Trials@uspto.gov
571-272-7822

Paper 41
Entered: March 17, 2017

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

PALO ALTO NETWORKS, INC.,
Petitioner,

v.

FINJAN, INC.,
Patent Owner.

Case IPR2015-02001
Case IPR2016-00157
Patent 8,225,408 B2[1]

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

BOUCHER, *Administrative Patent Judge*.

FINAL WRITTEN DECISION
35 U.S.C. § 318(a) and 37 C.F.R. § 42.108

[1] These proceedings have been consolidated ("the consolidated proceeding"). Cases IPR2016-00955 and IPR2016-00956 have been consolidated and joined with the consolidated proceeding.

Palo Alto Networks, Inc. ("Petitioner") filed two Petitions pursuant to 35 U.S.C. §§ 311–319 to institute *inter partes* reviews challenging claims of U.S. Patent No. 8,225,408 B2 ("the '408 patent"). IPR2015-02001, Paper 2 ("Pet. 2001"); IPR2016-00157, Paper 2 ("Pet. 157"). After consideration of respective Preliminary Responses filed by Finjan, Inc. ("Patent Owner"), the Board consolidated the proceedings and instituted review of claims 1, 3–7, 9, 12–16, 18–23, 29, and 35. Paper 7[2] ("Dec."), 24. Patent Owner's Request for Rehearing of that decision was denied. Paper 13. Subsequent to institution, we granted Motions for Joinder filed by Blue Coat Systems, Inc., who is therefore also a party to this proceeding. Paper 21.

During the trial, Patent Owner timely filed a Response, and Petitioner timely filed a Reply to Patent Owner's Response. Paper 19 ("PO Resp."); Paper 27 ("Reply"). A consolidated oral hearing was held on January 5, 2017, and a copy of the transcript entered into the record. Paper 40 ("Tr."). In addition, both parties filed Motions to Exclude certain evidence, with corresponding oppositions and replies. Papers 31, 32, 34–37.

We have jurisdiction under 35 U.S.C. § 6. This Decision is a Final Written Decision under 35 U.S.C. § 318(a) as to the patentability of the claims on which we instituted trial. Based on the record before us, Petitioner has not shown, by a preponderance of the evidence, that any of claims 1, 3–7, 9, 12–16, 18–23, 29, and 35 is unpatentable.

[2] Unless otherwise indicated, citations are to the record of IPR2015-02001.

I. BACKGROUND

A. *The '408 Patent*

The '408 patent relates to network security, including scanning content that includes "mobile code" to produce a diagnostic analysis of potential exploits, such as viruses, within the code. Ex. 1001, col. 1, ll. 19–20, col. 1, ll. 59–64. Figure 2 of the '408 patent is reproduced below.

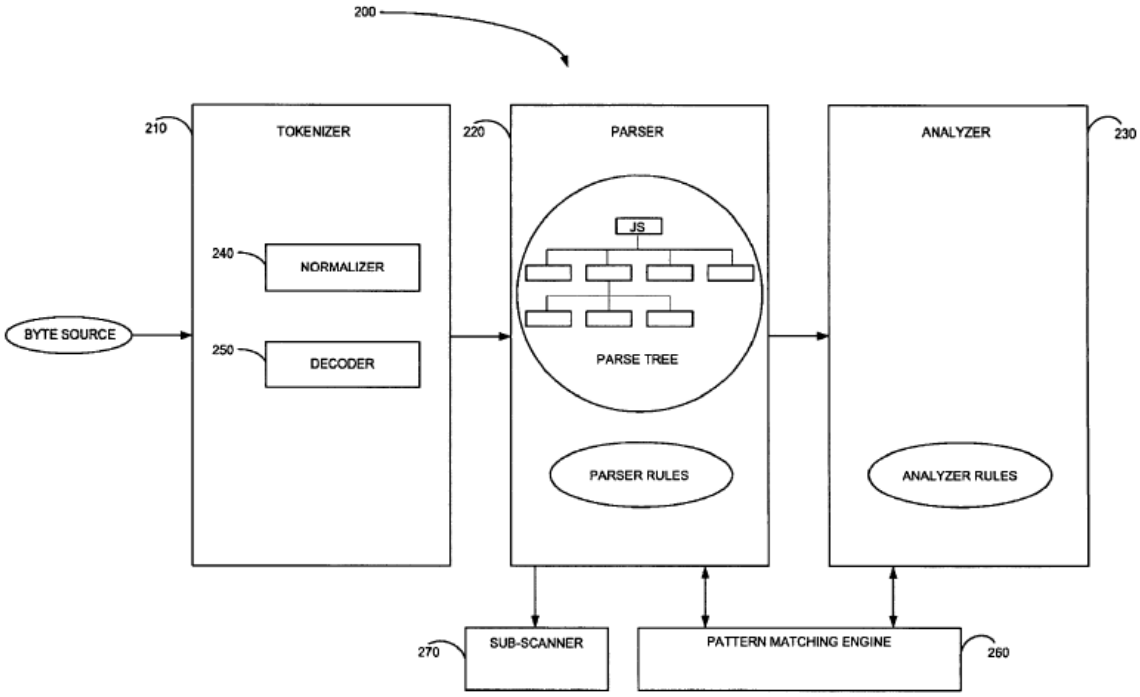

FIG. 2

Figure 2 provides a simplified block diagram of an adaptive rule-based content scanner system. *Id.* at col. 6, ll. 14–17.

The '408 patent explains that the adaptive rule-based scanner of Figure 2 "is preferably designed as a generic architecture that is language-independent, and is customized for a specific language through use of a set of language-specific rules." *Id.* at col. 6, ll. 17–20. In addition, "security

violations, referred to as exploits, are described using a generic syntax, which is also language-independent." *Id.* at col. 6, ll. 28–30. Adaptive rule-based scanner 200 includes three main components: (1) tokenizer 210, which recognizes and identifies constructs (i.e., "tokens") within a byte source code; (2) parser 220, which controls the process of scanning incoming content, such as with a parse tree data structure that represents the incoming content; and (3) analyzer 230, which checks for exploits by searching for specific patterns of content that indicate an exploit. *Id.* at col. 6, ll. 50–54, col. 8, ll. 18–27, col. 9, ll. 19–22. Sub-scanner 270 is another adaptive rule-based scanner used to scan a subsection of input being processed by scanner 200. *Id.* at col. 9, ll. 7–8. Pattern matching engine 260 performs pattern matching for both parser 220 and analyzer 230, such as by accepting an input list of regular-expression elements describing a pattern of interest and an input list of nodes from the parse tree to be matched against the pattern of interest, and outputting a Boolean flag indicating whether a pattern is matched. *Id.* at col. 9, ll. 44–58.

Using a "scanner factory," such adaptive rule-based scanners may be produced "on demand" for different types of content. *Id.* at col. 15, ll. 15–16. The scanner factory "instantiates" a scanner repository, which produces a single instance of multiple scanners, such as "a scanner for HTML content, a scanner for JavaScript content, and a scanner for URI content," each "able to initialize itself and populate itself with the requisite data." *Id.* at col. 15, ll. 31–41. When content is downloaded, a pool of thread objects is created and stores the scanner-factory instance as member data. *Id.* at col. 15, ll. 53–55. When a thread object has content to parse, it requests an appropriate

scanner from its scanner-factory object; when the thread finishes scanning the content, it returns the scanner instance to its scanner factory, "to enable pooling the [adaptive rule-based] scanner for later use." *Id.* at col. 15, ll. 56–63.

B. Illustrative Claim

Independent claim 1 of the '408 patent is illustrative of the claims at issue.

> 1. A computer processor-based multi-lingual method for scanning incoming program code, comprising:
>
> receiving, by a computer, an incoming stream of program code;
>
> determining, by the computer, any specific one of a plurality of programming languages in which the incoming stream is written;
>
> instantiating, by the computer, a scanner for the specific programming language, in response to said determining, the scanner comprising parser rules and analyzer rules for the specific programming language, wherein the parser rules define certain patterns in terms of tokens, tokens being lexical constructs for the specific programming language, and wherein the analyzer rules identify certain combinations of tokens and patterns as being indicators of potential exploits, exploits being portions of program code that are malicious;
>
> identifying, by the computer, individual tokens within the incoming stream;
>
> dynamically building, by the computer while said receiving receives the incoming stream, a parse tree whose nodes represent tokens and patterns in accordance with the parser rules;
>
> dynamically detecting, by the computer while said dynamically building builds the parse tree, combinations of nodes in the parse tree which are indicators of potential exploits, based on the analyzer rules; and

> indicating, by the computer, the presence of potential exploits within the incoming stream, based on said dynamically detecting.

Ex. 1001, col. 19, l. 45–col. 20, l. 7.

C. Instituted Grounds of Unpatentability

Petitioner relies on the following references:

Chandnani	US 7,636,945 B2	Dec. 22, 2009	Ex. 1003
Kolawa	US 5,860,011	Jan. 12, 1999	Ex. 1004
Walls	US 7,284,274 B1	Oct. 16, 2007	Ex. 1005
Huang	US 6,968,539 B1	Nov. 22, 2005	Ex. 1062 in IPR2016-00157

We instituted trial based on 35 U.S.C. § 103(a) over the following combinations of references. Dec. 24.

References	Claims Challenged
Chandnani and Kolawa	1, 3–5, 9, 12–16, 18, 19, 22, 23, 29, and 35
Chandnani, Kolawa, and Walls	1, 3–5, 9, 12–16, 18, 19, 22, 23, 29, and 35
Chandnani, Kolawa, and Huang	6, 7, 20, and 21
Chandnani, Kolawa, Walls, and Huang	6, 7, 20, and 21

In support of these asserted grounds, Petitioner also relies on the testimony of its expert, Aviel D. Rubin, Ph.D. Ex. 1002. In response, Patent Owner relies on the testimony of its expert, Nenad Medvidovic, Ph.D. Ex. 2007. Both experts were cross-examined during the trial, and transcripts of their depositions are in the record. Exs. 2009, 2010 (Rubin depositions); Ex. 1062 (Medvidovic deposition). Patent Owner further relies on the testimony of S. H. Michael Kim and Harry Bims, Ph.D., to support its positions

regarding secondary considerations of obviousness. Exs. 2012, 2013. Both Mr. Kim and Dr. Bims were cross-examined during the trial, and transcripts of their depositions entered into the record. Ex. 1065 (Bims deposition); Ex. 1066 (Kim deposition).

E. Related Proceedings

The parties assert that the '408 patent is the subject of the following district-court proceedings: (1) *Finjan, Inc. v. Palo Alto Networks, Inc.*, No. 3-14-cv-04908 (N.D. Cal.); (2) *Finjan, Inc. v. FireEye, Inc.*, No. 4-13-cv-03113 (N.D. Cal.); (3) *Finjan, Inc. v. Proofpoint, Inc.*, No. 3-13-cv-05808 (N.D. Cal.); (4) *Finjan, Inc. v. Blue Coat Systems, Inc.*, No. 5-15-cv-03295; and (5) *Finjan, Inc. v. Websense, Inc.*, No. 5:13-cv-04398 (N.D. Cal.). Pet. 2001, 2; Paper 5, 1; Pet. 157, 2; Paper 8 (IPR2016-00157), 1.[3]

II. ANALYSIS

A. Claim Construction

In an *inter partes* review, the Board interprets claims of an unexpired patent using the broadest reasonable construction in light of the specification of the patent in which they appear. *See* 37 C.F.R. § 42.100(b); *In re Cuozzo Speed Techs., LLC,* 793 F.3d 1268, at 1278 (Fed. Cir. 2015) ("We conclude that Congress implicitly approved the broadest reasonable interpretation standard in enacting the AIA"), *cert. granted sub nom., Cuozzo Speed*

[3] Petitioner omits identification of the *Websense* matter from the statement in its Petition in IPR2015-02001, and Patent Owner omits identification of the *Websense* matter from the statement in its mandatory notices in IPR2016-00157.

Techs., LLC v. Lee, 84 U.S.L.W. 3218 (U.S. Jan. 15, 2016) (No. 15-446); Office Patent Trial Practice Guide, 77 Fed. Reg. 48,756, 48,766 (Aug. 14, 2012).

In the Institution Decision, we made the following preliminary claim constructions. Dec. 7–12.

Claim Term	Construction
"parse tree"	a hierarchical structure of interconnected nodes built from scanned content
"dynamically building"	requires that a time period for dynamically building overlap with a time period during which the incoming stream is being received
"dynamically detecting"	requires that a time period for dynamically detecting overlap with a time period during which the parse tree is built
"instantiating . . . a scanner for the specific programming language"[4]	substituting specific data, instructions, or both into a generic program unit to make it usable for scanning the specific programming language

We have considered these constructions in light of the full trial record. We see no compelling reason to alter these constructions in light of the parties'

[4] We repeat our observation from the Institution Decision that claim 9 recites "*detecting* any one of a plurality of programming languages in which the incoming stream is written" (emphasis added), but requires instantiating the scanner "in response to said *determining*" (emphasis added), without apparent antecedent basis. We consider "said determining" in claim 9 as intended to refer to the previously recited "detecting," in parallel with the structure of claims 1 and 22. Dec. 11 n.4.

positions as developed during the trial, and we therefore adopt them for this Final Written Decision. *See* PO Resp. 12–13 ("Although Patent Owner maintains that the terms 'dynamically building' and 'dynamically detecting' do not require construction, it does not contest these constructions for purposes of this proceeding, and they are, therefore, controlling."); Reply 3 (noting Patent Owner does not contest the preliminary constructions of "dynamically building" and "dynamically detecting").

B. Asserted Grounds of Unpatentability

1. Legal Principles

A claim is unpatentable under § 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 skill in the art; and (4) objective evidence of non-obviousness, i.e., secondary considerations. *Graham v. John Deere Co.*, 383 U.S. 1, 17–18 (1966).

Additionally, the obviousness inquiry typically requires an analysis of "whether there was an apparent reason to combine the known elements in the fashion claimed by the patent at issue." *KSR*, 550 U.S. at 418 (citing *In re Kahn*, 441 F.3d 977, 988 (Fed. Cir. 2006) (requiring "articulated reasoning with some rational underpinning to support the legal conclusion of

obviousness")); *see In re Warsaw Orthopedic, Inc.*, 832 F.3d 1327, 1333 (Fed. Cir. 2016) (citing *DyStar Textilfarben GmbH & Co. Deutschland KG v. C. H. Patrick Co.*, 464 F.3d 1356, 1360 (Fed. Cir. 2006)).

2. Level of Skill in the Art

Based on its expert's testimony, Petitioner contends that a person of ordinary skill in the art at the time of the claimed invention would have held 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. Pet. 2001, 14 (citing Ex. 1002 ¶ 33). Patent Owner does not dispute Petitioner's contention, and its expert's opinion on the level of ordinary skill is generally similar to that of Petitioner's expert, differing only by about a year in minimum work experience and suggesting an advanced degree as a substitute for work experience. *See* Ex. 2007 ¶ 37. Furthermore, Patent Owner's expert, Dr. Medvidovic, asserts that the opinions stated in his Declaration "would be the same if rendered from the perspective of a person of ordinary skill in the art set out by [Petitioner's expert,] Dr. Rubin." *Id.* at ¶ 40. Based on the expert testimony in the record, we agree with and adopt the level of skill advocated by Petitioner.

3. Scope and Content of the Prior Art

a. Chandnani

Chandnani describes tools "for detecting script language viruses by performing a lexical analysis of a data stream on a computing

device/system." Ex. 1003, col. 4, ll. 32–35. The data stream may be generated from a potentially infected file, which "may be stored on a storage medium, such as hard disk or floppy disk, or received via a network, such as the Internet, or a wired or wireless transmission medium, such as telephone landlines or RF airwaves." *Id.* at col. 4, ll. 36–40. The data stream may be converted to a stream of tokens using lexical analysis, with the tokens corresponding to respective language constructs. *Id.* at col. 3, ll. 65–67. Although further lexical analysis may be performed on the stream of tokens, and is described by Chandnani, the procedure for tokenizing a data stream is of particular relevance to this Final Written Decision. Such a procedure may be understood with reference to both Figures 2 and 6 of Chandnani.

Figure 2 of Chandnani is reproduced below.

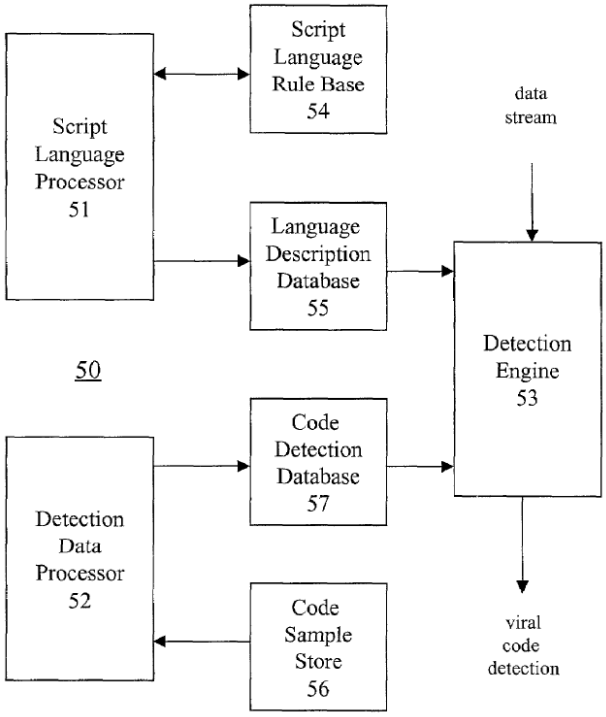

FIG. 2

Figure 2 of Chandnani provides a block diagram of a script language virus detection apparatus. *Id.* at col. 4, ll. 11–13. Figure 6 of Chandnani is reproduced below.

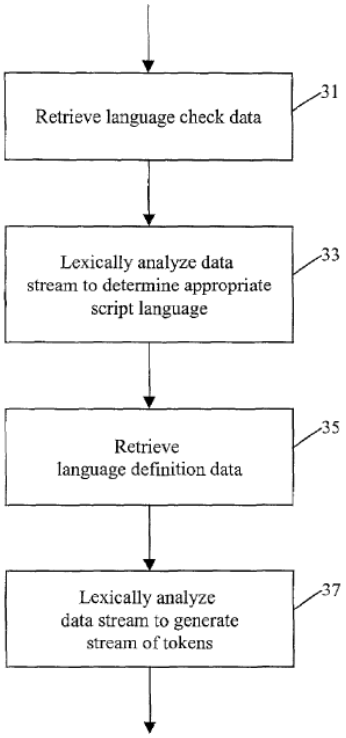

FIG. 6

Figure 6 is a flowchart illustrating a process for generating a stream of tokens. *Id.* at col. 4, ll. 23–25.

The script language virus detection apparatus shown in Figure 2 is organized generally around detection engine 53, which receives a data stream "corresponding to a file to scan." *Id.* at col. 8, l. 4. Language definition rules and language check rules are defined for respective target script languages and stored in rule base 54. *Id.* at col. 5, ll. 41–44. Such rules are processed by script language processor 51 to generate language

description data for respective target script languages, stored in language description data module 55. *Id.* at col. 5, ll. 44–49.

In tokenizing the data stream, detection engine 53 receives the language check data from language description module 55, as indicated at step 31 of Figure 6. *Id.* at col. 7, ll. 61–63. The language check data are used to lexically analyze the data stream at step 33 to determine the appropriate script language. *Id.* at col. 7, ll. 63–65. The language definition data for the script language determined in step 33 are retrieved from language description module 55 at step 35 to perform a second lexical analysis that generates the stream of tokens at step 37. *Id.* at col. 7, l. 67–col. 8, l. 3 ("the data stream is *again* lexically analyzed to generate a stream of tokens") (emphasis added). Chandnani provides the following summary of the tokening procedure to explain the function of a "lexical analyzer" that is not shown in the drawings. We highlight certain portions of the description for emphasis.

> The data stream corresponding to a file to scan is tokenized by lexical analysis. The data stream is fed to a lexical analyzer (not shown) in the detection engine which generates a stream of tokens. To tokenize the data stream, a script language used in the data stream is analyzed using the language check data. *The data stream is analyzed* using the language check data to select the language definition data to use for the detection process. *Next*, the selected language definition data and the data stream are supplied to the lexical analyzer. *The data stream* is lexically analyzed *again*, *this time* using the language definition data, to generate a stream of tokens. As mentioned above, each generated token corresponds to a specific language construct, and may be a corresponding unique number or character.

Id. at col. 8, ll. 4–17 (emphasis added).

b. Kolawa

Kolawa "relates to a method and system for automatically checking computer source code quality." Ex. 1004, col. 1, ll. 19–20. Figure 1 of Kolawa is reproduced below.

FIG. 1

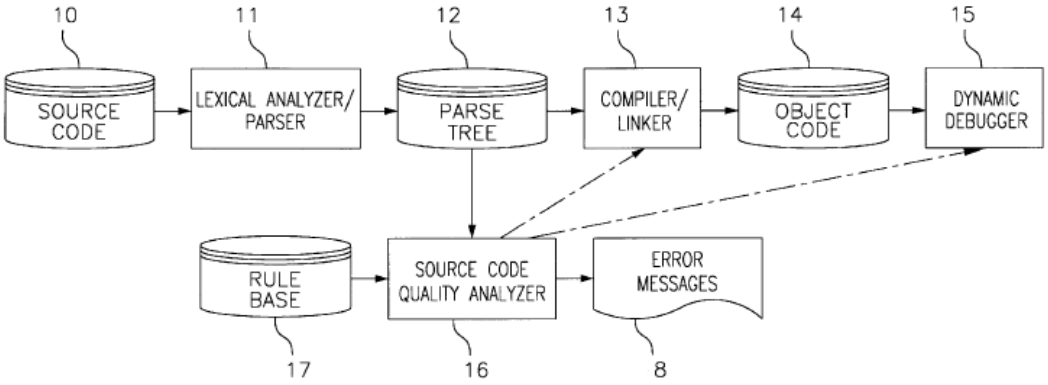

Figure 1 of Kolawa is a functional block diagram of a system for automatically checking computer source code quality. *Id.* at col. 3, ll. 47–50. Source code 10, which contains a series of programming language instructions, is preferably organized into files stored on a secondary device. *Id.* at col. 3, ll. 50–53. Source code 10 is read as input to lexical analyzer/parser 11, which scans source code 10 and groups instructions into tokens. *Id.* at col. 3, l. 66–col. 4, l. 2.

The parser performs a hierarchical analysis that groups the tokens into grammatical phrases represented by parse tree 12, which is preferably organized into files stored also stored on a secondary device. *Id.* at col. 4, ll. 2–6. Parse tree 12 is read by compiler/linker 13, which transforms the parse tree into executable object code 14, and is read by source code quality analyzer 16. *Id.* at col. 4, ll. 14–17. The compiler translates the grammatical

phrases stored in parse tree 12 into individual object modules that the linker links and combines with external library function modules to create object code 14, also preferably organized into files stored on a secondary storage device. *Id.* at col. 4, ll. 17–23. Other aspects of Figure 1 are not relevant to our analysis in this Final Written Decision.

c. Walls

Walls "provides a certification process for application software that enables organizations to deploy software in essential applications with greater confidence." Ex. 1005, col. 6, ll. 62–64. In particular, Walls describes a "pipelined" approach for software certification, with distinct components assembled into a pipeline such that the results of one component are used as input for the next component. *Id.* at col. 7, ll. 3–6. Figure 2 of Walls is reproduced below.

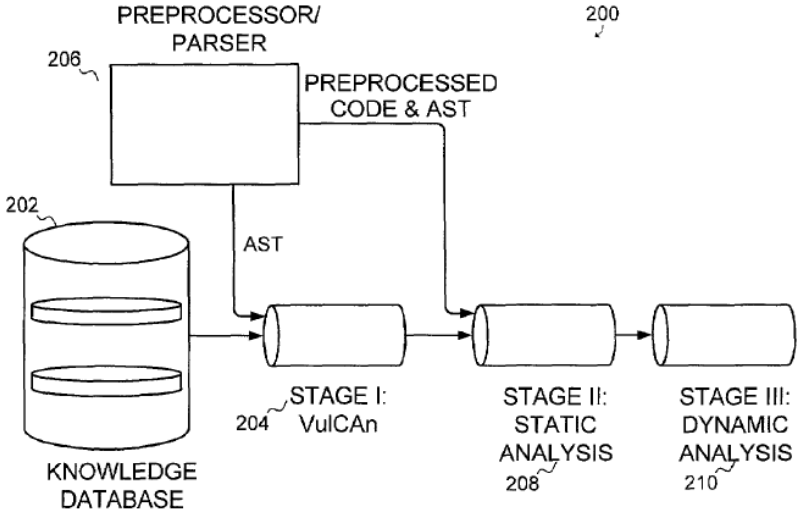

FIG. 2

Figure 2 is a schematic diagram that illustrates the process flow for software analysis and certification. *Id.* at col. 6, ll. 53–55. An "Abstract Syntax

Tree" of code being examined is generated by preprocessor/parser module 206 for input into Stages I and II of pipeline 200. *Id.* at col. 7, ll. 28–31. Knowledge database 202 stores information regarding various fault classes to be scanned for. *Id.* at col. 7, ll. 31–32. Information from knowledge database 202 is fed into Stage I, which uses that input together with the Abstract Syntax Tree, to run a context-sensitive algorithm on the input. *Id.* at col. 7, ll. 32–37. Resulting flagged vulnerabilities are passed to Stage II, which performs static analysis, also using the Abstract Syntax Tree and preprocessed code to obtain various graphs used as a basis for analysis. *Id.* at col. 7, ll. 38–41.

4. Comparison of Claimed Subject Matter and Prior Art

a. Obviousness Over Chandnani and Kolawa

Petitioner challenges claims 1, 3–5, 9, 12–16, 18, 19, 22, 23, 29, and 35 as unpatentable under 35 U.S.C. § 103(a) over Chandnani and Kolawa, addressing the challenged independent claims in IPR2015-02001 and the challenged dependent claims in IPR2016-00157. Pet. 2001, 19–56; Pet. 157, 21–52. In IPR2016-00157, Petitioner additionally challenges claims 6, 7, 20, and 21 as unpatentable over Chandnani, Kolawa, and Huang. Pet. 157, 52–54.

The dispositive issue is whether Petitioner demonstrates, by a preponderance of the evidence, that the combination of Chandnani and Kolawa teaches or suggests "dynamically building" a parse tree "while" receiving an incoming stream of program code. Although the specific language differs in immaterial respects by reciting or omitting the structural

elements of a "computer" or a "receiver," such a limitation appears in each of independent claims 1, 9, 22, 23, 29, and 35 through a combination of recitations that include receipt of the incoming stream of programming code and "dynamically building" the parse tree "while" receiving the incoming stream.

Petitioner contends that all limitations of independent claims 1, 9, 22, 23, 29, and 35 are taught by Chandnani "except for the use of a parse tree to store and analyze programming code," and that "[i]t would have been obvious to combine Chandnani's scanner with the parse-tree teachings of Kolawa" to meet the full set of limitations of each independent claim. Pet. 2001, 19–20. Although the parties agree that Chandnani describes lexical analysis of a "data stream" to generate a stream of tokens, they disagree whether Chandnani discloses that such lexical analysis occurs *while* receiving the incoming data stream, as recited in each of the independent claims. That is, the parties disagree whether Chandnani discloses a time period for generation of the token stream *that overlaps* a time period during which the incoming stream is received, as required by our construction of "dynamically building."

Petitioner contends that "Chandnani generates tokens by examining each character in the data stream, checking for a match against a state transition table and, depending on the result of that comparison, outputting a token—all before moving on to the next character in the data stream." Pet. 2001, 30 (citing Ex. 1003, col. 8, ll. 17–31). From this contention, Petitioner and its expert, Dr. Rubin, reason that "[b]ecause additional characters in the data stream are still queued up for receipt while earlier characters are

identified and stored as tokens, Chandnani creates and stores tokens during the time period during which the incoming data stream is received." *Id.* at 31 (citing Ex. 2003, col. 8, ll. 17–31; Ex. 1002 ¶ 162).

Patent Owner's expert, Dr. Medvidovic, specifically disagrees with this reasoning, and points to Figure 6 of Chandnani, which is reproduced above. Noting Chandnani's explanation that the data stream is lexically analyzed "again," with the language definition data determined, at step 33 of Figure 6 to generate the stream of tokens, Dr. Medvidovic opines that a person of ordinary skill in the art would recognize that each step in Figure 6 "take[s] an input, process[es] the input, and produce[s] an output without interleaving the step with any upstream or downstream tasks." Ex. 2007 ¶ 72 (citing Ex. 1003, col. 7, l. 65–col. 8, l. 3). "Chandnani's process requires that the . . . 'detection engine' lexically analyze the data stream once (at step 33) before tokenizing the data stream at step 37." Ex. 2007 ¶ 73 (citing Ex. 1003, Fig. 6, col. 7, l. 60–col. 8, l. 3). According to Dr. Medvidovic, "[t]he practical implication is that Chandnani's tokenization process does not occur until the data stream is fully resident within the detection engine and so it cannot 'create and store[] tokens during the time period during which the incoming data stream is being received as required under the Board's construction of the term 'dynamically building.'" *Id.*

We have considered the opinions of both experts and the underlying bases for those opinions. In light of specific disclosures in Chandnani, we conclude that Patent Owner's expert, Dr. Medvidovic, articulates the more compelling position. Of particular relevance is the following disclosure from Chandnani:

> The script language virus detection methodologies described herein may be performed by a computer in one or a combination of the following circumstances. The script language virus detection methodologies may be performed periodically (e.g., once/twice per day) or at selected times (e.g., each time the computer is powered up or restarted) by the computer on all relevant electronic files. In addition, *the script language virus detection methodologies may be performed on a file (or a data stream received by the computer through a network)* ***before the file is stored/copied/executed/opened*** *on the computer.*

Ex. 1003, col. 9, ll. 6–16 (emphasis added). In considering this passage, Petitioner's expert, Dr. Rubin, makes the conclusory statement that "[t]hus, under [Patent Owner's] own interpretation of the claim language, the Chandnani+Kolawa combination teaches the 'dynamically building' limitation." Ex. 1002 ¶ 167. We accord such a conclusory assertion little weight. *See Verlander v. Garner*, 348 F.3d 1335, 1371 (Fed. Cir. 2003) (Board has discretion to accord little weight to broad conclusory statements from expert witness).

Context for the passage is provided by other disclosures in Chandnani, such as the following introduction to its discussion of detecting viral code using lexical analysis:

> The file to be scanned, which may be stored on a storage medium, such as hard disk or floppy disk, or received via a network, such as the Internet, or a wired or wireless transmission medium, such as telephone landlines or RF airwaves, is converted to a data stream.

Ex. 1003, col. 7, ll. 30–34. This disclosure makes clear that a "data stream" as used by Chandnani results from conversion of a file, and is not a data stream actively being received over a network—whether the file is stored on

a hard disk, stored on a floppy disk, or was itself received via a network. *See* Tr. 45:12–25. As Dr. Medvidovic explains, a person of skill in the art "would understand that even if the embodiment [] involves 'a data stream received by the computer through a network,' Chandnani would still temporarily store the entire data stream in memory at least between the first and second lexical analyses (steps 33 and 37 of FIG. 6), precluding a result in which the data stream is tokenized while the data stream is being received." Ex. 2007 ¶ 74 (citing Ex. 1003, col. 9, ll. 12–16, col. 7, l. 60–col. 8, l. 17, Fig. 6).

Consequently, Dr. Medvidovic opines, and we agree, that "simply because Chandnani's tokenizer operates on a data stream does not demand or even imply that the data stream is being received while being tokenized." *Id.* As Patent Owner contends, because Chandnani discloses that operation of the tokenizer "only occurs on the ***second pass*** through the data stream, . . . the data stream must have already been received before the stream of tokens is generated." PO Resp. 21. "[W]hile it is conceivable (though not explicitly disclosed) that a data stream might be 'incoming' while Chandnani uses the 'language check data to select the language definition data to use for the detection process,' it is not possible for the data stream to still be 'incoming' when the data stream 'is lexically analyzed again, this time using the language definition data, to generate a stream of tokens.'" *Id.* (citing Ex. 1003, col. 8, ll. 4–17; Ex. 2007 ¶¶ 73–74).

We are persuaded by Patent Owner's arguments. We have considered, but are not persuaded by, Petitioner's counterarguments that Patent Owner "misinterprets Chandnani by incorrectly treating a 'data

stream' as a single discrete object, rather than a byte-by-byte flow of packetized data." Reply 7. Petitioner contends that "Chandnani's disclosure taught or suggested that the packetized data in an incoming stream should be passed from one analytical step to the next without waiting for each step to analyze the entire stream." *Id.* at 8. This contention is premised on a reading of Chandnani that ascribes file analysis to two different embodiments: analysis of a file already resident on a computer and analysis of a file received via a network. *See id.* at 8–9. But to the extent that Chandnani discloses different embodiments directed to the underlying identification of the file to be scanned—whether stored on a hard or floppy disk, or received via a network—Chandnani still requires multiple passes through the file, first to determine the appropriate script language and then to lexically analyze the data stream to generate the stream of tokens. Ex. 1003, Fig. 6; Ex. 2007 ¶¶ 73–74.

Accordingly, we conclude that Petitioner has not demonstrated, by a preponderance of the evidence, that the combination of Chandnani and Kolawa teaches or suggests "dynamically building" a parse tree "while" receiving an incoming stream of program code.

b. Obviousness Over Chandnani, Kolawa, and Walls

Petitioner challenges claims 1, 3–5, 9, 12–16, 18, 19, 22, 23, 29, and 35 as unpatentable under 35 U.S.C. § 103(a) over Chandnani, Kolawa, and Walls, addressing the challenged independent claims in IPR2015-02001 and the challenged dependent claims in IPR2016-00157. Pet. 2001, 56–60; Pet. 157, 54–58. In IPR2016-00157, Petitioner additionally challenges claims 6,

7, 20, and 21 as unpatentable over Chandnani, Kolawa, Walls, and Huang. Pet. 157, 58–59. Petitioner asserts that such challenges provide "an alternative ground for finding that two limitations—the 'dynamically building' and 'dynamically detecting' elements common to every Petitioned Claim—would have been obvious to a [person of ordinary skill in the art] at the time of the alleged invention claimed in the '408 patent." Pet. 2001, 56–57. As explained below, Petitioner's challenges based on the combination of Walls with Chandnani and Kolawa suffer from the same deficiencies as its challenges based on Chandnani and Kolawa alone, in that Petitioner does not sufficiently establish that the prior art it relies on discloses the temporal interleaving required by our construction of "dynamically building."

In addressing these limitations, Petitioner draws a correspondence between the "parse tree" recited in the claims and the "Abstract Syntax Tree" disclosed by Walls, and contends that "Walls explicitly teaches receiving a data stream and building a parse tree in parallel." Pet. 2001, 57, 58. Petitioner relies on testimony by Dr. Rubin to support its assertion that "Walls teaches parsing and analyzing one part of a data stream while other parts of the stream are still being received, as required by the 'dynamically building' limitations" of the challenged claims. *Id.* at 58 (citing, *inter alia*, Ex. 1002 ¶¶ 168–173). Dr. Rubin asserts that a person of ordinary skill in the art "would have understood that there would be additional code information waiting to be parsed . . . in order to keep the pipeline filled. Otherwise, the pipelined architecture would not achieve the goal of both pipelining and Walls: increased parallelism." Ex. 1002 ¶ 172. We find Dr. Rubin's testimony provides insufficient support for Petitioner's assertion

because it expresses a conclusion that is neither a necessary logical consequence of Dr. Rubin's testimony nor consistent with the full record. Specifically, like Chandnani, Walls discloses "a pipelined approach for certifying software wherein distinct components are assembled into a pipeline *such that the results of one component are used as input for the next component*." Ex. 1005, col. 7, ll. 3–9 (emphasis added). As Patent Owner's expert, Dr. Medvidovic explains, "each system component has a defined task, and the output of each task is used as the input of the next task without successive tasks being interleaved together." Ex. 2007 ¶ 97.

Consequently, Dr. Medvidovic further testifies that "like Chandnani and Kolawa, Walls'[s] staged analysis techniques are antithetical to the interleaved 'dynamically building' and 'dynamically detecting' processes disclosed and claimed in the '408 Patent." Ex. 2007 ¶ 93. According to Dr. Medvidovic, "using the results of one component as the input for the next component precludes the temporal overlap required by plain language of the claims and the Board's construction." *Id.*

Patent Owner provides the following characterization of Walls: "Walls'[s] sole disclosure relating to building its [Abstract Syntax Tree] omits all mention of how the tree is built, where the 'code being examined' comes from, and the timing for building the parse tree in relation to obtaining such code, stating only that the [Abstract Syntax Tree] 'is generated.'" PO Resp. 51. Based on our review of Walls, we find this characterization accurate. In an attempt to address the omission of such details in Walls, Petitioner's expert, Dr. Rubin, testifies that "[u]sing the pipelining approach . . . , Walls builds an 'abstract syntax tree' (i.e., a type

of expanded parse tree) from *an already-received code stream* to feed its first pipeline stage . . . even as other upstream portions of code . . . are waiting to be received." Ex. 1002 ¶ 171 (emphasis added). We agree with Patent Owner that "[i]f Walls builds the [Abstract Syntax Tree] 'from an already-received code stream,' that code stream is not 'being received' at a time period that overlaps with building the [Abstract Syntax Tree]." PO Resp. 54. We further agree with Patent Owner that reading the claim term "'incoming code' so broadly as to encompass code received at Walls'[s] preprocessor/parser that is unrelated to and will not become part of the [Abstract Syntax Tree] generated by the preprocessor/parser simply does not meet the construction of the term 'dynamically building.'" *Id.*

Accordingly, we conclude that Petitioner has not demonstrated, by a preponderance of the evidence, that the combination of Chandnani, Kolawa, and Walls teaches or suggests "dynamically building" a parse tree "while" receiving an incoming stream of program code.

5. Conclusion

Having considered the arguments and evidence presented by both parties, we conclude that Petitioner has not shown, by a preponderance of the evidence, that any of challenged claims 1, 9, 22, 23, 29, or 35 would have been obvious over the combination of Chandnani and Kolawa, or would have been obvious over that combination in further combination with Walls. Because the challenges of the dependent claims suffer from the same deficiencies, we further conclude that Petitioner has not shown, by a preponderance of the evidence, that those dependent claims would have been

obvious over the combinations of art it presents. Because we determine that the art does not disclose "dynamically building" a parse tree "while" receiving an incoming stream of program code, we need not consider whether a further conclusion of nonobviousness is warranted because of evidence of secondary considerations.

C. Motions to Exclude

1. Petitioner's Motion to Exclude

Petitioner moves to exclude Exhibit 2006 and paragraphs 14, 15, and 18–23 of Exhibit 2013 (Bims Declaration). Paper 32. We deny the Motion as moot because this Final Written Decision does not rely on that evidence in reaching our conclusion that Petitioner has not met its burden in demonstrating that the challenged claims are unpatentable.

2. Patent Owner's Motion to Exclude

Patent Owner moves to exclude portions of Exhibit 1062 (deposition transcript of Dr. Medvidovic), Exhibit 1063 (Exhibit 4 to the deposition of Dr. Medvidovic), and portions of Exhibit 1065 (deposition transcript of Dr. Bims). Paper 31.

At the oral hearing, we expressed surprise that a party would seek to exclude the cross-examination testimony of its own expert witness. *See* Tr. 49:19–52:8. Patent Owner's counsel explained that it "provided a motion to exclude based on the way that the evidence was used in the reply." Tr. 50:1–6. But the Board has previously explained that "neither a motion to strike nor a motion to exclude is a proper mechanism to present argument

that a reply contains new arguments." *Ultratec, Inc. v. Sorenson Communications, Inc.*, Case IPR2013-00288, Paper 38 at 2 (PTAB May 23, 2014). Rather, an evaluation of the propriety of, and weight to be accorded to, arguments made in a reply is left to the determination of the Board, with parties afforded various opportunities to provide context for the evidence. *See* 37 C.F.R. §§ 42.23, 42.65. We, therefore, consider Patent Owner's arguments as going to the weight that should be given to the cross-examination testimony and related deposition exhibit, not to the admissibility of that evidence. *See Liquid Dynamics Corp. v. Vaughan Co.*, 449 F.3d 1209, 1221 (Fed. Cir. 2006) ("Vaughan's challenge goes to the weight of the evidence rather than the admissibility of Lueptow's testimony and analysis."). As indicated above, we have considered and weighed the testimony provided by Dr. Medvidovic and have not relied on the testimony of Dr. Bims in reaching our decision.

Accordingly, we deny Patent Owner's Motion to Exclude.

III. ORDER

It is

ORDERED that, based on a preponderance of the evidence, claims 1, 3–7, 9, 12–16, 18–23, 29, and 35 of U.S. Patent No. 8,225,408 B2 have not been shown to be unpatentable;

FURTHER ORDERED that Petitioner's Motion to Exclude (Paper 32) is denied as moot;

FURTHER ORDERED that Patent Owner's Motion to Exclude (Paper 31) is denied; and

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

PETITIONER:

Orion Armon
oarmon@cooley.com

Christopher Max Colice
mcolice@cooley.com

Jennifer Volk
jvolkfortier@cooley.com

Brian Eutermoser
beutermoser@cooley.com

Michael Rosato
mrosato@wsgr.com

Andrew Brown
asbrown@wsgr.com

PATENT OWNER:

James Hannah
jhannah@kramerlevin.com

Jeffrey Price
jprice@kramerlevin.com

Michael Kim
mkim@finjan.com