

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

PALO ALTO NETWORKS, INC.,
Petitioner,

v.

FINJAN, INC.,
Patent Owner.

Case IPR2015-01974¹
Patent 7,647,633 B2

Before, THOMAS L. GIANNETTI, MIRIAM L. QUINN, and
PATRICK M. BOUCHER *Administrative Patent Judges*.

QUINN, *Administrative Patent Judge*.

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

¹ Case IPR2016-00480 (filed by Blue Coat Systems, Inc.) has been joined with this proceeding.

Palo Alto Networks, Inc. and Blue Coat Systems, Inc. (“Petitioner”) each filed a Petition to institute *inter partes* review of claims 1–4, 6–8, 13, 14, 19, 28, and 34 of U.S. Patent No. 7,647,633 B2 (“the ’633 patent”) pursuant to 35 U.S.C. § 311–319. IPR2015-01974, Paper 1 (“Pet.”); IPR2016-00480, Paper 3. Finjan, Inc. (“Patent Owner”) filed a Preliminary Response in both proceedings. IPR2015-01974, Paper 6; IPR2016-00480, Paper 8. Upon consideration of the information submitted by the parties at the preliminary stage, we instituted trial only as to claims 14 and 19 of the ’633 patent. Paper 7 (“Dec.”). We also granted Blue Coat Systems, Inc.’s motion requesting joinder of IPR2016-00480 with this proceeding. Paper 17. We terminated Case IPR2016-00480, and ordered consolidation of all Petitioner filings in this proceeding. *Id.* at 10.

During trial, Patent Owner filed a Patent Owner Response (Paper 22 (“PO Resp.”)); and Petitioner filed a Reply (Paper 31 (“Reply”)). Both parties filed Motions to Exclude, Oppositions, and Replies in connection with those Motions. Papers 35, 36, 39, 40, 42, and 43. We held oral argument on January 5, 2017. Paper 48 (“Tr.”).

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 14 and 19 of the ’633 patent are unpatentable.

I. BACKGROUND

A. RELATED MATTERS

Petitioner identifies the '633 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, 13-cv-03133, 13-cv-03999, 5-13-cv-04398, 13-cv-05808, and 5-15-cv-01353). Pet. 2. Petitioner also states that petitions for *inter partes* review have been filed regarding other patents assigned to Patent Owner. *Id.*

The '633 patent is also the subject of two *ex parte* reexamination proceedings with Control Nos. 90/013,016 and 90/013,652. Paper 46 (Patent Owner updated mandatory notice pursuant to 37 C.F.R. § 42.8). Neither of these *ex parte* reexamination proceedings involves the claims-at-issue in this *inter partes* review.

B. THE '633 PATENT (EX. 1001)

The '633 patent relates to a system and a method for protecting network-connectable devices from undesirable downloadable operation. Ex. 1001, 1:30–33. The patent describes that “Downloadable information comprising program code can include distributable components (e.g. Java™ applets and JavaScript scripts, ActiveX™ controls, Visual Basic, add-ins and/or others).” *Id.* at 1:60–63. Protecting against only some distributable components does not protect against application programs, Trojan horses, or zip or meta files, which are other types of Downloadable information. *Id.* at 1:63–2:2. The '633 patent “enables more reliable protection.” *Id.* at 2:27–28. According to the Summary of the Invention,

In one aspect, embodiments of the invention provide for determining, within one or more network “servers” (e.g. firewalls, resources, gateways, email relays or other devices/processes that are capable of receiving-and-transferring a Downloadable) whether received information includes executable code (and is a “Downloadable”). Embodiments also provide for delivering static, configurable and/or extensible remotely operable protection policies to a Downloadable-destination, more typically as a sandboxed package including the mobile protection code, downloadable policies and one or more received Downloadables. Further client-based or remote protection code/policies can also be utilized in a distributed manner. Embodiments also provide for causing the mobile protection code to be executed within a Downloadable-destination in a manner that enables various Downloadable operations to be detected, intercepted or further responded to via protection operations. Additional server/information-destination device security or other protection is also enabled, among still further aspects.

Id. at 2:39–57.

C. CHALLENGED CLAIMS

Challenged claims 14 and 19 are reproduced below.

14. A computer program product, comprising a computer usable medium having a computer readable program code therein, the computer readable program code adapted to be executed for computer security, the method comprising:

providing a system, wherein the system comprises distinct software modules, and wherein the distinct software modules comprise an information re-communicator and a mobile code executor;

receiving, at the information re-communicator, downloadable-information including executable code; and

causing mobile protection code to be executed by the mobile code executor at a downloadable-information destination such that one or more operations of the executable

code at the destination, if attempted, will be processed by the mobile protection code.

19. The method of claim 14, wherein the re-communicator is at least one of a firewall and a network server.

Id. at 21:58–22:5, 22:15–16.

D. INSTITUTED GROUNDS

We instituted *inter partes* review of claims 14 and 19 based on the following grounds (Dec. 13–16):

Reference(s)	Basis	Claims
Shin ²	§ 103	14 and 19
Poison Java ³ and Brown ⁴	§ 103	14 and 19

Petitioner supports its contentions of unpatentability with a declaration from Dr. Aviel Rubin. Ex. 1002. Patent Owner supports its contentions of patentability with a declaration from Dr. Michael Goodrich. Ex. 2019. Patent Owner also proffers as support a declaration from Dr. Harry Bims (Ex. 2020) and Michael Kim (Ex. 2021). The cross-examinations of Drs. Rubin, Goodrich, and Bim are in the record as Exhibits 2022, 1097, and 1098, respectively.

² Insik Shin, et al., *Java Bytecode Modification and Applet Security* (Technical Report, Computer Science Dept., Stanford University, 1998), <https://web.archive.org/web/19980418130342/http://www-cs-students.stanford.edu/~ishin/reserach.html> (Ex. 1009) (“Shin”).

³ Eva Chen, *Poison Java*, IEEE SPECTRUM, August 1999 at 38 (Ex. 1004).

⁴ Mark W. Brown, et al., SPECIAL EDITION USING NETSCAPE 3, (Que Corp. 1996) (Ex. 1041) (“Brown”).

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). 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).

In our Decision on Institution, we did not construe expressly any claim terms. During trial, Patent Owner proposed a construction for the following phrase: “causing mobile protection code to be executed by the mobile code executor at a downloadable-information destination such that one or more operations of the executable code at the destination, if attempted, will be processed by the mobile protection code.” PO Resp. 12–13. Patent Owner urges the Board to construe the phrase according to its plain and ordinary meaning, with the explanation that “the mobile protection code was communicated to the downloadable-information destination without modifying the executable code.” *Id.* Patent Owner argues that a District Court has construed the phrase and concluded that “[i]t is clear from the specification that the MPC [(mobile protection code)], does not modify

the executable code.” *Id.* at 13 (citing Ex. 1036, 9). In particular, Patent Owner points out the District Court’s reliance on a stated advantage of the ’633 patent invention: protecting against malicious operations without modifying the mobile code and the specification’s statements distinguishing the invention from the prior art on this basis. *Id.* at 13–14 (citing Ex. 1001 at 4:12–16, 10:39–44). Patent Owner also relies on various expert declarations submitted in connection with the District Court litigation. *Id.* at 15–16.

Petitioner counters that the Board should construe the phrase under the broadest reasonable interpretation, and that under that interpretation the claim language should be given its plain and ordinary meaning. Reply 2–3. Petitioner, however, argues that the “without modification” aspect of Patent Owner’s proposed construction is supported by neither the claim language nor the specification. *Id.* at 3–5. Moreover, Petitioner argues that a Finjan expert, Dr. Harry Bims, although opining on issues of secondary considerations of nonobviousness (Ex. 2020), confirms that Patent Owner’s proposed construction is at odds with the plain and ordinary meaning of the phrase. *Id.* at 5–7.

Claim 14 is directed to a computer program product adapted to execute three steps. The first step is directed to providing a system of software modules, one of which is a re-communicator, the other being a mobile code executor. In the second step, the re-communicator receives “downloadable-information including executable code.” In the third and final step, the executable code is processed at the destination by the mobile protection code, if one or more operations in that code are attempted. The issue before us is whether the executable code at the destination could be a

modified code, in comparison to the executable code that the re-communicator received. Having reviewed the evidence and arguments proffered by both parties on this issue, we determine that the claimed “executable code” is not modified prior to being processed by the mobile protection code at the destination.

First, we note that the claim language, itself, is silent regarding the modified or unmodified status of the “executable code” at the destination. The omission, however, is not dispositive because the language of claim 14 focuses on the functionality of the mobile protection code vis-à-vis the executable code. For example, the claim focuses on execution of the mobile protection code at a downloadable-information destination to process the “one or more operations of the executable code at the destination, if attempted.” Ex. 1001, 22:1–4. The claim language, therefore, implies that the protection from malicious code is provided by the mobile protection code processing the attempted operations of the downloaded executable code. Considering the objective of the invention is to protect a computer from “malicious” operations, we understand claim 14 to require processing the executable code which may result in malicious operations, as received, at the intended computer destination via the mobile protection code. In other words, the claim language may be understood to require that the “executable code” received at the re-communicator is the same code whose “one or more operations” are being processed at the destination by the mobile protection code. See PO Resp. 14–15 (citing Ex. 2019, the Goodrich Declaration, ¶ 36).

Petitioner argues that the broader interpretation consistent with a plain and ordinary meaning may not require that the entire “executable code”

received at the re-communicator is the same executable code processed at the destination. Reply 3. At oral argument, Petitioner agreed that the claimed “executable code” at the destination recites an antecedent that refers back to the received executable code, and is the same code. Tr. 9:8–14. The claim language, according to Petitioner, although referring to the same code, does not resolve the issue whether some of the executable code processed at the destination may be modified. *Id.* at 9:15–25; 10:22–11:10 (Petitioner arguing that the claimed “one or more operations” of the executable code at the destination “could refer to either modified or unmodified.”). We agree with Petitioner that although the claim language refers to the executable code being received at the destination, the question of whether operations of the code may be modified is unanswered by the claim language alone. Our inquiry, however, is not based on the claim language in a vacuum, as the appropriate meaning of the claim language emerges when viewing the claim in the context of the specification. *See Innova/Pure Water, Inc. v. Safari Water Filtration Sys., Inc.*, 381 F.3d 1111, 1116 (Fed. Cir. 2004) (stating that the specification provides context and may be relied on to understand the meaning of the claim).

We acknowledge that the specification does not define “executable code” or any of the words of the disputed claim phrase. With regard to the executable code and its processing, we note that the background of the invention fairly describes it as “program code”: “It is observed by this inventor for example, that Downloadable information comprising program code can include distributable components” Ex.1001, 1:60–63. There is also a reference in the specification to “application programs,” and programs in various forms: “[Downloadable information] can also include,

for example, application programs, Trojan horses, multiple compressed programs such as zip or meta files, among others.” *Id.* 1:63–66; *see also* 2:29–33 (“remotely operable code that is protectable against can include downloadable application programs, Trojan horses and program code groupings, as well as software ‘components,’ such as Java™ applets, ActiveX™ controls, JavaScript™/Visual Basic scripts, add-ins, etc., among others.”). From these descriptions, we conclude that it would be apparent to a person of ordinary skill that the claimed “executable code” was meant to include programs or computer instructions that are executed by a computer. The executable nature of the “executable code” is essential to the nature of the problem addressed by the ’633 patent claims, i.e., virus protection. According to the background of the invention, a virus is “potentially system-fatal or otherwise damaging *computer code*.” *Id.* at 1:40–44 (emphasis added).

In describing the problems with the virus protection systems in the prior art, the inventors recognized known techniques for protecting a sub-set of executable code, namely only Java applets and ActiveX controls. *Id.* at 1:65–2:2. But more importantly, the inventors disclosed that this prior art, specifically U.S. Patent No. 5,983,348 to Shuang, Ji (Ex. 2006, “Shuang-Ji”), was resource intensive, used high bandwidth static and operational analysis of the downloadable content, and modified the Downloadable component. *Id.* at 2:2–4.⁵ According to the inventors,

⁵ Shuang-Ji is directed to scanning an applet at a server to identify problematic instructions, which are then instrumented by inserting special code before and after each problematic instruction. Ex. 2006, 3:16–31. Alternatively, the problematic instruction may be replaced. *Id.*

Shuang-Ji also lacked the efficiency and flexibility afforded by the invention. *Id.* at 2:10–13. The embodiments of the '633 patent all involve receiving at, or sending to, a downloadable destination, a mobile protection code and the downloadable information that includes the executable code. *Id.* at 2:39–3:62.

The mobile protection code is installed at the user device (downloadable destination) and monitors resource access attempts by the executable code in the downloadable information, such that those attempts may be dealt with in a variety of ways. *Id.* at 3:63–4:10. One of those ways includes the option of “modifying the Downloadable operation.” *Id.* However, it is important to note that the mobile protection code, being executed at the client, is not described as modifying the executable code itself before or after the executable code is received at the destination. The relevance of this passage is that, notwithstanding the threat of the malicious operations, the destination executes the instructions in the executable code. The mobile protection code intercepts the potentially malicious operations resulting from the execution of those instructions. In sum, all the embodiments described in the '633 patent defer the processing of the malicious operations to the destination, with one of those processing options being the modification of the operation. However, no embodiment describes modification of the executable code itself.

Nevertheless, the instrumentation occurs at a server, and the instrumented applet is downloaded to the client. *Id.* at 3:32–35. In other words, the applet, or executable code, is modified at the server before delivery to the client.

In addition to providing the above descriptions of the executable code, we find that the specification expressly describes an advantage of non-modification of the executable code as follows:

Advantageously, systems and methods according to embodiments of the invention enable potentially damaging, undesirable or otherwise malicious operations by even unknown mobile code to be detected, prevented, modified and/or otherwise protected against *without modifying the mobile code*. Such protection is further enabled in a manner that is capable of minimizing server and client resource requirements, does not require pre-installation of security code within a Downloadable-destination, and provides for client specific or generic and readily updateable security measures to be flexibly and efficiently implemented.

Ex. 1001, 4:11–21 (emphasis added). This passage uses the phrase “mobile code” not “executable code.” We find, however, that the “mobile code” being referred to here is the code that may result in the potentially malicious operations sought to be prevented. *Id.* (referring to enabling detection of “potentially damaging, undesirable or otherwise malicious operations by even unknown *mobile code*”) (emphasis added). Thus, this passage is consistent with the statements regarding the difference between Shuang-Ji’s modification of the downloadable component, i.e. executable code. *See e.g., id.* at 1:63–2:6 (referring to Shuang-Ji requiring modification of the “Downloadable component,” which is a Java applet or ActiveX controls components included in the Downloadable information). In sum, while the prior art modifies the downloadable component, the claimed invention does not. And this is an advantage touted by the inventors in describing that communicating the mobile protection code separate from the downloadable that includes executable code is more accurate and “far less resource

intensive than, for example, performing content and operation scanning, modifying a Downloadable, or providing completely Downloadable-destination based security.” *Id.* at 10:39–45.

Notwithstanding the specification passages analyzed above, Petitioner asserts that embodiments of the ’633 patent specification disclose modification of the executable code. Reply 4–5. Specifically, Petitioner points out the following passage: “[T]he IAT [(API Import Address Table)] can be modified so that any call to an API can be redirected to a function within the MPC [mobile protection code].” *Id.* at 4 (citing Ex. 1001, 17:51–53); Tr. 12:11–13:8. As further evidence of modification of executable code embodiments, Petitioner argues the description of compression, encryption, or encoding of executable code discloses modified executable code. *Id.* (citing Ex. 1001, 13:29–30). As support, Petitioner alleges that Patent Owner’s expert agrees that compression comprises modification. *Id.* at 5 (citing Ex. 1100, 173:22–174:16).

We are not persuaded by Petitioner’s arguments. First, we find that the embodiment concerning the IAT table modification does not disclose modification of executable code. The embodiment Petitioner refers to describes the functionality of the mobile protection code after installation at the destination. Ex. 1001, 17:13–19 (describing an embodiment shown in Figure 7a of the client receiving a sandbox package and initiating a mobile code installer, which initiates the mobile protection code); 17:28–42 (describing that the mobile protection code intercepts an operation, and, as one example, limits access to resources, such as address space, in the event the operation is malicious). As explained above, the specification does describe modification of operations performed at the destination when the

mobile protection code detects malicious code. But the modifications are (1) to the operation resulting from execution of the executable code; and (2) at the destination, after the executable code has been received at the destination. The IAT embodiment Petitioner points to is one such embodiment, where the mobile protection code modifies an operation at the destination. For instance, the mobile protection code loads in memory the executable code, shown in Figure 7a as “XEQ” 343. *Id.* at 17:43–47. The IAT, illustrated as “API” 731 in Figure 7b, is loaded in memory also so that the mobile protection code can access the table and divert the access attempts of the executable code to an analyzer. *See id.* at 17:58–63 (stating that initial access to the IAT provides for diverting, evaluating, and responding to attempts by the downloadable to utilize system APIs); 18:20–24 (describing that a resource access diverter modifies that IAT entries causing access attempts to be diverted to the resource access analyzer). In short, the ’633 patent specification describes modification of the IAT after the executable code in the downloadable has been received and as part of the processing of the malicious operations by the mobile protection code.

Given the above disclosure, arguments by the parties regarding whether the IAT is executable and whether the IAT is part of the downloadable are not germane to our determination. *See* Tr. 14:13–15:5 (Petitioner asserting that the IAT is part of the downloadable); 43:6–13 (Patent Owner arguing that the IAT is not executable code and the modification occurs at the client once the mobile protection code has access to it). As stated above, we find that the IAT disclosure in the ’633 patent describes the claimed processing by the mobile protection code of the

attempted operations of the executable code. This disclosure does not support Petitioner's assertion that the specification contemplates modification of the executable code prior to sending it to the client for processing there.⁶ Indeed, such a disclosure would be contrary to the stated advantages and distinctions discussed above of mobile protection code that detects, prevents, and modifies malicious operations of executable code, without modifying such executable code.

Finally, we address whether the alleged disclosure of compression, encryption, and encoding supports Petitioner's contention that the '633 patent specification discloses modification of executable code. Reply 4–5 (citing Ex. 1001 at 13:29–30). Patent Owner contends that the disclosure of compression, encryption, and encoding does not disclose modification of the executable code. Tr. 39:3–40:15. One of the confusing aspects of this case is that evidence proffered by Petitioner as supporting its claim construction position is taken from Patent Owner's evidence submitted during the District Court litigation, before the Court there ruled against Patent Owner on this issue. For instance, Petitioner points to Dr. Rubin's testimony during cross-examination as supporting the contention that compression constitutes modification. Reply 4–5 (citing Ex. 2022, 108:4–11). Upon close inspection, however, we understand that Dr. Rubin read into the record portions of Finjan's brief on claim construction submitted in the District Court litigation. Ex. 2022, 107:4–108:11 (referring to Plaintiff Finjan, Inc.'s

⁶ We also disagree with Petitioner that Dr. Goodrich's testimony supports Petitioner's position on this issue. Dr. Goodrich testified that although the '633 patent describes modifying the IAT, that modification does not constitute modifying the downloadable. Ex. 1097, 25:7–27:19.

Opening Claim Construction Brief (Exhibit 1039), page 17 (Petitioner's pagination in the footer of the exhibit), sentence starting at line 15).⁷ The reality here is that Patent Owner was not successful in persuading the District Court that the disputed '633 patent claims (including claim 14) are broad enough to encompass communicating both un-modified and modified executable code to a client. *See* Ex. 1036, 8–15; Tr. 41:10–42:14 (Patent Owner explaining that Finjan adopted the district court's construction, did not object to it even though it was not the construction it proffered, and that it has not raised this issue on appeal). The intrinsic record is sufficiently clear that we have no need to rely on expert testimony presented in District Court.⁸ And the intrinsic record does not show that operations of compression, encryption, and encoding constitute a modification of executable code. The '633 patent discloses “restoring” the downloadable, not modifying it. Ex. 1001, 13:29–32. That is, if a downloadable is received in a compressed format, it is decompressed to detect whether there is executable code. *Id.* The downloadable then is *restored* to its compressed format for transmission and subsequent processing at the client. *Id.* This is not teaching modification of executable code because the code itself remains unchanged.

To conclude, our determination is based on the plain and ordinary meaning of the claims within the context of the specification. First,

⁷ Similarly, Petitioner relies on testimony from the deposition of Finjan's expert in the District Court litigation, Dr. Nenad Medvidovic, Exhibit 1100. Reply 5 (citing Ex. 1100).

⁸ We do not rely on expert testimony also because extrinsic evidence may not be used to contradict claim meaning that is unambiguous in light of the intrinsic record. *Summit 6, LLC v. Samsung Elecs. Co.*, 802 F.3d 1283, 1290 (Fed. Cir. 2015).

according to the entire disclosure of the '633 patent specification the executable code is received at the destination without modification. *See Vitronics Corp. v. Conceptronic, Inc.*, 90 F.3d 1576, 1582 (Fed. Cir. 1996) (the specification is “the single best guide to the meaning of a disputed term,” and “acts as a dictionary when it expressly defines terms used in the claims or when it defines terms by implication.”). “The specification may define claim terms ‘by implication’ such that the meaning may be ‘found in or ascertained by a reading of the patent documents.’” *Bell Atl. Network Servs., Inc. v. Covad Communications Group, Inc.*, 262 F.3d 1258, 1268 (Fed.Cir.2001) (quoting *Vitronics*, 90 F.3d at 1582, 1584 n. 6). We also confirm the scope of the “executable code” in light of the advantage of the invention, e.g., to protect against malicious code without modifying the executable code, thereby minimizing resources. *See Toro Co. v. White Consol. Indus., Inc.*, 199 F.3d 1295, 1301 (Fed. Cir. 1999) (construing a ring as permanently attached to the cover because all embodiments read together suggest this relationship and the specification describes the advantages of the unitary structure). Although the specification describes additional advantages, such as not requiring pre-installation of security code within the destination, the claims need not encompass all stated advantages. *See Golight, Inc. v. Wal-Mart Stores, Inc.*, 355 F.3d 1327, 1331 (Fed. Cir. 2004) (“[P]atentees [are] not required to include within each of their claims all of [the] advantages or features described as significant or important in the written description.”). Finally, all the embodiments are consistent with and support the inventors’ characterization of the problem solved: avoiding the resource intensive modification by communicating to the destination the mobile protection code and the detected executable code. *See Ex. 1001*,

10:39–45; 1:63–3:6; *see also O.I. Corp. v. Tekmar Co.*, 115 F.3d 1576, 1581 (Fed. Cir. 1997) (relying on the consistent disclosure of structures as being either non-smooth or conical and the statements distinguishing prior art where structures were smooth). In light of the disclosure, the plain and ordinary meaning of the disputed claim language emerges: the executable code whose operations are processed by the mobile protection code at the destination is the same as the executable code received, i.e., it undergoes no modification.

In sum, we determine that the “executable code” at the destination is not modified.

B. PRINTED PUBLICATION ISSUES

The three asserted references are non-patent documents. Shin (Ex. 1009) is an article that Petitioner alleges was publicly available via the Internet in 1998. Pet. 29. Poison Java (Ex. 1004) is an article that Petitioner asserts was published in the August 1999 issue of the IEEE Spectrum magazine. Pet. 25. And Brown (Ex. 1041) is a guide to the Netscape browser and appears to be a book that Petitioner asserts was published in 1996. Pet. 27. Patent Owner argues that Petitioner did not meet its burden in showing that the three references are printed publications. Based on our review of the evidence of record, we determine that Shin, Poison Java, and Brown are printed publications as described further below.

1. *Shin*

On the face of this article, we note that there are no indicia of publication or dissemination. *See* Ex. 1009. Petitioner, however, has provided other evidence establishing that Shin was publicly available no later than April 1998. A printout of the website directed to Mr. Shin’s

research at Stanford, and archived by archive.org, shows that the article's title was listed and that the article was publicly available through the Internet as of April 14, 1998. Ex. 1095. As explained in the affidavit proffered by Christopher Butler, Office Manager of Internet Archive, the URL format for captured website, "19980418130342/http://www-cs-students.stanford.edu/~ishin/research.html," establishes that the website posting the abstract and the title of the article with a link to the article itself were archived on April 18, 1998 at 13:03:42. *Id.* The article was available as a PDF file, downloadable from the same page. *Id.* (identifying in a cover page the URL showing that Ex. 1009 was available from www-cs-students.stanford.edu/~ishin/paper.ps, as of April 18, 1998). We find this evidence credible and undisputed.

Finally, there is also undisputed evidence that Dr. Rubin read Shin in 1998. Ex. 2022, 70:20–71:13. Accordingly, we find that Shin was publicly available and sufficiently accessible online to persons interested in applet security no later than April 14, 1998.

2. *Poison Java*

On the face of this reference, we note that Poison Java contains multiple indicia that it is part of a periodical, IEEE Spectrum, published in August 1999. Ex. 1004. This fact is confirmed by the declaration and testimony of Gerard P. Grenier, Senior Director of Publishing Technologies and custodian of certain records at IEEE. Exs. 1005, 2023. In particular, we note that Mr. Grenier testified that, although the electronic copies of Poison Java are now maintained in IEEE Xplore, the article itself was published in Volume 36, Issue 8 of IEEE Spectrum in August 1999. Ex. 1005. The printouts attached to Mr. Grenier's declaration show that the database IEEE

Xplore deems Poison Java officially published in August 1999. *Id.* at Ex. A; Ex. 2023, 29:19–20, 32:7–34:4 (testifying that IEEE Spectrum is a flagship publication of IEEE mailed to members every month and that it is publishing practice to mail by no later than the first day of the month published on the cover).

Patent Owner makes much about the fact that the IEEE Xplore database would not have been available until June 2000. PO Resp. 23–24. This argument, however, ignores that the article was not *only* available as a download in the IEEE database. The evidence shows that the article was included in a monthly publication of the magazine IEEE Spectrum and, according to the publishing practices of IEEE, the magazine issue containing Poison Java would have been received by subscribers by August 1999. Accordingly, we find that Poison Java is a printed publication, disseminated to subscribers of the IEEE Spectrum in August 1999.

3. *Brown*

Petitioner asserts that Brown (Ex. 1041) was published in 1996, as confirmed by a declaration of one of the book's authors and the Internet Archive. Exs. 1082, 1092. Peter Kent, one of the authors, testifies that he received a copy of the book shortly after publication in December 1996, and that the publisher sent periodic sales updates beginning in 1996. Ex. 1082 ¶¶ 3–6. An affidavit from Christopher Butler, of Internet Archive, provides evidence that a website listing books by Que, the publisher of Brown, listed the book for sale as of the date of the archived page, December 19, 1996. See 1092 (showing capture of webpage <http://www.mcp.com/que/et/se-net3/>). On cross-examination, Mr. Kent testified that the book was actually published on September 1996, not

December 1996. Ex. 2024, 20:20–22:25. The testimony leads to the conclusion that the book was printed in 1996, notwithstanding the misunderstanding regarding whether September or December was the first month of publication.

Furthermore, we find persuasive the evidence from the Internet Archive that Brown was available for sale on the Internet as of December 16, 1996. Notwithstanding Patent Owner’s argument that the link to buy the book has not been shown to work, we find persuasive that Brown was available for purchase over the Internet, which was searchable using a browser. *See* Ex. 2024, 33:12–17. In summary, the record evidence on the whole persuades us that Brown was published and offered for sale in the United States in 1999. We are persuaded that by December 1999, anyone with interest in understanding the use of the Netscape browser would have been able to buy it. *See In re Wyer*, 655 F.2d 221, 227, (CCPA 1981) (“Accordingly, whether information is printed, handwritten, or on microfilm or a magnetic disc or tape, etc., the one who wishes to characterize the information, in whatever form it may be, as a ‘printed publication’ ... should produce sufficient proof of its dissemination or that it has otherwise been available and accessible to persons concerned with the art to which the document relates and thus most likely to avail themselves of its contents.”) (citations omitted).

C. 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).

D. 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.” Pet. 24; Ex. 1002 ¶ 29. 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. Michael Goodrich, offers a level of ordinary skill that is different from Petitioner’s. Ex. 2019 ¶ 25. In Particular, Dr. Goodrich 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’s experience or the proposed alternative of 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. Goodrich’s opinions would not change if he had considered instead the level of ordinary skill in the art proffered by Dr. Rubin. *Id.* ¶ 28.

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).

E. OBVIOUSNESS GROUND BASED ON SHIN

Petitioner asserts that claims 14 and 19 are unpatentable as obvious over Shin. Pet. 38–44. Specifically, with regards to Shin, Petitioner contends that Shin discloses Python or Java code (*id.* at 38–39) with software modules (*id.* at 39) implemented in an HTTP proxy server or “re-communicator” and a Java Virtual Machine within the HTTP client (*id.* at 40) to receive “messages” from the web server and to send those “messages”

to the client (*id.* at 41). At the HTTP client, according to Petitioner, Shin executes safeguarding code incorporated in modified applets to process the operations attempted by the applet, “such as window attacks, network accesses, and uniform resource locator (URL) spoofing, by performing security checks and raising exceptions if those checks fail.” *Id.* at 42–43. Therefore, Shin, according to Petitioner, teaches all the limitations recited in claim 14. As for claim 19, Petitioner alleges that Shin teaches the further limitation of “at least one of a firewall and a network server” because Shin teaches the use of an HTTP proxy server that modifies classes before they are received by the browser. *Id.* at 44.

1. *Overview of Shin (Ex. 1009)*

Shin’s title is “Java Bytecode Modification and Applet Security.” Ex. 1009, 1. By its very title, Shin describes modifying bytecode. The reason for the modification is to protect client machines from applets that a Java Virtual Machine alone cannot protect. *Id.* at 3–4 (describing denial of service attacks, disclosure of confidential information attack, spoofing attack, and annoyance attack). Shin describes restricting the applet’s operations by “inserting safeguarding code,” that monitors and controls resource usage and that limits the functionality of the applet. *Id.* at 4. In practice, Shin substitutes one “executable entity, such as a class or a method, with a related executable entity that performs additional run time tests.” *Id.* The applet is modified at the HTTP proxy server, which sits between a web server and a client browser. *Id.*

Shin describes two ways to implement the modification: class-level modification and method-level modification. *Id.* at 4–5. Shin states that “[i]n Java, all references to strings, classes, fields, and methods are resolved

through indices into the constant pool of the class file, where their symbolic names are stored.” *Id.* at 5 (citation omitted). For a Java class-file change, Shin modifies the constant pool. For example, a “Window” class may be replaced by a sub-class, “Safe\$Window,” that sets a limit on the number of new windows that an applet may create. *Id.* That substitution is shown in Figure 2 of Shin, reproduced below.

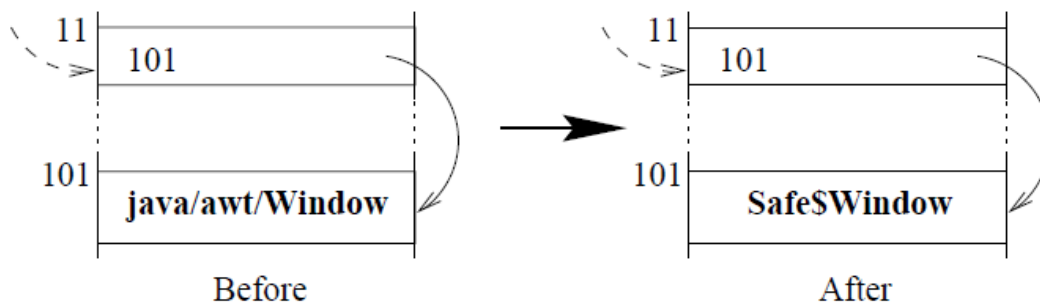


Figure 2: Class-level modification substitutes class reference

Figure 2 illustrates that the constant pool is altered to replace the entry 101, the name of the “class java/awt/Window,” with the name of the new class, “Safe\$Window.” *Id.* at 6.

As for the method-level modification, Shin also replaces a method with a related method. *Id.* at 6. As in the case of the class-level modification, Shin describes making substitutions in the constant pool. *Id.* at 7–8. The method reference entry is modified and new entries are added to reference the new method. *Id.* 8–9. Yet another technique that Shin discloses is the method invoking modification. *Id.* at 9. In this technique, the new method, such as “Safe\$Thread.setPriority(Ljava/lang/Thread;I)V” is added into the bytecode program. *Id.* at 9. In short, Shin’s class-level modification requires constant pool modifications, while method-level

modifications requires constant pool modifications and instruction modifications. Regardless of which modification is desired, Shin explains that the techniques involve “modified executable entities [that] are inserted in Java bytecode.” *Id.* at 4.

2. Analysis

The issue for us to decide is whether Petitioner has shown by a preponderance of the evidence that Shin teaches or suggests the recited “executable code” as we have construed the term. Patent Owner argues that Shin teaches modification of executable code, because Shin describes modifying the Java applets at the HTTP proxy server with “safeguarding code” before forwarding the modified applet to the client. PO Resp. 27–28. In response, Petitioner argues that Shin’s applet modification is “essentially the same as the application programming interface (API) import address table (IAT) modification taught in the ’633 patent.” Reply 16 (citing Ex. 1001, 17:43–67). Specifically, Petitioner asserts that the Shin modification, like the IAT modification, involves “modifying pointers to resources used by the device executing the executable code.” *Id.* (citing Ex. 2022, Dr. Rubin’s cross-examination testimony, 138:16–139:1). During the oral argument, Petitioner clarified its contention that the constant pool in Shin is modified and the constant pool is “located in the Java Bytecodes of the Java applet.” Tr. 23:7–20. In sum, Petitioner contends that Shin’s constant pool does not constitute executable code, even though it is part of the downloadable delivered to the client. *Id.* at 23:21–24:24.

We find that Shin’s constant pool is not the same as the IAT embodiment described in the ’633 patent. As stated above, the mobile protection code modifies the IAT at the destination while processing the one

or more operations of the executable code. In contrast, Shin's applet is modified at the proxy server, and its modification constitutes rewriting the bytecode of the applet, which is executable. *See* Ex. 1009, 2 ("we propose a technique, called *bytecode modification*, through which we put restrictions on applets by inserting additional bytecode instructions that will perform the necessary run-time tests"), 4 ("modified executable entities are inserted in Java bytecode"). As explained by Shin, the constant pool is part of the class file of the applet. *Id.* at 5. Further, the bytecode modification is not performed at run-time, like the IAT modification table scenario. *See id.* at 2 ("HTTP proxy server []modifies classes before they are received by the browser"), 4 (Shin explaining that the "safety mechanism must be applied before the applet is executed").

The differences are significant, for the modification of the applet at the server is precisely the type of modification that the '633 patent sought to avoid. The applet is undisputedly executable code, and is what Petitioner pointed to in Shin as disclosing executable code. *See* Pet. 43 ("Shin also discloses that the safeguarding code processes operations attempted by the applet ('executable code') . . ."). By describing that the server modifies the constant pool in the applet, we find that Shin teaches modifying the executable code before the applet is processed at the client browser. Therefore, Petitioner has failed to show by a preponderance of the evidence that Shin teaches or suggests that the applet is "executable code" as we have construed the claim language, *supra*.

We are not persuaded by Petitioner's arguments to the contrary. For instance, Petitioner argues that the modification of Shin's applet does not change "operations of the original bytecode," and, therefore, original

executable code is not altered. Reply 17 (citing Ex. 2022, Rubin cross-examination testimony, 26:6–17). First, we do not agree with Petitioner that the construction requiring no modification of executable code “only preclude[s] modifying the operations of the executable code[,] rather than [modification of] the code itself.” *Id.* Petitioner has this backward. The claim language, as we stated above, may encompass modification of the operations, as this is expressly described in the ’633 patent when the mobile protection code processes those operations at the destination. However, the modification of the operation during processing, allowed by the claim, is distinct from any modification of the code at a server. The ’633 patent, again, ensured the former, while avoiding the latter.

Second, we glean from Petitioner’s argument above that the applet Petitioner pointed to in the Petition is alleged to include both the “executable code” and the “mobile protection code.” Pet. 42 (“Shin also discloses a Java Virtual Machine (‘mobile code executor’) that can run applets that are modified to include safeguarding code . . .”). Petitioner, however, has the burden of proving that Shin’s applet constitutes both executable code and mobile protection code as claimed. Faced with the evidence of Shin’s constant pool modification, Petitioner argued that the constant pool portion of the applet file is not executable code. Reply 16–17 (“Shin’s applet modification . . . does not involve modifying problematic executable code—it only modifies the lookup table referenced when a method or class is called.”). Although we have found this argument unpersuasive, we note that even if we had not, Petitioner fails to explain whether Shin’s constant pool is mobile protection code or something else entirely. For example, at oral argument, Petitioner belatedly argued that the substitution with a class

Safe\$Window is “the only modification that the class-level modification makes in Shin[, and] that modification effectuates the mobile protection code’s protection.” Tr. 22:16–23. As discussed above, we understand Shin to implement its safeguarding code by rewriting the constant pool portion of the applet file. Ex. 1009, 4–5 (explaining that “modified executable entities are inserted in Java bytecode” and that “the modifications may be divided into two general forms,” one of which is the class-level modification that modifies the new class reference in the constant pool). Thus, if we were to identify Shin’s constant pool with the claimed mobile protection code, we find that such a mapping would fail. According to the claim language, the mobile protection code is executed, and according to Petitioner, the constant pool is only a lookup table (Reply 16–17). Further, the claim language requires that the mobile protection code process the attempted operations of the executable code, and Petitioner has not shown that the constant pool processes any operations.

According to our determination that the claimed executable code is not modified, and after review of the parties’ proffered evidence and arguments, we determine that Petitioner has not shown by a preponderance of the evidence that Shin teaches the claimed “executable code” and that Shin’s applet constitutes both “executable code” and “mobile protection code.”

3. Conclusion

Petitioner has relied solely on Shin as teaching or suggesting all the limitations of claim 14. Claim 19 depends from claim 14. Consequently, based on our findings and conclusions stated above, we determine that

Petitioner has failed to show by a preponderance of the evidence that claims 14 and 19 are unpatentable over Shin.

F. OBVIOUSNESS GROUND BASED ON POISON JAVA AND BROWN

With regard to Poison Java, Petitioner contends that AppletTrap is a system including code (Pet. 51) in distinct software modules (*id.*) where a web browser at the client receives instrumented applets and the HTML page (*id.* at 52). Petitioner also contends that Brown discloses running applets in a Java-enabled web browser. *Id.* In particular, Petitioner contends that Poison Java teaches wrapping monitoring code around the applet, such that the monitoring code can monitor and intercept potentially malicious code. *Id.* at 53–54. Petitioner argues that Poison Java “processes—and if appropriate, blocks—the executable code’s operations.” *Id.* at 54. Therefore, according to Petitioner, Poison Java teaches all the claim limitations of claim 14 with the added teaching of a Java-enabled web browser from Brown. As for claim 19, Petitioner also contends that Poison Java teaches the further limitation of “at least one of a firewall and a network server” because Poison Java describes a proxy server. *Id.*

1. Overview of Poison Java (Ex. 1004)

Poison Java describes “a hybrid solution to supplementing Java security” that integrates client- and sever-based solutions. Ex. 1004 at 42.⁹ The solution is called InterScan AppletTrap. *Id.* AppletTrap pre-filters applets at the server, which performs more thorough examination of applets that include calls to system resources. *Id.* Poison Java describes the preparation process as instrumentation. *Id.* During this instrumentation,

⁹ Citations to Poison Java refer to the pagination in the original article.

“AppletTrap wraps monitoring code around the applet and attaches the security policy that determines what system resources it can access.” *Id.* Poison Java also describes instrumentation as “addition of extra code” that “deprives the applets of their original signatures.” *Id.*

Poison Java states that “[t]he HTML page, along with the instrumented applets, is then delivered to the client and displayed on its Web browser.” *Id.* at 43. AppletTrap’s monitoring code may permit the applet’s actions to take place, may block the action, and/or notify the user. *Id.* Poison Java describes AppletTrap with an advantage of “centralized protection, with control, deployment, and management functions at the server.” *Id.*

2. Analysis

With regard to Poison Java, the issue for us to decide is whether applets instrumented at the AppletTrap server constitute “executable code,” as claimed. We determine that they do not. Petitioner points to applets received at the proxy server as the executable code received at the re-communicator. Pet. 53. Petitioner then points to the monitoring code that is wrapped around the applet during instrumentation as mobile protection code. *Id.* at 54. Petitioner explains instrumentation as follows:

Code rewriting or instrumentation involves inserting extra protection code into the mobile code, e.g., at a server that receives the mobile code for transmission to a client computer. Instrumentation is performed after mobile code is filtered or scanned but before it is executed in the sandbox. Instrumentation often involves taking advantage of the fact that mobile code is usually divided into functions for better organization, code-reuse, and readability. Mobile code can be rewritten to provide an additional layer of

security by inserting code right at the very beginning of a function body. Every time that function is called, the newly inserted code is executed first.

Id. at 9–10 (internal citations omitted).

Patent Owner contends that the instrumentation in Poison Java modifies the executable code. PO Resp. 36. We agree with Patent Owner's contention and supporting evidence. In particular, Patent Owner points out that instrumentation is rewriting the applet, which is executable code. Petitioner's description of instrumentation is consistent with Patent Owner's characterization of Poison Java. As the Petition states, instrumentation adds extra protection code to the mobile code. Pet. 9. The addition of this code is performed through code rewriting. *Id.* (citing Ex. 1020, 2). Code rewriting, for a Java applet, involves transforming a binary program into a different but functionally equivalent program. Ex. 1020 at 2; *see also* Ex. 2019 ¶ 69. Having reviewed the facts presented and the arguments made on the record, we find that the instrumentation of the Java applet in Poison Java is a modification of the executable code, as the applet's *binary code is rewritten at the server* to add functionality that monitors the applet's behavior.

Our determination is further supported by our finding that the instrumentation described in Poison Java is the same type of modification that the '633 patent described and distinguished from Shuang-Ji (Ex. 2006). As observed in our claim construction discussion, Shuang-Ji, in contrast with the invention described in the '633 patent, modifies the downloadable component at the server. Shuang-Ji, as stated above, identifies problematic instructions, which "are then each instrumented, e.g., special code is inserted before and after each problematic instruction, where the special code calls

respectively a prefilter and a post filter.” Ex. 2006, 3:24–29. Poison Java, although more generic in its description of instrumentation, describes the applet instrumentation at the server, the same technique described in Shuang-Ji. Therefore, we determine that Petitioner has failed to demonstrate by a preponderance of the evidence that Poison Java discloses that “executable code” is not modified in accordance with our claim construction, *supra*.

Petitioner’s arguments to the contrary are not persuasive. In its Reply, Petitioner argues that Poison Java’s applet instrumentation is “essentially the same as the application programming interface (API) import address table (IAT) modification taught in the ’633 patent.” Reply 21 (internal citations omitted). In support of this statement, Petitioner argues that “[b]oth involve modifying pointers to resources used by the device executing the executable code.” *Id.* Petitioner cites expert testimony of Dr. Bims, one of Patent Owner’s experts. *Id.* (citing Ex. 1098, 64:19–68:18). None of the evidence proffered by Petitioner supports the contention that Poison Java involves pointers to resources. Furthermore, the cited testimony of Dr. Bims does not address Poison Java at all. Finally, Petitioner’s argument that Poison Java does not change operations at the destination is misplaced. Reply 21–22. As we discussed above with respect to Shin, the ’633 patent claims do not preclude modification of operations processed by the mobile operation code. Indeed, modification of the operations is the only modification discussed in the ’633 patent. The dispositive issue is whether the executable code at the destination is modified. And, as stated above, rewriting or instrumentation of the Java applet at the server is a modification of executable code.

Therefore, Poison Java does not disclose, teach, or suggest the “executable code” phrase according to our construction, *supra*.

3. Conclusion

Petitioner has relied solely on Poison Java as disclosing the “executable code” phrase of claim 14. Brown is not alleged to cure the deficiencies noted above. Claim 19 depends from claim 14. Consequently, based on our findings and conclusions stated above, we determine that Petitioner has failed to show by a preponderance of the evidence that claims 14 and 19 are unpatentable over Poison Java and Brown.

III. MOTIONS TO EXCLUDE

Petitioner moves to exclude various portions of the record as follows:

- 1) Paragraph 57 of Ex. 2019, Goodrich Declaration, on the basis of insufficient factual support and unreliability. Paper 35, 1–3.
- 2) Paragraphs 13–27 and 30–34 of Ex. 2020, Bims Declaration, on the basis of improper expert testimony. *Id.* at 4–8.
- 3) Exs. 2004, 2007, 2018, 2012, 2014, 2016, 2027, 2028, 2029, 2037, and 2048 in general as irrelevant, potentially confusing, and prejudicial. *Id.* at 8–15.

Petitioner’s Motion to Exclude is denied as moot, because that evidence objected to is not relied upon in reaching our determination that Petitioner has not met its burden of showing that claims 14 and 19 are unpatentable.

Patent Owner moves to exclude Exhibits 1002, 1004, 1005, 1006, 1007, 1008, 1009, 1041, 1082, 1092, 1093, 1095, 1099, 1101, and 1035. Paper 36 (“Mot.,” “PO Motion to Exclude”).

1. Exhibit 1002, Dr. Rubin's Declaration

Patent Owner urges that the Board exclude paragraphs 79–83 of Dr. Rubin's Declaration regarding claim construction on the basis that Patent Owner alleges Petitioner did not permit Patent Owner to question Dr. Rubin regarding those opinions. Mot. 1–3. We deny Patent Owner's request as moot, because we do not rely on any testimony from Dr. Rubin's Declaration regarding claim construction.

2. Exhibit 1005, Declaration of Mr. Grenier

Patent Owner seeks exclusion of the Grenier Declaration, Exhibit 1005, on the basis of lack of authentication, relevance, and because the witness lacks personal knowledge. Mot. 3–4. Patent Owner alleges that Mr. Grenier's testimony has no relevance to the prior art at issue. *Id.* We do not agree. Mr. Grenier testifies regarding the accessibility, either in print or via database of the Poison Java article published in the IEEE Spectrum magazine. His testimony is highly relevant to Poison Java's status as prior art. Furthermore, we do not agree with Patent Owner that Mr. Grenier's testimony should be excluded for lack of personal knowledge. The portion of the deposition testimony Patent Owner relies on for this assertion does not support Patent Owner's argument. To the contrary, we find that Mr. Grenier has demonstrated, as Senior Director of Publishing Technologies at IEEE, that he has knowledge of the business practices of IEEE with regard to publishing and maintaining accurate records of its publications. We also agree with Petitioner that Mr. Grenier's testimony confirms that the printed publication of August 1999 of the magazine article is reflected in the database entries, which are performed according to business practices of

IEEE. Paper 39, 2–3. The objections to Exhibit 1005 are, therefore, overruled. Patent Owner’s request to exclude this Exhibit is denied.

3. Exhibit 1006, “Author’s Webpage”

Patent Owner seeks exclusion of Exhibit 1006 because it is allegedly hearsay, irrelevant, and lacks authentication. Mot. 4–5. Specifically, Patent Owner argues the lack of “independent authentication” and that this exhibit cannot demonstrate that Shin was publicly accessible. *Id.* Patent Owner also argues that the date of publication is hearsay and is irrelevant to establish the date of Shin. *Id.* We disagree, but need not address each of the above superficial arguments because we do not rely on Exhibit 1006 to render our decision that Shin is a printed publication as stated above. Patent Owner’s request is denied as moot.

4. Exhibit 1007, Filewatcher.

Patent Owner seeks exclusion of Exhibit 1007 on the basis of lack of authentication, hearsay, and irrelevance. Mot. 5–6. For the same reasons as stated above with regard to Exhibit 1006, we deny this request as moot.

5. Exhibit 1008, Kava Paper

Patent Owner seeks exclusion of Exhibit 1008 for lack of authentication and relevance. *Id.* at 5. For the same reasons stated above, with regard to Exhibit 1006, we deny this request as moot.

6. Exhibit 1082, Kent Declaration

Patent Owner contends that Mr. Kent lacks personal knowledge to testify regarding publication of Brown. *Id.* at 6–7. We do not agree. Mr. Kent has personal knowledge of Brown’s status as a printed publication. Whether Mr. Kent’s testimony is credible based on his reliance on different versions of Brown is an issue of weight, not admissibility. Indeed, Patent

Owner cross-examined the witness, and through that cross-examination (Ex. 2024), we find that the declaration testimony was amended to reflect more accurately what Mr. Kent perceived as the earliest date of publication of Brown. Patent Owner's objections to Exhibit 1082 are overruled, and the request to exclude it is, therefore, denied.

7. Exhibit 1093, Sherfesse Affidavit

Patent Owner makes several arguments in support of its contention that Exhibit 1093 is inadmissible. *Id.* at 7–8. We do not address any of those arguments for we do not rely on Exhibit 1093 to render our decision. Patent Owner's request is, therefore, denied as moot.

8. Exhibits 1092 and 1095, Christopher Butler Affidavits

Patent Owner seeks exclusion of Exhibits 1092 and 1095, which are declarations of Mr. Christopher Butler, the Office Manager at the Internet Archive, regarding two archived websites proffered as evidence tending to show public access and publication of Brown and Shin, respectively. *Id.* at 8. The basis for the request is alleged lack of personal knowledge of Mr. Butler. *Id.* We do not agree with Patent Owner's contention. Mr. Butler declares that he has personal knowledge of the facts testified to in his declaration, and none of Patent Owner's arguments and evidence show that this is not the case. Exs. 1092 and 1095 ¶ 1. Furthermore, Patent Owner cross-examined Mr. Butler (Ex. 2025), and confirmed through that cross-examination that Mr. Butler's responsibilities at the Internet Archive include processing requests for authentication of records from the website archive.org (*id.* at 5:10–21). Mr. Butler also testified more particularly about archived webpages, the timestamps for the webpages accessible through the URLs, and the common location of the URL at the footer of a

printout. *See, e.g., id.* at 21:1–13. Mr. Butler has established how archive.org maintains the archived web pages and how, in its ordinary course of business, the printouts of archived webpages provide URLs with timestamps in accordance with the information in the affidavit. We find the testimony sufficient to show that Mr. Butler has personal knowledge and that the affidavit supports Petitioner’s contentions regarding the publications of Brown and Shin. *See supra*, Section II.B. Therefore, Patent Owner’s objections are overruled. Patent Owner’s request to exclude Exhibits 1092 and 1095 is denied.

9. *Portions of Exhibits 1099, 1101, 1035, and 2022, and
Petitioner’s Reply.*

Patent Owner requests exclusion of the above-identified exhibits and the Petitioner’s Reply on the basis that they introduce arguments and evidence outside the proper scope of a reply. Mot. 9–11. The request is denied. 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.¹⁰ 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

¹⁰ *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).

§ 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 are not considered, and the request to exclude portions of Exhibits 1099, 1101, 1035, 2022, and the Petitioner's Reply as outside the scope is denied.

*10. Exhibits 1009 (Shin), 1004 (Poison Java), and 1041
(Brown)*

Patent Owner argues that Shin, Poison Java, and 1041 should be excluded based on lack of authentication, relevance, and hearsay. Mot. 12–15. We deny this request. First, we fail to understand the relevance argument. Patent Owner argues that the date on the face of the documents is irrelevant. Patent Owner mistakes relevancy with how much weight the evidence should be credited with. If a reference, such as Poison Java contains dates and other indicia showing that it was published by a certain timeframe, the fact that those dates appear on the face of the reference is relevant, unless it is untrustworthy. Second, these three references (Shin, Poison Java, and Brown) have been shown to be prior art printed publications through various sources. Shin, for example, has been shown to be a printed publication though the evidence of the archived webpages showing that Shin was available publicly to anyone with Internet access and a search engine. Petitioner did not rely on Shin alone. For Poison Java and Brown, other evidence has been proffered to show their state as prior art printed publications. Again, Petitioner did not rely on dates on the face of the references alone. Notwithstanding the above analysis, we also note that Shin, Poison Java, and Brown are not hearsay, as they are being considered only for what they describe 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).

Finally, Patent Owner's argument that the references have not been authenticated is conclusory and lacks sufficiently persuasive reasoning. As the movant, Patent Owner has the burden of proof to establish that it is entitled to the requested relief. 37 C.F.R. § 42.20(c). This burden cannot be met by conclusory arguments and bare assertions of inadmissibility. The objections to Shin, Poison Java, and Brown are overruled. Patent Owner's request with regard to Exhibits 1009, 1004, and 1041 is denied.

IV. CONCLUSION

For the foregoing reasons, we conclude that Petitioner has failed to show by a preponderance of the evidence that claims 14 and 19 are unpatentable.

We also deny as moot the entirety of Petitioner's Motion to Exclude. Patent Owner's Motion to Exclude is denied as to Exhibits 1005, 1082, 1092, 1095, 1009, 1004, and 1041. The remainder of Patent Owner's Motion to Exclude is either denied as moot or not considered.

V. ORDER

In consideration of the foregoing, it is hereby:

ORDERED that claims 14 and 19 of the '633 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 as to Exhibits 1005, 1082, 1092, 1095, 1009, 1004, and 1041 is

denied, and the remainder of the Motion is denied as moot or not considered;
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.

IPR2015-01974
Patent 7,647,633 B2

PETITIONER:

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

Michael Rosato
Andrew Brown
WILSON SONSINI GOODRICH & ROSATI
mrosato@wsgr.com
asbrown@wsgr.com

PATENT OWNER:

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

Michael Kim
FINJAN INC.
mkim@finjan.com