Paper No. _____

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

ESET, LLC and ESET spol s.r.o
Petitioners

v.

FINJAN, Inc.
Patent Owner

Patent No. 7,975,305
Issue Date: July 5, 2011
Title: METHOD AND SYSTEM FOR ADAPTIVE RULE-BASED CONTENT SCANNERS FOR DESKTOP COMPUTERS

Inter Partes Review No. 2017-01738

PETITION FOR *INTER PARTES* REVIEW UNDER 35 U.S.C. §§ 311-319 AND 37 C.F.R. § 42.100 *ET. SEQ.*

TABLE OF CONTENTS

PAGE

EXHIBIT LIST

Ex. #	Exhibit
1001	U.S. Patent No. 7,975,305 ("the '305 patent")
1002	Image File Wrapper for '305 Patent
1003	Examination of U.S. Patent No. 7,975,305, Non-Final Office Action, dated June 15, 2010
1004	Examination of U.S. Patent No. 7,975,305, September 15, 2000 Response to Non-Final Office Action, dated June 15, 2010
1005	Examination of U.S. Patent No. 7,975,305, Notice of Allowance, dated December 20, 2010
1006	Declaration of Eugene Spafford, Ph.D. and *Curriculum Vitae*
1007	90/013,660 *Ex Parte* Reexamination of U.S. Patent No. 7,975,305, Final Office Action, dated August 24, 2016
1008	90/013,660 *Ex Parte* Reexamination of U.S. Patent No. 7,975,305, Notice of Appeal, dated November 22, 2016
1009	Plaintiff Finjan, Inc.'s Initial Disclosure of Asserted Claims and Infringement Contentions and Document Production Pursuant to Patent Local Rules 3-1 and 3-2 dated December 9, 2016 in Case No. 3:16-cv-03731-JD (N.D. Cal).[1]
1010	Plaintiff Finjan, Inc.'s Amended Infringement Contentions Pursuant to Patent Local Rules 3-1 and 3-6 dated June 12, 2017.
1011	U.S. Patent No. 6,092,194 ("the '194 patent")
1012	U.S. Patent No. 8,225,408 ("the '408 patent")
1013	U.S. Patent No. 7,636,945 ("Chandnani")
1014	U.S. Patent No. 5,987,611 ("Freund")

[1] Following the exchange of these disclosures the case was transferred to the Southern District of California and assigned case No. 3:17-cv-0183-CAB-BGS.

NOTICE OF LEAD AND BACKUP COUNSEL

Lead Counsel: Nicola A. Pisano (Reg. No. 34,408) **Tel:** 858.847.6877

Backup Counsel: Christopher C. Bolten (Reg. No. 61,531) **Tel:** 858-847-6887

Address: Foley & Lardner LLP, 3579 Valley Centre Drive, San Diego, CA 92130.

Fax: 858-792-6773

NOTICE OF EACH REAL-PARTY-IN-INTEREST

The real-parties-in-interest of Petitioner are ESET, LLC and its foreign parent ESET spol s.r.o. ("ESET spol") (collectively "ESET").

NOTICE OF RELATED MATTERS

U.S. Patent No. 7,975,305 ("the '305 patent") is currently asserted in *Finjan, Inc., v. ESET, LLC, et al.,* Civil Action No. 3:17-cv-00183-CAB-BGS (S.D. Cal.). The '305 patent is currently the subject of an *ex parte* reexamination petition by Proofpoint Inc. ("Proofpoint"): Reexamination Control No. 90/013,660.

NOTICE OF SERVICE INFORMATION

Please address all correspondence to the lead counsel at the address above. Petitioner consents to electronic service at: npisano@foley.com and cbolten@foley.com.

GROUNDS FOR STANDING

Petitioner hereby **certifies** that the patent for which review is sought is available for *inter partes* review and that the Petitioner is not barred or estopped from requesting

an *inter partes* review. Petitioner has paid all fees believed to be due for this Petition. The Commissioner is hereby authorized to charge any additional fees which may be required regarding this Petition under 37 C.F.R. §§ 1.16-1.17, or credit any overpayment, to Deposit Account No. 19-0741.

STATEMENT OF MATERIAL FACTS

1. Finjan, Inc. ("Finjan") is the purported patent owner by virtue of an assignment executed on November 24, 2009 and recorded at the United States Patent Office at Reel/Frame 023556/0853.

2. The '305 patent issued on July 5, 2011 from U.S. patent application Serial No. 11/009,437, based upon a Notice of Allowance mailed December 20, 2010.

3. In an Office action dated June 15, 2010, the Examiner rejected application claims 1, 2, 5, 6, 8-12, 13, 17, 18 and 20-25 as anticipated by Freund U.S. Patent 5,987,611 and rejected claims 3, 4, 7, 9 and 14-16 as obvious over Freund in view of the skill in the art. (Ex. 1002, Ex. 1003.)

4. In response to the June 15, 2010 Office action, applicant amended application claims 1, 13 and 25 to recite that 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." No other changes were made to the pending application claims by applicant. (Ex. 1004.)

5. On December 20, 2010, a Notice of Allowability and Examiner's Amendment was mailed, wherein the Notice of Allowability states at recites at pages 3-4 that the prior art fails to disclose features, in combination with the 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." (Ex. 1005.)

STATEMENT OF PRECISE RELIEF REQUESTED

Petitioner respectfully requests the Board initiate an *inter partes* review and cancel Claims 1-25 of the '305 patent as unpatentable pursuant to 35 U.S.C. § 311(b) based on the two grounds of unpatentability discussed in detail herein.

THRESHOLD REQUIREMENT FOR *INTER PARTES* REVIEW

A petition for *inter partes* review must demonstrate "a reasonable likelihood that the petitioner would prevail with respect to at least one of the claims challenged in the petition." (35 U.S.C. § 314(a).) The Petition meets this threshold. Each of the elements of Claims 1-25 of the '305 patent is taught in the prior art as explained below and in the accompanying Declaration of Eugene H. Spafford, Ph.D. (Ex. 1006). Also,

where the claims are rendered obvious by combinations of prior art, the reasons to combine are established under 35 U.S.C. § 103(a).

STATEMENT OF REASONS FOR RELIEF REQUESTED

I. Introduction to the Technology of the '305 Patent

The '305 patent is entitled "Method and System for Adaptive Rule-Based Content Scanners for Desktop Computers" and generally is directed to anti-virus software for analyzing the behavior of executable program code downloaded from the Internet to determine whether the program code will engage in harmful or malicious behavior. (Ex. 1006 at ¶ 23.) Rather than analyzing the program code simply to identify a sequence of bytes known to be malicious, a so-called "virus signature," the '305 patent describes a rule-based system in which a lexical analysis of the code is conducted, i.e., an analysis that takes into account the grammar and punctuation using rules developed for a specific language, such as HTML or JavaScript. (*Id.* ¶ 24.)

The '305 patent describes that unlike prior art scanners "that are hard-coded for one particular type of content" the disclosed "adaptive rule-based scanners" ("ARB scanners") "can be enabled to scan any specific type of content by providing appropriate rule files, without the need to modify source code. (Ex. 1001 at 2:10-18.) "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." (*Id*. at 2:20-27; Ex. 1006 at ¶ 25.)

As described in the '305 patent, the rules for specific languages are stored in a rules database, which may be periodically updated with a rule update manager to incorporate new rules that become available. (Ex. 1001 at 2:37-52; Ex. 1006 at ¶ 25.)

II. The Claims of the '305 Patent

Claim 1 is directed to a security system for scanning content within a computer, the system comprising a network interface; a database of parser and analyzer rules, where the rules describe the potential compute exploits as patterns of types of tokens where the tokens include a punctuation type, an identifier type, and a function type; a network traffic probe that selectively diverts the incoming content to the rule-based content scanner, and a rule update manager for updating the database of parser and analyzer rules. (Ex. 1001.) Dependent claims 2-12 add non-inventive aspects to the alleged invention of claim 1, which were initially rejected over the prior art during prosecution, and subsequently allowed based on amendments to the independent claim. (Ex. 1004, 1005, Ex. 1006 at ¶ 28.)

Independent claim 13 recites method equivalent limitations of the system limitations of claim 1, while independent claim 25 is directed to a computer readable medium that stores program code for performing the method steps recited in

independent claim 13. Dependent claims 14-24 recite method equivalents of the system limitations recited in claims 2-12, respectively. (Ex. 1006 at ¶¶ 29-30.)

Table 1 below illustrates the one-to-one correspondence of the limitations of claims 1-12 to claims 13-24, in which certain limitations of claim 13 have been rearranged to demonstrate such correspondence. (Ex. 1006 at ¶ 31.) Table 2 below illustrates the one-to-one correspondence of the limitations of claims 1 and claim 25, in which certain limitations of claim 25 have been rearranged to demonstrate such correspondence. (*Id.*) To reduce redundancy and improve readability of the Petition, corresponding limitations of the various claims, as designated in Tables 1 and 2 below, will be used for purposes of establishing correspondence between the limitations of the claims of the '305 patent and the prior art.

Table 1

Correspondence Between Limitations of Claim Sets 1-12 and 13-24	
1(a). A security system for scanning content within a computer, comprising:	**13(a)**. A method for scanning content within a computer, comprising:
1(b) a network interface, housed within a computer, for receiving incoming content from the Internet on its destination to an Internet application running on the computer;	**13(b)** receiving, at the computer, incoming content from the Internet on its destination to an Internet application;
1(c)(1) a database of parser and analyzer rules corresponding to	**13(e)(1)** 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,	computer exploits, computer exploits being portions of program code that are malicious,
1(c)(2) 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;	**13(e)(2)** 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
1(d) 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;	**13(d)** scanning, by the computer, the selectively diverted incoming content to recognize potential computer exploits therewithin, based on **[13(e)(1) and (e)(2)]** ; and
1(e) a network traffic probe, operatively coupled to said network interface and to said rule-based content scanner, for selectively diverting incoming content from its intended destination to said rule-based content scanner; and	**13(c)** selectively diverting, by the computer, the received incoming content from its intended destination;

1(f) 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.	**13(f)** updating the database of parser and analyzer rules periodically to incorporate new behavioral rules that are made available.
2. The security system of claim 1 wherein said database of parser and analyzer rules stores parser and analyzer rules in the form of pattern-matching engines.	**14.** The method of claim 13 wherein said database of parser and analyzer rules stores parser and analyzer rules in the form of pattern-matching engines.
3. The security system of claim 2 wherein the pattern-matching engines are deterministic finite automata.	**15.** The method of claim 14 wherein the pattern-matching engines are deterministic finite automata.
4. The security system of claim 2 wherein the pattern-matching engines are non-deterministic finite automata.	**16.** The method of claim 14 wherein the pattern-matching engines are non-deterministic finite automata.
5. 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.	**17.** The method of claim 13 further comprising preventing incoming content having a computer exploit that was recognized by said scanning from reaching its intended destination.

6. The system of claim 1 wherein the incoming content received from the Internet by said network interface is HTTP content.	**18.** The method of claim 13 wherein the incoming content received from the Internet by said network interface is HTTP content.
7. The system of claim 1 wherein the incoming content received from the Internet by said network interface is HTTPS content.	**19.** The method of claim 13 wherein the incoming content received from the Internet by said network interface is HTTPS content.
8. The system of claim 1 wherein the incoming content received from the Internet by said network interface is FTP content.	**20.** The method of claim 13 wherein the incoming content received from the Internet by said network interface is FTP content.
9. The system of claim 1 wherein the incoming content received from the Internet by said network interface is SMTP content.	**21.** The method of claim 13 wherein the incoming content received from the Internet by said network interface is SMTP content.
10. The system of claim 1 wherein the incoming content received from the Internet by said network interface is POP3 content.	**22.** The method of claim 13 wherein the incoming content received from the Internet by said network interface is POP3 content.
11. The system of claim 1 wherein the destination Internet application is a web browser.	**23.** The method of claim 13 wherein the destination Internet application is a web browser.
12. The system of claim 1 wherein the destination Internet application is an e-mail client.	**24.** The method of claim 13 wherein the destination Internet application is an e-mail client.

Table 2

Correspondence Between Limitations of Claims 1 and 25	
1(a). A security system for scanning content within a computer, comprising:	**25(a)**. A computer-readable storage medium, the medium excluding signals, storing program code for causing a computer to perform the steps of:
1(b) a network interface, housed within a computer, for receiving incoming content from the Internet on its destination to an Internet application running on the computer;	**25(b)** receiving incoming content from the Internet on its destination to an Internet application;
1(c)(1) 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,	**25(e)(1)** a database of parser and analyzer rules corresponding to computer exploits, computer exploits being portions of program code that are malicious,
1(c)(2) 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;	**25(e)(2)** wherein the parser and analyzer rules describe 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;
1(d) a rule-based content scanner that communicates with said database of parser and analyzer rules, operatively	**25(d)** scanning the selectively diverted incoming content to recognize potential exploits therewithin, based on **[25(e)(1)**

coupled with said network interface, for scanning incoming content received by said network interface to recognize the presence of potential computer exploits therewithin;	**and (e)(2)]**; and
1(e) a network traffic probe, operatively coupled to said network interface and to said rule-based content scanner, for selectively diverting incoming content from its intended destination to said rule-based content scanner; and	**25(c)** selectively diverting the received incoming content from its intended destination;
1(f) 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.	**25(f)** updating the database of parser and analyzer rules periodically to incorporate new parser and analyzer rules that are made available.

III. The Ex Parte Reexamination History of the '305 Patent

Claims 1, 2, 5, and 13 of the '305 patent stand rejected in *ex parte* reexamination as obvious over U.S. Patent No. 8,140,660 to Wells and obvious over U.S. Patent Application No. 2005/0172338 to Sandu in view of Wells. (Ex. 1007 at 3, 22.) Reexamination was not requested for any other claims. Patent Owner has appealed the final rejection to the Board. (Ex. 1008.)

IV. Priority Date of the Claims for Which Review Is Sought

The '305 patent issued from U.S. Patent Application Serial No. 11/009,437, filed Dec. 9, 2004, which is a continuation-in-part of U.S. Patent Application Serial No. 10/930,884, filed on Aug. 30, 2004, now Patent No. 8,225,408 ("the '408 patent"), which is a continuation-in-part of U.S. Patent Application Serial No. 09/539,667, filed on Mar. 30, 2000, now Patent No. 6,804,780, which is a continuation of application No. 08/964,388, filed on Nov. 6, 1997, now Patent No. 6,092,194 ("the '194 patent"). (Ex. 1001 at cover.) New disclosure was added in each of the continuation-in-part applications filed on August 30, 2004 and December 9, 2004. (Ex. 1006 at ¶¶ 11-12.)

In ongoing litigation between Petitioner and Patent Owner, Patent Owner twice has represented that the claims of the '305 patent are entitled to a priority date of December 9, 2004. (Ex. 1009 at 11 ("each of the asserted claims [1, 2, 5-7, 10-14, 17-19, 22-25] of the '305 [sic] Patent is entitled to the priority date of December 9, 2004."); Ex. 1010 at 15 ("each of the asserted claims [1, 2, 7, and 10] of the '305 [sic] Patent is entitled to the priority date of December 9, 2004.")) Also, during the ongoing *ex parte* reexamination, Patent Owner did not challenge the prior art status of the two prior art references cited in rejecting the claims; that prior art had priority dates of July 21, 2003 and January 30, 2004. Accordingly, there is no dispute that claims for which review is sought here are entitled to no earlier date than December 9, 2004. (Ex. 1006 at ¶¶ 13-14.)

In addition, during the prosecution of the ’305 patent, in a Non-Final Office Action mailed June 15, 2010, the Examiner rejected all of the pending application claims under 35 U.S.C. 112(a) as lacking written description. Specifically, the application claims reciting “patterns of types of tokens” were rejected as “depart[ing] from the recitations found within the applicant’s disclosure and are not standard among those of ordinary skill in the art.” (Ex. 1003 at 4.) In a response filed September 15, 2010, applicant responded by amending, *inter alia*, independent claim 1 to recite “and types of tokens comprising a punctuation type, an identifier type and a function type.” (Ex. 1004 at 2-5; Ex. 1006 at ¶¶ 15-16.)

A detailed review of the specification of the ’194 patent (the earliest possible priority claim) reveals no support for the limitation “types of tokens comprising a punctuation type, an identifier type and a function type” as set forth in all of the issued claims. (Ex. 1006 at ¶ 17.) Indeed, the word “token” appears nowhere in the ’194 patent. (*Compare* Ex. 1001 *with* Ex. 1011; Ex. 1006 at ¶ 17.) Furthermore, none of the Figures in the ’305 patent (especially those relating to parsing and tokens, *e.g.*, Figures 2-6, and 8) appear in the ’194 patent. (*Compare* Ex. 1001 *with* Ex. 1011; Ex. 1006 at ¶ 17.) None of the disclosure of the specification cited by the Applicant as providing support for the “types of tokens comprising a punctuation type, an identifier type and a function type” is found in the specification of the ’194 patent. (*Compare* Ex. 1004 *with* Ex. 1011; Ex. 1006 at ¶ 17.) Instead, the portions of the specification of

the ’305 patent cited as supporting the “types of tokens” limitation appear only in the specification of the ’408 patent, filed August 30, 2004. (*See* Ex. 1004; Ex. 1006 at ¶ 17.)

Irrespective whether the priority date should be December 9, 2004 (as Patent Owner alleges in its infringement contentions) or August 30, 2004, based on the filing date of the ’408 patent, the references that Petitioner relies upon herein qualifies as prior art as discussed below.

V. Construction of the Claims

A claim subject to IPR is given its “broadest reasonable construction in light of the specification.”[2] (*See* 37 C.F.R. § 42.100(b)). As the Federal Circuit stated “the PTO must give claims their broadest reasonable construction consistent with the specification. Therefore, we look to the specification to see if it provides a definition for claim terms, but otherwise apply a broad interpretation.” (*In re ICON Health and Fitness, Inc*. at 496 F.3d 1374, 1379 (Fed. Cir. 2007).) Petitioner identifies explanations of certain claim terms as agreed upon in the pending litigation between the parties and/or in other Patent Office proceedings involving the ’305 patent.

[2] Petitioner reserves the right to pursue different constructions in litigation. *In re Zeltz*, 893 F.2d 319, 321 (Fed. Cir. 1989).

A. **"database"**

Petitioner and Patent Owner have agreed in the pending district court litigation that the term "database" means "a collection of interrelated data organized according to a database schema to serve one or more applications." (Ex. 1009 at 34-35.) Petitioner proposes that such a construction is consistent with the understanding of a person of ordinary skill in the art as of the effective filing date of the '305 patent discussed above. (Ex. 1006 at ¶ 33.)

B. **"parser rules"**

"Parser rules" are "rules that identify patterns of tokens." The specification explains that "parsing rules" are used to "identify groups of tokens as a single pattern." (Ex. 1001 at 10:53-55.) In addition, the specification teaches that parsing rules "recognize a pattern" and that the parser rules must minimally include "a pattern." (*Id.* at 11:6-7; 12:27.) Examples of parser rules are provided in Appendix A and the rules are drafted as patterns of tokens. (*Id.* at 43-45; 21:50-30:41.) Moreover, Figures 3-5 teach parsing data by means of pattern recognition, while Figure 6 expressly performs a check on an incoming pattern to see if "is there a pattern match with a parser rule." (*Id.* at Figs. 3-6.) Finally, claim 1 recites that parser rules "describe computer exploits **as patterns of types of tokens.**" *Id.* at claim 1 (emphasis added). This construction is consistent with the understanding of a person of ordinary skill in the art as of the effective filing date of the '305 patent. (*See also* Ex. 1006 at ¶ 34.)

C. **"analyzer rules"**

"Analyzer rules" are rules that provide "a generic syntax pattern of tokens that indicates a potential exploit." As described in the specification of the '305 patent, "an analyzer rule specifies a generic syntax pattern in the node's children that indicates a potential exploit." (*Id.* at 12:60-62.) Claim 1 confirms such an interpretation by requiring that analyzer rules "describe computer exploits as patterns of type of tokens." (*Id.* at claim 1.) The foregoing construction of the term "analyzer rules" is consistent with how a person of ordinary skill would have understood that term as defined in the specification of the '305 patent. (*See also* Ex. 1006 at ¶ 35.)

D. **"function type"**

A "function type" token is an "identity token that represents a function name." First, the '305 patent does not expressly describe a token that is a "function type" as set forth in the claims. Instead, the '305 patent discloses three types of tokens (1) an identity token; (2) a punctuation token; and (3) a keyword token. (Ex. 1001 at 10:39-42.) Specifically, the text of the specification states "tokens may include inter alia, an **IDENT token** for the name of a variable or function, various **punctuation tokens**, and **tokens for keywords** such as NEW, DELETE, FOR, and IF." (*Id.* (emphasis added).) Thus, while the specification explains that the identity tokens may contain the name of a variable or a function it does not expressly call an identity token that represents a function a "function token." Instead, the specification implies that a function token is

`a subset of the identity token. For example, the specification teaches that "if a node represents an IDENT token for the name of a variable, then the value of the node is the variable name; and if the node represents a rule regarding a pattern for a function signature, then the value of the node is the function name." (*Id.* at 11:2-5; Ex. 1006 at ¶ 36.) Moreover, Appendix A explains that a function signature (i.e. FuncSig) consists of a pattern of tokens: "(FUNCTION) (IDENT?) (list)." (Ex. 1001 at 23-24.) Here, the FUNCTION token is followed by zero or one IDENT tokens followed by a list of additional tokens. (Ex. 1006 at ¶ 36.) One of skill in the art would have recognized this pattern to be associated with the typical function call syntax including a function name followed by a list of arguments. (*Id.*) Thus, one of skill in the art would have understood that a function type token was simply a type of identity token that represented a function name. (*Id.*)

However, if a "function type" token is not construed to be a subset of the identity token, then the other possible interpretation of a "function type" token is that it corresponds to the "keyword type" token as discussed in the specification. (Ex. 1001 at 10:39-42.) But, doing so would inappropriately rewrite the claim language.

Irrespective of the construction the Board chooses, the prior art teaches a "function type" token that is a type of identity token representing a function name as well as a keyword type token. (Ex. 1006 at ¶ 37.)

VI. Claim-By-Claim Explanation of Grounds for Unpatentability

Unpatentability of claims 1-25 of the ’305 patent are set forth below and discussed in the declaration by Eugene H. Spafford, Ph.D whose *curriculum vitae* is presented as Exhibit A in Ex. 1006.

Ground 1. Chandani Anticipates Claims 1-25 of the ’305 Patent

U.S. Patent No. 7,636,945 to Anjali Chandnani et al. (“Chandnani”), entitled “Detection of polymorphic script language viruses by data driven lexical analysis,” (Ex. 1013) expressly or in combination with the skill in the art, anticipates at claims 1-25 of the ’305 patent. Chandnani teaches each and every limitation of these claims.

A. Chandnani Is Prior Art Under 35 U.S.C. § 102(e)

Chandnani is prior art under 35 U.S.C. 102(e) because it was filed on July 14, 2001, before either the December 9, 2004 or the August 30, 2004 effective priority dates to which the ’305 patent is entitled. (*See* § III.) Chandnani is not cumulative to the prior art of record and was not cited or addressed during the prosecution of the application for the ’305 patent. (*See* Ex. 1013.)

B. Independent Claims 1, 13 and 25

(1) Preamble – claim limitations 1(a), 13(a), and 25(a)

The preamble of claim 1 recites “A security system for scanning content within a computer” while claim 13 recites “A method of scanning content within a computer” (Table 1). Claim 25 recites “A computer-readable storage medium, the medium excluding signals, storing program code for causing a computer to perform the

[recited] steps." (Table 2). Chandnani discloses a system that meets each of these preambles. (Ex. 1006 at ¶¶ 40-43.)

Chandnani describes "the detection of polymorphic script language viruses by using data driven lexical analysis" and more particularly, "a method and apparatus for detecting script language … [that] includes a script language processor, a detection data processor and a detection engine ... [wherein] the detection engine lexically analyzes a data stream using the language description data and the detection data to detect the viral code." (Ex. 1013 at 1:16-18, Abstract, Figs. 1-2 (illustrating systems) and Figs. 3-7 (illustrating methods).) (Ex. 1006 at ¶ 41.)

In Chandnani, the content is "scanned" to detect the presence of viral code by tokenizing a file into a data stream and comparing the data stream to language definition data. (Ex. 1013 at 8:42 ("The data stream corresponding to a file to scan is tokenized by lexical analysis. The data stream is fed to a lexical analyzer (not shown) in the detection engine which generates a stream of tokens … The data stream is lexically analyzed again, this time using the language definition data, to generate a stream of tokens"), 7:42-45 (discussing various types of scanning of the data stream, such as pattern matching, pattern searching and/or virus or CRC signature checking), Fig. 2 (depicting a script language virus detection apparatus).) (Ex. 1006 at ¶ 42.)

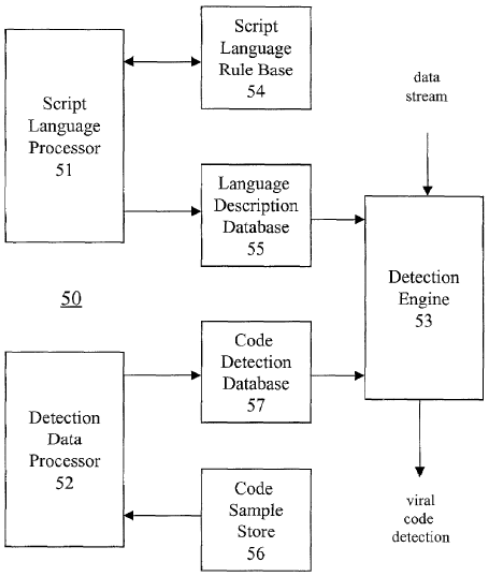

FIG. 2

Chandnani discloses that its systems and methods are directed to computer security: "provides tools (in the form of apparatus, systems and methods) for detecting script language viruses by performing a lexical analysis of a data stream on a computing device/system." (Ex. 1013 at 4:32-35.) Chandnani teaches the preambles of each of claims 1, 13 and 25. (Ex. 1006 at ¶ 43.)

(2) Claim limitations 1(b), 13(b), and 25(b) – receiving incoming content

Claim limitation 1(b) recites "a network interface, housed within a computer, for receiving incoming content from the Internet on its destination to an Internet application running on the computer," while claim limitations 13(b) and 25(b) recite similar limitations of "receiving, at the computer, incoming content from the Internet on its destination to an Internet application" (13(b)) and "receiving incoming content

from the Internet on its destination to an Internet application" (25(b)). Chandnani discloses these limitations. (Ex. 1006 at ¶¶ 44-48.)

Chandnani's security system includes a network interface housed within a computer: "[c]omputer system 70 comprises . . . a wired or wireless connection to a network 78." (Ex. 1013 at 4:44-49 and Fig. 1 (network interface 78), 9:12-16.) Chandnani discloses that the "network can be, for example, a LAN, a WAN, an intranet, an extranet, the Internet, and/or any combinations of such networks." (*Id*. at 4:49-51; Ex. 1006 at ¶ 45.)

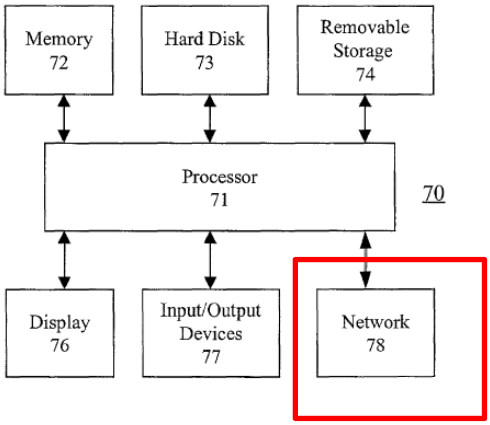

FIG. 1

The network interface receives incoming content from the Internet on its destination to an Internet application running on the computer: "a subject file may be downloaded to the computer system or computer through network 78." (Ex. 1013 at 4:61-62.) Chandnani explains that "[the] data stream may be generated from a potentially infected file," which may be "received via a network, such as the Internet."

(*Id*. at 4:35-38.) In Chandnani, virus detection may be performed on the content as it comes into the computer from the network, as opposed to some later time: "the script language virus detection methodologies may be performed on a file (or a data stream received by the computer through a network) before the file is stored/copied/executed/ opened on the computer." (*Id.* at 9:12-16.) Further, Chandnani teaches that viruses may be downloaded in script languages such as VBScript or JavaScript: "The advantages of scripts have motivated the proliferation of languages, such as VBScript (Visual Basic Script) and JavaScript specifically designed for expressing computer scripts. A script language virus is written using a script language and targeted at scripts expressed in the same script language." (Ex. 1013 at 3:3-8). Accordingly, a person of ordinary skill would have understood that Chandnani described receiving a file from the Internet (*e.g.*, over HTTP or HTTPS) that may contain a JavaScript and was destined for an Internet application (*e.g.*, a web browser). (Ex. 1006 at ¶¶ 46-47.)

Accordingly, Chandnani meets claim limitations 1(b), 13(b), and 25(b) by teaching a network interface, housed within a computer (*e.g.*, network interface 78 inside computer 70) for receiving incoming content (*e.g.*, an infected file) from the Internet on its destination to an Internet application (*e.g.*, a web browser) running on the computer (*e.g.*, computer system 70). (Ex. 1006 at ¶ 48.)

(3) Claim limitations 1(c)(1), 13(e)(1), and 25(e)(1) – database of parser and analyzer rules

Claim limitation 1(c)(1) requires "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…" Claim limitations 13(e)(1) and 25(e)(1) differ from 1(c)(1) in that they omit the phrase "stored within the computer" (inherent in the preamble of claim 25). Chandnani discloses all of these limitations. (Ex. 1006 at ¶¶ 49-57.)

Chandnani teaches "parser rules" (referred to in Chandnani as "language definition rules" and "language description data") that are used to detect computer exploits (*e.g.*, "viral code"). As discussed below in Section VI.Ground 1.b.(4) with respect to claims limitations 1(c)(2), 13(c)(2) and 25(c)(2), "the parser … rules describe computer exploits as patterns of types of tokens, tokens being program code constructs." Chandnani's "language definition rules" – which are used to create language description data that is supplied to the detection engine – describe computer exploits as patterns of tokens, where the tokens are program code constructs. (Ex. 1006 at ¶¶ 49-51; *see also* Ex. 1013 at 6:24-34; Abstract ("detection engine lexically analyzes a data stream using the language description data and the detection data to detect the viral code. The language description data may correspond to language definition rules and language check rules."), Fig. 3 ("Lexically analyze data stream using language description data and detection data"), Fig. 7 ("Run lexical analysis on token stream using pattern match detection data and language description data").

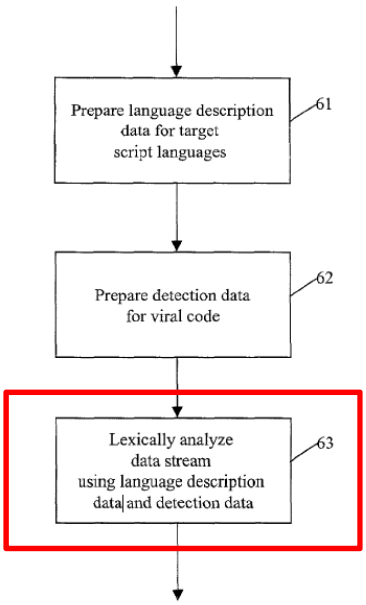

FIG. 3

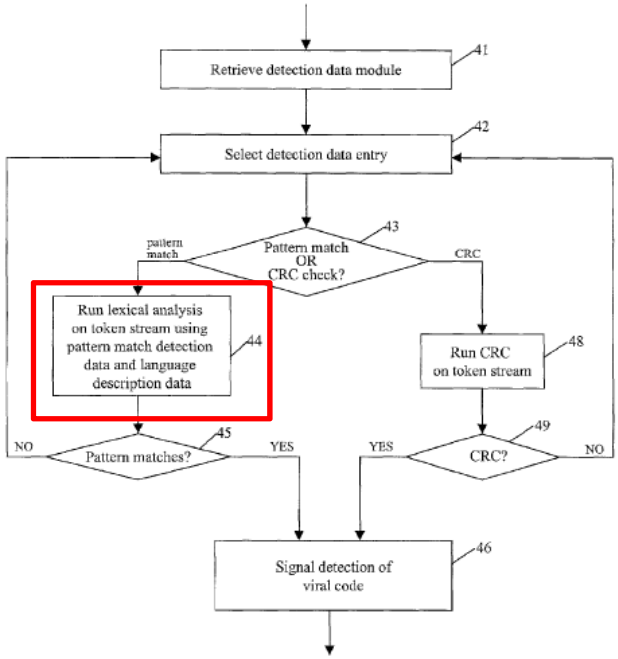

FIG. 7

Chandnani teaches that its parser rules (*i.e.*, language definition rules) consist of patterns of tokens, which correspond to "constructs": "tokens may correspond to respective language constructs" and are generated from a data stream "using lexical analysis." (Ex. 1013 at 3:65-67; *see also id.* at 5:15-19 ("The constituent parts/constructs of a target script language include, for example, operator symbols …, identifiers …, delimiters …, keywords …, numbers, blank spaces, etc.").) The '305 patent likewise describes tokens as "lexical constructs for a specific language." (Ex. 1001 at Abstract.) Chandnani further explains that "[l]anguage definition rules [*i.e.*, parser rules] for a target script language describe the constructs of the target script language and any relations between the constructs (*e.g.*, in relation R1[,] construct c1 is followed by construct c2)." (Ex. 1013 at 5:24-27.) The "relations between the

constructs" correlates to the "patterns" of tokens described in limitation 1(c)(1), 13(e)(1), and 25(e)(1). (*See id.* at 5:24-27, 3:51-52 ("The lexical analysis may include one or more pattern matches based on the language definition rules."), 5:53-68 ("[E]ach script language definition may be a corresponding set of pattern matching rules.").) Accordingly, Chandnani's "language definition rules," which define patterns in the data stream in terms of lexical constructs (tokens), meet the definition of "parser rules" recited in the claims of the '305 patent. (Ex. 1006 at ¶¶ 51-53.)

Chandnani also discloses "analyzer rules" (referred to as "viral code detection data") that are used to detect computer exploits (*e.g.*, viral code). According to claim limitations 1(c)(2), 13(e)(2), and 25(e)(2), "analyzer rules describe computer exploits as patterns of types of tokens, tokens being program code constructs." (*See* Tables 1 and 2, *infra*.) Chandnani says the "viral code detection data" is created by a detection regimen with "layers of token pattern matching and/or CRC signature checking." (Ex. 1013 at 6:57-67, 3:44-49.) Detection data "may include multiple layers of tests," each of which "may be specified as a token pattern match methodology or a combination of token pattern match and CRC signature check methodologies." (*Id*. at 7:1-7.) "Token pattern match methodologies" are rules for identifying potentially malicious program code, which are computer exploits described as a pattern of tokens (program code constructs). (*See id*. at 7:8-15.) Just as in the '305 patent, Chandnani teaches that the patterns can be represented as Dynamic Finite Automata ("DFAs"). (*Compare id.* at

7:16-28 *with* Ex. 1001 at 11:9-21; Ex. 1006 at ¶¶ 54-55.) Moreover, by referring to "token pattern matching" as distinct from CRC signature checking, Chandnani highlights that "token pattern matching" differs from traditional signature scanning techniques. (Ex. 1006 at ¶ 54.)

Chandnani's detection engine uses these pattern-matching rules to identify potential exploits: "[t]he detection engine lexically analyzes a data stream using . . . the detection data to detect the viral code." (Ex. 1013 at 3:44-49; Fig. 2 ("Code Detection Database") and 8:43-60, Fig. 3 ("Prepare detection data for viral code"), 7:8-15 (example pattern-matching rules) and 9:24-34 (describing detection methodology as "a rule-based approach").) Chandnani provides an "example of a pattern match methodology corresponding to identifying characteristics of a sample viral code which corrupts system macros: (a) search for a pattern p1 corresponding to access to a system macro; and (b) if pattern p1 is found in (a), search for one or more patterns corresponding to modification, replacement or deletion of the system macro." (*Id.* at 8:8-15.) In the foregoing example, the computer exploit is "corrupt system macros" and it is described by a pattern of tokens (*i.e.*, (a) followed by (b) where (a) contains the token for p1 and (b) contains the token for a modification, replacement, or deletion. Thus, Chandnani's disclosure of "viral code detection data" that includes pattern matching rules based on tokens meets claim limitations 1(c)(2), 13(e)(2) and 25(e)(2). (Ex. 1006 at ¶ 56.)

Chandnani's parser rules (*i.e.*, "language description data …[which] correspond to language definition rules") and analyzer rules ("viral code detection data") are stored in the computer in a "database" as shown in Fig. 2, reproduced below:

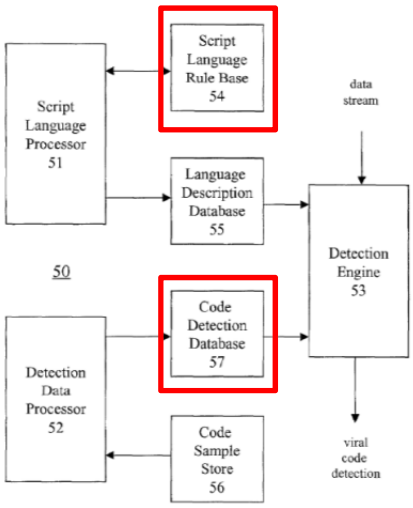

FIG. 2

(*See* Ex. 1013 at 5:41-49 and 6:66-57.) A person of ordinary skill would have understood that terms "Script Language Rule Base 54" and "Code Detection Database 57" depicted in Fig. 2 both refer to databases, *i.e.*, a collection of interrelated data organized according to a database schema to serve one or more applications. (Ex. 1006 at ¶ 57.) One of skill in the art would have further recognized that these two databases might actually be a single database, without any loss of functionality. (*Id.*) Chandnani teaches a "database of parser and analyzer rules" as required by claim limitations 1(c)(1), 13(e)(1), and 25(e)(1).

(4) Claim limitations 1(c)(2), 13(e)(2), and 25(e)(2) – patterns of types of tokens

Claim limitations 1(c)(2), 13(e)(2), and 25(e)(2) each require that "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." Chandnani discloses this limitation. (*Id.*at ¶¶ 58-64.)

As detailed above in § VI., Ground 1.B.(3), Chandnani teaches "parser and analyzer rules [that] describe computer exploits as patterns of types of tokens, tokens being program code constructs." Chandnani discloses that the types of tokens include "a punctuation type, an identifier type and a function type." Chandnani teaches that tokens may include keywords, symbols and identifiers. (Ex. 1013 at 5:15-19 "**identifiers** (e.g., "x1", "y2", etc.)," (emphasis added).). Chandnani further illustrates this by describing parsing and storing the expression that appears between an IF and THEN statement as its own token. (*Id.* at 6:16-23: "(3) store the expression between the keyword ''IF' found in (1) and the keyword 'THEN' found in (2), as an expression to be parsed.") A person of ordinary skill would have understood that an expression appearing between an IF and THEN could be a variable. (Ex. 1006 at ¶ 59.) For example, a common script block may take the form:

```
IF variable_name THEN {
        Expression_1;
        Expression_2;
}
```

(*Id.*) Chandnani teaches that the expression between the IF and THEN could be tokenized into its own token, e.g. "variable_name." (Ex. 1013 at 6:16-23.) These teachings of Chandnani correspond to the disclosure in the '305 patent that one possible "IDENT token" could comprise "the name of a variable." (Ex. 1001 at 10:39-41). Therefore, Chandnani's language definition rules includes parser rules that comprise an identifier type. (Ex. 1006 at ¶ 59.)

Chandnani also teaches that parser rules (i.e., language definition rules) consist of a "function type" token. (*Id.* at ¶ 60.) Chandnani teaches that patterns of tokens, correspond to "constructs": "tokens may correspond to respective language constructs" and are generated from a data stream "using lexical analysis." (Ex. 1013 at 3:65-67; *see also id.* at 5:15-19 ("The constituent parts/constructs of a target script language include, for example, operator symbols …, identifiers …, delimiters …, keywords …, numbers, blank spaces, etc.").) One skilled in the art would have recognized that functions would be included within "the constituent parts/constructs of a target script language" and would thus be recognized "using lexical analysis." (Ex. 1006 at ¶ 60.) Appendix A of the '305 patent shows how a function is recognized by parsing a FUNCTION followed by an identifier, (Ex. 1001 at 24-25), both of which are explicitly described as recognized by Chandnani: "Language definition rules for a target script language describe the constructs of the target script language and **any** relations between the constructs (e.g., in relation R1 construct c1 is followed by

construct c2)." (Ex. 1013. at 5:24-27) (emphasis added).) One of skill in the art would have recognized, therefore, that Chandnani teaches that tokens may comprise a function type because Chandnani teaches parsing an identity token c1 that is a function name when in the format c1 followed by a list c2. (Ex. 1006 at ¶ 60.)

If, however, "function type" is interpreted to mean "keyword type" then Chandnani also teaches this limitation because it teaches parsing the tokens for keywords such as IF or THEN. (*Id.; see also* Ex. 1013 at 6:10-17 ("(1) search for the keyword 'IF'; (2) search for the first instance of the keyword 'THEN'.)

The '305 patent explains that a punctuation type delimits a sequence of characters. (Ex. 1001 at 9:14-16: "A token is generally a sequence of characters delimited on both sides by a punctuation character, such as a white space.") As an example of a punctuation type token, the '305 patent expressly identifies the left curly bracket as a punctuation types. (Ex. 1001 at 23:30: "LBRACE "[!left_curly_bracket!]" punct;"), *id.* 9:14-16; *see also* Ex. 1006 at ¶ 62.) Similarly, Chandnani explains the need to parse and identify delimiter tokens and expressly notes that the left curly bracket is a key constituent part of a target script language. (Ex. 1013 at 10-19: "Before the analysis is commenced, target script languages, including their constituent parts, ... are identified/defined. The constituent parts/constructs of a target script language include, for example, operator symbols (*e.g.*,"+","=", etc.), identifiers (*e.g.*, "x1", "y2", *etc.*), ***delimiters*** (*e.g.*,"**{** ... }", "BEGIN ... END", *etc.*), keywords (*e.g.*, "IF

... THEN", "GOTO", *etc*.), numbers, blank spaces, *etc*.") (emphasis added).) Furthermore, Chandnani explains that its rules include instructions such as "search for a statement terminator after the keyword "THEN" [*e.g.*, a function type]." (*Id.* at 6:19-20.) With respect to the above example, a person of ordinary skill would have understood Chandnani as teaching to look for the left and right curly brackets after the THEN to determine that expression_1 and expression_2 are to be executed when the IF condition evaluates to true. (Ex. 1006 at ¶ 61.) Accordingly, Chandnani teaches rules based on tokens that include a punctuation type.

Chandnani's viral code detection data ("analyzer rules") also comprise "a punctuation type, an identifier type and a function type." Chandnani expressly teaches searching for and detecting script language viruses in a data stream involving "pattern matching" the data stream against both parser rules (*i.e.*, language description data) and analyzer rules (*i.e.*, viral code detection data). (*See* Ex. 1013 at 7:38-45.) Because Chandnani parses the incoming data stream and then matches the parsed tokens against the viral code detection data, Chandnani teaches that its analyzer rules require the same constituent part information as its parsing rules: the "data stream corresponding to a file to scan is tokenized by lexical analysis…which generates a stream of tokens. To tokenize the data stream, a script language used in the data stream is determined…the data stream is lexically analyzed again, this time using the language definition data, to generate a stream of tokens. As mentioned above, each generated token corresponds to

a specific language construct…" (*Id.* at 8:4-17.) Chandnani then teaches looking for a "pattern match…on the generated token stream." (*Id.* at 39-41.) Because Chandnani teaches tokenizing the data stream based on the parser rules and then pattern matching the tokens in the analyzer rules, the two sets of rules must contain the same types of tokens (*e.g.*, punctuation, function, and identifiers). (Ex. 1006 at ¶ 63.)

Accordingly, Chandnani also teaches that both the analyzer and parser rules consist of tokens and the types of tokens that comprise "a punctuation type, an identifier type and a function type" as required by claim limitations 1(c)(2), 13(e)(2), and 25(e)(2). (Ex. 1006 at ¶ 64.)

(5) Claim limitations 1(d), 13(d), and 25(d) – a rule-based content scanner

Claim limitation 1(d) recites "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" while 13(d) and 25(d) recite "scanning the selectively diverted incoming content to recognize potential computer exploits therewithin based on [the database of parser and analyzer rules discussed in limitations 13(e)(1) and (2) and 25(e)(1) and (2)]." (*See* Tables 1 and 2, *infra*.) Chandnani discloses these limitations. (Ex. 1006 at ¶¶ 65-68.)

Chandnani's scanner 44 depicted in Fig. 7 uses the databases discussed above with respect to Fig. 2, and constitutes a "rule-based content scanner for scanning

incoming content." (Ex. 1013 at Fig. 7.) Chandnani describes that the "token stream is analyzed lexically using the pattern match detection data and language description data." (*Id.* at 8:51-52; 3:65-67 and Fig. 6 and Fig. 7.) The lexical analysis uses rules stored in the databases to scan incoming data stream by attempting to pattern match tokens generated from the lexical analysis of the data stream. (*Id*. at 5:63-6:2; Fig. 2; Ex. 1006 at ¶ 66.)

Chandnani teaches using the rule based scanner to "recognize[] the presence of potential computer exploits therewithin." Specifically, Chandnani's detection engine performs "pattern match[ing] ... on the generated token stream," where a match indicates that "viral code has been detected." (Ex. 1013 at 8:36-42; Fig. 2.) The rule-based scanner communicates with the database of parser and analyzer rules: Fig. 2 shows Detection Engine in communication with Code Detection Database 57 (analyzer rules) and Language Description Database 55 (generated from the Script Language Rule Base 54 – parser rules). Chandnani explains that "the language description data provided by language description module 55 and the viral code detection data provided by code detection module 57 are used by detection engine 53 to analyze lexically the data stream and decide if the data stream contains viral code." (*Id.* at 7:38-45, Fig. 2.) (Ex. 1006 at ¶ 67.)

Chandnani also teaches that its rule-based scanner is "operatively coupled with [the] network interface" to scan "incoming content received by [the] network

interface." Chandnani's network interface is housed within computer system 70 for receiving incoming content (*e.g.*, "subject file") from the Internet. (*See* § III.) Chandnani explains that a "data stream may be generated from a … 'subject file'" (Ex. 1013 at 4:35-36) and "fed to a lexical analyzer in the detection engine" (Ex. 1013 at 8:4-17). Accordingly, Chandnani teaches all of claim limitations 1(d), 13(d), and 25(d). (Ex. 1006 at ¶ 68.)

(6) Claim limitations 1(e), 13(c), and 25(c) – a network traffic probe

Claim limitation 1[e] recites "a network traffic probe, operatively coupled to said network interface and to said rule-based content scanner, for selectively diverting incoming content from its intended destination to said rule-based content scanner." Claim limitations 13(c) and 25(c) omit the "network traffic probe" but recite a step of "selectively diverting … the received incoming content from its intended destination." Chandnani discloses these limitations. (Ex. 1006 at ¶¶ 69-71.)

Chandnani teaches two types of selective diversion of incoming content, either of which satisfies the claim limitations of the '305 patent. First, Chandnani expressly states that the "disclosure relates to the detection of ***script language*** viruses..." (Ex. 1013 at 1:15-16 (emphasis added); Abstract ("method and apparatus for detecting script languages."), Fig. 6 ("Lexically analyze data stream to determine appropriate script languages.")) Thus, Chandnani could be viewed as teaching diversion of only script languages to its rule-based scanner. (*See* Ex. 1006 at ¶ 70.)

Chandnani also teaches selectively diverting only a subset of incoming data to the lexical analysis engine: "a data stream received by the computer through a network" can be scanned by the rule-based content scanner (*e.g.*, item 44) "before the file is stored/copied/executed/opened on the computer." (Ex. 1013 at 12-16.) Thus, before an incoming file reaches its intended destination it may be scanned by the rule-based content scanner. In this case, decision box 43 of Fig. 7 serves as a network traffic probe by selectively diverting only some content to the rule-based content scanner, depending upon the result of the CRC check: at "step 43 . . . it is determined whether a selected check is a pattern match or virus or CRC signature check. If the check is a pattern match, the token stream is analyzed lexically... If it is determined in step 43 that the check is a CRC signature check, the CRC is run on the token stream (step 48)." (*Id.* at 8:48-55.) Thus, depending on the determination made by the network traffic probe, which has intercepted the file prior to it being stored/copied/executed/opened on the computer by step 43, only some data is diverted to the lexical analyzer before it is allowed to reach its intended destination. (*See* § V., Ground 1, ii.,(2); *see also* Ex. 1006 at ¶ 71.) Chandnani therefore teaches claim limitations 1(e), 13(c) and 25(c).

(7) Claim limitations 1(f), 13(f), and 25(f) – a rule update manager

Claim limitation 1(f) recites "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." Similarly, claim limitations 13(f) and 25(f) require "updating the database of parser and analyzer rules periodically to incorporate new behavioral rules that are made available." Chandnani teaches these limitations. (Ex. 1006 at ¶¶ 72-74.)

Specifically, Chandnani discloses that "[w]hen the script language virus detector determines that a data stream includes viral code, the positive identification of viral code may be fed back to the learning component for fortifying the rules in the rule base and/or adding additional rules." (Ex. 1013 at 9:29-33.) Chandnani teaches updating Script Language Rule Base 54 and Code Detection Database 57 of Fig. 2 (*i.e.*, the parser and analyzer rules) and thus satisfies claim limitations 1(f), 13(f) and 25(f). (Ex. 1006 at ¶ 73.)

Chandnani therefore anticipates each and every element of claims 1, 13 and 25 of the '305 patent. (Ex. 1006 at ¶ 74.)

C. **Dependent Claims 2-4 and 14-16 – pattern matching engines**

Claims 2 and 14 each recites "wherein said database of parser and analyzer rules stores parser and analyzer rules in the form of pattern-matching engines." Claim 2 depends from claim 1; claim 14 depends from claim 13. As discussed above, Chandnani teaches all of the limitations of claims 1 and 13. As further discussed in detail above, Chandnani expressly teaches that its "lexical analysis may include one or more pattern matches based on the language definition rules." (*Id.* at 3:52-53.)

Chandnani therefore teaches a "pattern matching engine," satisfying claims 2 and 14. (Ex. 1006 at ¶ 75.)

Claims 3 and 15 each recites "wherein the pattern-matching engines are deterministic finite automata." As described above, Chandnani describes that Dynamic Finite Automata may be used to conduct pattern matches, and corresponds to the "deterministic finite automata" described for conducting pattern matches in the '305 patent. (*Cf.*, Ex. 1013 at 3:55-60 and 7:16-27 to Ex. 1001 at 11:15-19 and 12:1-26; Ex. 1006 at ¶ 76.) Chandnani therefore teaches use of DFA as a "pattern matching engine" satisfying claims 3 and 15.

Claims 4 and 16 each recites "wherein the pattern-matching engines are non-deterministic finite automata." The '305 patent describes that DFA and NFA (non-deterministic finite automata) "are well known in the art of compilers as finite-state machines for pattern matching." (Ex. 1001 at 11:14-21.) Accordingly, a person of ordinary skill would have readily understood, in view of Chandnani's explicit teaching of the use of DFA for pattern matching, that substitution of NFA would have been obvious or obvious to try where exact evaluation of candidate rules is desired. (Ex. 1006 at ¶ 77). Accordingly, use of NFA pattern-matching techniques as required by claims 4 and 16 would have been obvious over Chandnani's teaching of use of DFA as a "pattern matching engine."

D. **Dependent Claims 5 and 17 – blocking infected content**

Claim 5 requires "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 17 requires a method equivalent step of "preventing incoming content having a computer exploit that was recognized by said scanning from reaching its intended destination." Claim 5 depends from claim 1; claim 17 depends from claim 13. As discussed above, Chandnani teaches all of the limitations of claims 1 and 13. (Ex. 1006 at ¶ 78.)

The '305 patent describes the content blocker with respect to Fig. 9, and states: "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." (Ex. 1001 at 19:48-51.) It is plain from the discussion in Chandnani of the "Description of Related Art" that the patent is directed to identifying the presence of "computer virus[es] … [that] perform unwanted, and also possibly harmful, actions on components of the computer and/or information stored on the computer." (Ex. 1013, 1:30-35.) Chandnani also discusses that "[v]irus scanner programs generally are successful only at ***eliminating*** viruses that are known to the scanner program." (*Id.* at 2:22-23 (emphasis added).) In view of these teachings and the general knowledge of a person of ordinary skill, such as person would have understood that "detection of a viral code is signaled" in Chandnani, the infected code would be prevented from

executing at the destination computer. (Ex. 1006 at ¶ 79.) Accordingly, a person of ordinary skill would have found it obvious, after use Chandnani's method of lexical analysis to identify viral code, to prevent such incoming infected content from reaching its intended destination as recited in claims 5 and 17. (*Id.*)

E. **Dependent Claims 6-10 and 18-22 – content types**

Claims 6-12 and 18-24 variously recite that "incoming content received from the Internet by said network interface" is HTTP content (claims 6 and 18), HTTPS content (claims 7 and 19), FTP content (claims 8 and 20), SMTP content (claims 9 and 21), and POP3 content (claims 10 and 22). All of claims 6-10 depend from claim 1, while claims 18-22 depend from claim 13. As discussed above, Chandnani teaches all of the limitations of claims 1 and 13. (Ex. 1006 at ¶ 80.)

Chandnani discloses communication between a computer system (*e.g.*, computer system 70) and a network (*e.g.*, a LAN, a WAN, an intranet, an extranet, the Internet). A person of ordinary skill therefore would have understood that Chandnani's reference to the Internet encompassed all forms of communication over the World Wide Web, including HTTP, HTTPS, FTP, SMTP and POP3 protocols, all of which were notoriously well-known by the effective filing date of the '305 patent. Chandnani expressly teaches that viruses may be downloaded in script languages such as VBScript or JavaScript. (Ex. 1013 at 3:3-8 ("The advantages of scripts have motivated the proliferation of languages, such as VBScript (Visual Basic Script) and JavaScript

specifically designed for expressing computer scripts.") And because JavaScript files are generally destined for a web browser or e-mail (*e.g.*, Internet Application) a person of ordinary skill would have understood that Chandnani discloses receiving data via any of the well-known HTTP, HTTPS, FTP, SMTP and POP3 protocols. (Ex. 1006 at ¶ 81.)

Accordingly, a person of ordinary skill would have found it obvious to use Chandnani's lexical analysis system to analyze content downloaded from the Internet using any of HTTP, HTTPS, FTP, SMTP and POP3 protocols, as recited in claims 6-12 and 18-24. (Ex. 1006 at ¶ 82.)

F. **Dependent Claims 11-12 and 23-24 – destination applications**

Claims 11 and 23 recite that "the destination Internet application is a web browser" while claims 12 and 24 recite that "the destination Internet application is an e-mail client." Claims 11 and 12 depend from claim 1; claims 23 and 24 depend from claim 13. Chandnani teaches all of the limitations of claims 11-12 and 23-24. (Ex. 1006 at ¶ 83.)

Chandnani teaches that viruses may be downloaded in script languages such as VBScript or JavaScript. (Ex. 1013 at 3:3-8 ("The advantages of scripts have motivated the proliferation of languages, such as VBScript (Visual Basic Script) and JavaScript specifically designed for expressing computer scripts.")) Chandnani also describes a "virus may spread from one computing machine to another by attaching to data

transmissions between the computer machines via a network or other transmission medium." (*Id.* at 2:3-5.) A person of ordinary skill would have understood that JavaScript files received over a network generally were destined for a web browser or e-mail (*e.g.*, Internet application) and that a conventional way of transmitting computer viruses is by e-mail. (Ex. 1006 at ¶¶ 81, 84.) Accordingly, it would have been obvious to use the lexical analysis method described in Chandnani for scanning incoming content destined for web browsers and e-mail clients. Thus, Chandnani renders claims 11-12 and 23-24 obvious. (*Id.*)

Ground 2. Freund in View of Chandnani Renders Claims 1-25 Invalid as Obvious under 35 U.S.C. § 103.

U.S. Patent No. 5,987,611 by Gregor Freund ("Freund"), entitled "System and methodology for managing internet access on a per application basis for client computers connected to the internet," (Ex. 1014) in view of Chandnani, renders obvious claims 1-25 of the '305 patent. Freund in view of Chandnani teaches each and every limitation of these claims. (Ex. 1006 at ¶ 85.) As discussed above in Section IV, *supra*, Freund was cited during prosecution of the '305 patent in an Office action mailed June 15, 2010 as anticipating or rendering obvious all of application claims 1-25. Applicant responded by amending independent claims 1, 13 and 25 to recite that the "types of tokens compris[e] a punctuation type, and identifier type and a function type" and further argued that Freund does not disclose "selectively diverting" incoming content, after which the application was allowed. (Ex. 1006 at ¶ 86.)

A. **Freund Is Prior Art Under 35 U.S.C. § 102(e)**

Freund is prior art under 35 U.S.C. 102(e) because it claims priority to an application filed on December 31, 1996, before the earliest effective filing date in 2004 to which the claims of the '305 patent are entitled. (*See* § III.)

B. **Independent Claims 1, 13 and 25**

(1) Preamble – claim limitations 1(a), 13(a), and 25(a)

The preamble of claim 1 recites "A security system for scanning content within a computer" while claim 13 recites "A method of scanning content within a computer" (Table 1). Claim 25 recites "A computer-readable storage medium, the medium excluding signals, storing program code for causing a computer to perform the [recited] steps." (Table 2.) Freund discloses a system that meets each of these preambles. (Ex. 1006 at ¶¶ 88-90.)

Freund teaches "system and methods for regulating access and maintaining security of individual computer systems and local area networks (LANs) connected to larger open networks (Wide Area Networks or WANs), including the Internet" and more specifically, "a system comprising one or more access management applications that set access rules for the entire LAN for one or more workgroups or individual users, a client-based filter application (installed at each client), and a central supervisor application that maintains the access rules for the client based filter." (Ex. 1014 at 1:24-30, 3:60-66, Figs. 1-7K (illustrating systems) and Figs. 8A-14 (illustrating

methods); *see also* Ex. 1006 at ¶ 89.)

In Freund, content is "scanned" to detect viruses and Trojan Horse programs by "filtering incoming data, including binary files." (Ex. 1014 at 9:14-16; *see also id.* 29:59-30:1 (The content is parsed to detect "syntax elements that are known or suspected to cause security or network problems. ") Freund therefore teaches the preambles of each of claims 1, 13, and 25. (Ex. 1006 at ¶ 90.)

(2) Claim limitations 1(b), 13(b), and 25(b) – receiving incoming content

Claim limitation 1(b) recites "a network interface, housed within a computer, for receiving incoming content from the Internet on its destination to an Internet application running on the computer," while claim limitations 13(b) and 25(b) recite similar limitations of "receiving, at the computer, incoming content from the Internet on its destination to an Internet application" (13(b)) and "receiving incoming content from the Internet on its destination to an Internet application" (25(b)). Freund discloses these limitations. (Ex. 1006 at ¶¶ 91-95.)

Freund's computer system includes a network interface housed within a computer: "System 100 comprises . . . a network interface card or controller 111 (*e.g.*, Ethernet), and a modem 112 (*e.g.*, 28.8K baud modem or ISDN modem)." (Ex. 1014 at 7:36-43 and Fig. 1 (Network Controller 111); Ex. 1006 at ¶ 92.)

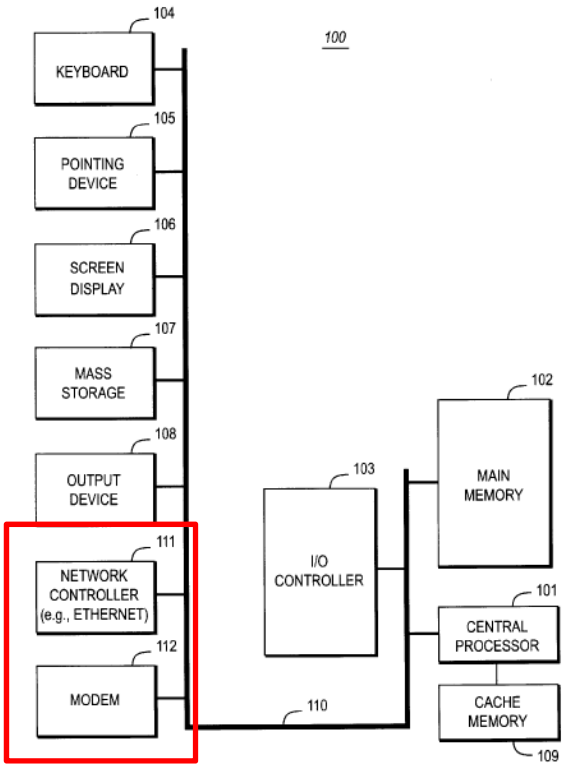

FIG. 1

The network interface receives incoming content from the Internet that is destined for an Internet application running on the computer: "The system itself communicates with other systems via a network interface card 111 … and/or modem 112." (Ex. 1014 at 7:53-54.) The system may communicate with the Internet through an internet browser to download data (*e.g.*, "foo.html"): "The system includes one or more clients, each operating applications or processes (*e.g.*, Netscape Navigator or Microsoft Internet Explorer browser software) requiring Internet (or other open network) access (*e.g.*, an Internet connection to one or more Web servers)" (*id*. at Abstract); "[a] Web browser also might implement other protocols, such as the older File Transfer Protocol (FTP) for downloading data" (*id.* at 12:22-24 "); "the

application contacts the host using Winsock Connect" (*id. at* Fig. 12A); "the application calls Winsock Send() with HTTP command 'Get foo.html'" (*id. at* Fig. 12B). (*See also* Ex. 1006 at ¶ 93.)

Further, the system of Freund may include a "mail client" application that receives e-mail from "an Internet-based mail server" and a "news collecting program" that receives news from the Internet. (Ex. 1014 at 10:20-27). The system includes "communication layer or driver 241 (*e.g.*, Winsock)" enabling the client application software to communicate with the Internet. (*Id.* at 8:8:10, Fig. 2 (Application Program(s) Browser 245), 10:16-20; *see also* Ex. 1006 at ¶ 95.)

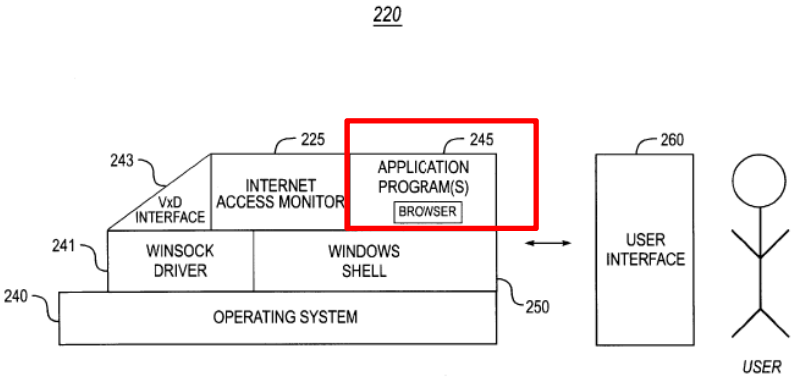

FIG. 2

Accordingly, Freund meets claim limitations 1(b), 13(b), and 25(b) by teaching a network interface, housed within a computer (*e.g.*, network interface card or controller 111) for receiving incoming content (*e.g.*, "foo.html") from the Internet on its destination to an Internet application (*e.g.*, a browser, ftp client, or e-mail client) running on the computer (*e.g.*, system 100). (Ex. 1006 at ¶ 95.)

(3) Claim limitations 1(c)(1), 13(e)(1), and 25(e)(1) – database of parser and analyzer rules

Claim limitation 1(c)(1) requires "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…" Claim limitations 13(e)(1) and 25(e)(1) differ from 1(c)(1) in that they omit the phrase "stored within the computer" (inherent in the preamble of claim 25). Freund discloses all of these limitations.

Freund teaches a database of parser rules (referred to in Freund as "rules database or knowledgebase 570") that are used to detect computer exploits (*e.g.*, suspected "syntax elements" of a file): "To intelligently determine the action which it should undertake for a given message, the module 560 refers to a rules database or knowledgebase 570. Here, the various rules which define permitted activity in the Internet-based system are stored in a format which is readily accessible by the data interpretation module 560. Contained within the rules database 570 are individual databases fully characterizing the administrator-specified rules for the system." (Ex. 1014 at 21:26-37, 29:67-30:1; Fig. 5 ("rules knowledgebase 570"); *see also* Ex. 1006 at ¶ 97.)

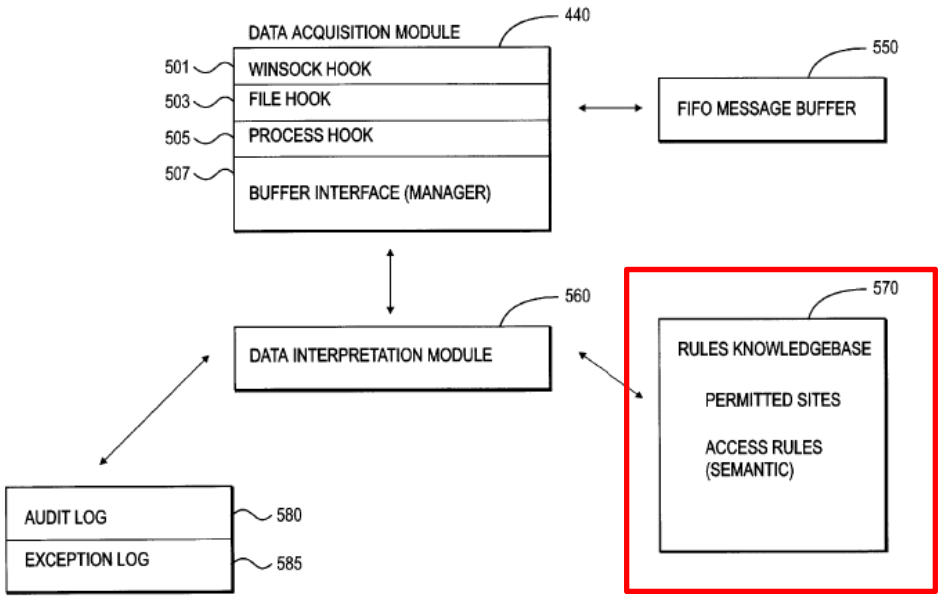

FIG. 5

Freund explains that its parser rules (*i.e.*, rules stored in "rules database or knowledgebase 570") are used to parse the contents of a file. That is, Freund teaches that a host (*i.e.*, "Web Server 350") sends the contents of foo.html to a client. (Ex. 1014 at 29:56-57, Fig. 12C (step 1218); *see also* 14:37-38 ("Host is a Third party server program that can be contacted through the Internet"); Ex. 1006 at ¶ 98.)

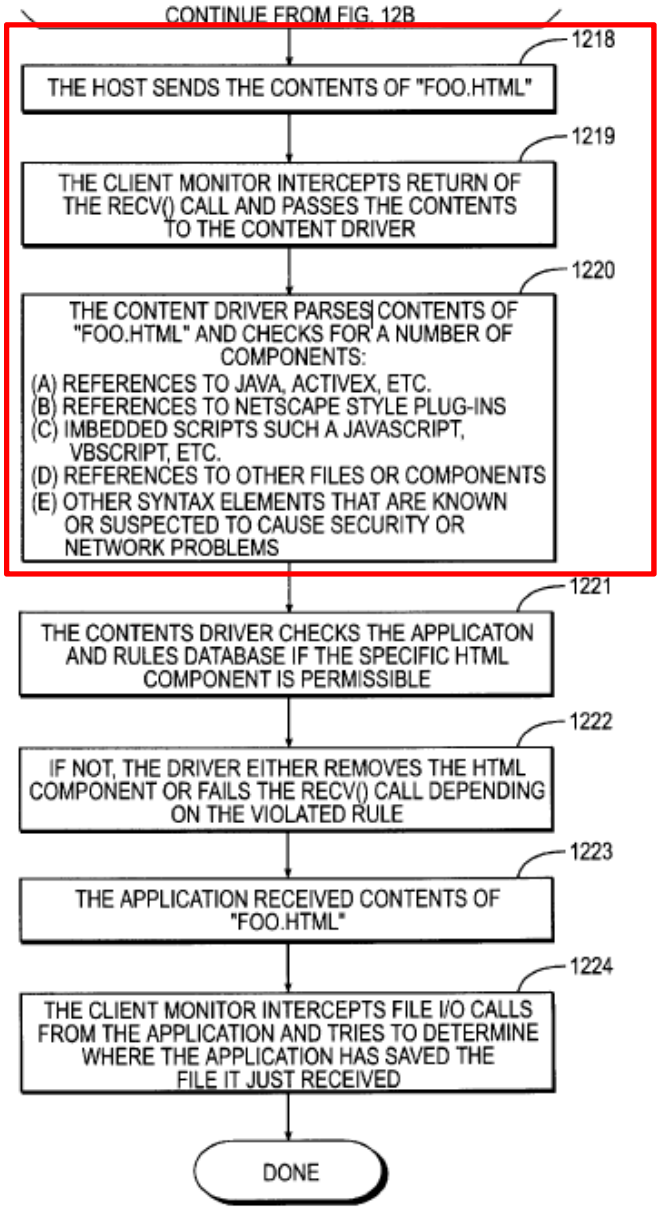

FIG. 12C

Freund clarifies that the "Client Monitor" then passes the contents of foo.html to the "content driver" to be parsed and checked for HTML components – which include applet tags, object tags, embed tags, imbedded scripts, references to files or components (*e.g.*, system tags), and syntax elements – that are known or suspected to cause security or network problems:

> "At step 1216, the Client Monitor loads the content driver for ".html" files … At step 1218, the Host sends the contents of "foo.html". At step 1219, the Client Monitor intercepts the return of the recv() call and passes the contents to the content driver. At step 1220, the content driver parses the contents of "foo.html" and checks for the following components: (a) References to Java™, ActiveX, and the like (<APPLET> or <OBJECT> tags); (b) References to Netscape style plug-ins (<EMBED> tag); (c) Imbedded scripts such as Java Script™, VBScript, and the like (<SCRIPT> tag); (d) References to other files or components (<A HREF=\f "Symbol">, or <IMG SRC=\f "Symbol"> tags); and (e) Other syntax elements that are known or suspected to cause security or network problems."

(*Id*. at 29:54-67; Fig. 12C (steps 1218-1220); *see also* Ex. 1006 at ¶ 99.)

Freund's system includes a "driver supporting HTTP, FTP, SMTP, and POP3 protocols" for parsing a variety of content including, "HTTP files, executable files, ZIP files, ActiveX controls, or Java classes." (Ex. 1014 at 23:46-54; *see also* Ex. 1006 at ¶ 100.)

Freund also discloses a database of analyzer rules (referred to in Freund as "rules database or knowledgebase 570") that is used to detect computer exploits (*e.g.*, suspected "syntax elements" of a file): "At step 1221, the Contents driver checks the application and rules database to see if the specific HTML component is permissible." (Ex. 1014 at 30:1-9; *see also* Ex. 1006 at ¶ 101.)

Freund's content driver uses its database of parser and analyzer rules to check a file for applet tags, object tags, embed tags, imbedded scripts, references to files or components, and syntax elements corresponding to computer exploits. (Ex. 1014 at 29:54-30:1; Fig. 12C (step 1221); *see also id*. at 30:1-9 ("Contents driver checks the

application and rules database to see if the specific HTML component is permissible."); Ex. 1006 at ¶ 102.)

Freund's database of parser and analyzer rules is stored within the computer: "Internet-based (client/server) system 300 … includes multiple clients 310 (*e.g.*, clients 310a, 310b, 310c, each of which comprises a personal computer or workstation." (Ex. 1014 at 14:52-57.) "Each client includes a client-side monitoring component for monitoring Internet . . . , as specifically shown at 311a, 311b, and 311c." (*Id.* at 14:59-62, Fig. 3A (depicting a client-side monitor (*e.g.*, "Monitor 311a") housed within "Client 310a"; *see also* Ex. 1006 at ¶ 103.)

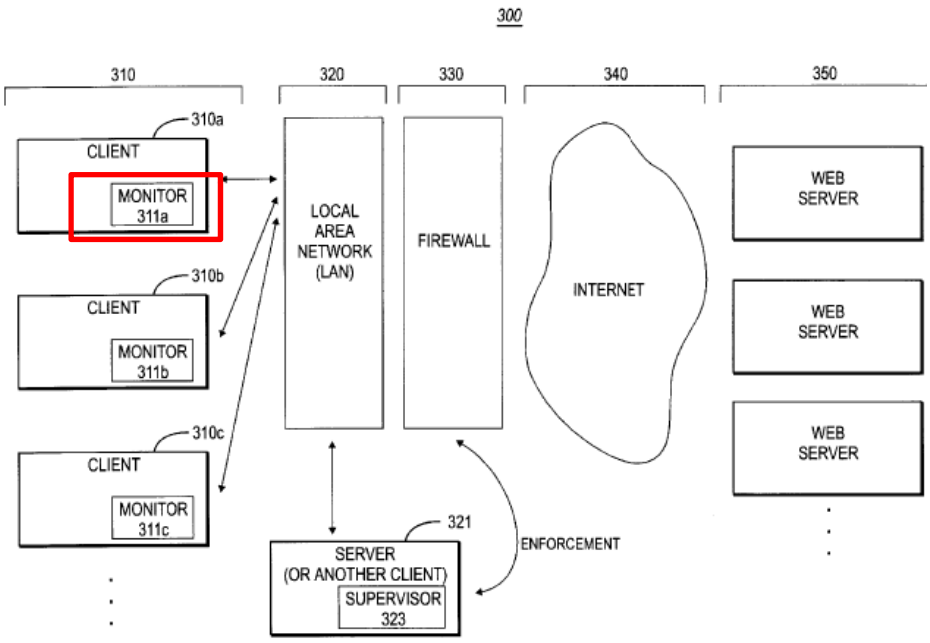

FIG. 3A

Each client-side monitor (*e.g.*, Monitor 311a) comprises its own database of parser and analyzer rules (*i.e.*, "rules database or knowledgebase 570"): "Fig. 5 is a block

diagram illustrating a client-side monitor or data acquisition module." (Ex. 1014 at 6:42-43, Fig. 5 ("Rules Knowledgebase 570").) Freund thus teaches databases of parser and analyzer rules stored within the computer. (Ex. 1006 at ¶ 104.)

Accordingly, Freund meets claim limitations 1(c)(1), 13(e)(1), and 25(e)(1) by teaching a database of parser and analyzer rules (*e.g.*, "rules database or knowledgebase 570") corresponding to computer exploits (*e.g.*, suspected "syntax elements" of a file), stored within the computer (*e.g.*, client 310a, 310b, 310c), computer exploits being portions of program code that are malicious. (Ex. 1006 at ¶ 105.)

(4) Claim limitations 1(c)(2), 13(e)(2), and 25(e)(2) – patterns of types of tokens

Claim limitations 1(c)(2), 13(e)(2), and 25(e)(2) each require that "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." Freund teaches or suggests this limitation.

As detailed above in Section VI., Ground 2.B.(3), Freund teaches parsing and analyzing the content of an HTML file based on its components, such as applet tags, object tags, embed tags, imbedded scripts, references to files or components (*e.g.*, system tags), and syntax elements. (Ex. 1006 at ¶ 107.)

To the extent Freund is deemed not to explicitly teach that its parser and analyzer rules include a punctuation type, an identifier type, and a function type, it

would have been obvious to one of ordinary skill in the art at the time of the invention that Freund could be readily modified by adding these features, which are disclosed by Chandnani. (*See* § VI., Ground 1.B.(4).) (*See also*, Ex. 1006 at ¶ 108.)

As further discussed above, Chandnani's language definition rules include parser rules that comprise an identifier type. (Ex. 1006 at ¶ 111.) Chandnani also teaches that parser rules (i.e., language definition rules) consist of a "function type" token and rules based on tokens that include a punctuation type. (*Id.* at ¶¶ 109-112.)

As further discussed above, Chandnani's viral code detection data ("analyzer rules") also comprise "a punctuation type, an identifier type and a function type." Chandnani parses the incoming data stream and then matches the parsed tokens against the viral code detection data, and Chandnani teaches that its analyzer rules require the same constituent part information as its parsing rules. Because Chandnani teaches tokenizing the data stream based on the parser rules and then pattern matching the tokens in the analyzer rules, the two sets of rules must contain the same types of tokens (*e.g.*, punctuation, function, and identifiers). (Ex. 1006 at ¶ 113.)

Accordingly, Chandnani teaches that both the analyzer and parser rules consist of tokens and the types of tokens that comprise "a punctuation type, an identifier type and a function type" as required by claim limitations 1(c)(2), 13(e)(2), and 25(e)(2). One of ordinary skill would have found the use of such tokens to be an obvious modification (if not already inherent) in the parser and analyzer rules disclosed in

Freund. (*Id*.at ¶ 114.)

(5) Claim limitations 1(d), 13(d), and 25(d) – a rule-based content scanner

Claim limitation 1(d) recites "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" while 13(d) and 25(d) recite "scanning the selectively diverted incoming content to recognize potential computer exploits therewithin based on [the database of parser and analyzer rules discussed in limitations 13(e)(1) and (2) and 25(e)(1) and (2)]" *See* Tables 1 and 2, *infra*. Freund discloses these limitations. (Ex. 1006 at ¶ 115.)

Freund teaches a rule-based content scanner (referred to in Freund as a "content driver") that scans incoming content: "At step 1216, the Client Monitor loads the content driver for ".html" files." (Ex. 1014 at 29:54-55.) Freund explains that "the Client Monitor passes the contents [of 'foo.html'"] to the content driver." (*Id*. at 29:57-59.) The content driver "checks" for HTML components, such as applet tags, object tags, embed tags, imbedded scripts, references to files or components (*e.g.*, system tags), and syntax elements. (*See id.* at 29:54-67; Ex. 1006 at ¶ 116.)

Freund's content driver recognizes potential computer exploits in the incoming content by identifying HTML components that are "known or suspected to cause security or network problems." (*See* Ex. 1014 at 29:54-30:1; step 1221 of Fig. 12C.)

Freund's content driver communicates with its database of parser and analyzer rules: "checks the application and rules database [*i.e.*, rules database or knowledgebase 570] to see if the specific HTML component is permissible." (*Id.* at 30:1-9; Ex. 1006 at ¶ 117.)

Freund teaches that its content driver is operatively coupled with a network interface that receives incoming content. As detailed above in Section VI., Ground 2.B.(2), Freund teaches a network interface (*e.g.*, "network interface card or controller 11") housed within a computer (*e.g.*, "computer system 100") for receiving incoming content (*e.g.*, "foo.html") from the Internet. Freund explains that a host (*e.g.*, "Web Server 350") sends the contents of foo.html to a client whose respective client-side monitor (*e.g.*, monitor 310a) component passes to content driver. (*Id*. at 29:50-60 and Fig 12C.) Accordingly, Freund discloses all of claim limitations 1(d), 13(d), and 25(d). (Ex. 1006 at ¶ 118.)

(6) Claim limitations 1(e), 13(c), and 25(c) – a network traffic probe

Claim limitation 1[e] recites "a network traffic probe, operatively coupled to said network interface and to said rule-based content scanner, for selectively diverting incoming content from its intended destination to said rule-based content scanner." Claim limitations 13(c) and 25(c) omit the "network traffic probe" but recite the step of "selectively diverting … the received incoming content from its intended destination." Freund discloses these limitations.

Freund teaches that its "Client Monitor" selectively diverts incoming content to its rule-based scanner (*i.e.*, "content driver") and that an application has a right to download from the Internet via HTTP commands. In particular, Freund discloses that "[e]ach client includes a client-side monitoring component for monitoring Internet . . . , shown at 311a, 311b, and 311c." (Ex. 1014 at 14:59-62 and Fig. 3A.) Freund explains that the "Application calls WinSock send() with the HTTP get command (*e.g.*, 'GET foo.html')" to download foo.html. (*Id*. at 29:43-44.) However, before the Application can receive the downloaded content, "Client Monitor intercepts the call and … checks the rules and application database to see if the Application has the right to use HTTP." (*Id*. at 29:47-49.) Freund clarifies that Client Monitor will "fail[] or redirect[] the send() call" if it determines based on the rules and application database that the Application does not have the right to download content from the Internet via HTTP commands. (*Id*. at 29:52-53.) Conversely, if the Application is allowed to download the content, then Client Monitor will "pass[] the contents to the content driver" for parsing and checking for HTML components that are known or suspected to cause security or network problems. (*Id.* at 29:54-30:1.) Contrary to the applicant's arguments during prosecution of the '305 patent, Freund does indeed teach a "network probe" for "selectively diverting" incoming content as required by claim limitations 1(e), 13(c), and 25(c). (Ex. 1006 at ¶¶ 120-121.)

(7) Claim limitations 1(f), 13(f), and 25(f) – a rule update manager

Claim limitation 1(f) recites "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." Similarly, claim limitations 13(f) and 25(f) require "updating the database of parser and analyzer rules periodically to incorporate new behavioral rules that are made available." Freund teaches these limitations.

Freund teaches that each time "the data interpretation module 560 is first loaded at a given client machine, it attempts to download from the supervisor module into its local knowledgebase 570 a copy of those system rules pertinent to the client." (Ex. 1014 at 21:33-37.) Freund explains that if "such a copy is not available (or has not changed since last download), the data interpretation module 560 employs the last downloaded local copy." (*Id.* at 21: 37-40.) A person of ordinary skill in the art plainly would have understood Freund as teaching in the above passages that if a revised version of the knowledgebase 570 is available, it will be downloaded to the client machine from the supervisor module. Accordingly, Freund teaches use of an update manager to periodically update the database of parser and analyzer rules. (Ex. 1006 at ¶ 123.)

Freund, either alone or in combination with Chandnani, therefore renders obvious each of claims 1, 13 and 25 of the '305 patent. (Ex. 1006 at ¶ 124.)

C. **Dependent Claims 2-4 and 14-16 – pattern matching engines**

Claims 2 and 14 each recites "wherein said database of parser and analyzer rules stores parser and analyzer rules in the form of pattern-matching engines." Claim 2 depends from claim 1; claim 14 depends from claim 13. As discussed above, the combination of Freund and Chandnani teaches all of the limitations of claims 1 and 13.

As discussed above, Freund teaches that content driver checks whether any of the parsed components of a content file matches specific patterns of HTML components, such as applet tags, object tags, embed tags, imbedded scripts, references to files or components (*e.g.*, system tags), and syntax elements. (*See* Ex. 1014 at 29:54-67; Ex. 1006 at ¶ 126.) Freund therefore teaches a "pattern matching engine," satisfying claims 2 and 14.

Claims 3 and 15 each recites "wherein the pattern-matching engines are deterministic finite automata." Claims 4 and 16 each recites "wherein the pattern-matching engines are non-deterministic finite automata."

During prosecution of the '305 patent, the Examiner rejected application claims 3, 4, 15 and 16 as obvious over Freund in view of the skill in the art as "it was well known in the art for DFA and NFA to be used as engines for pattern matching (e.g., see admission by applicant, Applicant's specification, par. 73)." (Ex. 1003 at 8:12-15). Applicant did not traverse those rejections. Moreover, as discussed above in Section VI., Ground 1.C, Chandnani expressly teaches the use of finite state machines for pattern matching. Accordingly, it would have been obvious to one of ordinary skill in

the art at the time of the invention to have used DFA and NFA as taught and suggested in Chandnani for performing the pattern matching discussed in Freund. (Ex. 1006 at ¶ 128.)

D. **Dependent Claims 5 and 17 – blocking infected content**

Claim 5 requires "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 17 requires a method equivalent step of "preventing incoming content having a computer exploit that was recognized by said scanning from reaching its intended destination." Claim 5 depends from claim 1; claim 17 depends from claim 13. As discussed above, the combination of Freund and Chandnani teaches all of the limitations of claims 1 and 13.

Freund teaches that its rule-based scanner (*i.e.*, content driver) "checks the application and rules database to see if the specific HTML component is permissible [and] if it is not permissible, the driver either removes the HTML component or fails the recv() call depending on the violated rule." (Ex. 1014 at 30:1-6.) Freund further clarifies that a remedial action for any violated rule includes terminating the communication: "the prescribed remedial action for any violated rule is performed, including logging an exception log entry and, depending on the rules the TCP/IP activity, the communication is either terminated, redirected, modified, or continued."

(*Id*. at 4:58-62.) Such termination prevents the incoming content from reaching the Freund's intended destination (*e.g.*, "Application Program(s) Browser 245" in Fig. 2). (Ex. 1006 at ¶ 130.)

Accordingly, the combination of Freund and Chandnani teaches all of the limitations of claims 5 and 17 of the '305 patent.

E. **Dependent Claims 6-10 and 18-22 – content types**

Claims 6-12 and 18-24 variously recite that "incoming content received from the Internet by said network interface" is HTTP content (claims 6 and 18), HTTPS content (claims 7 and 19), FTP content (claims 8 and 20), SMTP content (claims 9 and 21), and POP3 content (claims 10 and 22). All of claims 6-10 depend from claim 1, while claims 18-22 depend from claim 13. As discussed above, the combination of Freund and Chandnani teaches all of the limitations of claims 1 and 13.

Freund's system includes a "driver supporting HTTP, FTP, SMTP, and POP3 protocols" for parsing "HTTP files, executable files, ZIP files, ActiveX controls, or Java classes." (*Id.* at 23:46-54.) While Freund does not explicitly state that its driver supports HTTPS (*e.g.* hyper-text transfer protocol secure), a person of ordinary skill, however, would have understood that Freund's reference to a "secure communication protocol" encompassed HTTPS protocols. (*Id*. at 5:15-20, 14:7-13; Ex. 1003 at 8:5-10; Ex. 1006 at ¶ 133.) The combination of Freund and Chandnani therefore teaches all of the limitations of claims 6-12 and 18-24 of the '305 patent.

F. Dependent Claims 11-12 and 23-24 – destination applications

Claims 11 and 23 recite that "the destination Internet application is a web browser" while claims 12 and 24 recite that "the destination Internet application is an e-mail client." Claims 11 and 12 depend from claim 1; claims 23 and 24 depend from claim 13. The combination of Freund and Chandnani teaches all of the limitations of claims 11-12 and 23-24.

Freund teaches that the "system includes one or more clients, each operating applications or processes (*e.g.*, Netscape Navigator ™ or Microsoft Internet Explorer™ browser software) requiring Internet (or other open network) access (*e.g.*, an Internet connection to one or more Web servers)." (Ex. 1014 at Abstract.) Freund explains that the "application" (*e.g.*, browser software) opens an internet socket to download the incoming content (*e.g.*, "foo.html"): "the Application calls WinSock send() with the HTTP get command (*e.g.*, 'GET foo.html')." (*Id*. at 29:43-44.) Freund also discloses receiving content over a network via "SMTP, and POP3 protocols." (*Id*. at 23:44-49.) A person of ordinary skill would have understood that content received over a network via SMTP and POP3 protocols generally were destined for an e-mail client. (Ex. 1006 at ¶ 135.) Accordingly, the combination of Freund and Chandnani teaches all of the limitations of claims 11-12 and 23-24 of the '305 patent.

G. Motivation to Combine Freund and Chandnani

As explained in the accompanying Declaration of Eugene Spafford, the concepts

of parser and analyzer rules were well-known in the field of computer security prior to the earliest effective filing date that can be accorded the claims of the '305 patent. As discussed in the Summary of the Description portion of the '305 patent, content scanners that parse and analyze software to determine the presence of malicious exploits were known. (Ex. 1006 at ¶ 136.) The '305 patent attempts to distinguish its "adaptive rule-based (ARB) scanners" from prior art systems by arguing that "ARB scanners differ from prior art scanners that are hard-coded for one particular type of content" and "can be enabled to scan any specific type of content by providing appropriate rule files without the need to modify source code." (Ex. 1001 at 2:14-18). Apparently unbeknownst to the alleged inventors of the '305 patent, other rule-based systems, such as Chandnani and Freund, did in fact store parser and analyzer rules in databases, as required by the claims of the '305 patent. (Ex. 1006 at ¶ 136.)

More importantly, however, in 2004, one of ordinary skill in the art would have understood that prior art hard-coded content scanners also relied upon parsing techniques to tokenize the incoming content, and then use well-known finite state machines to pattern match and detect the presence of malicious code. (*Id.* at 11:15-21; Ex. 1003 at 5:14-17, Ex. 1006 at ¶ 137.) Such as person of ordinary skill also would have understood that in parsing and tokenizing the incoming data, it would have been necessary to identify token types corresponding to punctuation, function types and identifiers (e.g., expressions), as taught in Chandnani. *See, supra*, Section VI.B.(4).

Accordingly, a person of ordinary skill would have expected that the pattern-matching of Freund inherently to involve the use of tokens having types of at least punctuation, identifier and function types, or at the very least, that it would have been obvious to use such token types in implementing the security measures for incoming content described in Freund. (Ex. 1006 at ¶ 137.)

CONCLUSION

For the foregoing reasons, Petitioner respectfully requests that Trial be instituted and that Claims 1-25 of the '305 patent be canceled.

Respectfully submitted,

Dated: July 4, 2017

By: /s/ Nicola A. Pisano
Nicola A. Pisano
Reg. No. 34,408
Counsel for Petitioners

CERTIFICATE OF SERVICE

The undersigned hereby certifies that a CD containing copies of the foregoing Petition for *Inter Partes* Review together with all exhibits and other papers filed therewith was served on July 4, 2017, via Federal Express, directed to the attorneys of record for the patent at the following address:

Dawn-Marie Bey
Bey & Cotropia PLLC
213 Bayly Court
Richmond, VA 23229

By: */Nicola A. Pisaon/*
Nicola A. Pisano
Reg. No. 34,408
Counsel for Petitioners