruses that may be received in applications received outside of the host. Rather, we find that Khazan's disclosure of Internet, as the network that gives a host's access to data storage and other hosts, reasonably teaches that in the embodiment in which libraries, such as security DLLs, are instrumented and stored at one host during pre-processing static analysis, an instrumented library is

received over a network for dynamic analysis at *another host*. *See* ¶ 75, and claim 31 (static analysis is performed on a first host and static analysis results are made available to a second host on which said application is executed). The same disclosure is applicable to instrumented applications that are *distributed* to the executing host for dynamic analysis. *See, e.g.*, Ex. 1003, claim 33 ("the results of said static analysis are distributed together with the said application").

Finally, we address Patent Owner's argument regarding the libraries not being "directly executable," like the "application executable," and therefore not "content," as identified by Petitioner. PO Resp. 22. As stated above, we understand the Petition to assert that both instrumented applications and libraries are the recited "content." Furthermore, under our claim construction, *see supra* section II.A, "content received over a network" means data or information which has been modified and is received over a network. Instrumented applications and libraries both fall under the scope of the term, as both are data or information that has been modified. And, as stated above, we find that Khazan teaches or suggests instrumented applications and libraries received over a network.

c. The Content Including a Call to a First Function

Claims 1 and 4 recite the "content including a call to a first function." Petitioner contends that both Khazan and Sirer disclose this limitation. With regard to Khazan, Petitioner contends that the function added by instrumentation is the first function included in the content. Pet. 20 (citing, for example, Ex. 1007, Fig. 7). Petitioner further contends that Sirer discloses "instrumented content" in more detail than Khazan. *Id.* Petitioner also argues that Sirer discloses remote dynamic analysis such that

substituting Sirer's instrumentation and dynamic analysis for Khazan's would make it more clear that it would have been obvious for instrumented content (including a function call) to be instrumented remotely from a client computer. *Id.* at 20–21. In particular, Petitioner explains that Sirer's distributed architecture with a centralized network security service parses and rewrites incoming applications to insert calls to the enforcement manager in accordance with a network security policy. Pet. 21 (citing Ex. 1004, 6). Petitioner's argument, in summary, is that Sirer, much like Khazan, uses "static" analysis to parse an application and insert a call that implements a "dynamic" analysis in order to check the security of the application. *Id.* (citing Ex. 1004, 3–6).

Petitioner offers three separate rationales for the motivation to combine the teachings of Khazan and Sirer. Pet. 21–27. For instance, Petitioner argues that a person of ordinary skill in the art would recognize the advantages of instrumenting an application at a proxy server (as done in Sirer) before the client receives it in order to use "the powerful network processor rather than the weaker client processor." *Id*. at 23 (citing Ex. 1004, Abs. 5; Ex. 1002 ¶ 89). For another rationale, Petitioner asserts that Sirer's instrumentation at the centralized proxy server was a known method and an obvious substitution for instrumentation performed at the client (such as disclosed with respect to some embodiments in Khazan), yielding a predictable result. *Id.* at 25–26 (citing Ex. 1002 ¶ 96 and discussing factors supporting the predictable substitution). Finally, Petitioner asserts that there were a limited number of locations in which to perform instrumentation: the client executing the application and a remote system. *Id.* at 27 (citing Ex. 1002 ¶ 97). And even without Sirer's teachings of instrumenting at a proxy

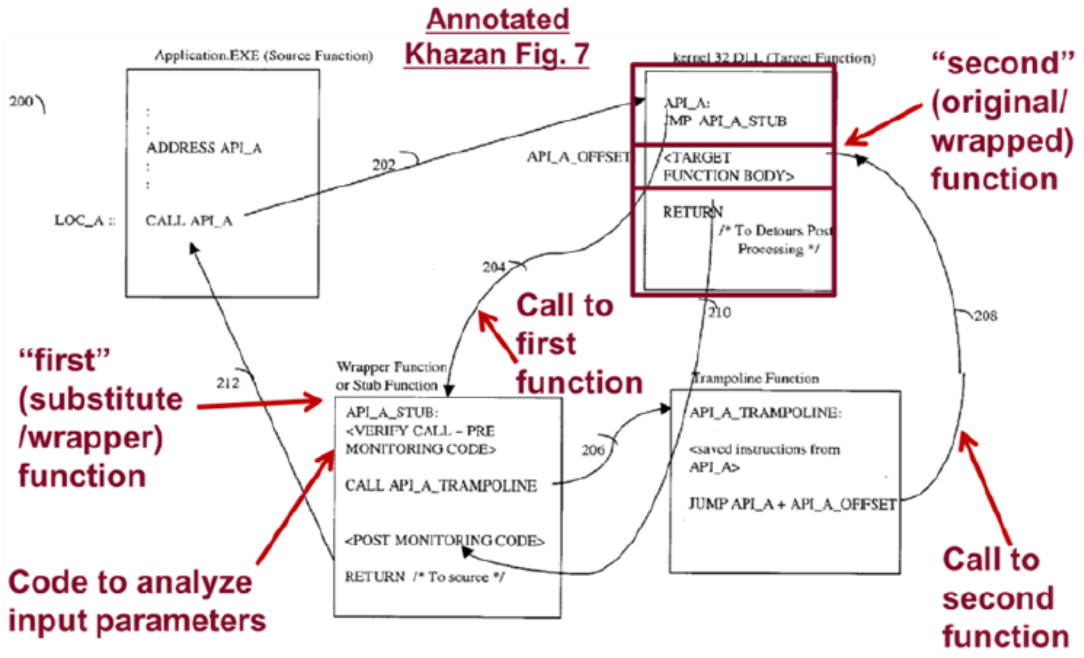
server, Petitioner argues that it would have been obvious to try instrumentation at the remote system. *Id*.

Discussion of Khazan's Teachings

Khazan, according to Patent Owner, does not disclose a "call to a first function" because Khazan implements a "jump" instruction, not a "call" to a function. PO Resp. 25–27. A "jump" is a "low-level computer instruction rather than the type of high-level 'function call' that would be found in the type of content described in the '154 patent." *Id.* at 26 (citing Ex. 2002 ¶¶ 108–109). Dr. Medvidovic, Patent Owner's expert, opines that a person of ordinary skill in the art would not understand a jump ("JMP") instruction and a function call to be same for three reasons. Ex. 2002 ¶¶ 108–111. By way of summary, these reasons focus on the different manner in which a jump instruction transfers control and data in a program in comparison with a "function call," differences which, for a jump instruction, may require additional instructions in order to handle transfer of control back to the calling function and various transfers of data. *Id.* Dr. Rubin, Petitioner's expert, also testifies to the similarities and differences between a "jump" and a "call," stating they can be the same "when you call a function that involves jumping to the location in memory where that function code is, but you can also just jump in the code without calling a function." Ex. 2005, 83:6–20 (also testifying that "in order to execute a call you have to have a Jump").

Having considered the arguments and evidence presented by both parties, we find that Khazan discloses the "content including a call to a first function." We credit the testimony of Dr. Rubin that the flow of control shown in Khazan's Figure 7 illustrates that Khazan includes a call to a first function in the instrumented content. Pet. 24 (citing Ex. 1002 ¶ 91). In

particular, we find that the annotated Figure 7, reproduced below as proffered by Petitioner, conveys that Khazan's instrumentation causes an instrumented library to be rewritten to execute a JMP instruction that transfers control to the wrapper function (the first function).



In particular, the annotated Figure 7, above, illustrates that Petitioner identifies the transfer of control 204 to a wrapper function API_A_STUB, as a "call to first function." *Id.* at 24–25. For example, Khazan explains that the call to the function API_A (call to the original function) is intercepted using the instrumentation. Ex. 1003 ¶ 82. In other words, by intercepting the original function, the program does not execute the body of that original function, but, instead, executes another function altogether, i.e., the wrapper function.

Although Figure 7 does not illustrate the instrumentation of the application itself, we do not agree that the example in the embodiment is inapplicable to instrumenting applications. As stated above, we find that

Khazan expressly discloses that the instrumentation techniques are applicable to both applications and libraries. *See, e.g.*, *id.* ¶ 114.

Furthermore, the Figure 7 embodiment's use of a JMP instruction, rather than a CALL instruction, does not persuade us that Khazan's teaching with respect to transfer of control is limited to a JMP instruction. Although Figure 7 implements a JMP instruction together with a Trampoline function to transfer control to and from a wrapper function (first function), we find that Khazan recognized that the transfer of control technique would be effected with either a JMP or CALL instruction. For example, we find instructive Khazan's explanation that monitoring for call instructions includes also jump instructions, or "other types of instructions transferring control from the application as may be the case for various routines being monitored." Ex. 1003 ¶ 46. We recognize that this statement in Khazan addresses the instructions monitored before instrumentation occurs. Nevertheless, the discussion regarding how a "jump" and a "call" are both instructions that transfer control from one function to another supports the finding that Khazan suggests its teachings are not limited to the use of a jump instruction when discussing transfer of control in executing code. *See* Ex. 1003 ¶ 90 ("the first instructions or instructions just saved from the current target are replaced by instructions which transfer control to the stub or wrapper for the current call").

The discussion of transfer of control is important, as we further find that Khazan teaches that the instrumented content requests the service of the first function, i.e., includes a call to a first function. In particular, as noted above, the transfer of control results in execution of the stub or wrapper function. *See id.* ¶ 83 ("The first instruction of the target function API_A

includes a transfer or jump instruction to the wrapper or stub function as described elsewhere herein. This transfer is indicated by arrow 204."). That transfer of control, in essence, involves the execution of an instruction requesting that the wrapper function verify the intercepted call. *See id.* ¶ 84 ("Within the pre-monitoring portion of the wrapper function, the intercepted call is verified. As used herein, the pre-monitoring code portion refers to that portion of code included in the wrapper or stub function executed prior to the execution of the body of the intercepted routine or function."). We also note that Khazan broadly teaches using any instruction that transfers control to the wrapper function. *Id.* ¶ 88 (describing instrumentation as dynamically modifying libraries "in which the instruction or instructions of the API of the target function are replaced with a jump instruction *or other transfer instruction* transferring control to the wrapper function.") (emphasis added). Accordingly, we find that Khazan's transfer of control to the stub or wrapper function to execute that function, as illustrated in Figure 7 by the arrow 204, teaches or suggests "a call to a first function" as we have construed the term.

Discussion of Sirer's Teachings

Patent Owner challenges Petitioner's assertion that Sirer also teaches content including "a call to a first function." PO Resp. 32–33. In particular, Patent Owner argues that Sirer's dynamic service component is not a "function," but rather, it is a component that "provide[s] service functionality during the execution of applications." *Id.* at 33 (citing Ex. 1004, 3). We find Patent Owner's argument unpersuasive.

As discussed above, a "call to a first function" is a statement or instruction in the content, the execution of which causes the function to provide a service. We find that Sirer describes its dynamic service

components in alignment with the definition of the term. For example, Sirer describes the dynamic service components as providing "service functionality" during execution of applications. Ex. 1004, 3 (also stating that "[d]ynamic service components provide service functionality to clients during run-time as necessary"). These dynamic service components are code that is delivered to the client from the central proxy server on demand. *Id.* at 4 ("[t]he code for the dynamic service components resides on the central proxy and is distributed to clients on demand."). Sirer performs a dynamic service by inserting a call to the corresponding dynamic service component. *Id.* at 3; *see also id.* at 5 ("[T]he verification service modifies the code to perform the corresponding checks at runtime by invoking a simple service component (Figure 3)."). The call insertion is performed by Sirer's static service components at a proxy server. *Id.* at 4 ("[t]he proxy transparently intercepts code requests from clients, parses JVM bytecodes and generates the instrumented program in the appropriate binary format"). In particular, Sirer teaches rewriting application code during static service when "encounter[ing] data-dependent operations that cannot be performed statically." *Id.* at 3. One example of data-dependent operations checked dynamically is verifying program safety. *Id.* Another example is a security check for checking user-supplied arguments to system calls. *Id.*

Based on the foregoing, we find that Sirer teaches that content in need of a security check is instrumented at a proxy server where a call to dynamic service components is inserted. We find that this call to dynamic service components is a call to a first function because Sirer teaches that the call requests a particular service provided by the code comprising the dynamic service components. With respect to the example of performing a security

check, for example, we understand Sirer to teach that the call to the dynamic service component will be inserted into the application to check whether the user-supplied arguments are secure. *See, e.g.*, *id.* at 5 (disclosing that, as shown in Figure 3, the verification service modifies the code to perform the corresponding checks at runtime by invoking a simple service component). One such particular example is provided in Figure 3 of Sirer, reproduced below.

```
class Hello {
    static boolean __mainChecked = false; // Inserted by the verifier
    public static void main() {
        if(__mainChecked == false) {  // Begin automatically generated code
            RTVerifier.CheckField("java.lang.System","out",
                                  "java.io.OutputStream");
            RTVerifier.CheckMethod("java.io.OutputStream","println",
                                   "(Ljava/lang/String)V");
            __mainChecked = true;
        }                                 // End automatically generated code
        System.out.println("hello world");
    }
}
```

Figure 3 provides the "hello world example" after it has been processed by the distributed verification service. The security checks deferred to execution time are shown in italics. *Id.* at 5. This example supports Petitioner's contention and our finding that Sirer's content includes a call to a first function. In particular, the italicized code, which is the instrumented portion of the program, shows that the program invokes a verifier function *RTVerifier.CheckMethod*, for example, that requests verification that class OutputStream implements a method "println" to print a string. *Id.*

Accordingly, we agree with Petitioner's contention that Sirer teaches a call to a first function. Patent Owner's arguments to the contrary are not persuasive.

Combination of Khazan and Sirer

In connection with the limitation "the content including a call to a first function," Petitioner asserts that it would have been obvious to combine Sirer's teachings of a proxy server's instrumentation of applications (for including calls to the dynamic service components) with Khazan's teachings. Pet. 21–25. We have already summarized Petitioner's various contentions in this regard. These contentions appear applicable insofar as Khazan discloses instrumenting the application on a "host." Pet. 22. We determined above, however, that Khazan teaches "content received over a network" and the "content including a call to a first function." It is, therefore, unnecessary to determine if a person of ordinary skill in the art would have been motivated to combine the teachings of Sirer's instrumentation at a proxy server with the teachings of Khazan resulting in the "content including a call to a first function."

d. The Call Including an Input

Claims 1 and 4 require that the call to a first function include an input. Petitioner offers four contentions as to how the prior art teaches the limitation. First, Petitioner argues that Khazan's "parameters" included in the wrapper function satisfy the limitation. Pet. 27–28. Second, Petitioner relies on Khazan's description of the Microsoft Detours package, which "requires the original function parameter to be passed to the wrapper function." *Id.* at 28 (citing Ex. 1002 ¶¶ 101–02; Ex. 1012 at 5). Third, alluding to instrumentation occurring at a proxy server, such as in Sirer, Petitioner asserts that a person of ordinary skill in the art would have passed the parameters for checking and verification to the substitute (wrapper) function. *Id.* (citing Ex. 1002 ¶¶ 101, 81–82). Finally, Petitioner argues that

it would have been obvious to a person of ordinary skill in the art for the wrapper function to include the parameters from the wrapped function because otherwise, the wrapper function could not verify the parameter information. *Id.* at 28–29 (citing Ex. 1002 ¶ 101).

In addition to the disclosure of Detours, the relevant Khazan disclosures Petitioner points to describe that the pre-monitoring code, which is part of the stub or wrapper function, performs verification of parameter information, "including type and value of some parameters." Ex. 1003 ¶ 87. As an example, Khazan states that the parameters associated with the target call would have been also the subject of static analysis. *Id.* Dr. Rubin proffers that a function "input" is often called a function "parameter." Ex. 1002 ¶ 100. Therefore, it appears reasonable to conclude that Khazan, when referring to the parameter verification in the wrapper function, refers to verifying "inputs" to the function.

Patent Owner argues that because Khazan discloses a jump instruction, and jumps do not include an input, Khazan does not disclose a "call including an input." PO Resp. 28–29. Further, Patent Owner argues that Detours also uses jumps rather than function calls. *Id.* at 29–31. As we discussed above, we are not persuaded that the teachings of Khazan are limited to the use of only jump instructions. But, rather, Khazan discloses broadly the use of any instructions that transfer control to a wrapper function. Indeed, we credit Dr. Rubin's explanation that parameters (or inputs) would be passed from the wrapped function to the wrapper function in order to verify the parameter information, as taught by Khazan. Ex. 1002 ¶¶ 100–02. Dr. Rubin also explains that the Detours package passes "the identical parameters from the calling code to the detoured function and then

into the original 'target' function." *Id.* ¶ 101. From this testimony, we understand Khazan's transfer of control to the wrapper function (call to a first function) to include the parameters (input) that will be verified during pre-monitoring. This understanding extends not only to the operation of Detours (which checks API calls), but also for the verification of parameters in instrumented scripted programs. *See id.*

As to Patent Owner's further arguments that Khazan verifies parameters without using a call including an input, we are not persuaded. *See* PO Resp. 31 (Patent Owner arguing that "it may be appreciated that Khazan is able to '[verify] the parameter information' despite not utilizing a call to the first function or a call including an input."). Patent Owner's argument focuses narrowly on the specific embodiments of Khazan. As stated above, Khazan broadly teaches using any instruction that transfers control to the wrapper function. Ex. 1003 ¶ 88 (describing instrumentation as dynamically modifying libraries "in which the instruction or instructions of the API of the target function are replaced with a jump instruction *or other transfer instruction* transferring control to the wrapper function.") (emphasis added).

We do not see such a broad disclosure as limiting Khazan's technique to jump instructions or to using the Detours package. To the contrary, as we have determined above, Khazan's disclosure as a whole teaches or suggests that calls would be used, just as jump instructions, to transfer control. From Khazan's verification of parameters, description of transfers of control, and Dr. Rubin's testimony on this issue, we find that when using a call to effectuate the transfer of control, Khazan teaches or suggests that the call includes inputs in order to pass parameters to the wrapper function.

e. Invoking a Second Function With the Input

Claim 1 recites that the content processor invokes "a second function with the input, only if a security computer indicates that such invocation is safe." Claim 4 similarly recites "invoking a second function with the input only if the indicator indicates that such invocation is safe."

Khazan's Disclosures

Petitioner argues that Khazan teaches that the "second function," i.e., the original or target function, is invoked after verification. Pet. 29–30. In particular, Petitioner proffers an annotated Figure 9 from Khazan, reproduced below, showing the recited invocation. *Id.*

Petition Figure 2

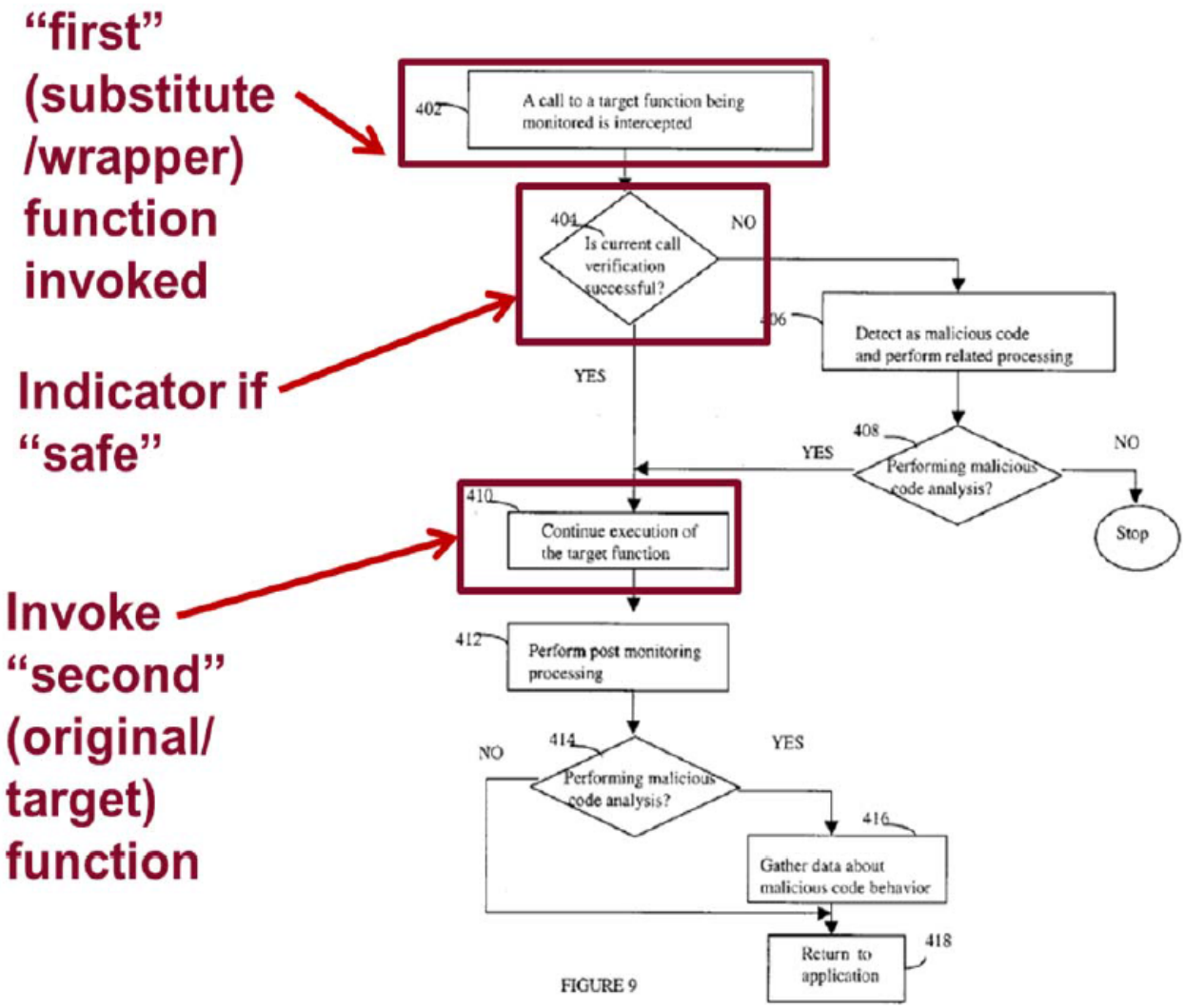


Figure 9 of Khazan is a flowchart of method steps summarizing the run-time processing performed by the dynamic analyzer. Ex. 1003 ¶ 27. According to the annotated figure, Petitioner asserts that Khazan invokes two functions: (1) step 402 is evidence of the invocation of the "first function";[13] and (2) step 410 is evidence of the invocation of the "second

[13] We note that the claims require a call to a first function, but are silent regarding "invocation" of the first function. The distinction, however, is not

function." *Id.* Step 402 of Figure 9, however, does not refer to invocation of a function, but instead refers to intercepting a call to a target function being monitored. *Id.* at Fig. 9 ("a call to a target function being monitored is intercepted"). As discussed above, Khazan intercepts the call to the target function by executing the jump instruction that transfers control to the stub or wrapper function, i.e. the first function. Thus, we understand that the act of intercepting the call is what Petitioner points to as invoking the "first" or wrapper function. The problem here is that, as we explain further below, for Khazan to transfer control or jump to the wrapper function, Khazan must call first the target function, which Petitioner maps to the "second function." Petitioner's pointing to the "second function" invoked at step 410 does not solve the problem, because the claims require invocation of the second function *only if* a security computer or the indicator indicates that the invocation is safe.

Patent Owner's arguments correctly point out this problem in Petitioner's contentions. Patent Owner argues that the description of Khazan's dynamic analyzer shown in Figure 9 does not disclose the limitation. PO Resp. 35–36. Specifically, Khazan, according to Patent Owner, *always* invokes the second function. *Id.* (arguing that the CALL API_A in Application.EXE is always invoked). We are persuaded by this argument. Khazan explains that "[b]eginning with the source function of the application's binary, a call is made to the target function API_A from the invocation address LOC_A." Ex. 1003 ¶ 83. Khazan further explains that

---

relevant to our discussion of Khazan's invocation of the target function, i.e., second function.

the "first instruction of the target function API_A includes a transfer or jump instruction to the wrapper or stub function." *Id.* That is, in order for the stub or wrapper function to be executed, the target function must be invoked first. Indeed, Khazan's instrumentation rewrites the target function to include therein the transfer of control to the stub or wrapper function, indicating, therefore, that the target function (recited "second function") must be invoked. The claims require, however, that the second function be invoked *only if* it is safe.

Petitioner, in reply, explains that the invocation of the target function (API_A) in the trampoline routine is the invocation of the second function. Reply 15–16. We find this explanation insufficient to rebut Patent Owner's argument and contrary to the facts of Khazan. First, Khazan describes the execution of the second function after the verification check as "continuing" execution. Ex. 1003, Fig. 9 (step 410: "*Continue* execution of the target function") (emphasis added); ¶ 94 ("control proceeds to step 410 to *continue* execution of the target routine") (emphasis added). Second, as described above, the target function must be invoked in order for control to transfer to the wrapper function. We find that this would be the case even if dealing with an instrumented application invoking internal target functions. When Khazan describes intercepting the target function, it refers to *invoking* the target function first, in order for the code inserted in the instrumented content to transfer control to the wrapper function. *See* Ex. 1003 ¶82 ("Referring now to FIG. 7, shown is the logical flow of control in one embodiment when an external *target function*, such as a Win32 API function, *is invoked at run time* from the application using a call instruction. The external call is intercepted using the instrumentation techniques

described herein.") (emphasis added); ¶ 90 ("the first instruction or instructions just saved from the current target are replaced by instructions which transfer control to the stub or wrapper for the current call"); ¶ 92 ("the code of the target function is modified in memory rather than on a storage device"); ¶ 93 ("Every invocation of a Win32 API may be intercepted in the foregoing instrumentation technique. When one of the Win32 API calls is intercepted, this particular instance or *invocation* is checked against the list.") (emphasis added); ¶ 94 ("a call to a target routine being monitored is intercepted"). Third, although we agree with Petitioner that the target function is verified during pre-monitoring and execution is suspended, the verification only occurs after invocation of the target function. Petitioner has failed to point out any teaching in Khazan where the target function is not invoked first. Accordingly, we find that Khazan does not teach or suggest the limitation "invoking a second function with the input only if" a security computer or the indicator indicates that such invocation is safe.

Sirer's Disclosures

Patent Owner argues that the Petition maps to Khazan only the limitation of "invoking a second function with the input." PO Resp. 39–40; Tr. 71:24–72:23. We agree that the Petition addresses only Khazan in connection with the limitation "invoking a second function with the input." Pet. 29–30. We note, however, that Petitioner relies on Sirer for the portion of the limitation requiring invocation of the second function "only if a security computer indicates that such invocation is safe." *Id.* at 30–34. According to Petitioner, Khazan "discloses ***locally*** invoking the intercepted function only if the pre-monitoring code verifies the function and its parameters (e.g., input) for safety." *Id.* at 30 (emphasis in original). The

Petition then addresses Sirer's teachings on "remote verification" given Khazan's failure to disclose a remote computer for performing the verification. *Id.* Petitioner, therefore, does not rely on any of Sirer's teachings to disclose that invocation of the second function occurs *only if* it is safe to do so. Accordingly, there is no need to address Sirer's disclosures or the asserted combination of Sirer with Khazan, because, as stated above, we find that Khazan does not disclose, teach or suggest that the second function is invoked *only if* it is safe to do so.

f. Transmitting the Input . . . , When the First Function is Invoked

Petitioner argues that Sirer teaches transmitting the function input by disclosing that the "security service may check user-supplied arguments to system calls." Pet. 35. Sirer, according to Patent Owner, does not disclose any timing for the transmission of the user-supplied arguments. PO Resp. 41. Patent Owner argues that Khazan also is silent regarding when the alleged input to the first function is transmitted. *Id.*

Patent Owner's arguments are not persuasive. Instead, we agree with Petitioner's contention that Sirer teaches verification when the function is executed. Reply 16–17 (citing Ex. 1004 at 3–5, Figs. 1–4; Ex. 1002 ¶¶ 107–109). For example, Sirer describes that in order to perform a runtime verification, the "verification service modifies the code to perform the corresponding checks at runtime by invoking a simple service component." Ex. 1004 at 5. Sirer also describes that a security service, which is a dynamic service component, checks user-supplied arguments to system calls. *Id.* at 3. Dr. Rubin opines that the "system call" is the intercepted call and the wrapper function (we read here the modified code) contains the

access checks that query the security service. Ex. 1002 ¶107–109. From this discussion, we find that in order to perform the security service checks, the modified code or wrapper function (as identified by Dr. Rubin) would be invoked in order to execute the call to the applicable dynamic service component. Accordingly, any transmission of inputs in Sirer would occur "when the first function is invoked." We also find persuasive Petitioner's argument and evidence that Khazan's verifications take place when the pre-monitoring code is executed, which timing also meets the claim language of transmitting an input, when the first function is invoked. Pet. 36 (citing Ex. 1003 ¶ 84).

g. Receiving an Indicator . . . Whether it is Safe to Invoke the Second Function With the Input

Petitioner points to Sirer as receiving information from querying the security service during execution of the application. Pet. 37–38 (citing Ex. 1002 ¶¶ 110–11). Petitioner also points out that Sirer checks the user-supplied arguments to system calls, ensuring that the arguments do not violate the security policy. *Id.* at 38 (citing Ex. 1004 at 4–5). Petitioner asserts that it would have been obvious to combine Sirer and Khazan to gain the benefits of performing a run-time analysis on a network server (as in Sirer) to receive the information about that analysis. *Id.* at 38–39 (citing Ex. 1002 ¶ 111). Patent Owner challenges Petitioner's assertions in this regard. PO Resp. 42–43. We are not persuaded by Petitioner's argument.

According to the mapping provided by Petitioner, Sirer's client computer, which executes the application with the modified code, calls the security server to verify the security identifiers and permissions it maintains. Pet. 37–38 (citing Ex. 1004 at 6, Fig. 4). The verification Sirer performs

results in a query of the security service which is a lookup performed by the security service. Reply 17; Ex. 1004 at 6. The client caches the results of the lookup. *Id.* That is, Sirer teaches receiving the lookup results and providing access (e.g., allowing or disallowing access to a requested file). We find, therefore, that Sirer's client receives an indicator from a security computer whether it is safe to invoke the second function (the operation that is being checked) with the input (e.g., user-supplied arguments).

Nevertheless, Petitioner relies on the combination of Khazan and Sirer as teaching this limitation. The Petition explains that a person of ordinary skill in the art would have been motivated to obtain the benefits of analyzing the input at a remote computer, as taught by Sirer. Pet. 38. The premise is based on Khazan's teaching that the pre-monitoring code performs the verification of its parameters locally (not at a security computer, as required by the claims). *See* Pet. 30.

As discussed above, however, we are not persuaded that Khazan teaches the limitation of invoking the second function only if the invocation is safe. Khazan *continues* the operation of the second function, depending on the verification check performed by the pre-monitoring code. Ex. 1003, Fig. 9 (step 410: "*Continue* execution of the target function") (emphasis added); ¶ 94 ("control proceeds to step 410 to *continue* execution of the target routine") (emphasis added). It follows, therefore, that any combination of teachings of Khazan with Sirer would result in the second function being invoked, as taught by Khazan, upon execution of the instrumented content, but not "only if" the invocation is safe, after receiving the indicator. Accordingly, we are not persuaded that Petitioner has shown that the combination of Khazan and Sirer teaches or suggests this limitation.

h. Motivation to Combine Teachings of Khazan and Sirer

Patent Owner challenges the proffered rationale for the asserted combinations of Khazan and Sirer. PO Resp. 47–50. In particular, Patent Owner argues that the combination alters the principles of operation of Khazan. *Id.* And further, Patent Owner asserts that the combination of Khazan and Sirer would be inoperable. *Id.* at 50. In light of our determination that Khazan fails to disclose, teach, or suggest invoking a second function, as recited, we need not address Patent Owner's additional arguments regarding the rationale for the asserted combination of teachings.

i. Conclusion Regarding Claims 1–5

Independent claims 1 and 4 recite the "invoking a second function" limitations addressed above. Having found that Khazan does not disclose, teach, or suggest the limitation, we determine that Petitioner has failed to establish by a preponderance of the evidence that claims 1 and 4, and claims 2, 3, and 5, dependent therefrom, are unpatentable over the combination of Khazan and Sirer. In light of our determination, we, therefore, do not address additional arguments and evidence proffered by Patent Owner regarding claims 2 and 3, and secondary considerations of nonobviousness.

E. GROUND BASED ON KHAZAN, SIRER, AND BEN-NATAN

This ground addresses claims 6–8, 10, and 11. Claims 6 and 10 are independent claims. Petitioner contends that the "modified input variable" recited in claims 6 and 10 is taught by Ben-Natan. *See, e.g.*, Pet. 48 ("Ben-Natan discloses 'a modified input variable' in the form of a 'result data access statement.'"). For the remaining limitations of these claims,

Petitioner relies on Khazan and Sirer. Pet. 46–54. For example, claim 6 recites that a content processor calls "a second function with a modified input variable," which Petitioner maps to Khazan's execution, post-verification, of the target function combined with the teachings of Ben-Natan's modification of a data access statement, in an SQL query. *Id.* at 48. Patent Owner challenges the combination with Ben-Natan on the basis that Ben-Natan is not analogous art and does not disclose the limitation. PO Resp. 51–53, 56–58. Patent Owner argues also that there is no motivation to combine Ben-Natan with Khazan and Sirer and that the combination would be inoperable. *Id.* at 54–56, 58.

1. Overview of Ben-Natan (Ex. 1005)

Ben-Natan is titled "System and methods for nonintrusive database security." Ben-Natan describes "configurations of the invention [that] provide a nonintrusive data level security mechanism for intercepting database access streams." Ex. 1005, 6:32–34. "Such an implementation deploys a security filter between the application and database, and observes, or 'sniffs' the stream of transactions between the application and the database." *Id.* at 6:38–41. "If the 'sniffed' transactions indicate restricted data items, the security filter modifies the transaction to eliminate only the restricted data items, and otherwise allows the transaction to pass with the benign data items." *Id.* at 6:50–54.

2. Discussion

Petitioner asserts that Khazan discloses identifying potentially malicious function parameters. Pet. 50, 53. According to Petitioner, Khazan performs two actions when identifying the existence of malicious code: (1) stop execution and return an error code; and (2) continue to run the

application to monitor the behavior of the malicious code. *Id.* at 50. Thus, Petitioner contends, Ben-Natan's limiter operation, which modifies the input of an SQL query, would allow for a program in Khazan to execute without harming the client computer, instead of stopping. *Id.* (citing Ex. 1002 ¶¶ 127–28). Petitioner further argues that given the limited number of known techniques for handling potentially malicious function inputs, it would have been obvious to try modifying Khazan's input as taught by Ben-Natan, to allow safe execution. *Id.* at 51. Finally, Petitioner asserts that the addition of Ben-Natan to the teachings of Khazan and Sirer is "a natural progression," resulting in a "system in which the security service of Sirer not only checks the function inputs, but modifies them if they are potentially malicious, to allow the downloaded application to execute safely (i.e., without violating the security policy)." *Id.* at 53.

Patent Owner challenges Petitioner's proffered rationale, arguing that a person of ordinary skill in the art would not be motivated to modify Khazan to make the inputs or parameters safe because Khazan would not perform the disclosed behavior analysis of detected malicious code. PO Resp. 54–55. We agree with Patent Owner's argument, and find that the alleged combination of teachings would so alter Khazan's operation that a person of ordinary skill in the art would not be motivated to combine the teachings as Petitioner alleges.

First, in order to combine the teachings of Khazan, Sirer, and Ben-Natan to achieve the claimed requirements of a modified input variable, a number of modifications appear necessary, and not all are identified or explained by Petitioner. Khazan's pre-monitoring code would need to be rewritten to transmit the input variable of the target function to a network

server or proxy that performs analysis of the input variable (as Petitioner alleges in Sirer). Additionally, Sirer's dynamic analysis components would need modification to include the limiting technique taught in Ben-Natan in order to modify the input variable. Further, and unexplained by Petitioner, Sirer would need to modify its server communication stream with the client devices to transmit the modified input variable, instead of sending the results of the lookups. Further still, and also unexplained by Petitioner, Khazan would need to be modified to receive the modified input variable, and replace the parameters of the target function with the modified ones.

We find that Petitioner has not explained sufficiently how the reference's teachings would be combined in order to achieve the claimed limitations. For instance, Petitioner's assertion that the combination is predictable because the references continue to do what they did prior to the combination (Ex. 1002 ¶ 125) is conclusory and unreasonable in light of the various and necessary, yet unexplained modifications of Khazan's teachings for combinability with those of Sirer and Ben-Natan.

Particularly noteworthy, Petitioner relies, for this ground, on the combinations of Sirer and Khazan made with respect to the previous ground. *See* Pet. 49 ("As discussed above, it would have been obvious to the POSA to combine the teachings of Sirer with Khazan. (§§X.A.1.d.1, X.A.1.g.)"). But the previous ground addresses claims (1–5) that do not recite any modifications to the input or input variables. The rationale for the combination of Khazan and Sirer for those claims, therefore, does not address any rationale for obviousness concerning either Sirer or Khazan handling modified input variables. Indeed, at most, Sirer is alleged in the previous ground to produce an indicator indicating whether it is safe to

invoke the second function with the input. *See* Pet. 33 ("The POSA would be familiar with developing the software for performing the security analysis on a remote computer and would expect the predictable result of returning a security indicator from the remote computer regarding whether the input is safe to execute in the original function.").

The instant ground, however, addresses claims that recite receiving the "modified input variable," for which Sirer is relied on as teaching the centralized or remote verification. *See id.*at 52 (discussing the "receiver" limitation of claim 6); 53 (discussing the limitation regarding how the modified input variable is obtained and relying on Sirer as disclosing "the security computer in the form of a security service."); 58 (discussing the "receive" limitation of claim 10, which does not require a security computer, but nevertheless relying on Sirer providing a security service). As stated above, to meet the claims it would be necessary for Sirer's security service to send a modified input variable, not just an indicator that invocation with the input is safe. Further, it would be necessary for Khazan to substitute the modified input variable into the target function during runtime. Neither of these particulars are addressed in the reasoning provided for combining Khazan and Sirer in the ground concerning claims 1–5. The reasoning provided, as discussed above, focuses generally on Sirer providing a centralized or remote security service processing. No changes in either Khazan's or Sirer's operation or features were alleged with regard to the modified input variable, and no motivation has been asserted sufficiently to combine the teachings in a manner that achieves claims 6–8, 10, and 11. Therefore, we find that Petitioner's reliance on the rationales asserted for the ground concerning claims 1–5 are insufficient articulated reasoning with a

rational underpinning for the asserted combinations regarding claims 6–8, 10, and 11.

Furthermore, we find insufficient the reasoning Petitioner provides to combine Ben-Natan's teachings with those of Sirer and Khan. Petitioner's expert, Dr. Rubin, opines that, in addition to the alleged similarities of the prior art systems, "Ben-Natan's proposal to actually modify the inputs is a small and natural extension of the same operating principles that Khazan and Sirer use." Ex. 1002 ¶ 122. With regard to utilizing Sirer's security processing at a server, Dr. Rubin similarly opines that it is a "natural extension." *Id.* ¶ 124. Finally, Dr. Rubin asserts that Ben-Natan's contribution is "also a straight-forward and unsurprising addition." *Id.*

We find these explanations insufficient to show an articulated reason with a rational underpinning for why a person of ordinary skill in the art would be motivated to combine the references as asserted by Petitioner. Such statements of "straight-forward," "small," "natural," and "unsurprising" applications are generic, and fail to provide necessary factual support—they are akin to stating in a conclusory fashion that the combination "would have been obvious." *In re Van Os.*, 844 F.3d 1359, 1361 (Fed. Cir. Jan. 3, 2017) ("Absent some articulated rationale, a finding that a combination of prior art would have been 'common sense' or 'intuitive' is no different than merely stating the combination 'would have been obvious.' Such a conclusory assertion with no explanation is inadequate to support a finding that there would have been a motivation to combine.").

As for stating that it would have been "obvious to try," the rationale also lacks factual support. It is not enough to assert that the prior art

provides two options and that it would have been predictable to implement either. An obviousness rationale generally requires some identification of "a design need or market pressure to solve a problem" before looking at the "finite number of identified, predictable solutions." *See KSR*, 550 U.S. at 421. Accordingly, an obvious to try rationale requires that the design need or market pressure is what drives a person of ordinary skill in the art to consider the identified, predictable solutions. We find neither an assertion nor evidence proffered by the Petitioner concerning this need. The Petition states, with regard to the "obvious to try" rationale, that a person of ordinary skill in the art would expect "simply that the input would be modified to execute safely." Pet. 51. This alleged result identifies a solution, but does not address either a design need or market pressure.

Moreover, the result of Petitioner's asserted combinations would result in an alteration to Khazan that renders the disclosure inoperable for the analysis mode. *See* PO Resp. 54. In particular, we find persuasive Patent Owner's argument and evidence that if a parameter of a target function is modified to be "safe," Khazan would not operate in the analysis mode where the behavior of the malicious code is analyzed. *See* Ex. 1003 ¶ 99. In other words, after detecting malicious code, the technique of Khazan to conduct behavior analysis would not be possible, given that the malicious code, in the asserted combination, is excised by modifying the input variable. We also find persuasive Patent Owner's argument that Petitioner has not supported its assertion that Ben-Natan "discloses a known method for modifying [a] function input to allow for safe execution of the downloaded application" because Ben-Natan is not concerned with downloaded applications or safe execution of those applications. PO Resp.

55 (addressing Petitioner's Rationale "B" making the disputed assertion). Ben-Natan's alleged known method is limiting a database query to narrow the scope of the database search, but does not discuss any downloaded applications or implementation of the limiting query to an execution of applications. Ex. 1005, 13:27–14:24. Petitioner fails to explain how the Ben-Natan disclosure constitutes a "known method for modifying an input to allow for safe execution of the downloaded application," as asserted in the Petition.

Finally, we find unavailing Petitioner's assertion that the "combination [of the references] is nothing more than combining known techniques in a different way to produce predictable results." Pet. 54 (citing Ex. 1002 ¶¶ 122–129). The statement alone is not sufficient for Petitioner to carry its burden. The Federal Circuit has made clear that a petitioner in an *inter partes* review proceeding cannot "satisfy its burden of proving obviousness" by "employ[ing] mere conclusory statements" and "must instead articulate specific reasoning, based on evidence of record" to support an obviousness determination. *In re Magnum Oil Tools Int'l*, 829 F.3d 1364, 1380–81 (Fed. Cir. 2016). The "factual inquiry" into the reasons for "combin[ing] references must be thorough and searching, and the need for specificity pervades . . . ." *In re Nuvasive, Inc.*, 842 F.3d 1376, 1381–82 (Fed. Cir. 2016) (internal quotations and citations omitted). A determination of obviousness cannot be reached where the record lacks "explanation as to *how* or *why* the references would be combined to produce the claimed invention." *Trivascular, Inc. v. Samuels*, 812 F.3d 1056, 1066 (Fed. Cir. 2016); *see Nuvasive*, 842 F.3d at 1382–85; *Magnum Oil*, 829 F.3d at 1380–81. The Petition's statement that combining known techniques yields

predictable results relies exclusively on the paragraphs of Dr. Rubin's declaration discussed above, which we find conclusory and therefore unpersuasive. Furthermore, to the extent the statement is an attempt to invoke a rationale for finding obviousness asserted in *KSR*, that attempt fails, for *KSR* requires the known elements to be combined "according to known methods"—not "in a different way," as alleged by Petitioner. *See KSR*, 550 at 416.

3. Conclusion Regarding Claims 6–8, 10, and 11

Having considered the arguments and evidence presented by both parties, we determine that Petitioner has not shown by a preponderance of the evidence that claims 6–8, 10, and 11 would have been obvious over the combination of Khazan, Sirer, and Ben-Natan. As stated above, the proffered rationales to combine the references lack factual support or rational underpinning supporting the reasoning. Given our findings above, which address the assertions made with regard to independent claims 6 and 10, we find that the challenged claims dependent therefrom also have not been shown to be unpatentable.

F. MOTIONS TO EXCLUDE

Petitioner moves to exclude Exhibits 2009 and 2011–2013 based on various objections as to relevance and hearsay. Paper 46 ("Pet. Motion to Exclude"). Petitioner's Motion to Exclude is denied as moot, because the evidence objected to is not relied upon in reaching our determination that Petitioner has not met its burden of showing that claims 1–8, 10, and 11 are unpatentable.

In turn, Patent Owner moves to exclude various exhibits in the record:

a) Exhibits 1036, 1039–1042, 1044–1045 as outside the scope of Petitioner's Reply. Paper 48 ("PO Motion to Exclude").
b) Exhibit 1008, the Sirer Declaration, as hearsay and for lack of foundation. *Id.* at 5–8.
c) Exhibit 1036, Declaration of Mr. Mel DeSart, for lack of foundation and because opinions are conclusory and unreliable. *Id.* at 8–9.
d) Exhibits 1004 and 1024, Sirer reference, as hearsay, irrelevant, and lack of authentication. *Id.*at 10–14.
e) Exhibit 1012 and Annotated Figure 1–4 in the Petition, as prejudicial. *Id.* at 14–15.

Patent Owner's motion is denied. From the outset, we have stated repeatedly that a motion to exclude is not a vehicle for arguing that Petitioner's arguments and supporting evidence are outside the proper scope of a reply.[14] A motion to exclude evidence filed for the purpose of striking or excluding an opponent's brief and/or evidence that a party believes goes beyond what is permitted under 37 CFR § 42.23 is improper. An allegation that evidence does not comply with 37 CFR § 42.23 is not a sufficient reason under the Federal Rules of Evidence for making an objection and requesting exclusion of such evidence. Accordingly, these arguments in Patent

[14] *See Valeo v. Magna Elecs., Inc.*, Case IPR2014-00227, Paper 44 (PTAB Jan 14, 2015); *Carl Zeiss SMT GmbH v. Nikon Corp.*, Case IPR2013-00362, Paper 23 (PTAB June 5, 2014); *Ultratec, Inc. v. Sorenson Commc'ns, Inc.*, Case IPR2013-00288, Paper 38 at 2 (PTAB May 23, 2014); *Primera Tech., Inc. v. Automatic Mfg. Sys., Inc*., Case IPR2013-00196, Paper 33 (PTAB Feb. 10, 2014); *ZTE Corp. v. Contentguard Holdings Inc.*, Case IPR2013-00133, Paper 42 (PTAB Jan. 21, 2014).

Owners' Motion to Exclude are not considered, and the request to exclude Exhibits 1036, 1039–1042, 1044–1045 as being outside the scope of a proper reply is denied.

With regard to Exhibit 1008, the Sirer Declaration, we agree that Patent Owner was unable to cross-examine Mr. Sirer. We stated above, *see supra* footnote 11, that we give no weight and do not rely on the Sirer Declaration. In that same footnote we discuss Exhibit 1024, to which Patent Owner objects. We do not rely on either Exhibit 1008 or 1024 in rendering our findings regarding whether Sirer is a printed publication. Accordingly, the request to exclude Exhibits 1008 and 1024 is denied as moot.

We deny on the merits Patent Owner's request to exclude the Declaration of Mr. Mel DeSart, Ex. 1036, and the Sirer reference, Ex. 1004. First, as to Exhibit 1036, the Board granted the request to submit the DeSart Declaration as supplemental information under 37 C.F.R. ¶ 123(b). *See* Ex. 1037 at 24:5–19. Second, Patent Owner conducted the cross-examination of Mr. DeSart and points to no persuasive evidence that Mr. DeSart's testimony is unreliable or lacks foundation. We agree with Petitioner that Mr. DeSart's testimony is based on personal knowledge of the business practices of the University of Washington Engineering Library. Paper 50, at 8–9. We overrule Patent Owner's objections to Exhibit 1036 and deny Patent Owner's request to exclude it.

As to the Sirer reference, Exhibits 1004 has not been shown to be either irrelevant or hearsay. Nor is there a lack of authentication of the Sirer reference. The Sirer reference is self-authenticating because it contains indicia sufficient to show that it is an ACM article as discussed *supra* at Section II.D.3 ("Whether Sirer is a Printed Publication"). *See* Paper 50 at

12–13 (Petitioner asserting the periodical and inscription information that show Sirer is self-authenticating). Further, the Sirer article is not hearsay, as it is being considered only for what it describes and not for truth. *See* Fed. R. Evid. 807(c); *Joy Techs., Inc. v. Manbeck*, 751 F.Supp. 225, 233 n.2 (D.D.C. 1990), *aff'd*, 959 F.2d 226 (Fed. Cir. 1992). Accordingly, Patent Owner's objections to Exhibit 1004 is overruled, and the requests to exclude it are denied.

With regard to Exhibit 1012 and annotated figures in the Petition, we adopt the reasons provided by Petitioner in its opposition to the Patent Owner motion to exclude. Paper 50 at 13–15. The objections to Exhibit 1012 are overruled, and the motion to exclude the exhibits and annotated figures is denied.

III. CONCLUSION

For the foregoing reasons, we conclude that Petitioner *has not shown* by a preponderance of the evidence that claims 1–8, 10, and 11 of the '154 patent are unpatentable. Petitioner's Motion to Exclude is denied as moot. Patent Owner's Motion to Exclude is denied.

IV. ORDER

In consideration of the foregoing, it is hereby:

ORDERED that claims 1–8, 10, and 11 of the ’154 patent have not been shown to be unpatentable;

FURTHER ORDERED that Petitioner’s Motion to Exclude is denied as moot;

FURTHER ORDERED that Patent Owner’s Motion to Exclude is denied; and

FURTHER ORDERED that, because this is a Final Written Decision, parties to the proceeding seeking judicial review of the decision must comply with the notice and service requirements of 37 C.F.R. § 90.2.

PETITIONER:

Orion Armon
Christopher Max Colice
Jennifer Volk
Brian Eutermoser
COOLEY LLP
oarmon@cooley.com
mcolice@cooley.com
jvolkfortier@cooley.com
beutermoser@cooley.com

Nathaniel Hamstra
QUINN EMANUEL URQUHART & SULLIVAN, LLP
nathanhamstra@quinnemanuel.com

PATENT OWNER:

James Hannah
Jeffrey H. Price
Michael Kim
KRAMER LEVIN NAFTALIS & FRANKEL LLP
jhannah@kramerlevin.com
jprice@kramerlevin.com
mkim@finjan.com