EXHIBIT F

Attorney Docket No. FINREXM0012

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re *Ex Parte* Reexamination of U.S. Patent No. 7,975,305 to Rubin, et al.	Technology Center: 3992
Application No.: 90/013,660	Group Art Unit: 3992
Filed: December 11, 2015	Confirmation No.: 5600
Patent Owner: Finjan, Inc.	CRU Examiner: Majid A. Banankhah

For U.S. Patent No. 7,975,305 – METHOD AND SYSTEM FOR ADAPTIVE RULE-BASED CONTENT SCANNERS FOR DESKTOP COMPUTERS.

Submitted Electronically

Mail Stop *Ex Parte* Reexam
Attn: Central Reexamination Unit
Commissioner for Patents
United States Patent & Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

RESPONSE TO FINAL OFFICE ACTION

Dear Sir:

In response to the pending Office Action dated August 24, 2016, please consider the following remarks. Prior to taking action responsive hereto, the Patent Owner respectfully requests an interview with the Examiner pursuant to the Interview Request and Proposed Agenda filed and faxed on October 21, 2016.

I. OVERVIEW

Patent Owner respectfully requests the Examiner withdraw the Final Office Action (FOA) as improper and confirm patentability of the rejected claims based on a number of errors.

First, in the FOA, the Examiner interprets key elements of the claims in a manner inconsistent with the law. For example, the Examiner improperly cites to extrinsic evidence regarding a non-claim term, "parsing," in order to define the claim term "parser rules" as "rules related to the process of analyzing a string of symbol in computer language [sic]." FOA, pgs. 48–49. Yet, in U.S. Patent No. 7,975,305 ("the '305 Patent") and the claims, parser rules "*describe computer exploits as patterns of types of tokens.*" The Examiner's definition is thus inconsistent with the specification and legally improper. *See Microsoft Corp. v. Proxyconn, Inc.*, 789 F. 3d 1292 (Fed. Cir. 2015) ("Even under the broadest reasonable interpretation, the Board's construction cannot be divorced from the specification and the record evidence and must be consistent with the one that those skilled in the art would reach.") (citations omitted). Here, the Examiner legally erred by using extrinsic evidence for a definition to "parsing," which is ***not*** a term used in the claims – i.e., "parser rules," nor supported in the '305 Patent where parser rules describe computer exploits as patterns of types of tokens.

Second, the Examiner interprets key elements of the claims in a manner inconsistent with the specification of '305 Patent and the reasons for allowance distinguishing over prior art. Specifically, the allowance of application no. 11/009,437 (now the '305 Patent) in December of 2010 is directly tied to *at least* the following pivotal claim language:

> *computer exploits being portions of program code that are malicious, wherein the parser and analyzer rules describe computer exploits as patterns of types of tokens, tokens being program code constructs, and types of tokens comprising a punctuation type, and identifier type and a function type*

See Notice of Allowance, Pages 3-4. Indeed, the Notice of Allowance, with accompany reasons for allowance, was responsive to Patent Owner's detailed arguments filed in September of 2010 wherein Patent Owner stated: "a point of novelty of the claimed invention is <u>describing and recognizing computer exploits from patterns of types of tokens</u>, which is not a known concept." *See* Response to Non-Final Rejection, September 15, 2010, Pages 7-8 (emphasis in original). One of ordinary skill would recognize at least this same point of novelty distinguishes the claims of the '305 Patent over the cited prior art and, in particularly, is clearly absent from Wells, Sandu and any combination thereof. *See* Declaration of Nenad Medvidovic ("Medvidovic Dec.") ¶ 20; *see also* '305 Patent at Col. 1, l. 64 - Col. 2, l. 27.

Patent Owner respectfully submits that the Examiner continues to misinterpret both the claim elements and the cited references, Wells and Sandu, from the vantage of one of ordinary skill. For example, one of ordinary skill would not equate the content pattern recognition language (CPRL) of Wells with the potentially malicious "program code" recited in the '305 Patent claims. This is a critical and fundamental error as explained by one of at least ordinary skill—Dr. Medvidovic. *See* Medvidovic Decl. ¶¶ 29-30, 32, 33. Importantly, Wells' CPRL cannot be program code that includes a computer exploit because the CPRL in Wells is shown as part of the scanner, which scans program code for exploits, and not the program code itself. That is, the CPRL cannot be both the scanner and what is being scanned "program code" rendering the FOA erroneous in asserting obviousness over the '305 Patent. *Id.* ¶¶ 26-40.

Additionally, Sandu's signature generation and matching process does not disclose or suggest the claimed "*parser and analyzer rules*" which "*describe computer exploits as patterns of types of tokens*." Medvidovic Decl. ¶¶ 46-51. The claimed "*parser rules*" cannot be equated with the Examiner's cited portions of Sandu which more accurately overlap with the pre-parser rule steps taken by the Tokenizer/Normalizer/Decoder of the '305 Patent. *See* '305 Patent, Col. 9, l. 5 - Col. 10, l. 44. Further, Sandu is completely devoid of any description of scanning or rules to teach or suggest the claimed "parser and analyzer rules. Sandu's singular action is a static comparison of a generated script signature to known malware signatures; **<u>without identifying any exploits therewithin</u>**. Medvidovic Dec. ¶47, row 26. In contrast, the '305 Patent states that:

> The present invention enables behavior analysis of content. As distinct from prior art approaches that search for byte patterns [like Sandu], the approach of the present invention is to analyze incoming content of its programmatic behavior. Behaviour analysis is an automated process that parses and diagnoses software program, **<u>to determine if such program can carry out an exploit</u>**.

'305 Patent at Col. 1, l. 64 – Col. 2, l. 3 (emphasis added). This feature of the '305 Patent, which is explicitly recited in the claims appears to be ignored by the Examiner in evaluating the claims over the cited prior art.

Third, the Examiner cannot simply summarily dismiss the underlying factual basis of a 37 C.F.R. § 1.132 Declaration without giving some consideration to it. Indeed, Dr. Medvidovic is a renowned expert in the field of computer science and security. His opinions and underlying factual bases, which are presented from the perspective of one of ordinary skill and distinguish the claims over the prior art, are offered as a rebuttal to an obviousness rejection and must be considered. Importantly, Dr. Medvidovic opinions cite to specific teachings underlying the prior art and the '305 Patent. *See, e.g.*, Medvidovic Dec. ¶32 - ¶51 (Tables pointing to specific teachings of the cited prior art and '305 Patent in support and as the underlying basis to his opinions). Also, contrary to the Examiner's implication that Dr. Medvidovic did not "present evidence," his discussion of the meaning of the term "exploit" as understood by one of skill in the art and in the context of the '305 Patent—precisely the type of evidence the Examiner purports to seek—was completely ignored. *See, e.g.*, Medvidovic Decl., ¶¶ 20–22. The Examiner commits ***reversible*** error by ignoring and not considering the underlying basis to Dr. Medvidovic's opinions concerning the cited prior art and '305 Patent. *Ashland Oil, Inc. v. Delta Resins & Refractories, Inc.*, 776 F.2d 281, 294 (Fed. Cir. 1985); *Ex Parte Malone* (BPAI 2009).

Fourth, the Examiner improperly disregards Finjan's objective evidence of nonobviousness, in contravention of the clearly laid out requirements for a proper obviousness analysis. *See, e.g., Wbip, LLC v. Kohler Co.*, 2015-1038 (Fed. Cir. July 19, 2016). It would seem that the Examiner has fallen victim to the hindsight bias trap "develop[ing] a hunch that the claimed invention was obvious, and then construct[ing] a selective version of the facts that confirms that hunch." *In re Cyclobenzaprine Hydrochloride Extended-Release Capsule Patent Litig.*, 676 F.3d 1063, 1079 (Fed. Cir. 2012). The evidence presented in Mr. Kim's declaration supports a strong nexus between the **exact claims at issue** in this reexamination and the licenses. *See* Kim Dec., ¶¶ 6, 7, Exhibits A and B. Importantly, in Exhibits A and B, the '305 Patent was expressly identified and noticed and a claim chart provided to licensees mapping infringement to an accused product, which eventually led to licenses for the '305 Patent. *Id.* ¶¶ 6, 7. Such objective evidence weighs heavily in favor of non-obviousness. This nexus cannot be ignored by the Examiner in determining the patentability of the '305 Patent over the cited art of record and to do so is improper.

For these and further reasons discussed below, the undersigned respectfully submits that this *Ex Parte* Reexamination proceeding is now in condition for confirming the patentability of all of the original claims of the '305 Patent.

II. ARGUMENTS

A. The Underlying Basis Supporting Dr. Medvidovic's Opinion Cannot Be Ignored

Initially, the undersigned wishes to address the Examiner's misinterpretation and *de facto* dismissal of Dr. Medvidovic's Declaration. The Examiner asserts:

> the evidence and arguments presented by Medvidovic fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically point out <u>how the language of the claims</u> patentably distinguishes them from the references.

Final Office Action ("FOA"), pg. 61(D). This assertion is simply false. The Declaration presents paragraph after paragraph and chart after chart which describe the differences between the '305 Patent claim language and the applied art to Sandu and Wells from the perspective of one of ordinary skill. *See, e.g.,* Medvidovic Dec. ¶¶ 21–23, 27–50, including the right-hand column of all charts presented therein. Specifically, Dr. Medvidovic ties his opinions with particularity to the underlying teachings in Sandu and Wells and the specification and claims of the '305 Patent. This type of analysis is precisely what a declaration pursuant to 37 C.F.R § 1.132 is intended to convey and, importantly, is exactly what Dr. Medvidovic does convey in his Declaration.

Moreover, the Examiner's own statement "[w]hile an opinion as to a legal conclusion is not entitled to any weight, ***the underlying basis for the opinion may be persuasive***" supports precisely this use of Dr. Medvidovic's Declaration. *Id.* (*citing In re Chilowsky*, 306 F.2d 908 (CCPA 1962)(emphasis added)). Accordingly, whether or not Dr. Medvidovic provided an opinion that the '305 Patent is not obvious, that opinion does not somehow render all of Dr. Medvidovic's supporting facts and underlying bases moot. The Examiner commits error to baldly summarize and label the numerous factual assertions in Dr. Medvidovic's Declaration as "general allegations." And regardless of how the Examiner wishes to categorize Dr. Medvidovic's statements, it is reversible error to dismiss them out of hand.

Indeed, the Federal Circuit held that "[o]pinion testimony rendered by experts must be given consideration, and while not controlling, generally is entitled to some weight." *Ashland Oil, Inc. v. Delta Resins & Refractories, Inc.*, 776 F.2d 281, 294 (Fed. Cir. 1985). Similarly, the BPAI (predecessor to the PTAB) has been clear on this issue:

> After a prima facie case of obviousness has been made and rebuttal evidence submitted, all the evidence must be considered anew." *In re Eli Lilly & Co.*, 902 F.2d 943, 945 (Fed. Cir. 1990) (citing *In re Piasecki*, 745 F.2d 1468, 1472 (Fed. Cir. 1984)); *Piasecki*, 745 F.2d at 1472 ("Prima facie obviousness is a legal conclusion, not a fact. Facts established by rebuttal evidence must be evaluated along with the facts on which the earlier conclusion was reached, not against the conclusion itself. (internal cites omitted)); *see also* MPEP § 716.01(d).

Ex Parte Malone (BPAI 2009), pg. 4. And the BPAI goes on to hold:

> The Examiner's response to Nykerk Declaration is largely dismissive. ***In fact, even though Appellants' Briefs place extensive reliance on the Nykerk Declaration to overcome the prima facie case, the Examiner's Answer never addresses it in any detail. This is improper.*** Whether the claimed invention would have been obvious cannot be determined without considering evidence attempting to rebut the prima facie case. Manifestly, the Examiner's consideration and treatment of the Nykerk declaration is improper, since the Examiner has not reweighed the entire merits of the matter. Rather, he has dismissed the evidence of nonobviousness in a cursory manner. Since the Examiner did not properly consider the submitted evidence, the rejection cannot be sustained.

Id. at 4-5 (emphasis added). In the FOA, the Examiner commits legal error by ignoring Dr. Medvidovic's Declaration and, in particular, ignoring specific and numerous underlying facts including charts in his Declaration tied directly to teachings in the cited art of record and the '305 Patent. For example, in ¶¶ 32-51, Dr. Medvidovic provides detailed tables tying his opinions to specific teachings in Wells and Sandu and explains how they do not teach the claims on an element by element basis thereby laying out his underlying basis for his opinions. Moreover, Dr. Medvidovic provides a detailed overview of the features of the '305 Patent and the differences between the '305 Patent and Wells and Sandu citing specifically to teachings in the references themselves. Medvidovic Dec. ¶¶ 19-51. With respect to Dr. Medvidovic's Declaration, the Examiner gave no weight to these important facts supporting Dr. Medvidovic's opinions, nor addressed them in the FOA, which renders the present rejection improper.

B. The Examiner's Interpretation of the Claim Term "parser rules" is Incorrect and Contrary to the Law

"[C]laims subject to reexamination will 'be given their broadest reasonable interpretation ***consistent with the specification***.'" *In re Yamamoto*, 740 F.2d 1569 (Fed. Cir. 1984) (emphasis added); MPEP § 2258(I)(G). Under BRI, "claims should always be read in light of the specification and teachings in the underlying patent". *In re Suitco Surface, Inc.*, 603 F.3d 1255, 1260 (Fed. Cir. 2010). "Moreover, when the specification is clear about the scope and content of a claim term, there is no need to turn to extrinsic evidence for claim interpretation." MPEP 2111.01(III) (citing *3M Innovative Props. Co. v. Tredegar Corp.*, 725 F.3d 1315, 1326–28, (Fed. Cir. 2013). Here, the Examiner's interpretation of the claim term "parser rules" is improper because it ignores the specification and teachings of the patent, relies on extrinsic evidence despite the specification being clear about its meaning, and is inconsistent with the specification. The Federal Circuit most recently rejected just such an improper claim interpretation by the Office in *PPC Broadband, Inc. v. Corning Optical Commc'ns RF, LLC*, 815 F.3d 747 (Fed. Circ. 2016):

> The Board seems to have arrived at its construction by referencing the dictionaries cited by the parties and simply selecting the broadest definition therein. And it does appear that among the many definitions contained in the dictionaries of record "in the immediate vicinity of; near" is the broadest. While such an approach may result in the broadest definition, it does not necessarily result in the broadest reasonable definition in light of the specification. The Board's approach in this case fails to account for how the claims themselves and the specification inform the ordinarily skilled artisan as to precisely which ordinary definition the patentee was using.

On remand, the PTAB reversed its decision, concluding in view of the Federal Circuit's claim interpretation "that Corning has not demonstrated by a preponderance of the evidence that claims 10–25 of the '060 patent are unpatentable under § 103(a) over the combination of Matthews and Tatsuzuki." Final Opinion, IPR2013-00342 (Oct. 12, 2016). Patent Owner submits that the Examiner's interpretation of "*parser rules*" is similarly incorrect.

The '305 Patent discloses "parser rules" or "parsing rules" as "patterns of tokens that form syntactical constructs of program code" that "identify groups of tokens as a single pattern." '305 Patent at 2:22–24, 10:53–54. These descriptions are fully consistent with the claim language, which recites "parser and analyzer rules [that] describe computer exploits as patterns of types of tokens." *See id.* at claim 1. Patentee alerted the Examiner of these descriptions throughout the Response to the Non Final Office Action. *See, e.g.*, Response to NFOA, pg. 4 ("patterns of tokens that form syntactical constructs of program code, referred to as parsing rules"); *id.* ("(2) identify groups of tokens as a single pattern (e.g. parser rules that group tokens into phrases)"); *id.* at 14 ("The claimed "*parser rules*" operate on tokens to identify groups of tokens as a single pattern or as claimed to "*describe computer exploits as patterns of types of tokens.*").

Ignoring the intrinsic record evidence, the Examiner first sought out extrinsic evidence in the form of a non-cited definition of a non-claim term, "parsing or syntactic analysis" and then applied that extrinsically based definition to arrive at a construction for the term "parser rule":

> Examiner notes that in the specification of the '305 patent, parsing is referred to its [sic] ordinary meaning in the compiler art *e.g.*, "**Parsing** or **syntactic analysis** is the process of analyzing a string of symbols, either in natural language or in computer languages, conforming to the rules of a formal grammar." The term "parser rules" constitute rules related to the process of analyzing a string of symbol in computer language [sic].

FOA at 48–49 (emphasis in original). The Examiner improperly implies this definition of "parsing or syntactic analysis" comes from the specification of the '305 patent, which is simply untrue. According to a Google Scholar search, this ***unattributed quotation*** appeared first in Montecchi, et al., *Searching in Cooperative Patent Classification: Comparison between keyword and concept-based search,* Advanced Engineering Informatics 27.3 (2013): 335-345, not Patent Owner's specification. In any case, the Examiner's use of this extrinsic evidence is erroneous as a matter of law because it ignores the intrinsic evidence and contradicts the claim language. *Vitronics Corp. v. Conceptronic, Inc.*, 90 F. 3d 1576, 1585 ("However, as we have recently re-emphasized, extrinsic evidence in general, and expert testimony in particular, may be used only to help the court come to the proper understanding of the claims; it may not be used to vary or contradict the claim language. Nor may it contradict the import of other parts of the specification.").

As a result of the Examiner's legally incorrect claim interpretation, the Examiner arrived at a meaning that is inconsistent with the specification and the understanding of a person skilled in the art. The '305 patent never describes parser rules as "rules related to the process of analyzing a string of symbol in computer language" as suggested by the Examiner. Indeed, the Examiner's interpretation is more closely related to the "rule files for a language describe character encodings, ***sequences of characters*** that form lexical constructs of the language, referred to as tokens" than the "***patterns of tokens*** that form syntactical constructs of program code, referred to as parsing rules" described in the '305 Patent. '305 Patent at 2:20–24. The Examiner's interpretation is also inconsistent with the claim language, which requires that "parser… rules describe computer exploits as patterns of types of tokens." *See id.* at claim 1.

Thus, the Examiner's interpretation is inconsistent with the '305 Patent rendering the FOA improper.

C. The Examiner Fails to Support a Prima Facie Case of Obviousness

Here, in reexamination, the Examiner has failed to demonstrate a *prima facie* case of obviousness. *In re: Natural Alternatives, LLC,* Dckt. 2015-1911 (Fed. Cir. August 31, 2016); *Kennametal, Inc. v. Inger-sol Cutting Tool Co.*, 780 F.3d 1376, 1384 (Fed. Cir. 2015) (noting that the Patent Office "bears the initial burden of showing a prima facie case of obviousness"). To support a *prima facie* case of obviousness under the seminal Supreme Court decision in *Graham v. John Deere* 383 U.S. 1 (1966), the Examiner must first make the following factual inquiries: (i) the scope and content of the prior art; (ii) the differences between the prior art and the claims at issue; (iii) the level of ordinary skill in the field of the invention; and (iv) relevant secondary considerations. *KSR Int'l Co. v. Teleflex, Inc.*, 550 U.S. 398, 406 (2007); *Graham*, 383 U.S. at 17–18. Based on these inquiries, claims are only determined to be legally obvious "if the differences between the subject matter sought to be patented 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." 35 U.S.C. § 103(a). Furthermore, as stated in the Manual of Patent Examination Procedure (MPEP) § 2143(A):

> In order to reject a claim based on this rationale, Office personnel must resolve the *Graham* factual inquiries. Then, Office personnel must articulate the following:
>
> - (1) a finding that the prior art included each element claimed, although not necessarily in a single prior art reference, with the only difference between the claimed invention and the prior art being the lack of actual combination of the elements in a single prior art reference;
> - (2) a finding that one of ordinary skill in the art could have combined the elements as claimed by known methods, and that in combination, each element merely performs the same function as it does separately;
> - (3) a finding that one of ordinary skill in the art would have recognized that the results of the combination were predictable; and
> - (4) whatever additional findings based on the *Graham* factual inquiries may be necessary, in view of the facts of the case under consideration, to explain a conclusion of obviousness.

In the FOA, the Examiner does not articulate Graham factor (1) and has effectively dismissed and ignored evidence from the perspective of one of ordinary skill in the art which clearly rebuts any findings under Graham factors (2) and (3).

1. Wells does not Disclose at Least "Tokens," "Types of Tokens," "Patterns of Types of Tokens"

   i. Predicates are not "*Tokens*"

Patent Owner has made every attempt to follow the Examiner's rejection and arguments and it remains quite clear that the Examiner is maintaining his position that the predicates of Wells, which are the "basic roots or components of a CPRL" (Wells, Col. 4, ll. 57-58), can be equated with the claimed "*tokens*" "that form syntactical constructs of program code." '305 Patent, Col. 2, l, 23. As explained by Dr. Medvidovic, Predicates cannot be equated with "*tokens*." Medvidovic Dec. ¶¶ 29-30, 36-37.

In equating Wells' predicates with the "tokens" described and claimed in the '305 Patent, the Examiner exhibits a fundamental misunderstanding of the meaning of the term "token." As defined in the '305 patent, "tokens" are "sequences of characters that form lexical constructs of the language." '305 Patent at 2:21–22. This definition is consistent with the way the term is used in the claims. *See, e.g., id.* at claim 1 ("tokens being program code constructs"). In stark contrast, Wells discloses that a predicate is an element of the CPRL language that "is compiled into a byte stream that controls a logic" of a processor by performing functions associated with the predicate. Response to NFOA, pg. 8 (citing Wells at 5:8–11). That is, Wells' predicates are not "lexical constructs of the [CPRL] language," but rather "basic roots or components of a CPRL" that indicate how incoming network traffic is to be processed." Wells at 4:55–58; 5:8–11.

Moreover, on the bottom of page 53 of the FOA, the Examiner appears to argue that because predicates could be interpreted as program code constructs of the CPRL that this somehow makes them read on the claimed "*tokens*"—it does not. The claimed "*tokens*" are unique to the "*program code*" that is being scanned for "*computer exploits*." The CPRL is never scanned for "*computer exploits*," as the CPRL is, in effect, performing the scanning. The Examiner's statement on page 54 that "[t]he PO however does not make any argument as to why a function that is part of a program code (programming code) is not scanned in Wells or cannot be scanned for that matter" is not well-received. First, the burden is on the Examiner to present a *prima facie* case of unpatentability. Second, this statement is simply not true. Finjan has clearly and unequivocally argued that the CPRL is not scanned. *See* Medvidovic Dec. in at least ¶¶ 29, 36, 37. Simply put, on of ordinary skill would not equate CPRL as "program code" that contains in the claims of the '305 Patent—as would be required under the Examiner's obviousness theory—because the CPRL is used to examined the incoming program code for potential computer exploits and is ***not*** the incoming program code itself. Medvidovic Dec. ¶¶ 29–30.

ii. The Examiner Improperly Conflates Tokens and Parser Rules

From pages 48-52[1] of the Final Office Action ("FOA"), initially it seems that the Examiner argues, *inter alia*, that **predicates** = **parser rules**. *See* FOA, Page 49, ¶1 ("Wells clearly discloses a 'parser rule' when discloses an 'A' predicate that parses the buffer holding the content and tests for the presence of a <u>particular string passed as an argument to the predicate</u>.") (emphasis in original). But then on page 50, the Examiner contradicts his definition for parser rule and argues, "<u>Wells' teaches parser rules is used for determining type and form of predicates</u>, and not the same as predicate, since based on the parser rule the type of predicate is determined as explained." *See* FOA, Page 50, 2nd ¶ (emphasis in original). Notably, the Examiner offers no citation from Wells to support this latter assertion. And Finjan fails to see how these disparate statements by the Examiner can be reconciled. Is a predicate a parser rule, or is a parser rule used to determine the type of predicate?

Confusing the issue, the Examiner also states, "[m]oreover, to compile signatures into instructions for detecting malware, Wells teaches <u>parser rules to verify the logical elements (*e.g.*, predicates) making up the signature</u>. *See* FOA, Page 50, 3rd ¶ (emphasis in original). The Examiner's statement "[a]ccordingly, much like the '305 patent, Wells teaches parser rules for compiling and determining the validity of the CPRL signatures" (FOA, Page 50, ¶3) is not even remotely close to being an accurate comparison. The '305 Patent, and specifically the claims at issue, are most certainly not directed to "compiling and determining the validity of the CPRL signatures." The '305 Patent scans incoming content for the presence of potential computer exploits, i.e., "patterns of types of tokens, tokens being program code constructs, … ." CPRL signatures are not incoming content and CPRL signatures are not scanned for computer exploits. On the contrary, CPRL signatures actually facilitate the scanning of network traffic content; they aren't the content being scanned. *See* Wells, Col. 6, l. 53-Col. 7, l. 37. The Examiner admits as much when he equates **CPRL** = **analyzer rules**. *See* FOA, Page 49, ¶1 ("Wells also discloses

[1] Pages 3-47 of the FOA are a duplicate of the rejection in the NFOA.

that that the malicious content, *e.g.*, 'computer exploit' is detected using content pattern recognition language (CPRL) (*e.g.*, analyzer rules)").

Thus, the Examiner essentially argues that (1) individual predicates, such as the 'A' predicate, qualify as "parser rules," (2) some unspecified set of "rules" for choosing between available types of predicates qualify as "parser rules," and (3) some unspecified set of "rules" to verify predicates making up a CPRL signature qualify as "parser rules." However, the identification of individual predicates as "parser rules" irreconcilably conflicts with the Examiner's identification of predicates as "*tokens*," and the Examiner never even attempts explain how these alleged "*parser rules*" "describe computer exploits as patterns of types of tokens," as explicitly recited in the claims of the '305 Patent.

Not until page 51, does the Examiner attempt to address where Wells discloses the claimed "*tokens,*" "*types of tokens*" and "*patterns of types of tokens*" of the "*program code*" that is to be scanned using the aforementioned CPRL. But the Examiner appears to conflate parser rules and tokens arguing, "Wells' teaches CPRL based signatures and <u>parser rules which employ the use of various predicates. These predicates represent tokens</u>, which as the '305 claims specify are 'program code constructs.'" *See* FOA, Page 51, ¶1 (emphasis in original). And as discussed previously, the Examiner also states: "Wells clearly discloses a 'parser rule' when discloses an 'A' predicate that parses the buffer holding the content and tests for the presence of a <u>particular string passed as an argument to the predicate</u>." *See* FOA, Page 49, ¶1 (emphasis in original). The predicates cannot be both the claimed parser rules and the claimed tokens.

As claimed, part of the scanning of the *incoming content* is the application of *parser rules* thereto in order to identify the *exploits, exploits* being described both as portions of *program code* that are malicious and as patterns of types of *tokens*. Under the Examiner's interpretation, the predicates (e.g., parser rules) of Wells are applied to the predicates (e.g., tokens) of Wells, which simply doesn't make sense. In particular, and as noted above in Section II.B, the Examiner's rejection relies upon an interpretation of the term "parser rules" that contradicts both the claims and the specification of the '305 Patent. This contradiction was identified and discussed extensively in the expert declaration of Dr. Nenad Medvidovic in <u>at least</u> ¶¶29, 36:

> A predicate is, therefore, not a "token" or even a "type of token" as described and claimed in the '305 Patent because it is not a "program code construct," where the program code is the code in which potential computer exploits are found. *See* '305 Patent, independent claims 1 and 13 ("computer exploits being portions of program

> code that are malicious… tokens being program code constructs"). That is, while tokens are constructs that <u>make up</u> the program code being scanned for potential computer exploits, predicates describe functions in a wholly separate language that <u>are performed on</u> program code being scanned. Put yet another way, "predicates are the basic roots or components of a <u>CPRL</u>," not program code that contains potential computer exploits as claimed in the '305 Patent. Wells at 4:55–58 (emphasis added).
>
> I also disagree that individual predicates correspond to types of tokens. As noted above, predicates are not tokens or types of tokens. While a particular predicate might ***execute*** a macro or process, it is not itself function type of token. Similarly, a predicate that can test content script for characters is not itself a type of character. Nor is a predicate that is ***represented by*** a letter or punctuation mark, or that includes a letter or punctuation mark, a type of token. *See* Office Action at 16. Rather a predicate "represents a function to be performed by the processor." Wells at claim 1.

(emphasis in original). The claimed "*tokens*" are "*program code constructs*" which means they make up the code that is being scanned. *See* Medvidovic Dec. ¶37, row 12.

Further, since predicates are not "*tokens*" it matters not that predicates may have multiple forms. *See* FOA, Page 51, ¶2-Page 52, ¶1. Nor does it matter that a CPRL signature may be comprised of multiple predicates. *See* FOA, Page 52, ¶2. The Examiner's careful deconstruction and description of the predicates exemplified in FIG. 5 of Wells is simply irrelevant to the invention and claims of the '305 Patent because, *inter alia*, "FIG. 5 is a table showing examples of predicates that can be used to create a signature of content desired to be detected. Column **502** shows identifications of ***predicates that are the basic roots or components of a CPRL***." *See* Wells, Col. 4, ll. 55-58 (emphasis added). CPRL and CPRL signatures are not the claimed "*program code*." And as explained by Dr. Medvidovic:

> This portion of Wells describes "predicates, which "can be used to create a signature of content desired to be detected." Wells at 4:55–58. Predicates are not tokens. As described in the specification of the '305 Patent, as claimed in independent claims 1 and 13, and as understood by a person of ordinary skill in the art, a token is "a program code construct." In the context of the '305 Patent, a person of skill in the art would understand the "program code" to be the code of the "incoming content" that is selectively diverted to the "rule-based content scanner." It is this "program code" that includes computer exploits, "portions of program code that are malicious." On the other hand, the predicates disclosed in Wells are the "basic roots or components of a CPRL" (i.e. a content pattern recognition language) that is used to scan network traffic content, but which is otherwise unrelated to that content.

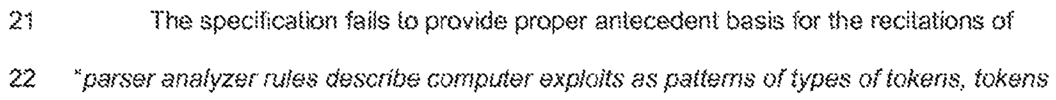
The claimed "*tokens,*" "*types of tokens,*" and "*patterns of types of tokens*" of the "*program code*" are a critical component of the claims as stated by Finjan during the original prosecution: "a point of novelty of the claimed invention is <u>describing and recognizing computer exploits from patterns of types of tokens</u>, which is not a known concept." *See* Response to Non-Final Rejection September 15, 2010, Pages 7-8 (emphasis in original). It is this same point of novelty that is clearly absent from Wells as discussed herein and in Finjan's response to non-final rejection.

It is most respectfully submitted that the Examiner has not adequately addressed these fundamental flaws previously identified in the rejection. The predicates of Wells, which are components of the CPRL signatures, which are a component of the scanner simply cannot also be the claimed "*tokens,*" "*types of tokens,*" "*patterns of types of tokens*" of the "*program code*" that is being scanned for "*computer exploits*". The Examiner cannot point to any disclosure in Wells that describes scanning the CPRL, the CPRL signatures or any portions thereof for computer exploits. Since predicates are unique to CPRL and indeed are the "basic roots or components of a CPRL" it necessarily follows that the predicates of Wells cannot read on the claimed "*tokens*" wherein "*computer exploits*" are defined "*as patterns of types of tokens.*" The fact of the matter is that Wells does not disclose certain claim elements that are critical to patentability. Accordingly, under *Graham* inquiries (i) and (ii), Wells does not contain a description of the claimed "*tokens*" and the Examiner cannot articulate the requisite finding that the prior art included each element claimed, although not necessarily in a single prior art reference, with the only difference between the claimed invention and the prior art being the lack of actual combination of the elements in a single prior art reference.

iii. Wells Does Not Disclose *Patterns of Types of* [Predicates]

Even if we assume, *arguendo*, that Wells' predicates can somehow be read on the claimed "*tokens,*" which they do not, this is not the end of the inquiry. The claims also require that "*parser and analyzer rules describe computer exploits as patterns of types of tokens*." Patent Owner emphasizes here that the FOA is internally inconsistent under any of the Examiner's theories as to how Wells allegedly teaches the claimed "parser rules." First, if the Examiner insists that "Wells clearly discloses a 'parser rule' when [sic] discloses an 'A' predicate that parses the buffer holding the content and tests for the presence of a <u>particular string passed as an argument to the predicate</u>," and that "predicates" are "*tokens*," this argument necessarily fails because the "A" predicate would correspond (under the Examiner's own reasoning) to a "token," not a pattern of types of tokens as required by the claims. *See* FOA pg. 49. Second, if the Examiner insists that Wells teaches "parser rules to parse both the predicates and analyzer rules for the purpose of compiling the signatures into instructions," this argument also fails simply because any such rules—which are never actually disclosed in Wells—are not patterns of types of tokens or even patterns of types of *predicates* (under the Examiner's reasoning). Nor does the Examiner even allege that these alleged "parser rules to parse both the predicates and analyzer rules" are patterns of types of tokens, as explicitly required by the claims. These inconsistencies render the FOA improper.

Moreover, it is most respectfully submitted that the interpretation of the claimed "*patterns of types of tokens*" has been clearly established in the underlying record and the Examiner is precluded from altering the interpretation during reexamination.[2] More particularly, during prosecution of the '305 Patent, the Examiner objected to this language asserting in the Non-Final Rejection mailed June 15, 2010:


The specification fails to provide proper antecedent basis for the recitations of "*parser analyzer rules describe computer exploits as patterns of types of tokens, tokens*

[2] *Tempo Lighting, Inc. v. TIVOLI, LLC*, 742 F. 3d 973 (Federal Circuit 2014) (This court also observes that the PTO is under no obligation to accept a claim construction proffered as a prosecution history disclaimer, which generally only binds the patent owner. However, in this instance, the PTO itself requested Tivoli rewrite the "non-photoluminescent" limitation in positive terms. Tivoli complied, and then supplied clarification about the meaning of the "inert to light" limitation. J.A. 1216.)

being program code constructs" (e.g. see claim 1 and as similarly recited within independent claims 13 and 25.

For example, the examiner notes that while the applicant appears to have support for parser rules for defining and identifying "tokens" or sequences of characters within a language and for analyzer rules for identifying the existence of patterns of tokens (e.g. Specification, par. 53, 54, 63-65, appendix A), there is no support for the present language of "patterns of types of tokens". The examiner respectfully points out that language parsing and analyzing are basic and well known concepts within the art, involving the parsing of character sequences into individual tokens and the analysis of the token combinations or patterns for their meaning. It is respectfully noted that there appears to be no support, nor reason for the applicant's present recitations. For the purpose of examination, the examiner interprets such recitations as referencing the parsing rules for parsing of data into tokens and analysis rules for analyzing the meaning of patterns of tokens, according to the known meaning by those of ordinary skill in the art.

See Pages 2-3. The Examiner also rejected the claims under 35 USC 112, second paragraph as follows:

Regarding claims 1 – 25, the examiner notes that they comprise recitations of "patterns of types of tokens". Such recitations depart from the recitations found within the applicant's disclosure and are not standard among those of ordinary skill in the art. Furthermore, in argument for such recitations, the applicant points only to portions of the specification describing what is standard and known prior art teaching for parsing and analyzing language according to parsing rules and analyzing rules. Thus, the examiner notes that such recitations as they are distinctly recited render the scope of the claims unclear. For the purpose of examination the examiner interprets such recitations as referencing the parsing rules for parsing of data into tokens (i.e. sequences of characters) and analysis rules for analyzing the meaning of patterns of tokens - such as disclosed by the applicant.

Responsive to these objections and rejections, Finjan both amended the claims and provided detailed arguments regarding the construction of "*patterns of types of tokens*" stating:

Specification

On pages 2 and 3 of the Office Action, the Examiner has objected to the specification as failing to provide proper antecedent basis for the claimed subject matter. Specifically, the Examiner has indicated that there is no support for "*patterns of types of tokens*".

Applicants note that the appendix to the specification discloses that tokens are characterized into types. Thus, as defined on page 46,

IDENT "[A-Za-z[!underscore!][!dollarsign!]] [A-Za-z0-9[!underscore!][!dollarsign!]]*",

a token consisting of a character a-z or a character A-Z or an underscore or a dollar sign, followed by zero or more of a character a-z or a character A-Z or a number 0 – 9 or an underscore or a dollar sign, is of type IDENT. Similarly, as defined on page 47,

INTEGER_DECIMAL "[0-9]+",

a token consisting of one or more of the numbers 0 – 9, is of type INTEGER_DECIMAL; and

INTEGER_HEX "0[xX][0-9A-Fa-f]+",

a token consisting of 0x or 0X followed by one or more of the numbers 0 - 9 or the characters A-F or the characters a-f, is of type INTEGER_HEX.

Applicants respectfully submit that patterns of types of tokens appear throughout the specification. Inter alia, at par. [0067], the specification recites

A parse tree … uses parsing rules to identify groups of tokens as a single pattern.

Further, at par. [0085], the specification recites

For example, if a pattern "(IDENT) EQUALS NUMBER" is matched … if a matched pattern is "(1 2 3) 4 5" …

Further, at par. [0086], the specification recites

Reference is now made to FIG. 5, which is an illustration of a simple finite state machine … for a pattern
(IDENT) <val=="foo" & match(*):Rule1> ! <val=="bar"> EQUALS NUMBER.
Specifically, the pattern of interest specifies either an IDENT token with value "foo" and that matches Rule1, or a List with value "bar", followed by an EQUALS token and a NUMBER token.

> Further, at par. [0094] the specification recites
>
> > For example, the pattern in the rule for FuncSig
> > (FUNCTION) (IDENT?) (List)
> > describes a keyword "function", followed by zero or one IDENT tokens, and followed by a "List". In turn, the pattern in the rule for List
> > (LPAREN) ((Expr (COMMA Expr)*)? (RPAREN)
> > describes an LPAREN token and an RPAREN token surrounding a list of zero or more Expr's separated by COMMA tokens.
>
> Further, at par. [0098], the specification recites
>
> > Referring back to the example above, the pattern
> > (IDENT) ASSIGNMENT IDENT <val=="screen"> DOT IDENT <val=="width">
> > within the rule for ScrWidAssign describes a five-token pattern; namely (i) an IDENT token, followed by (ii) an ASSIGNMENT token, followed by (iii) an IDENT token that has a value equal to "screen", followed by (iv) a DOT token, followed by (v) an IDENT token that has a value equal to "width". Such a pattern … corresponds to the example exploit listed above …
>
> Clearly items (i) – (v) above form a pattern of token types IDENT ASSIGNMENT IDENT DOT IDENT.

See Response to Non-Final Rejection September 15, 2010, Pages 6-7 (emphasis in original). And further arguing:

> On page 9 of the Office Action, the Examiner has indicated that Freund teaches parsing data into recognizable tokens, wherein the tokens are not the same tokens and are distinct from one another. The Examiner is citing "*tokens*" in rejecting the claim limitations of "*patterns of types of tokens*". Applicants wish to point out that the phrases "tokens" and "patterns of types of tokens" have different meanings. In particular, as used in the subject specification, "types of tokens" refers to a categorization of tokens into types. A "type" is a category. For example, the constructs APPLET, OBJECT, EMBED, SCRIPT, HREF and IMAGE are distinct tokens; yet they are all of the same type IDENT. Similarly, the constructs 0x01, 0XC33, 0xGB and 0X24AD3 are distinct tokens; yet they are all of the same type INTEGER_HEX.
>
> Types of tokens disclosed in the subject specification include inter alia identifier tokens (say, type TYPE1), assignment tokens (say, type TYPE2), and punctuation tokens (say, type TYPE3). A pattern of types of tokens is, e.g., a pattern TYPE1 TYPE2 TYPE1 TYPE3 TYPE1; meaning, a token of type TYPE1 followed by a token of type TYPE2 followed by a token of type TYPE1 followed by a token of type TYPE3 followed by a token of type TYPE1; e.g., an identifier token followed by an assignment token followed by an identifier token followed by a punctuation token followed by an identifier token.

Directly responsive to these arguments and amendments, the Examiner allowed, *inter alia*, claims 1, 2, 5 and 13 stating in the Notice of Allowance:

> The following is an examiner's statement of reasons for allowance:
>
> The prior art fails to disclose the features, as found recited in combination with remaining claim limitations, of "*scanning, by the computer, the selectively diverted incoming content to recognize potential computer exploits therewithin, based on a database of parser and analyzer rules corresponding to computer exploits, computer exploits being portions of program code that are malicious, wherein the parser and analyzer rules describe computer exploits as patterns of types of tokens, tokens being program code constructs, and types of tokens comprising a punctuation type, an identifier type and a function type*".

See Notice of Allowance, Pages 3-4. As described in the '305 specification and further explained by Dr. Medvidovic, the claimed "*patterns of types of tokens*" represent the distinguishing difference between "simply recognizing previously known malware" using conventional signature based anti-virus detection and the ability to analyze incoming content in terms of its programmatic behavior. Per Dr. Medvidovic:

> 20. The '305 Patent explicitly states that the claimed invention is "distinct from prior art approaches that search for byte patterns" because the approach of the present invention is to analyze incoming content in terms of its programmatic behavior." *See* '305 Patent at 1:64–2:3; *see also id.* at 7:3–10 (disclosing the benefits of behavioral analysis over conventional signature based anti-virus detection). As opposed to byte patterns, this behavioral analysis is tied to the inventors' insight to describe computer exploits as patterns of types of tokens:
>
>> The present invention also utilizes a novel description language for efficiently describing exploits. This description language enables an engineer to describe exploits as logical combinations of patterns of tokens.
>>
>> Thus it may be appreciated that the present invention is able to diagnose incoming content for malicious behavior. As such, the present invention achieves very accurate blocking of content, with minimal over-blocking as compared with prior art scanning technologies.
>
> '305 Patent at 2:28–36. Describing exploits as patterns of types of tokens enables detection of exploits based on what a program does rather than the how underlying code is structured and written.

Medvidovic Dec. ¶ 20. In this reexamination, the Examiner, however, takes an interpretation of "*patterns of types of tokens*" inconsistent with the '305 patent and its prosecution history, which is error as a matter of law. *In re Am. Acad. of Sci. Tech. Ctr.*, 367 F.3d 1359, 1364 (Fed.Cir.2004) ("claims ... are to be given their broadest reasonable interpretation consistent with the specification, and ... claim language should be read in light of the specification as it would be interpreted by one of ordinary skill in the art." *In re Bond*, 910 F.2d 831, 833 (Fed. Cir. 1990); *accord Bass*, 314 F.3d at 577 (" [T]he PTO must apply the broadest reasonable meaning to the claim language, taking into account any definitions presented in the specification.")).

The following table is provided to highlight examples of the claimed "*patterns of types of tokens*" from the specification which support programmatic behavior analysis of incoming content compared to what the Examiner is improperly suggesting is the interpretation of the claimed "*patterns of types of tokens.*" In the '305 Patent examples, there are descriptions of patterns of types of tokens which, if found as patterned in incoming code, may indicate a behavioral exploit in incoming code. Whereas the sole example cited by the Examiner from Wells is simply a program in CPRL language which is run by a processor to operate on incoming byte streams to identify predetermined, static virus signatures. Medvidovic Dec. ¶ ¶ 28, 29. The two approaches are profoundly different and would not be adopted by one of ordinary skill as the same.

'305 Patent: "*patterns of types of tokens*"	Wells

For example, the pattern in the rule for FuncSig
(FUNCTION)(IDENT?)(List)
describes a keyword "function", followed by zero or one IDENT token, and followed by a "List". In turn, the pattern in the rule for List
(LPAREN)((Expr)(COMMA Expr)*)(RPAREN)
describes a LPAREN token and a RPAREN token surrounding a list of zero or more Expr's separated by COMMA tokens. In turn, the pattern in the rule for Expr
	S(68,SEEK_END) // Seek 68 (44h) bytes from eof
	T(OR, 0E, 8C) // Test first byte for 0E or 8C (increments sig ptr)
	R(0) // Reset pointer to first byte
	I(0E, L1) // IF 0E go to label L1 (test does not increment sig ptr)
	L(8C, CB) // Else test for CB [could skip(1) and use L(CB)]
	G(L2) // go to label L2
	L1: // do if first byte was 0E
	W(5, 1F) // search from sig base to locate 0x1F
	L2: // do for either match at start
	R(0) // reset under both conditions (0E or 8C)
	W(24, 60) // search 24d bytes for 60h)
	L(09) // test byte for 09h
	
	As can be seen from Wells Table above, the CPRL language is used to examine program code. It does not describe program code from incoming content, or computer exploits as program code constructs of patterns of types of tokens. In contrast to the examples in the '305 patent to the left, the patent describes examples of tokens and patterns of types of tokens that program code from incoming content may contain.

state machine for the pattern. Reference is now made to FIG. 5, which is an illustration of a simple finite state machine, used in accordance with a preferred embodiment of the present invention, for a pattern, (IDENT<val=="foo" & match(*): Rule1>|List<val=="bar">) EQUALS NUMBER Specifically, the pattern of interest specifies either an IDENT token with value "foo" and that matches Rule1, or a List with value "bar", followed by an EQUALS token and a NUMBER token.	
Referring back to the example above, the pattern (IDENT) ASSIGNMENT IDENT<val=="screen">DOT IDENT<val=="width"> within the rule for ScrWidAssign describes a five-token pattern; namely, (i) an IDENT token, followed by (ii) an ASSIGNMENT token, followed by (iii) an IDENT token that has a value equal to "screen", followed by (iv) a DOT token, and followed by (v) an IDENT token that has a value equal to "width". Preferably, the value of an IDENT (i.e., an identifier) is its name; thus such a pattern indicates use of a member reference "screen.width" within an assignment statement, and corresponds to the example exploit listed above in the discussion of FIG. 1. For example, it corresponds to an assignment of the form w=screen.width The action @($(1).val).attr+=ATTR_SCRWID within the ScrWidAssign rule assigns the attribute ATTR_SCRWID to the symbol table entry whose name is the value of the IDENT token on the left side of the pattern. Specifically, for the example above the attribute ATTR_SCRWID is assigned to the symbol table entry for w.	

Similarly, the pattern LPAREN Expr COMMA Expr COMMA Expr<attr?=ATTR_SCRWID>COMMA Expr<attr?=ATTR_SCRHGT>; within the rule for ScrWidHgtList identifies an eight-token pattern; namely, (i) an LPAREN token (i.e., a left parenthesis), followed by (ii) an expression Expr, followed by (iii) a COMMA token (i.e., a comma), followed by (iv) another Expr, followed by (v) another COMMA token, followed by (vi) an Expr with attribute equal to ATTR_SCRWID, followed by (vii) another COMMA token, and followed by (viii) an Expr with attribute equal to ATTR_SCRHGT). Such a pattern includes inter alia any pattern op.show(0,0, w, h, document.body)	

As discussed above in Section II.C.1.i, it has been clearly shown that predicates, which the Examiner has submitted are the claimed "*tokens*" are not constructs of the incoming content, but instead are constructs of the CPRL which is **never** scanned for "*computer exploits.*" This ***error is fatal*** and the FOA should be withdrawn accordingly.

As such, the entirety of the Examiner's arguments from pages 51-53 addressing the form, syntax, arguments and families of predicates is inapplicable to the claim language which requires that the actual content being scanned, i.e., the incoming content, be reviewed for "*patterns of types of tokens*" (which identify "*computer exploits*" within the incoming content). Predicates are not program code constructs of the incoming code in Wells. Ergo, the predicates are irrelevant to the claim language. The Examiner's attention is respectfully directed to the following portions of the declaration of Dr. Medvidovic found in the table of paragraph 37:

- ***Given that predicates are not tokens, tokens being program code constructs, Wells' CPRL signatures cannot be patterns of types of tokens. A CPRL signature is simply a set of instructions performed by a processor to match network traffic content to previously encountered content (i.e. content desired to be detected).*** (Row 12) (emphasis added)
- This portion of Wells describe the 'A' and 'B' predicates, both of which are "Test" type predicates. Wells at 15:27–16:54. Both the 'A' and 'B' predicates have a number of "formats" corresponding to differences in the number and type of arguments the predicate accepts. Id. The Office Action cites these portions of Wells to conclude that "a given predicate… represents a type of token because it may have multiple forms." Office Action at 14. ***A person of skill in the art would not reach the same conclusion because the hallmark of a token is that it is a "program code construct," not a processor instruction written in a "CPRL" that is unrelated to the program code. The fact that a given predicate can have more than one form does not transform a given predicate into a type of token.*** (Row 13) (emphasis added)

- The Office Action cites this portion of Wells as disclosing that the predicates are program code constructions. Office Action at 15. I disagree. The claims of the '305 Patent require that the program code is the code in which computer exploits are found. '305 Patent at claim 1 ("computer exploits being portions of program code that are malicious…tokens being program code constructs"). ***In contrast, Wells' predicates are "basic roots or components of a CPRL." Wells at 4:57–58. However, the CPRL is a specialized "content pattern recognition language"—a meta language used specifically for scanning—it is not program code that contains potential exploits.*** (Row 14) (emphasis added)
- The Office Action contends that this section of Wells teaches the "if" program code construct. Office Action at 15. I disagree because the 'I' predicate only compares bytes in a buffer. ***In other words, it performs an "if" operation on byte code, rather than identifying an "if" operation in the program code. This is because predicates are not "program code constructs" but rather CPRL components used to scan code.*** (Row 15) (emphasis added).
- These portions of the Wells reference indicate that Wells categorizes CPRL predicates into families based upon the type of function they perform on network traffic. ***But types functions to be performed on network traffic are not types of tokens, "tokens being program code constructs," they are "basic roots or components of a CPRL."*** Wells at 4:57–58. (Row 16) (emphasis added).
- The 'M' and 'P' predicates are each "a 'function' type predicate [that] provides instruction that causes the processor 24 to execute a prescribed function." ***These predicates do not represent a "function" type token in the program code.*** (Row 17) (emphasis added).
- This section of Wells only indicates the form that an "argument" of a predicate can take. To the extent that these arguments are used by the 'A' or 'U' predicates to "test the content script for characters" or "punctuation strings" as stated in the Office Action at 16–17, ***this only means that these particular predicates can be used to identify strings, not that the predicate is either a token or type of token. These predicates do not represent an "identifier" type token in the program code.*** (Row 18) (emphasis added).
- This portion of Wells only indicates that additional types of predicates may be added to the CPRL to increase its functionality. ***But predicates are not tokens, so the ability to expand the number of predicates in the CPRL has no bearing on the claims of the '305 Patent.*** (Row 19) (emphasis added).

The whole of the Examiner's attempt to address this fundamental difference between Wells and the claim language is as follows:

> The PO points to Medvidovic Dec. at ¶¶ 36-37 argues that predicate are not tokens or types of tokens. (Remarks at 12) Arguing that while a particular predicate might execute a macro process, it is not itself function type of token. Arguing that, a predicate that can test content script for characters is not itself a type of character. Nor is a predicate that is represented by a letter or punctuation mark, or that includes a letter or punctuation mark, a type of token.
>
> Examiner respectfully disagrees. The PO's statement is contradictory on its face. For example, the PO argue that predicate can execute a macro, which means can perform one or more instruction, or act like a function, but later argues that a predicate is not a "program code construct", which means it cannot be a part of a program code. As another example the PO argues that the predicate is not a token because, token as part of the program code can be scanned. The PO however does not make any argument as to why a function that is part of a program code (programming construct) is not scanned in Wells or cannot be scanned for that matter.

The fact that CPRL is *de facto* a type of program code does not mean that CPRL is the **claimed** "*program code*" which is scanned for "*computer exploits*" wherein said "*computer exploits*" are "*patterns of types of tokens, tokens being program code constructs, and types of tokens comprising a punctuation type, and identifier type and a function type.*" It most certainly is not. Here, the Examiner does not address the specific teachings cited to by Dr. Medvidovic in Wells, but only "disagrees" with the Patent Owner without any support or explanation in Wells. Such a disagreement with the Patent Owner without anymore is deficient to support any form of obviousness rendering the FOA improper.

Accordingly, no *prima facie* case of unpatentability can be maintained in view of Wells since there is no description, disclosure or suggestion in Wells of, *inter alia,* the required "*tokens,*" "*types of tokens*" and "*patterns of types of tokens*" of the "*program code*" as claimed. Finjan has clearly articulated how the predicates of Wells are not descriptive of the claimed "*tokens*" and thus Wells is not descriptive of the following required elements of claims 1, 2, 5 and 13:

> *computer exploits being portions of program code that are malicious, wherein the parser and analyzer rules describe computer exploits as patterns of types of tokens, tokens being program code constructs, and types of tokens comprising a punctuation type, and identifier type and a function type*
>
> *scanning, by the computer, the selectively diverted incoming content to recognize potential computer exploits therewithin, based on a database of parser and analyzer rules corresponding to computer exploits, computer exploits being*

> *portions of program code that are malicious, wherein the parser and analyzer rules describe computer exploits as patterns of types of tokens, tokens being program code constructs, and types of tokens comprising a punctuation type, an identifier type and a function type;*

As such, Patent Owner identifies a substantial gap between Wells and the claimed invention that is "so great as to render the [claim] nonobvious to one reasonably skilled in the art." *Dann v. Johnston,* 425 U.S. 219, 230, 189 USPQ 257, 261 (1976). In view of this gap, Patent Owner submits that the present claims would not be obvious to one of ordinary skill in the art in view of Wells.

//

//

//

2. Sandu in Combination with Wells does not Remedy the Deficiencies of Wells

Recognizing the deficiencies of Wells, the Examiner cites to a second reference to Sandu. *See* Office Action, pgs. 27-57. And as with Wells, Patent Owner respectfully submits that the Examiner does not and cannot find each element claimed in the combination of Wells and Sandu as is required to support a *prima facie* case of unpatentability. As discussed above, the following element includes numerous components, each of which must be identified in Wells and/or Sandu to sustain a rejection:

> *a database of parser and analyzer rules corresponding to computer exploits, stored within the computer, computer exploits being portions of program code that are malicious, wherein the parser and analyzer rules describe computer exploits as patterns of types of tokens, tokens being program code constructs, and types of tokens comprising a punctuation type, an identifier type and a function type.*

On page 57 of the FOA, the Examiner takes issue with Patent Owner's phrasing of the rebuttal to this rejection. Respectfully, neither Sandu, nor Wells, nor the combination of Sandu and Wells disclose the elements of the claims and thus no *prima facie* case of unpatentability can be maintain. Next, the Examiner suggests that certain phrasing used by Dr. Medvidovic is an improper importation of a limitation into claims 1 and 13. And, finally, the Examiner maintains his position regarding the teachings of Sandu and Wells. Finjan continues to strongly disagree with the Examiner's position and respectfully submits additional comments below.

Initially, primary Figures from the '305 Patent and Sandu are set forth below and annotated in an attempt to visually show how the components and processes overlap, on their face, and, more importantly, how they differ.

‘305 Patent – Figure 2

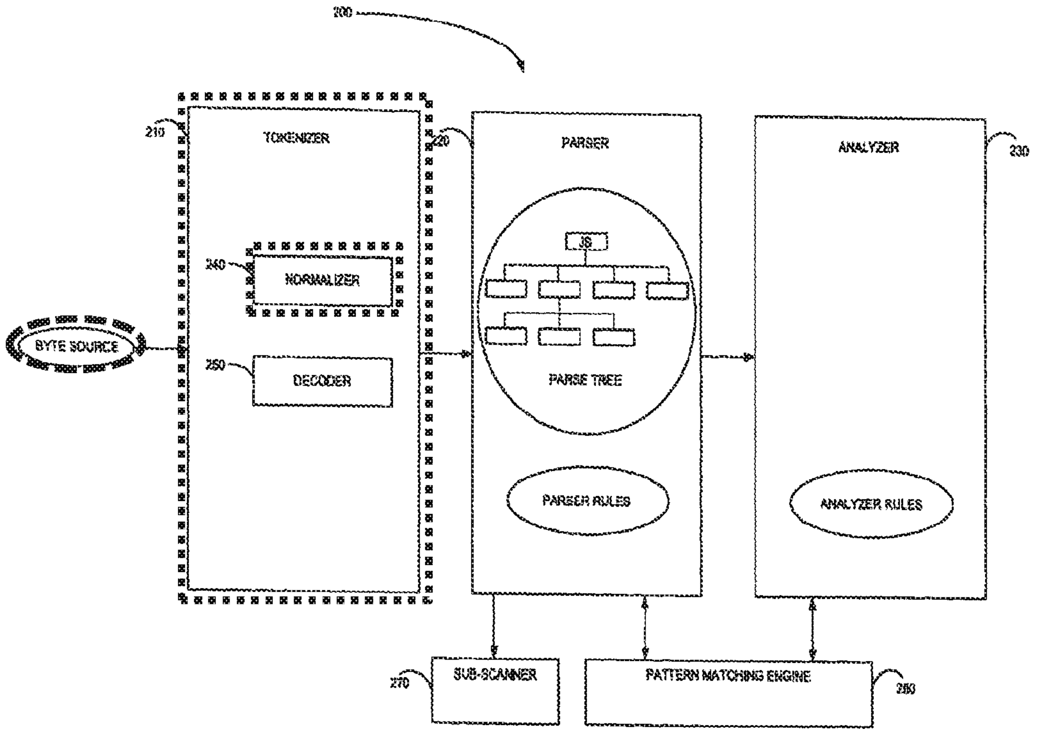


FIG. 2

Sandu – Figure 2

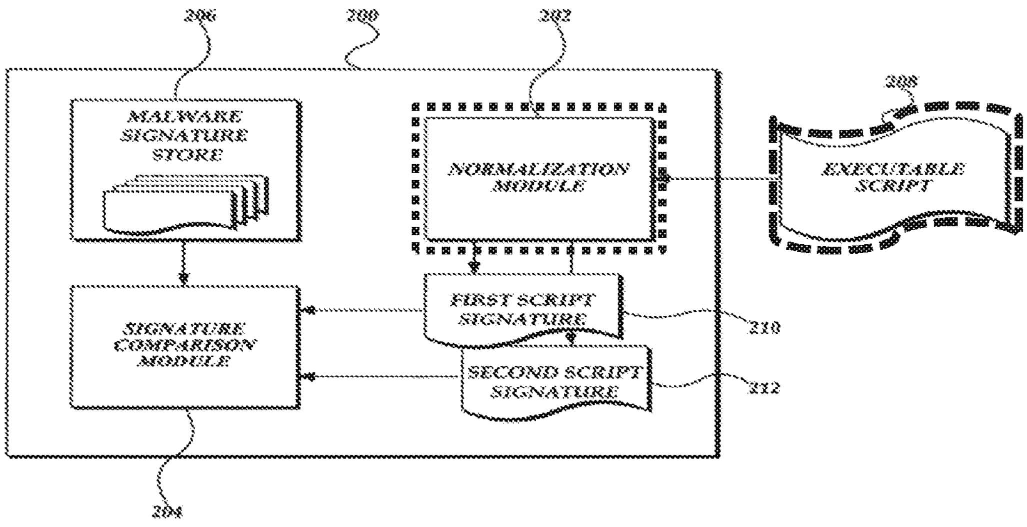


Fig. 2.

What is clear from this visual comparison is that any overlap between Sandu and the present claims ends at the "normalization module 202" of Sandu and the "Tokenizer 210" of the '305 Patent. The claimed "*parser rules*" and "*analyzer rules*" are absent from Sandu. The paragraphs from Sandu that are highlighted by the Examiner in the Office Actions, e.g., ¶¶ 37, 38, 40, and purported to disclose the claimed "*parser rules*" are, in fact, detailing what is occurring within the "normalization module 202" of Sandu. The "normalization module 202" of Sandu and accompanying description parallels in certain respects what is occurring in the Tokenizer/Normalizer/Decoder of the '305 Patent. *See* '305 Patent, Col. 9, l. 5 – Col. 10, l. 44. Although the nomenclature differs somewhat, what Sandu (and the Examiner) refers to as parsing and parser rules, are more appropriately compared with the "normalizer **240**" "normalization rules" and "decoders **250**" of the "tokenizer **210**" of the '305 Patent. These cited portions of Sandu and the '305 Patent are presented side-by-side in the table below and both stop short of the claimed parsing. Medvidovic Dec. ¶46. Thus, one skilled in the art can easily recognize the overlapping concepts of tokenization and normalization.

Sandu	**'305 Patent**
[0037] Those skilled in the art will recognize that most scripts include a "main" code segment. The main code segment may be located at the start of the script, such as is the case with Visual Basic script files, or in some other location, often with a label of "main." The main code segment is that body of code from an executable script which is first executed. It should be noted that while this main code segment is often not considered to be a "routine," for purposes of the present invention, the main code segment may be treated as a "routine." Accordingly, at block **404**, *the first routine in the executable script 208 is selected*. [0038] At block **406**, *the selected routine is normalized a first time, thereby generating a routine token set containing functionality tokens corresponding to the selected routine*. Normalizing a selected routine for a first time and generating a routine token set for the	The system of FIG. 2 includes three main components: a tokenizer **210**, a parser **220** and an analyzer **230**. *The function of tokenizer 210 is to recognize and identify constructs, referred to as tokens, within a byte source, such as JavaScript source code*. A token is generally a sequence of characters delimited on both sides by a punctuation character, such as a white space. *Tokens includes inter alia language keywords, values, names for variables or functions, operators, and punctuation characters, many of which are of interest to parser 220 and analyzer 230*. Preferably, *tokenizer 210 reads bytes sequentially from a content source, and builds up the bytes until it identifies a complete token*. For each complete token identified, tokenizer **210** preferably provides both a token ID and the token sequence.

selected routine is described below in regard to FIGS. 5A-5C. [0040] At block **506**, a first token from the selected routine is obtained. Obtaining tokens from an executable script is well known in the art as parsing, in this case parsing the selected routine. *Those skilled in the art will recognize that parsing identifies individual elements from the executable script. The individual elements are hereafter referred to as routine tokens.* These *routine tokens will comprise tokens of various types, including variables, operators, constants, execution directives, comments, subroutines, white space, and the like.* [0041] At block **508**, the current routine token is evaluated to determine its type, such as those token types described above. *At block **510**, a determination is made as to whether the routine token is a type of token that is to be ignored, i.e., one that is unimportant for comparison purposes and, correspondingly, not written to the routine token set.* According to one embodiment of the present invention, few routine token types are ignore tokens during the first normalization of the executable script **208**. For example, ignore tokens during the first normalization include comment tokens, execution directive tokens, and white space tokens.	Referring back to FIG. 2, tokenizer **210** preferably includes a normalizer **240** and a decoder **250**. In accordance with a preferred embodiment of the present invention, *normalizer 240 translates a raw input stream into a reduced set of character codes*. Normalized output thus becomes the input for tokenizer **210**. Examples of *normalization rules includes, inter alia* ○ *skipping character ranges that are irrelevant;* ○ *assigning special values to character codes that are irrelevant for the language structure but important for the content scanner;* ○ *translating, such as to lowercase if the language is case-insensitive, in order to reduce input for tokenizer **210**;* ○ *merging several character codes, such as white spaces and line ends, into one; and* ○ *translating sequences of raw bytes, such as trailing spaces, into a single character code.* Preferably, normalizer **240** also handles Unicode encodings, such as UTF-8 and UTF-16. In accordance with a preferred embodiment of the present invention, normalizer 240 is also implemented as a finite-state machine. *Each successive input is either translated immediately according to normalization rules, or handled as part of a longer sequence.* If the sequence ends unexpectedly, the bytes are preferably normalized as individual bytes, and not as part of the sequence. Preferably, normalizer **240** operates in conjunction with decoder **250**. Preferably, decoder **250** decodes character sequences in accordance with one or more character encoding schemes, including inter alia (i) SGML entity sets, including named sets and

	numerical sets; (ii) URL escape encoding scheme; (iii) ECMA script escape sequences, including named sets, octal, hexadecimal and Unicode sets; and (iv) character-encoding switches. Preferably, decoder **250** takes normalized input from normalizer **240**. In accordance with a preferred embodiment of the present invention, decoder **250** is implemented as a finite-state machine. The FSM for decoder **250** terminates when it reaches a state that produces a decoded character. If decoder **250** fails to decode a sequence, then each character is processed by tokenizer **210** individually, and not as part of the sequence. Preferably, a plurality of decoders **250** can be pipelined to enable decoding of text that is encoded by one escape scheme over another, such as text encoded with a URL scheme and then encoded with ECMA script scheme inside of JavaScript strings. *Id.* at Col. 9, l. 5 – Col. 10, l. 33.

Like Sandu, up to this point in the '305 Patent specification and figures, there has been no parsing or analysis in accordance with "*parser and analyzer rules [that] describe computer exploits as patterns of types of tokens.*" *See* Medvidovic Declaration, ¶47. The normalization and tokenization in the '305 Patent and Sandu are pre-parsing steps taken to prepare the raw incoming data stream for future action.

In Sandu, the output from the normalization and tokenization steps is a "script signature." Only after generation of the script signature does the static comparison step take place:

> With reference again to FIG. 3, after having generated a first script signature **210**, at block **304**, the first script signature is compared to known malware script signatures stored in the malware signature store **206**. Script signatures, such as script signature **210**, are compared on a routine basis, i.e., the signature comparison module **204** attempts to match routine token sets in the script signature **210** to routine token sets of known malware signature scripts stored in the script signature store **206**. According to one embodiment, the order of the routine token sets in a script signature **210** is unimportant.

Sandu, ¶60. Whereas, the output from the normalization and tokenization steps in the '305 Patent is subject to parsing and analysis in accordance with "*parser and analyzer rules [that] describe computer exploits as patterns of types of tokens.*" More particularly,

> In accordance with a preferred embodiment of the present invention, parser **220** controls the process of scanning incoming content. Preferably, parser **220** invokes tokenizer **210**, giving it a callback function to call when a token is ready. Tokenizer **210** uses the callback function to pass parser **220** the tokens it needs to parse the incoming content. Preferably, parser **220** uses a parse tree data structure to represent scanned content. ***A parse tree contains a node for each token identified while parsing, and uses parsing rules to identify groups of tokens as a single pattern. Examples of parsing rules appear in Appendix A, and are described hereinbelow***.

See, '305 Patent, Col. 10, ll. 45-54 (emphasis added). And further,

> Preferably, immediately after parser **220** performs a reduce operation, it calls analyzer **230** to check for exploits. Analyzer **230** searches for specific patterns of content that indicate an exploit.
>
> Preferably, parser **220** passes to analyzer **230** a newly-created parsing node. Analyzer **230** uses a set of analyzer rules to perform its analysis. An analyzer rule specifies a generic syntax pattern in the node's children that indicates a potential exploit. An analyzer rule optionally also includes one or more actions to be performed when the pattern of the rule is matched. In addition, an analyzer rule optionally includes a description of nodes for which the analyzer rule should be examined. Such a description enables analyzer **230** to skip nodes that are not to be analyzed. Preferably, rules are provided to analyzer **230** for each known exploit. Examples of analyzer rules appear in Appendix A, and are described hereinbelow.

To be clear, Sandu does not disclose the claimed:

> *database of parser and analyzer rules corresponding to computer exploits, stored within the computer, computer exploits being portions of program code that are malicious, wherein the parser and analyzer rules describe computer exploits as patterns of types of tokens, tokens being program code constructs, and types of tokens comprising a punctuation type, an identifier type and a function type.*

Further, the Examiner continues to argue that Sandu's "malware signatures… from known malware" correspond to the claimed "*analyzer rules*." FOA at 58. But again, the claimed "*analyzer rules*" correspond to "*computer exploits being portions of program code that are malicious*" and which are further claimed to be "*patterns of types of tokens.*" Sandu's singular, static action of comparing a generated script signature to known malware signatures; without identifying any exploits therewithin, can hardly be equated to the claimed "*analyzer rules.*" Further still, the claimed "*parser rules*" and "*analyzer rules*" are implemented by a claimed

> *rule-based content scanner that communicates with said database of parser and analyzer rules, operatively coupled with said network interface, for scanning incoming content received by said network interface to recognize the presence of potential computer exploits therewithin*

Respectfully, no such rule-based scanner is disclosed in Sandu. In fact, the words "*rule*" and "*scanner*" are **<u>completely absent</u>** from Sandu. And the solitary references in Sandu to "scan" and "scanning" are in Sandu's descriptions of the prior art. Sandu is enabled for a binary YES/NO "complete match" determination in comparing a generated script signature to known malware signatures; **<u>without identifying any exploits therewithin</u>**. Medvidovic Dec. ¶47, row 26. The claims require a "*database of parser and analyzer rules*" – rules being **<u>plural</u>**. Sandu's suggestion that it's process somehow determines a partial match from the comparison is completely and utterly unsupported by any enabling disclosure whatsoever and could hardly be considered a rule. In fact, Sandu provides no examples of any malware signatures or any examples of where a generated signature script matches, either completely or partially or not at all, a malware signature from the malware signature store. The entirety of Sandu's disclosure regarding "partial match" is found in ¶¶ 62, 76 and 77 which are set forth below:

> [0062] If there was not complete match between the script signature **210** and the known malware script signatures in the malware signature store **206**, at decision block **310**, an additional determination is made as to whether there was a partial match between the script signature and any of the known malware script signatures. Those skilled in the art will appreciate that often a discrete portion of a malware script actually performs its destructive process, while other portions of the malware script are not essential for that purpose. Thus, according to one embodiment, a partial match between the script signature **210** and a known malware script signatures may be indicative that the executable script is malware. Accordingly, at decision block **310**, if a partial match is made, at block **312**, a partial match flag is set. After having set the partial match flag, or if there is no partial match, at block **314**, a second script signature is generated. Generating a second script signature corresponding to a second normalization is described below in regard to FIG. 9.
>
> [0076] Alternatively, if there is not a complete match, a subsequent determination is made at decision block **320**, as to whether there was a partial match. If there was not a partial match, at decision block **322**, yet a further determination is made as to whether the partial match flag is set, indicating that there was a partial match between the first script signature **210** and corresponding malware script signatures in the script signature store **206**. If the partial match flag is set, or if, at decision block **320**, there was a partial match between the second script signature **212** and known malware signatures in the malware signature store **206**, at block **324**, the malware detection system **200** reports that a script signature for the executable

> script **208** partially matches a known malware script signature, indicating that the executable script is likely to be malware. Thereafter, the routine **300** terminates.
>
> [0077] Alternatively, if there was not a partial match at decision block **320** and the partial match flag is not set, at block **326**, the malware detection system **200** reports that the script signatures for the executable script **208** do not match any known malware script signatures, and that the malware detection system **200** is unable to determine that the executable script is malware.

This begs many questions, the most obvious being: how much of a match is enough for a partial match? And again, unlike the present claims, which specifically require "*parser and analyzer rules corresponding to computer exploits, stored within the computer, computer exploits being portions of program code that are malicious*" the best Sandu can offer with a complete match is that the executable script is malware and that a partial match indicates *likely* malware. *See* Sandu at [0076]; Medvidovic Dec. ¶47, row 26. These processes do not equate to the claimed identification of "*computer exploits.*" Under no circumstances does Sandu's process identify any individual exploits within an executable script and, therefore, the malware signatures disclosed in Sandu are not the claimed "*analyzer rules*" because they do not correspond to "*computer exploits*." *See* Medvidovic Dec. ¶47, row 26. As explained by Dr. Medvidovic,

> At best, Sandu can determine that there was some overlap between known malware and an executable script being analyzed. But Sandu cannot determine that the portion that overlaps, the partial match, corresponds to an exploit. Rather Sandu's system can only speculate that such a partial match "***may be indicative that the executable script is malware***." If Sandu's malware signatures actually corresponded to computer exploits, then the system would be able to determine that the match or partial match indicated the presence of a particular exploit in the code, which Sandu admittedly cannot accomplish.

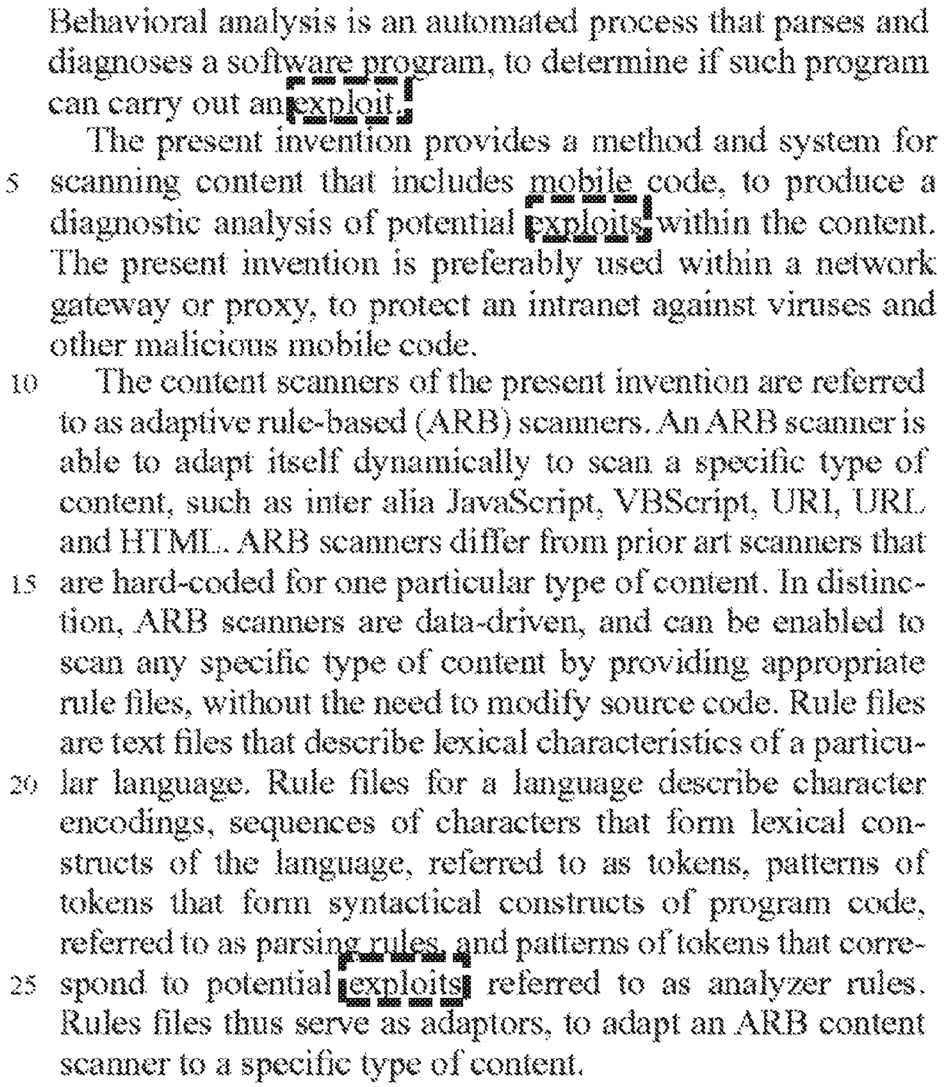
Id. at row 28.

On page 58 of the FOA, the Examiner submits:

> Second, as to the argument about "computer exploit" in the '305 patent and computer "malware" in Sandu, there is no explicit definition in the '305 patent regarding this term. The only place this term is defined is in the independent claims 1, 13 and 17 of the '305 patent that defines "**computer exploit**" as "**<u>being portions of program code that are malicious"</u>**. With the broadest reasonable interpretation by the Office **<u>"malware" is a form of "computer exploit"</u>** and identifying a malware" is in fact a reasonable interpretation of identifying a "computer exploit".

(Emphasis in original). This statement by the Examiner is not well-received. The word "exploit" is recited more than 60 times in the '305 Patent. And a specific example of an exploit is

<u>explicitly recited</u> in the Detailed Description. There are no recitations of the word "malware" in the '305 Patent. For the Examiner's benefit, key passages containing the word "exploit" are provided and annotated below for reference.

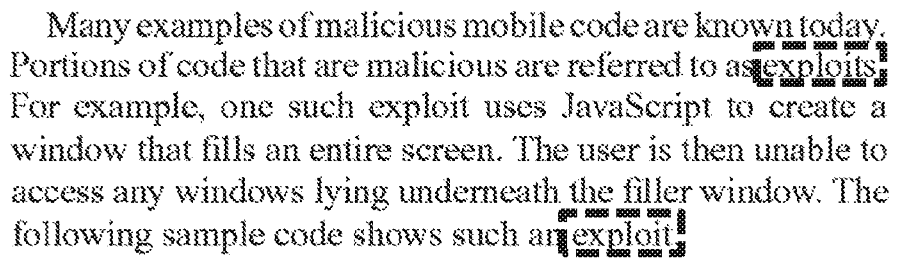


Behavioral analysis is an automated process that parses and diagnoses a software program, to determine if such program can carry out an exploit.

The present invention provides a method and system for scanning content that includes mobile code, to produce a diagnostic analysis of potential exploits within the content. The present invention is preferably used within a network gateway or proxy, to protect an intranet against viruses and other malicious mobile code.

The content scanners of the present invention are referred to as adaptive rule-based (ARB) scanners. An ARB scanner is able to adapt itself dynamically to scan a specific type of content, such as inter alia JavaScript, VBScript, URI, URL and HTML. ARB scanners differ from prior art scanners that are hard-coded for one particular type of content. In distinction, ARB scanners are data-driven, and can be enabled to scan any specific type of content by providing appropriate rule files, without the need to modify source code. Rule files are text files that describe lexical characteristics of a particular language. Rule files for a language describe character encodings, sequences of characters that form lexical constructs of the language, referred to as tokens, patterns of tokens that form syntactical constructs of program code, referred to as parsing rules, and patterns of tokens that correspond to potential exploits, referred to as analyzer rules. Rules files thus serve as adaptors, to adapt an ARB content scanner to a specific type of content.

'305 Patent, Col. 2, ll. 1-27.



Many examples of malicious mobile code are known today. Portions of code that are malicious are referred to as exploits. For example, one such exploit uses JavaScript to create a window that fills an entire screen. The user is then unable to access any windows lying underneath the filler window. The following sample code shows such an exploit.

EXAMPLE EXPLOIT

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>BID-3469</TITLE>
<SCRIPT>
      op=window.createPopup( );
      s='<body>foobar</body>';
      op.document.body.innerHTML=s;
      function oppop( )
      {
            if (!op.isOpen)
                  op.show(0,0, screen.width, screen.height, document.body);
      }
      function doit ( )
      {
            oppop( );
            setInterval("window.focus( ); {oppop( );}",10);
      }
</SCRIPT>
</HEAD>
<BODY>
<H1>BID-3469</H1>
<FORM method=POST action="">
<INPUT type="button" name="btnDoIt" value="Do It" onclick="doit( )">
</FORM>
</BODY>
</HTML>
```

The command op.show(0,0, screen.width, screen.height, document.body) is responsible for opening a window that fills the entire screen. It may be appreciated that there are many equivalents to such command. For example, the section of code

```
                {
                    w = screen.width;
                    h = screen.height;
                    op.show(0,0, w, h, document.body);
                }
```

performs the same action of opening a window that fills the entire screen; as also does the section of code

```
                {
                    a = screen.width;
                    b = screen.height;
                    w = a;
                    h = b;
                    op.show(0,0, w, h, document.body);
                }.
```

In distinction, although it appears similar, the section of code

```
        {
            w = screen.width;
            h = screen.height;
            w = 10;
            h = 10;
            op.show(0,0, w, h, document.body);
        }
```

does not fill the screen, and may be part of non-malicious content.

'305 Patent, Col. 5, l. 65-Col. 7, l. 2.

In accordance with a preferred embodiment of the present invention, network gateway **110** includes a content scanner **130**, whose purpose is to scan mobile code and identify potential exploits. Content scanner **130** receives as input content containing mobile code in the form of byte source, and generates a security profile for the content. The security profile indicates whether or not potential exploits have been discovered within the content, and, if so, provides a diagnostic list of one or more potential exploits and their respective locations within the content.

'305 Patent, Col. 7, ll. 11-20.

> Consider, for example, a complicated JavaScript file that is scanned and determined to contain a known exploit therewithin. An MD5 hash value of the entire JavaScript file can be stored in cache, together with a security profile indicating that the JavaScript file contains the known exploit. If the same JavaScript file arrives again, its hash value is computed and found to already reside in cache. Thus, it can immediately be determined that the JavaScript file contains the known exploit, without re-scanning the file.

'305 Patent, Col. 7, ll. 48-56.

> Moreover, in accordance with a preferred embodiment of the present invention, security violations, referred to as exploits, are described using a generic syntax, which is also language-independent. It is noted that the same generic syntax used to describe exploits is also used to describe languages. Thus, referring to Appendix A, the same syntax is used to describe the JavaScript parser rules and the analyzer exploit rules.
>
> It may thus be appreciated that the present invention provides a flexible content scanning method and system, which can be adapted to any language syntax by means of a set of rules that serve to train the content scanner how to interpret the language. Such a scanning system is referred to herein as an adaptive rule-based (ARB) scanner. Advantages of an ARB scanner, include inter alia:
>
> the ability to re-use software code for many different languages;
>
> the ability to re-use software code for binary content and EXE files;
>
> the ability to focus optimization efforts in one project, rather than across multiple projects; and
>
> the ability to describe exploits using a generic syntax, which can be interpreted by any ARB scanner.

'305 Patent, Col. 8, l. 53-Col. 9, l. 8.

> Preferably, immediately after parser **220** performs a reduce operation, it calls analyzer **230** to check for exploits. Analyzer **230** searches for specific patterns of content that indicate an exploit.

'305 Patent, Col. 12, ll. 54-57.

> Preferably, parser **220** passes to analyzer **230** a newly-created parsing node. Analyzer **230** uses a set of analyzer rules to perform its analysis. An analyzer rule specifies a generic syntax pattern in the node's children that indicates a potential exploit. An analyzer rule optionally also includes one or more actions to be performed when the pattern of the rule is matched. In addition, an analyzer rule optionally includes a description of nodes for which the analyzer rule should be examined. Such a description enables analyzer **230** to skip nodes that are not to be analyzed. Preferably, rules are provided to analyzer **230** for each known exploit. Examples of analyzer rules appear in Appendix A, and are described hereinbelow.

'305 Patent, Col. 12, l. 58 to Col. 13, l. 3.

> Referring back to the example above, the pattern
> (IDENT) ASSIGNMENT IDENT<val=="screen">DOT
> IDENT<val=="width">
> within the rule for ScrWidAssign describes a five-token pattern; namely, (i) an IDENT token, followed by (ii) an ASSIGNMENT token, followed by (iii) an IDENT token that has a value equal to "screen", followed by (iv) a DOT token, and followed by (v) an IDENT token that has a value equal to "width". Preferably, the value of an IDENT (i.e., an identifier) is its name; thus such a pattern indicates use of a member reference "screen.width" within an assignment statement, and corresponds to the example exploit listed above in the discussion of FIG. **1**. For example, it corresponds to an assignment of the form

'305 Patent, Col. 16, ll. 1-14.

Similarly, the pattern
IDENT<@(val).attr?=ATTR_WINDOW>DOT
FuncCall<val==“show” & matches(1):RULE(ScrWidH-gtList)>;
in the rule for WndShowScrnWidHgt1 corresponds to the command
op.show(0,0, w, h, document.body)
in the example exploit above; and the pattern
(IDENT) ASSIGNMENT IDENT<@(val).attr?=ATTR_WINDOW>DOT
FuncCall<val==“createPopup”>$;
in the rule for CreatePopup1 corresponds to the command
op=window.createPopup( ).
The action for the rule for Begin assigns attribute ATTR_WINDOW to the symbol table entry to “window”, and thus the action for CreatePopup1 assigns this attribute ATTR_WINDOW to the symbol table value for op. In turn, the rule for WndShowScrnWidHight1 recognizes that op satisfies the condition<@(val).attr?=ATTR_WINDOW.

It may thus be appreciated that exploits are generally described in terms of composite pattern matches, involving logical combinations of more than one pattern.

‘305 Patent, Col. 16, l. 51-Col. 17, l. 5.

> Desktop computer **900** preferably includes a network traffic probe **920**, which generally passes incoming network traffic to its destination, be it a browser, e-mail client or other Internet application. However, in accordance with a preferred embodiment of the present invention, network traffic probe **920** selectively diverts incoming network traffic to ARB scanner **930**. ARB scanner **930** scans and analyzes content to detect the presence of potential exploits. To this end, desktop computer **900** preferably maintains a database **940** of coded exploit rules in the form of deterministic or non-deterministic finite automata, which perform pattern matches appropriate to exploits under consideration. If ARB scanner **930** does not detect a match with a potential exploit, then the content is routed to its destination. Otherwise, if ARB scanner **930** detects the presence of potential exploits, then the suspicious content is passed to content blocker **950**, which removes or inoculates such content.
>
> In order to keep exploit rule database **940** current, desktop computer **900** preferably includes a rules update manager **960**, which periodically receives modified rules and new rules over the Internet, and updates database **940** accordingly.
>
> Reference is now made to FIG. **10**, which is a simplified block diagram of a rule server that updates rule databases for the desktop computer **900** of FIG. **9**, in accordance with a preferred embodiment of the present invention. Shown in FIG. **10** is a rules update server computer **1010**, which serves as a source for current exploit rules. Typically, when a rule is added for a new exploit, a rules compiler **1020** processes a semantic characterization of the exploit to produce an appropriate coded rule in the form of a deterministic or non-deterministic finite automaton. In turn, the newly coded rule is transmitted to desktop computer **900**, for incorporation into its local database **940**.

'305 Patent, Col. 19, ll. 35-67. In reviewing these passages, the definition of "exploit" is consistently described as portions of code that are malicious and generally described in terms of composite pattern matches, involving combinations of more than one pattern. This is simply not disclosed or suggested by Sandu.

Accordingly, it is most respectfully submitted that Sandu does not disclose either the claimed "*parser rules*" or the claimed "*analyzer rules*" and thus it follows that Sandu does not disclose the claimed "*database of parser and analyzer rules,*" the claimed "*rule-based content scanner that communicates with said database of parser and analyzer rules*" or the claimed "*rule* *update manager that communicates with said database of parser and analyzer rules.*" Likewise, Sandu does not disclose claim 5, which also requires recognition of a "*computer exploit ... by said rule-based content scanner.*" The deficiencies of Wells are discussed above and Sandu does not remedy these deficiencies.

Independent claim 13 was similarly rejected by the Examiner over Sandu in view of Wells. Like independent claim 1, claim 13 requires, *inter alia*: "*a database of parser and analyzer rules*" and "*updating the database of parser and analyzer rules periodically to incorporate new behavioral rules that are made available.*" It is respectfully submitted that for the reasons discussed above with respect to identical elements of claim 1, claim 13 is believed to be patentable over Sandu in view of Wells.

3. Examiner Failed to Consider Strong Evidence of Secondary Considerations

The Examiner must consider evidence of secondary considerations, i.e., objective evidence of nonobviousness, as part of the Examiner's obviousness analysis. This was reiterated most recently by the Federal Circuit in *Wbip, LLC v. Kohler Co.*, 2015-1038 (Fed. Cir. July 19, 2016):

> Indeed, we have repeatedly stressed that objective considerations of non-obviousness must be considered in every case. *Transocean Offshore Deepwater Drilling Inc. v. Maersk Drilling USA, Inc.*, 699 F.3d 1340, 1349 (Fed. Cir. 2012) ("[E]vidence rising out of the so-called 'secondary considerations' must always when present be considered en route to a determination of obviousness." (quoting *Stratoflex, 713 F.2d at 1538)).* This requirement is in recognition of the fact that each of the Graham factors helps to inform the ultimate obviousness determination. *Kinetic Concepts,* 688 F.3d at 1360; *Nike, Inc. v. Adidas,* 812 F.3d 1326, 1340 (Fed. Cir. 2016) (holding that evidence of secondary considerations must be examined to determine its impact on the first three Graham factors). Thus, the strength of each of the Graham factors must be weighed in every case and must be weighted en route to the final determination of obviousness or non-obviousness.

Contrary to the Federal Circuit's explicit requirement, this is precisely what the Examiner does; fails to consider properly presented evidence as part of the Examiner's obviousness analysis. Responsive to Patent Owner's evidence tying **the exact claims at issue** to multiple licenses and settlement, the Examiner states:

> In absence of any evidence beyond the listing of a few licenses, there is no nexus between the licensing activity and the merits of the claimed invention in the present situation. As such, the evidence of nonobviousness provided is given little if any weight, and for that reason the rejection of the claims are maintained.

FOA, page 61. The Examiner's complete and total failure to consider this evidence appears to stem from his reliance on a laundry list of potential inquiries set forth in a **non-precedential** BPAI opinion, *Ex Parte NTP, Inc.*, 2009 WL 3793380 (2009). FOA, page 60. Respectfully, this opinion is non-binding. It would seem that the Examiner has fallen victim to the hindsight bias trap "develop[ing] a hunch that the claimed invention was obvious, and then construct[ing] a selective version of the facts that confirms that hunch." *In re Cyclobenzaprine Hydrochloride Extended-Release Capsule Patent Litig.*, 676 F.3d 1063, 1079 (Fed. Cir. 2012).

The evidence presented in Mr. Kim's declaration supports an extremely strong nexus between the **exact claims at issue** in this reexamination and the licenses. *See* Kim Declaration, ¶6, 7, Exhibits A and B. In these exhibits, licensees' products are mapped to the elements of claims 1 and 13, respectively, in claim charts provided to the licensees. Thus indicating that the licenses arose from "recognition and acceptance" of the '305 by the licensees. *Stratoflex* at 1539. Further, the very requester of the present reexamination settled the concurrent litigation involving the '305 Patent on June 1, 2016, agreeing to pay Patentee $10.9 million. *Id.* at ¶8; *see also* Notice of Concurrent Proceedings filed June 7, 2016 including Stipulation and Order of Dismissal with Prejudice (June 7, 2016). In addition to the evidence presented and discussed in Mr. Kim's declaration, consideration should also be given to the fact that the very Assignee of the Sandu reference (which became USP 7707634), Microsoft, is a licensee of the '305 Patent. *See* Kim Declaration, Exhibits A ("Our non-confidential licensees include Microsoft, M86, Trustwave."). This begs the question: why take a license to the '305 patent if you already own rights to the technology (as is suggested by the Examiner)?

Accordingly, Finjan respectfully maintains that even if a *prima facie* case of unpatentability is presented with Wells or the combination of Sandu and Wells, such a case is readily rebutted by the objective evidence of non-obviousness provided for in the 37 C.F.R. § 1.132 Declaration of S. H. Michael Kim.

Patent Owner respectfully submits that this objective evidence must be considered by the Examiner in accordance with the requirements of *Graham v. John Deere Co.* which holds that obviousness is a question of law based on underlying factual findings: (1) the scope and content of the prior art; (2) the differences between the claims and the prior art; (3) the level of ordinary skill in the art; and ***(4) objective considerations of nonobviousness***. 383 U.S. 1, 17–18, 86 S.Ct.

684, 15 L.Ed.2d 545 (1966). A proper weighing of all of the evidence ultimately weighs in favor of non-obviousness.

III. CONCLUSION

In view of the foregoing, Patent Owner respectfully requests reconsideration of the positions taken in the Final Office Action and further submits that the present reexamination proceeding is in condition for a Notice of Intent to Issue a Reexamination Certificate confirming all original claims of the '305 Patent. The Examiner is respectfully requested to contact the undersigned by telephone at the below listed telephone number, in order to expedite resolution of any issues and to expedite prosecution of the present reexamination proceeding, if any comments, questions, or suggestions arise in connection with the present reexamination proceeding.

Please charge any shortage in fees due in connection with the filing of this communication to Deposit Account No. 50-6099 and please credit any excess fees to such deposit account.

Respectfully submitted,

Date: October 24, 2016

By: */Dawn-Marie Bey – 44,442/*
Dawn-Marie Bey (Reg. No. 44,442)
BEY & COTROPIA PLLC
213 Bayly Court
Richmond, VA 23229
(804) 441-8530
Attorneys for Patent Owner

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re *Ex Parte* Reexamination of
U.S. Patent No. 7,975,305 to Rubin, et al.

Application No.:	90/013,660	Technology Center:	3992
		Group Art Unit:	3992
Filed:	December 11, 2015	Confirmation No.:	5600
Patent Owner:	Finjan, Inc.	CRU Examiner:	Majid A. Banankhah

For U.S. Patent No. 7,975,305 – METHOD AND SYSTEM FOR ADAPTIVE RULE-BASED CONTENT SCANNERS FOR DESKTOP COMPUTERS.

Submitted Electronically

United States Patent and Trademark Office
Mail Stop *Ex Parte* Reexamination
Randolph Building
401 Dulany Street
Alexandria, VA 22314

CERTIFICATE OF SERVICE

I hereby certify that a true and correct copy of the foregoing Response to Final Office Action, filed on October 24, 2016, is to be served by first class United States mail on the 24th day of October, 2016 to:

Joseph J. Richetti
Bryan Cave LLP
1290 Avenue of the Americas
New York, New York 10104
(212) 541-2000
Attorney of Record for Third-Party Requester

Respectfully submitted,

Date: Oct. 24, 2016

By: */Dawn-Marie Bey – 44,442/*
Dawn-Marie Bey (Reg. No. 44,442)
BEY & COTROPIA PLLC
213 Bayly Court
Richmond, VA 23229
(804) 441-8530
Attorneys for Patent Owner

- updating the database of parser and analyzer rules periodically to incorporate new behavioral rules that are made available. (Claim 13)

Medvidovic Decl., ¶¶23, 24.

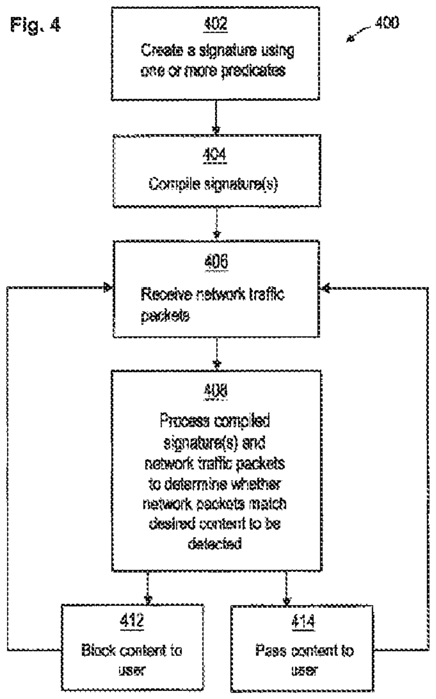
V. SUMMARY OF THE ASSERTED PRIOR ART

In rejecting the claims, the Office relies on two patent references in attempting to formulate a *prima facie* case of unpatentability based on obviousness. A brief summary of the references is provided below.

A. Wells

The first reference to Wells et al. U.S. Patent No. 8,140,660 ("Wells"), discloses a device for detecting known content. Wells at Abstract; *see also* Medvidovic Decl., ¶¶26-30. The device receives a signature that can be processed by a computer processor, receives network traffic to be screened, and executes functions, based on the signature, that determine whether the received network traffic "matches the content desired to be detected." Wells, 6:12–60; FIG. 4; *see also* Medvidovic Decl., ¶26.

Fig. 4


400
402
Create a signature using one or more predicates
404
Compile signature(s)
406
Receive network traffic packets
408
Process compiled signature(s) and network traffic packets to determine whether network packets match desired content to be detected
412
Block content to user
414
Pass content to user

Wells utilizes "a signature created using [a content pattern recognition language, "CPRL," which] represent[s] one or more instructions that control an operation of a processor being used to detect content." *Id.* at 5:50–53; *see also* Medvidovic Decl., ¶27. This CPRL signature is "a symbolic detection model" for content to be detected, such as a virus or a worm. *Id.* at 4:41–46; *see also* Medvidovic Decl., ¶29. In other words, a CPRL signature defines a series of scans, such as a worm scan, a conventional signature scan, a macro scan, or a heuristic scan, performed by a processor in an attempt to determine whether network traffic matches content to be detected. *See id.* at 6:53–7:50 (describing the different types of scans a CPRL signature will execute); *see also* Medvidovic Decl., ¶29. Such scans are carried out, as directed by a CPRL, in box 408 of FIG. 4, reproduced below. *See* Wells at 7:8–11 (describing running a signature scan that searches a "target file for byte-strings that are known to identify viruses"); *id.* at 7:11–18 (describing running a macro scan that searches a macro for "known macro virus strings" or "peculiar behavior"); *id.* at 7:11–18 (describing running a heuristic scan that searches "file for known byte strings that indicate the presence of a virus"); *see also* Medvidovic Decl., ¶29, n.1. Accordingly, rather than disclosing analyzer rules that "describe computer exploits as patterns of types of tokens, tokens being program code constructs," Wells' CPRL signatures are merely sets of instructions that inform a processor of the type of functions to perform on network traffic to identify known content. Medvidovic Decl., ¶29.

The CPRL signature described in Wells is created using a set of "predicates that are the basic roots or components of a CPRL." *Id.* at 4:55–58; *see also* Medvidovic Decl., ¶30. A predicate is an element of the CPRL language that "is compiled into a byte stream that controls a logic" of a processor by performing functions associated with the predicate. *Id.* at 5:8–11; *see also* Medvidovic Decl., ¶30. These functions called by the CPRL language are performed on the incoming network traffic content looking for a match. *see also* Medvidovic Decl., ¶30. To be clear, the predicates disclosed in Wells do not form any portion of the network traffic that is being scanned and the predicates are not compared to any portion of the network traffic that is being scanned by Wells. *Id.* Wells is not scanning the network traffic for predicates. The predicates are components of the instructions, i.e., CPRL, used to by a processer to perform scans of the network traffic. *Id.*

B. <u>Sandu</u>

Like the Wells reference, Sandu's technique is designed to determine whether a particular program—referred to as "an executable script"—is malware. See Sandu at [0011] ("After normalizing the executable script, the malware detection system compares the script signature corresponding to the executable script to the script signatures in the malware signature store, and accordingly determines whether the executable script is malware."); *see also* Medvidovic Decl., ¶¶41-45. To accomplish this task, Sandu describes a modified virus-signature scanner that can recognize known malware, which as a result of superficial changes to the code cannot be detected by traditional virus-signature scanners:

> As routine names, variable names, and the like may be easily modified in a superficial manner, yet functionally remain the same, the present invention looks past the arbitrarily assigned labels in an executable script 208, and instead looks at its functional contents in a normalized form.

Sandu at ¶ [0030]; *see also id.* at ¶ [0035] (describing superficially modifying executable script by "rearrang[ing] the location of the routines within the body of the executable script."); *see also* Medvidovic Decl., ¶41.

Sandu's technique for recognizing known malware that has been superficially modified involves normalizing an executable script to create a "script signature," and comparing the script signature to script signatures corresponding to known malware. See Sandu at [0011], [0029], [0031], [0032], [0060], [0061]; *see also* Medvidovic Decl., ¶42. The normalization of the executable script occurs "on a routine basis." Id. at [0035]; *see also* Medvidovic Decl., ¶42. Sandu states that it is appropriate to normalize executable scripts at the routine level because moving an entire routine from one section of the executable script to another generally does not affect the script's functionality while rearranging content within a routine changes functionality significantly. Id. The result of normalizing a single routine is a "routine token set." Id. at [0038]; *see also* Medvidovic Decl., ¶42. A script signature for the executable script is then generated as a collection of all of the routine token sets generated during the normalization process. Id. at [0059]; *see also* Medvidovic Decl., ¶42.

Upon comparing the script signature to known malware script signatures, Sandu discloses determining whether there was a complete or partial match between the script signature and any known malware script signatures. *Id.* at [0061]–[0062]; *see also* Medvidovic Decl., ¶43. This general procedure is illustrated in FIG. 3 of the Sandu reference:

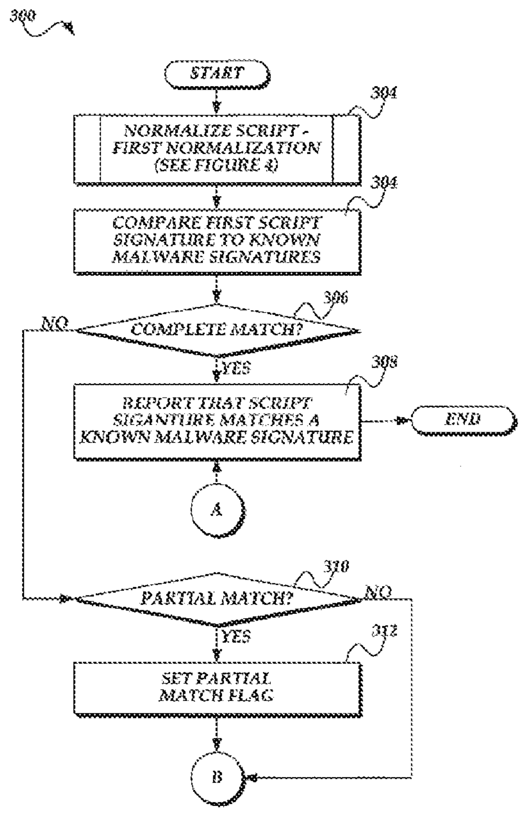

300
START
NORMALIZE SCRIPT - FIRST NORMALIZATION (SEE FIGURE 4)
304
COMPARE FIRST SCRIPT SIGNATURE TO KNOWN MALWARE SIGNATURES
304
COMPLETE MATCH?
306
NO
YES
REPORT THAT SCRIPT SIGNATURE MATCHES A KNOWN MALWARE SIGNATURE
308
END
A
PARTIAL MATCH?
310
NO
YES
SET PARTIAL MATCH FLAG
312
B

Fig.3A.

Sandu provides a modified version of a traditional "static" virus-signature scanner that simply compares normalized signatures from code being analyzed to similarly normalized signatures from known malware. Medvidovic Decl., ¶43.

VI. ARGUMENTS

A. Wells does not Disclose Required Elements of Claims 1, 2, 5 and 13

At least the following elements of Claims 1, 2, and 5 of the '305 Patent are neither disclosed nor suggested by Wells:

> *a database of parser and analyzer rules corresponding to computer exploits, stored within the computer, computer exploits being portions of program code that are malicious, wherein the parser and analyzer rules describe computer exploits as patterns*

> *of types of tokens, tokens being program code constructs, and types of tokens comprising a punctuation type, an identifier type and a function type*; (Claims 1, 2, 5)
>
> *a rule-based content scanner that communicates with said database of parser and analyzer rules, operatively coupled with said network interface, for scanning incoming content received by said network interface to recognize the presence of potential computer exploits therewithin*; (Claims 1, 2, 5)
>
> *a rule update manager that communicates with said database of parser and analyzer rules, for updating said database of parser and analyzer rules periodically to incorporate new parser and analyzer rules that are made available*. (Claims 1, 2, 5)

See Medvidovic Decl., ¶¶23, 26-40. To support a *prima facie* case of unpatentability under *Graham v. John Deere* 383 U.S. 1 (1966), the Examiner must present, *inter alia*, "(1) [] finding that the prior art included each element claimed, although not necessarily in a single prior art reference, with the only difference between the claimed invention and the prior art being the lack of actual combination of the elements in a single prior art reference." *See* Section 2143(A) of the Manual of Patent Examining Procedure (MPEP). It is most respectfully submitted that the Examiner does not and cannot find each element claimed in Wells as is required to support a *prima facie* case of unpatentability. More particularly, on pages 8-17 of the Office Action, the Office addresses the following element of claims 1, 2 and 5:

> *a database of parser and analyzer rules corresponding to computer exploits, stored within the computer, computer exploits being portions of program code that are malicious, wherein the parser and analyzer rules describe computer exploits as patterns of types of tokens, tokens being program code constructs, and types of tokens comprising a punctuation type, an identifier type and a function type*

The Office separates this element into two parts in the Office Action and attempts to address the two parts separately, which leads to the Examiner's misinterpretation and misapplication of Wells to the claim language. This is a single element which either implicitly or explicitly requires components for implementing tokenization, parsing and analyses. *See* Figure 2, '305 Patent; *see also* Medvidovic Dec. ¶¶19, 28. In the Office Action, the Examiner first suggests that the CPRL signatures of Wells disclose the claimed "*analyzer rules*" and that the predicates of Wells disclose the claimed "*parser rules*" of claims 1, 2 and 5. *See* Office Action, pgs. 8-11. The Examiner then goes on to suggest that these same predicates of Wells are the claimed "*tokens*." *Id.* These suggestions are not well-received. As discussed in the accompanying expert declaration of Dr. Nenad Medvidovic, the predicates cannot be both the "*parser rules*" and "*tokens*." Referring to ¶29 of the Medvidovic Dec.:

> A predicate is, therefore, not a "token" or even a "type of token" as described and claimed in the '305 Patent because it is not a "program code construct," where the program code is the code in which potential computer exploits are found. *See* '305 Patent, independent claims 1 and 13 ("computer exploits being portions of program code that are malicious… tokens being program code constructs"). That is, while tokens are constructs that <u>make up</u> the program code being scanned for potential computer exploits, predicates describe functions in a wholly separate language that <u>are performed on</u> program code being scanned. Put yet another way, "predicates are the basic roots or components of a <u>CPRL</u>," not program code that contains potential computer exploits as claimed in the '305 Patent. Wells at 4:55–58 (emphasis added).

See also Medvidovic Dec. ¶36 (I also disagree that individual predicates correspond to types of tokens. As noted above, predicates are not tokens or types of tokens. While a particular predicate might ***execute*** a macro or process, it is not itself function type of token. Similarly, a predicate that can test content script for characters is not itself a type of character. Nor is a predicate that is ***represented by*** a letter or punctuation mark, or that includes a letter or punctuation mark, a type of token. *See* Office Action at 16. Rather a predicate "represents a function to be performed by the processor." Wells at claim 1) (emphasis in original). The claimed "*tokens*" are "*program code constructs*" which means they make up the code that is being scanned. *See* Medvidovic Dec. ¶37, row 12. The predicates of Wells form the basis of the CPRL signatures that inform the processor what functions to perform on the network traffic, i.e., on the program code. *Id.* The predicates cannot be both the operating code and the code being operated on. *Id.*

Further, claims 1, 2 and 5 require that the "*computer exploits*" be "*patterns of types of tokens*." Again, there is no tokenization disclosed in Wells, and thus there can be no description of "*patterns of types of tokens*." *See* Medvidovic Dec. ¶¶27, 36-37. Wells runs predefined scans using CPRL signatures on the incoming network traffic to look for known byte string matches, i.e., known and previously identified content indicative of a virus. *Id.* The incoming network traffic in Wells is not tokenized. It is scanned as-is. *Id.* and Wells, Figure 4, Ref. 408. As such, the "*analyzer rules*" and "*parser rules*" of Wells cannot be found to correspond to "*computer exploits*" = "*patterns of types of tokens*." *Id.*

Further still, the claims require that the "*analyzer rules*" and "*parser rules*" be "*stored within the computer.*" Even if we assume, *arguendo,* that Wells' CPRL signatures disclose the claimed "*analyzer rules*" and that the predicates of Wells disclose the claimed "*parser rules,*" the predicates are never stored anywhere in Wells. *See* Medvidovic Dec. ¶35. At best, it could be said that the signatures may be stored for future use, but certainly not the predicates. *Id.*

Respectfully, Wells simply does not disclose or suggest:

> *a database of parser and analyzer rules corresponding to computer exploits, stored within the computer, computer exploits being portions of program code that are malicious, wherein the parser and analyzer rules describe computer exploits as patterns of types of tokens, tokens being program code constructs, and types of tokens comprising a punctuation type, an identifier type and a function type.*

See Medvidovic Dec. ¶¶33-35 (A CPRL signature is not a database of parser and analyzer rules corresponding to computer exploits stored within the computer, computer exploits being portions of program code that are malicious).

Similarly, without disclosure of the requisite "*database of parser and analyzer rules*" or "*computer exploits*" there can be no disclosure in Wells of the follow-on elements directed to:

> *a rule-based content scanner that communicates with said database of parser and analyzer rules, operatively coupled with said network interface, for scanning incoming content received by said network interface to recognize the presence of potential computer exploits therewithin* (Claim 1)
>
> *a rule update manager that communicates with said database of parser and analyzer rules, for updating said database of parser and analyzer rules periodically to incorporate new parser and analyzer rules that are made available.* (Claim 1)
>
> *The security system of claim 1 further comprising a content blocker, operatively coupled to said rule-based content scanner, for preventing incoming content having a computer exploit that was recognized by said rule-based content scanner from reaching its intended destination.* (Claim 5)

See Medvidovic Dec. ¶¶38-40.

Independent claim 13 was similarly rejected by the Examiner in view of Wells. Like independent claim 1, claim 13 requires components for implementing tokenization, parsing and analyses, including: "*a database of parser and analyzer rules corresponding to computer exploits, computer exploits being portions of program code…wherein the parser and analyzer* *rules describe computer exploits as patterns of types of tokens, tokens being program code constructs, and types of tokens comprising a punctuation type, an identifier type and a function type.*" Independent claim 13 also includes the follow-on element like that of claim 1 of "*updating the database of parser and analyzer rules periodically to incorporate new behavioral rules that are made available.*" It is respectfully submitted that for the reasons discussed above with respect to identical elements of claim 1, claim 13 is believed to be patentable over Wells. *See* Medvidovic Dec. ¶¶33-40.

B. Sandu does not Remedy the Deficiencies of Wells

Recognizing the deficiencies of Wells, the Examiner cites to a second reference to Sandu. *See* Office Action, pgs. 27-57. And as with Wells, Patentee respectfully submits that the Examiner does not and cannot find each element claimed in the combination of Wells and Sandu as is required to support a *prima facie* case of unpatentability. As discussed above, the following element includes numerous components, each of which must be identified in Wells and/or Sandu to sustain a rejection:

> *a database of parser and analyzer rules corresponding to computer exploits, stored within the computer, computer exploits being portions of program code that are malicious, wherein the parser and analyzer rules describe computer exploits as patterns of types of tokens, tokens being program code constructs, and types of tokens comprising a punctuation type, an identifier type and a function type.*

Initially, the Examiner first suggests that Sandu's identification of "the script's 'main' routine and generating a set of 'routine tokens' from the script's main body" corresponds to the claimed "*parser rules*." *See* Office Action at 34. The claimed "*parser rules*" operate on tokens to identify groups of tokens as a single pattern or as claimed to "*describe computer exploits as patterns of types of tokens.*" But Sandu simply receives all of the tokens for a given routine without applying any parser rules to identify groups of tokens as a single pattern. *See* Medvidovic Dec. ¶46. More particularly, as explained by Dr, Medvidovic,

> Rather than disclosing parser rules that "identify groups of tokens as a single pattern," Sandu simply tokenizes an entire routine. That is, Sandu simply describes obtaining all of the tokens for a selected routine but stops short of applying any "parser rules" after the tokens are obtained. In fact, Sandu's process has no need for parser rules that identify groups of tokens as a single pattern because it relies upon the comparison of token sets at the routine level rather than to identify exploits within program code.

Medvidovic Dec. ¶47, row 25.

Further, the Examiner suggests that Sandu's "malware signatures… from known malware" correspond to the claimed "*analyzer rules*." Office Action at 36. But again, the claimed "*analyzer rules*" correspond to "*computer exploits being portions of program code that are malicious*" and which are further claimed to be "*patterns of types of tokens*" whereas Sandu's malware signatures correspond to malware at the program level rather than to exploits, which may or may not be present for any given malware program. Medvidovic Dec. ¶46, row 25. Sandu only recognizes whether there is a complete or partial match (or no match) between a script signature and a malware script signature. Medvidovic Dec. ¶¶43-49. In the event that there is a complete match, the executable script associated with the script signature is deemed to match known malware—without identifying any exploits therewithin. Medvidovic Dec. ¶47, row 26. If there is a partial match, the script signature normalization process is repeated for the routine, but this time certain tokens are ignored by Sandu. *See* Sandu at [0066]; Medvidovic Dec. ¶47, row 26. The second normalized script signature is again compared to known normalized malware script signatures and after this second comparison, a complete match indicates malware and a partial match indicates *likely* malware. *See* Sandu at [0076]; Medvidovic Dec. ¶47, row 26. These processes do not equate to the claimed identification of "*computer exploits.*" Under no circumstances does Sandu's process identify any individual exploits within an executable script and, therefore, the malware signatures disclosed in Sandu are not "*analyzer rules*" because they do not correspond to "*computer exploits.*" *See* Medvidovic Dec. ¶47, row 26. As explained by Dr. Medvidovic,

> At best, Sandu can determine that there was some overlap between known malware and an executable script being analyzed. But Sandu cannot determine that the portion that overlaps, the partial match, corresponds to an exploit. Rather Sandu's system can only speculate that such a partial match "***may be indicative that the executable script is malware***." If Sandu's malware signatures actually corresponded to computer exploits, then the system would be able to determine that the match or partial match indicated the presence of a particular exploit in the code, which Sandu admittedly cannot accomplish.

Id. at row 28.

Accordingly, it is most respectfully submitted that Sandu does not disclose either the claimed "*parser rules*" or the claimed "*analyzer rules*" and thus it follows that Sandu does not disclose the claimed "*database of parser and analyzer rules,*" the claimed "*rule-based content* *scanner that communicates with said database of parser and analyzer rules*" or the claimed "*rule update manager that communicates with said database of parser and analyzer rules*." Likewise, Sandu does not disclose claim 5, which also requires recognition of a "*computer exploit ...by said rule-based content scanner.*" The deficiencies of Wells are discussed above and Sandu does not remedy these deficiencies.

Independent claim 13 was similarly rejected by the Examiner over Sandu in view of Wells. Like independent claim 1, claim 13 requires, *inter alia*: "*a database of parser and analyzer rules*" and "*updating the database of parser and analyzer rules periodically to incorporate new behavioral rules that are made available*." It is respectfully submitted that for the reasons discussed above with respect to identical elements of claim 1, claim 13 is believed to be patentable over Sandu in view of Wells.

C. <u>There is Strong Evidence of Secondary Considerations</u>

Objective indicia of non-obviousness plays a critical role in the obvious analysis. *Leo Pharmaceutical Products, Ltd. v. Rea*, 726 F. 3d 1346, 1358 (Fed. Cir. 2012). In fact, the court in Leo Pharmaceutical stated that objective indicia can be the most probative evidence of non-obviousness and enables a court to avert the trap of hindsight. *Id.*; see also *Plantronics, Inc. v. Aliph, Inc.*, 724 F. 3d 1343 (Fed. Cir. 2013) (the court reiterating the importance of objective indicia). These objective indicia of non-obviousness must be considered when present. *Sud-Chemie, Inc. v. Multisorb Techs., Inc.*, 554 F.3d 1001, 1008 (Fed. Cir. 2009). Accordingly, the objective indicia of non-obviousness provided below further demonstrates that the obviousness rejections should also be withdrawn.

Finjan respectfully submits that even if a *prima facie* case of unpatentability is presented with Wells or the combination of Sandu and Wells, such a case is weak and is readily rebutted by the evidence of secondary considerations provided for in the 37 CFR 1.132 Declaration of S. H. Michael Kim (attached hereto). Mr. Kim provides details regarding a highly successful patent licensing program which has been largely driven by the '305 Patent and, in particular, rejected claims 1 and 13. More particularly, Mr. Kim provides 2 different instances wherein the inventions of the '305 Patent, including one or more of the *identical* claims rejected by the Examiner, were presented to a potential licensee and resulted in two separate license agreements with royalty payments to Finjan totaling more than $5.5 million. *See* Kim Dec., ¶6, 7.

Attorney Docket No. FINREXM0012

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re *Ex Parte* Reexamination of U.S. Patent No. 7,975,305 to Rubin, et al.	Technology Center:	3992
Application No.: 90/013,660	Group Art Unit:	3992
Filed: December 11, 2015	Confirmation No.:	5600
Patent Owner: Finjan, Inc.	CRU Examiner:	Majid A. Banankhah

For U.S. Patent No. 7,975,305 – METHOD AND SYSTEM FOR ADAPTIVE RULE-BASED CONTENT SCANNERS FOR DESKTOP COMPUTERS.

Submitted Electronically

Mail Stop *Ex Parte* Reexam
Attn: Central Reexamination Unit
Commissioner for Patents
United States Patent & Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

RESPONSE TO FINAL OFFICE ACTION

Dear Sir:

In response to the pending Office Action dated August 24, 2016, please consider the following remarks. Prior to taking action responsive hereto, the Patent Owner respectfully requests an interview with the Examiner pursuant to the Interview Request and Proposed Agenda filed and faxed on October 21, 2016.

I. OVERVIEW

Patent Owner respectfully requests the Examiner withdraw the Final Office Action (FOA) as improper and confirm patentability of the rejected claims based on a number of errors.

First, in the FOA, the Examiner interprets key elements of the claims in a manner inconsistent with the law. For example, the Examiner improperly cites to extrinsic evidence regarding a non-claim term, "parsing," in order to define the claim term "parser rules" as "rules related to the process of analyzing a string of symbol in computer language [sic]." FOA, pgs. 48–49. Yet, in U.S. Patent No. 7,975,305 ("the '305 Patent") and the claims, parser rules "*describe computer exploits as patterns of types of tokens.*" The Examiner's definition is thus inconsistent with the specification and legally improper. *See Microsoft Corp. v. Proxyconn, Inc.*, 789 F. 3d 1292 (Fed. Cir. 2015) ("Even under the broadest reasonable interpretation, the Board's construction cannot be divorced from the specification and the record evidence and must be consistent with the one that those skilled in the art would reach.") (citations omitted). Here, the Examiner legally erred by using extrinsic evidence for a definition to "parsing," which is ***not*** a term used in the claims – i.e., "parser rules," nor supported in the '305 Patent where parser rules describe computer exploits as patterns of types of tokens.

Second, the Examiner interprets key elements of the claims in a manner inconsistent with the specification of '305 Patent and the reasons for allowance distinguishing over prior art. Specifically, the allowance of application no. 11/009,437 (now the '305 Patent) in December of 2010 is directly tied to *at least* the following pivotal claim language:

> *computer exploits being portions of program code that are malicious, wherein the parser and analyzer rules describe computer exploits as patterns of types of tokens, tokens being program code constructs, and types of tokens comprising a punctuation type, and identifier type and a function type*

See Notice of Allowance, Pages 3-4. Indeed, the Notice of Allowance, with accompany reasons for allowance, was responsive to Patent Owner's detailed arguments filed in September of 2010 wherein Patent Owner stated: "a point of novelty of the claimed invention is <u>describing and recognizing computer exploits from patterns of types of tokens</u>, which is not a known concept." *See* Response to Non-Final Rejection, September 15, 2010, Pages 7-8 (emphasis in original). One of ordinary skill would recognize at least this same point of novelty distinguishes the claims of the '305 Patent over the cited prior art and, in particularly, is clearly absent from Wells, Sandu