UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

SOPHOS, INC.

Petitioner
v.

FINJAN, INC.

Patent Owner

CASE IPR Unassigned

Patent No. 8,677,494

PETITION FOR
***INTER PARTES* REVIEW OF U.S. PATENT NO. 8,677,494**

Mail Stop "PATENT BOARD"
Patent Trial & Appeal Board
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

TABLE OF CONTENTS

Page

TABLE OF EXHIBITS

EXHIBIT	DESCRIPTION
1001	U.S. Patent No. 8,677,494 to Edery et al.
1002	Expert Declaration of Paul Clark
1003	Curriculum Vitae of Paul Clark
1004	File History for U.S. Patent No. 8,677,494
1005	U.S. Provisional Patent Application No. 60/030,639 to Touboul et al.
1006	*ThunderBYTE Anti-Virus Utilities User Manual*, ThunderBYTE B.V. (1995)
1007	Information Disclosure Statement filed in U.S. Patent Application No. 08/605,285 on October 25, 1996
1008	U.S. Patent No. 5,440,723 to Arnold et al.
1009	U.S. Patent No. 5,623,600 to Ji et al.
1010	U.S. Patent No. 5,951,698 to Chen et al.
1011	*Finjan, Inc. v. Sophos Inc.*, 14-cv-1197 – Finjan Opening Claim Construction Brief
1012	File History for U.S. Patent Application No. 11/370,114
1013	File History for U.S. Patent Application No. 09/861,229
1014	U.S. Patent No. 6,804,780
1015	U.S. Patent No. 6,092,194

1016	U.S. Patent No. 6,480,962
1017	Excerpts of *Finjan v. McAfee et al.*, Trial Transcripts
1018	Information Disclosure Statement filed in U.S. Patent Application No. 08/684,580 on October 25, 1996
1019	Excerpts of *Microsoft Press Computer Dictionary, Second Edition*, Microsoft Press (1994)
1020	*Mangosoft, Inc. v. Oracle Corp.*, 1:02-cv-545-SM – Claim Construction Order

I. INTRODUCTION

It is hereby requested that the United States Patent and Trademark Office proceed with an *inter partes* review of claims 1, 10, 14, and 18 of U.S. Patent No. 8,677,494 (Exhibit 1001) (hereinafter "the '494 patent").

The '494 patent relates generally to detecting suspicious computer operations in downloaded files. The claims are directed to receiving an incoming Downloadable, deriving security profile data for the Downloadable, and storing the security profile data in a database. The security profile data includes a list of suspicious computer operations that may be attempted by the Downloadable. As explained below, the claimed systems had been developed and were well known in the virus detection and computer security field long before the application for the '494 patent had been filed. Specifically, it was well known to receive downloaded files that could be infected with malware, examine the files to identify suspicious computer operations that may be attempted by the files, and store the identified data in a database. All of the claims of the '494 patent are therefore invalid over the prior art identified below, each of which was cited during prosecution of the '494 patent.

II. COMPLIANCE WITH FORMAL REQUIREMENTS

A. Mandatory Notices Under 37 C.F.R. §§ 42.8(b)(1)-(4)

1. Real Parties-In-Interest

Sophos Limited and wholly owned subsidiary, Sophos Inc., are the real parties-in-interest.

2. Related Matters

To the knowledge of Petitioner, a decision in this proceeding would affect or be affected by the following matters where Finjan is asserting the ’494 patent: (1) Finjan, Inc. v. Sophos Inc., 3:14-1197 (N.D. Cal.), (2) Finjan, Inc. v. Websense, Inc., Case No. 3:14-cv-1353 (N.D. Cal.), and (3) Finjan, Inc. v. Palo Alto Networks, Inc., Case No. 3:14-cv-4908 (N.D. Cal.).

3. Lead and Back-up Counsel

Lead Counsel for Petitioner is James M. Heintz, DLA Piper LLP (US), Reg. No. 41,828, who can be reached by email at Sophos-Finjan-494IPR@dlapiper.com, by phone at 703-773-4148, by fax at 703-773-5200, and by mail at 11911 Freedom Drive, Suite 300, Reston, VA 20190. Backup counsel for Petitioner is Nicholas Panno of DLA Piper LLP (US), Reg. No. 68,513, who can be reached by email at Sophos-Finjan-494IPR@dlapiper.com, by phone at 703-773-4157, by fax at 703-773-5157, and by mail at 11911 Freedom Drive, Suite 300, Reston, VA 20190.

4. Powers of Attorney and Service Information

Powers of attorney are being filed with the designation of counsel in accordance with 37 C.F.R. § 42.10(b). Service information for lead and back-up counsel is provided in the designation of lead and back-up counsel above. Service of any documents via hand delivery may be made at the postal mailing address of the respective lead and back-up counsel designated above.

B. Proof of Service on the Patent Owner

As identified in the attached Certificate of Service, a copy of this Petition in its entirety is being served on the Patent Owner's attorney of record at the address listed in the USPTO's records by overnight courier pursuant to 37 C.F.R. § 42.6.

C. Fee

The undersigned authorizes the Director to charge the fee specified by 37 C.F.R. § 42.15(a) and any additional fees that might be due in connection with this Petition to Deposit Account No. 50-1442.

III. GROUNDS FOR STANDING

In accordance with 37 C.F.R. § 42.104(a), the Petitioner certifies that the '494 patent is available for *inter partes* review and that the Petitioner is not barred or estopped from requesting an *inter partes* review challenging the patent claims on the grounds identified in this Petition.

IV. STATEMENT OF PRECISE RELIEF REQUESTED

In accordance with 37 C.F.R. § 42.22, the Petitioner respectfully requests that claims 1, 10, 14, and 18 of the ’494 patent be invalidated for the reasons set forth below.

V. IDENTIFICATION OF CHALLENGE

In accordance with 35 U.S.C. § 311 and 37 C.F.R. § 42.104(b), *inter partes* review of claims 1, 10, 14, and 18 of the ’494 patent is requested in view of the following grounds:

A. Claims 1, 10, and 18 are invalid under 35 U.S.C. § 103 (pre-AIA) as being obvious over ThunderBYTE Anti-Virus Utilities User Manual (“TBAV”) in view of U.S. Patent No. 5,623,600 to Ji et al. (“Ji”).

B. Claim 14 is invalid under 35 U.S.C. § 103 (pre-AIA) as being obvious over TBAV in view of Ji and U.S. Patent No. 5,951,698 to Chen et al. (“Chen”).

C. Claims 1, 10, 14, and 18 are invalid under 35 U.S.C. § 103 (pre-AIA) as being obvious over U.S. Patent No. 5,440,723 to Arnold et al. (“Arnold”) in view of Chen and Ji.

D. Claims 1, 10, 14, and 18 are invalid under 35 U.S.C. § 103 (pre-AIA) as being obvious over Chen in view of Arnold and Ji.

VI. LEVEL OF ORDINARY SKILL IN THE ART

A person of ordinary skill in the art at the time of the alleged invention of the ’494 patent would generally have a bachelor’s degree or the equivalent in

computer science, computer engineering, or a related degree, and three to four years of experience in the fields of network software and security, or equivalent work experience. This person would have been aware of the computer security technology discussed herein and would have been capable of understanding and applying the prior art references discussed herein, alone or in combination. Ex. 1003 ¶ 52.

VII. SUMMARY OF THE ’494 PATENT

A. Effective Filing Date

The ’494 patent, entitled “Malicious Mobile Code Runtime Monitoring System and Methods,” issued on March 18, 2014, and arises from U.S. patent application No. 13/290,708 (“the ’708 application”), which was filed on November 7, 2011. The patent was assigned upon issuance to Finjan, Inc. According to USPTO records, this patent is currently assigned to Patent Owner.

The ’494 patent claims the benefit of provisional applications No. 60/205,591 (filed May 17, 200) and No. 60/030,639 (filed November 8, 1996). Thus, assuming the ’494 patent correctly claims the benefit of the ’639 application, the earliest possible effective filing date of the ’494 patent is November 8, 1996. Ex. 1003 ¶¶ 47-51. Several other patents claiming the benefit of the same provisional application (U.S. Patent Nos. 6,092,194; 6,167,520; 6,480,962; and

7,058,822) have been invalidated as a result of Federal Court or PTAB proceedings.

B. Overview of the Prosecution History

The ’708 application was filed on November 7, 2011. A non-final rejection was mailed on July 23, 2012, which rejected the claims of the ’708 application for double patenting and under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent No. 5,983,348 to Ji. Ex. 1004, pp. 757-766. Applicants addressed the rejections by filing a terminal disclaimer and a petition to accept unintentionally delayed claim of priority to the ’639 application. Ex. 1004, pp. 721-256. This petition was dismissed on Novmber 27, 2012 because it was not accompanied by a copy of the’639 application. Ex. 1004, pp. 719-720. A final rejection was mailed on January 7, 2013, accepting the terminal disclaimer and citing the same U.S. Patent No. 5,983,348 to Ji. Ex. 1004, pp. 711-717. On May 7, 2013, Applicants filed a declaration by inventor Shlomo Touboul purporting to establish an invention date prior to the filing date of the Ji patent. Ex. 1004, pp. 182-190. Specifically, Touboul declares that his "sole invention was in [his] mind and developed by at least November 18, 1996," which is after the filing date of the ‘639 application. Ex. 1004, p. 189. A notice of allowance was mailed on August 29, 2013. Ex. 1004, pp. 118-124. Applicants subsequently filed a petition to withdraw from issue, a new petition to accept unintentionally delayed claim of priority to the ’639 application,

and a request for continued examination on October 1, 2013, and a corrected petition on December 6, 2013. Ex. 1004, ppp. 65-77, 94-102. The USPTO granted the petition for acceptance of a claim for late priority to the ’639 application on December 24, 2013, and the patent issued on March 18, 2014. Ex. 1001, p.1; Ex. 1004, pp. 53-54.

C. Patent Specification and Claims

The ’494 patent relates generally to protecting personal computers (PCs) or other devices from malicious software obtained over a network. The ’494 patent refers to executable files and software obtained over a network as “Downloadables” and focuses on identifying malicious Downloadables. The specification of the ‘494 patent itself has little specific teaching of the claimed subject matter beyond providing descriptions that are broadly in the same field as the claims. Ex. 1002 ¶39.

The disclosure of the ’639 application most closely resembles the subject matter claimed by the ‘494 patent. The ’639 application relates generally to a system and method for protecting computers from hostile Downloadables. Ex. 1005, p. 1, ll. 6-8; Ex. 1002 ¶40. A Downloadable is described as “an executable application program which is automatically downloaded from a source computer and run on the destination computer.” Ex. 1005, p. 2, ll. 1-4; Ex. 1002 ¶40. Examples of Downloadables include executables as well as applets for Java and

Active X. Ex. 1005, p. 2, ll. 4-7; Ex. 1002 ¶40. According to the ’639 application, viruses may be attached to Downloadables. Ex. 1005, p. 1, ll. 20-22; p. 2, ll. 7-9; Ex. 1002 ¶40.

The method taught by the ’639 application includes receiving a Downloadable, and, when the Downloadable does not get identified as a previously identified hostile Downloadable, deriving Downloadable security profile data from the received Downloadable. Ex. 1005, p. 3, ll. 13-21; Ex. 1002 ¶41.

Specifically, an internal network security system 110 examines Downloadables, determines whether they are hostile, and prevents hostile Downloadables from reaching an internal computer network 115 (e.g., a corporate local area network (LAN)). Ex. 1005, p. 6, ll. 2-17; Ex. 1002 ¶42. The internal network security system 110 is shown in FIG. 2 and includes a central processing unit (CPU) 205, communications interfaces 210, 225, input/output (I/O) interfaces 215, a data storage device 230, memory 235, and a signal bus 220. Ex. 1005, p. 6, l. 19 – p. 8, l. 4; Ex. 1002 ¶42. The security program 255 in memory 235 controls the operations of the internal network security system 110. Ex. 1005, p. 8, ll. 1-4; Ex. 1002 ¶42. The security database 240 in the data storage device 230 stores Downloadable security profiles (DSPs). Ex. 1005, p. 8, ll. 14-19; Ex. 1002 ¶42.

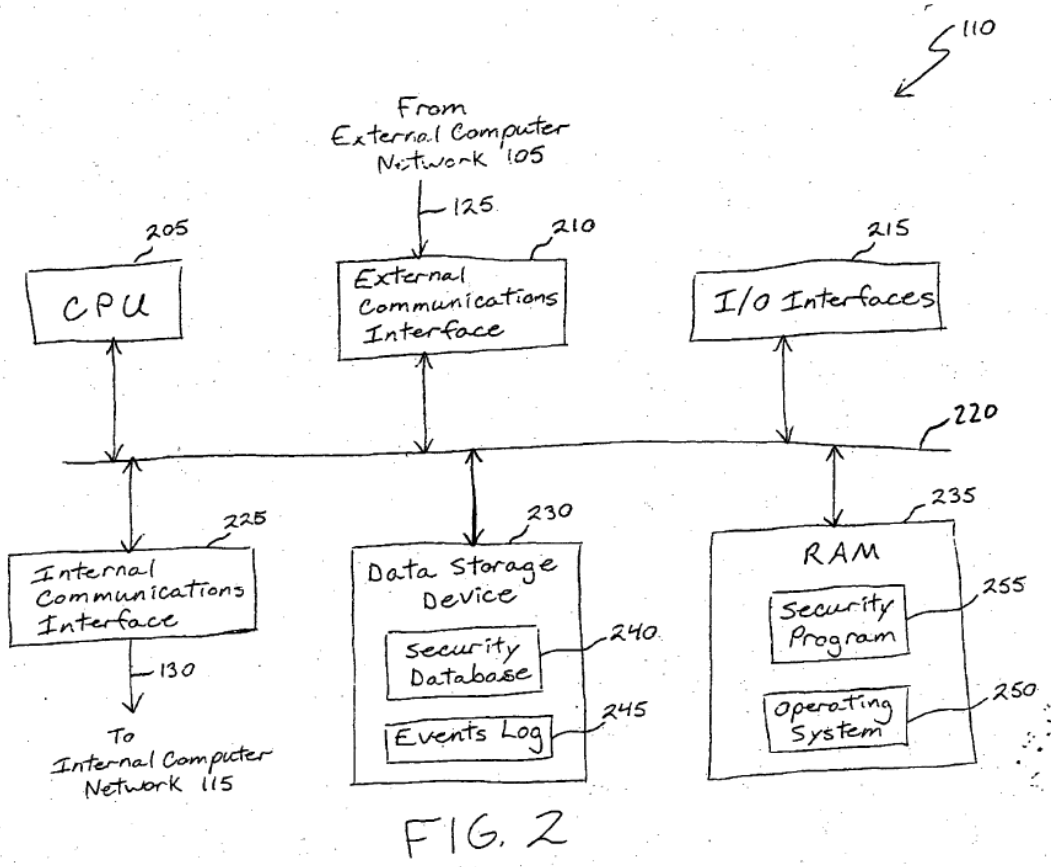

FIG. 2

FIG. 3 illustrates security program 255 details. An ID generator 315 of the security program 255 receives a Downloadable from external computer network 105. Ex. 1005, p. 9, ll. 14-16; Ex. 1002 ¶43.

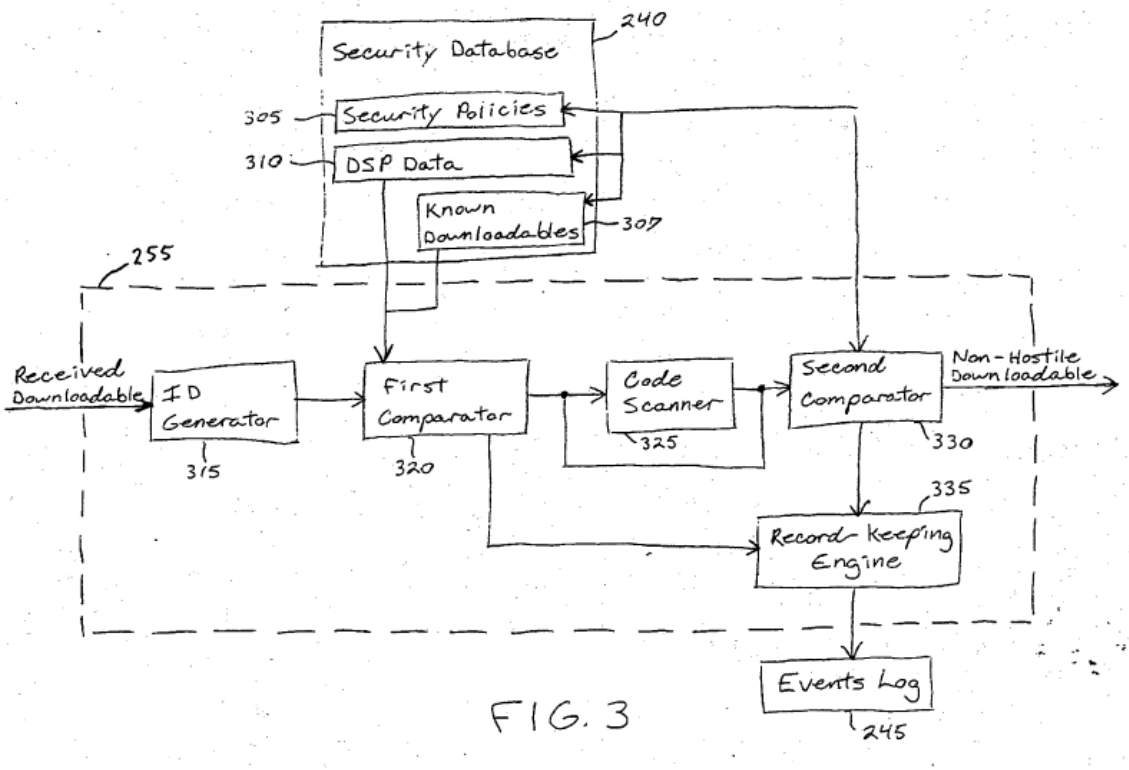

FIG. 3

A first comparator 320 of the security program 255 determines if the Downloadable is identical to a known non-hostile or hostile Downloadable 307 and discards the Downloadable if it is identical to a known hostile Downloadable. Ex. 1005, p. 9, l. 20 – p. 10, l. 5; Ex. 1002 ¶44. Known Downloadables 307 (hostile and non-hostile) along with corresponding DSP data 310 are stored in the data storage device 230. Ex. 1005, p. 8, ll. 13-19; Ex. 1002 ¶44. DSP data 310 includes the fundamental computer operations included in the known hostile Downloadables 307 (e.g., read, write, file management operations, system management operations, memory management operations, CPU allocation operations). Ex. 1005, p. 9, ll. 9-13; Ex. 1002 ¶44.

In the event that the Downloadable is unknown (either not identical to a known hostile Downloadable or not identical to a known non-hostile downloadable), a code scanner 325 of the security program 255 decomposes the code of the unknown Downloadable into DSP data and sends the Downloadable and DSP data to a second comparator 330 of the security program 255. Ex. 1005, p. 10, ll. 9-20; Ex. 1002 ¶45. Decomposing the code includes disassembling the machine code of the Downloadable, resolving a command in the machine code, and determining whether the resolved command is a suspect command (e.g., a memory allocation command or loop command). Ex. 1005, p. 15, l. 15 – p. 16, l. 2; Ex. 1002 ¶45. Code containing suspect commands is decoded, and the command and command parameters are registered as DSP data that can be stored in the data storage device 230; Ex. 1002 ¶45.

The second comparator 330 compares the DSP data against stored security policies 305 to determine whether the Downloadable includes a hostile operation. Ex. 1005, p. 10, l. 21 – p. 11, l. 5; Ex. 1002 ¶46. The known Downloadables 307 stored in the data storage device 230 include Downloadables that the second comparator 330 determines to be hostile. The DSP data 310 associated with these hostile Downloadables includes the fundamental operations performed by the Downloadables (i.e., the hostile operations). Ex. 1005, p. 9, ll. 3-13; Ex. 1002 ¶46.

VIII. CLAIM CONSTRUCTION

In accordance with 37 C.F.R. § 42.104(b)(3), Petitioner provides the following statement regarding construction of the ’494 patent claims.

A. Legal Standard

Claims in an *inter partes* review of an unexpired patent are to be given their “broadest reasonable interpretation in light of the specification.” 37 C.F.R. 42.100(b). This standard is often referred to as “BRI”.

B. Construction of the Terms Used in the Claims

Because the claim construction standard in this proceeding differs from that used in U.S. district court litigation, Petitioner expressly reserves the right to assert different claim construction positions under the standard applicable in district court for any term of the ’494 patent in any district court litigation. Claim construction is further discussed in Ex. 1002 ¶¶ 53-66.

1. “Downloadable” (claims 1, 10, 14, and 18): Under the BRI, this limitation should be understood to mean “an executable application program which is downloaded from a source computer and can be run on the destination computer”, as defined in the ’639 application. Ex. 1005, p. 2, ll. 1-4; Ex. 1002 ¶60. This construction is also consistent with the ’494 patent, which describes a Downloadable as “information including executable code.” This definition is what one of ordinary skill in the art would understand from the specification of the ’494

patent. Ex. 1001, col. 9, ll. 46-52; Ex. 1002 ¶60. This construction is also consistent with what was agreed upon by Petitoner and Patent Owner in related proceedings cited above. Ex. 1011, p. 4.

2. "suspicious computer operations" (claims 1 and 10): Under the BRI, this limitation should be understood to mean "computer instructions that are deemed to be potentially hostile". This construction is consistent with the disclosure in the '639 application regarding "potentially hostile operations" and "suspect commands." Ex. 1005, p. 8, l. 19 – p. 9, l. 3; p. 15, l. 19 – p. 16, l. 2; Ex. 1002 ¶62. This construction is also consistent with the '494 patent, which describes "harmful, undesirable, suspicious or other 'malicious' operations that might otherwise be effectuated by remotely operable code." Ex. 1001, col. 2, ll. 54-56; Ex. 1002 ¶62.

3. "database" (claims 1 and 10): Under the BRI, this limitation should be understood to mean "any structured store of data". The '639 application does not define "database", but one of ordinary skill in the art would understand the term "database" to have this meaning. See, e.g., Ex. 1020, p. 29 (wherein database files are given as examples of structured stores of data). This construction is consistent with the disclosure in the '639 application of a data storage device 230 that stores a security database 240. Ex. 1005, p. 7, ll. 16-20; Ex. 1002 ¶64. This construction is also consistent with the '494 patent, which teaches "[a]ny

suitable explicit or referencing list, database or other storage structure(s) or storage structure configuration(s) can also be utilized to implement a suitable user/device based protection scheme" Ex. 1001, col. 17, ll. 10-14; Ex. 1002 ¶64.

4. "program script" (claim 14): Under the BRI, this limitation should be understood to mean "a set of instructions that can be executed by a computer through an interpreter or some other program with a computer.". The '639 application does not define "program script", but one of ordinary skill in the art would understand the term "program script" to have this meaning. See, e.g., Ex. 1019, p. 350. This construction is consistent with the disclosure in the '639 application regarding Downloadables containing Java applets or ActiveX applets. Ex. 1005, p. 2, ll. 1-7; Ex. 1002 ¶66. This construction is also consistent with the '494 patent, which describes "downloadable application programs, Trojan horses and program code groupings, as well as software 'components', such as JavaTM applets, ActiveXTM controls, JavaScript/Visual Basic scripts, add-ins, etc." Ex. 1001, col. 2, ll. 59-64; Ex. 1002 ¶66. Those of ordinary skill in the art would understand that JavaTM applets, ActiveXTM controls, JavaScript/Visual Basic scripts contain instructions that can be executed by a computer through an interpreter or some other program. Ex. 1002 ¶66.

IX. STATE OF THE ART PRIOR TO THE ’494 PATENT

The state of the art prior to the ’494 patent is discussed generally in Ex. 1002 ¶¶35-38. By the time of the filing of the ‘494 patent, malware downloaded from a network was a well-known threat. Companies like Norton, McAfee, and Trend Micro were known to offer established products available to detect and prevent hostile downloadable software from being installed through the use of local and firewall resident scanners. In particular, these commercially available solutions generally included data files intended to assist an executable scanner in the identification of malware.

These companies and others continuously monitored developments in malware to identify new malware and update their product data files to enable identification of the new malware. Data file updates were distributed to end users periodically to keep their systems up to date.

However, it was widely known that new malware could appear on an end user system faster than it could be detected and incorporated into the data files by scanner software distributors. For example, it was widely known at the timein 1996 that files could be downloaded onto computers from networks. FTP, HTTP, and other network protocols were in general use, and computers used these protocols to interface with networks to download files, including executable files and files containing program script. See, e.g., Ex. 1009, col. 1, ll. 15-42; col. 2, l. 39 – col.

3, l. 16. Such files were known to be potentially contain suspicious computer instructions, such as malware. Often, malware was distributed to computers via networks in executable or script files before it could be detected by the distributors of malware protection software. See, e.g., Ex. 1006, p. 165.

To that end, it was well known by the time of the filing of the ‘494 patent to include modules for end user systems and other network-deployed systems to allow them to detect and respond to programs that contained suspicious instructions, and therefore potentially hostile software, in the absence of a signature in the scanner’s data file.

As discussed below, all of the elements of claims 1, 10, 14, and 18 of the ’494 patent are known in the prior art, and it would have been obvious to combine the prior art in the manner recited in those claims.

X. THERE IS A REASONABLE LIKELIHOOD THAT PETITIONER WILL PREVAIL WITH RESPECT TO AT LEAST ONE CLAIM OF THE ’494 PATENT

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). This Petition meets that threshold. There is a reasonable likelihood that Petitioner will prevail in establishing that at least one of claims 1, 10, 14, and 18 of the ’494 patent is invalid under 35 U.S.C. § 103 as explained below.

XI. DETAILED EXPLANATION OF THE GROUNDS FOR REJECTION

A detailed explanation of the pertinence and manner of applying the prior art references to claims 1, 10, 14, and 18 of the ’494 patent is provided below in accordance with 37 C.F.R. §§ 42.104(b)(4) and 42.104(b)(5). Except as noted, the detailed explanation set forth below assumes that the claims of the ’494 patent are construed consistent with the claim constructions set forth above.

A. Challenge to Claims 1, 10, and 18 based on TBAV in view of Ji

TBAV qualifies as prior art to the ’494 patent under 35 U.S.C. § 102(a) because TBAV has a copyright date of 1995 and was publicly available by at least October 25, 1996, as evidenced by the fact that it was disclosed by a patent applicant in an Information Disclosure Statement which was filed in U.S. Patent Application No. 08/605,285 on October 25, 1996 and an Information Disclosure Statement which was filed in U.S. Patent Application No. 08/684,580 on October 25, 1996. See Ex. 1007, pp. 2 and 6; Ex. 1018, pp. 2 and 65. October 25, 1996 is prior to the earliest possible effective filing date of the application of the ’494 patent on November 8, 1996. Ex. 1006, cover page; Ex. 1007, pp. 2 and 6. Ji qualifies as prior art to the ’494 patent under 35 U.S.C. § 102(e) because Ji was granted on an application filed in the U.S. on September 26, 1995, which is prior to the earliest possible ’494 patent effective filing date.

TBAV: TBAV is a user manual for the ThunderBYTE anti-virus software. TBAV describes a software program that protects computer systems against both known and unknown viruses. The TBAV software includes TbScan, a virus scanner, and TbScanX, a memory resident version of TbScan. Ex. 1006, pp. 1-2; Ex. 1002 ¶ 70.

TbScan is run upon user request or automatically (e.g., on startup via the AUTOEXEC.BAT file), and can be used to scan some or all files that have been downloaded or that have otherwise been stored on disks, fixed drives, and/or network drives. Ex. 1006, pp. 48-51; Ex. 1002 ¶ 71. TbScan includes a signature scanner and a heuristic scanner. The signature scanner detects known viruses whose signatures are stored in a signature file. The heuristic scanner disassembles files using a built-in disassembler and searches for suspicious instruction sequences within the files using a built-in code analyzer. Ex. 1006, pp. 1-2, 153-155; Ex. 1002 ¶ 71. The heuristic scanner disassembles files and interprets their instructions. The heuristic scanner disassembles the files to allow scanning to be restricted to an area of a file where a virus might reside, to make it possible to use algorithmic detection methods on encrypted viruses, and to make it possible to detect suspicious instruction sequences. Ex. 1006, p. 154; Ex. 1002 ¶ 71.

TbScan looks into a file's contents and interprets the file's instructions to detect their purpose (e.g., formatting a disk or infecting a file). Heuristic flags are

assigned to suspicious instructions, such as instructions that are common to viruses but uncommon to normal programs. TBAV discloses that a heuristic flag is a character indicating a specific type of suspicious instruction. Ex. 1006, pp. 155, 180-185; Ex. 1002 ¶ 72. For example, the flags include "# - Decryptor code found" (indicating that the file contains instructions that perform self-decryption operations), "A – Suspicious Memory Allocation" (indicating the program contains instructions that perform non-standard memory search and/or allocation operations), "B – Back to entry" (indicating the program contains instructions that perform endless loop operations), "D – Direct disk access" (indicating the program contains instructions that perform a write operation to a disk directly), "L – program load trap" (indicating the program contains instructions that perform an operation to trap the execution of other software), "S – Search for executables" (indicating the program contains instructions that perform a search operation for executable files), and "U – Undocumented system call" (indicating the program contains instructions that perform unknown DOS call or interrupt operations), among others. Ex. 1006, pp. 180-185; Ex. 1002 ¶ 72.

The heuristic flags are associated with scores, so that if the total score (i.e., the sum of heuristic flag scores) for a file exceeds a predefined limit, the file is assumed to contain a virus. Ex. 1006, pp. 153-155. Multiple predefined limits can be established, so that a file having a score above a sensitive limit "might" be

infected, and a file having a score above a higher limit is assumed to include a virus. Ex. 1006, p. 155; Ex. 1002 ¶ 73.

TbScan provides the option to output scan results to a log file. The results in the log file include the heuristic flags assigned to each file. Ex. 1006, p. 60; Ex. 1002 ¶ 74. If suspicious instructions are found in a file during a scan, the heuristic flags in the log file indicate their presence and describe the nature of the suspicious instructions. In addition to these flags, if a detailed log level is selected for the log, TbScan adds a description to the log file. Ex. 1006, pp. 76-77; Ex. 1002 ¶ 74.

TbScan scans files with filename extensions that indicate the file is a program file (e.g., .EXE, .COM, .BIN, .SYS, .OV?). TbScan can be configured to also scan files with filename extensions that suggest the file can be infected by viruses (e.g., .DLL, .SCR, .MOD, .CPL, .00?, and .APP). Ex. 1006, p. 57; Ex. 1002 ¶75. Files such as .DLLs contain executable code that is executed by other programs, so these files can be infected by malware, as those of ordinary skill in the art would understand. Ex. 1002 ¶75.

TBAV also includes a TbGenSig program that can be called by TbScan to define signature records for suspicious files. Ex. 1006, pp. 4, 57; Ex. 1002 ¶76. TbScan is used to extract instructions from files that are being examined, such as files that are believed to be infected with viruses. The signature comprises suspicious instructions form a signature for of the potential virus infecting the file.

Ex. 1006, pp. 166-168; Ex. 1002 ¶76. A virus name, one or more keywords, and the instructions are added to a file (e.g., USERSIG.DAT). Ex. 1006, p. 168; Ex. 1002 ¶76. Note that TBAV explicitly uses *.DAT files as databases (see also, ANTI-VIR.DAT). Ex. 1006, pp. 4, 36; Ex. 1002 ¶76. TbGenSig further adds the virus name, the keywords, and the suspicious instructions to the TBSCAN.SIG file. Ex. 1006, p.165; Ex. 1002 ¶76. Note that the signature instructions include suspicious instructions. TBAV shows an example signature for the Haifa.Mozkin virus wherein suspicious instructions are labeled (e.g., "?2*2" is a skip instruction to make it harder to define a signature, "4l" and "4h" are counter incrementation and decrementation, "75f1" is a jump instruction). Thus, TbGenSig identifies suspicious instructions and saves them, along with other identifying data, in the TBSCAN.SIG file. Ex. 1006, pp. 165, 173-174; Ex. 1002 ¶76.

TbScanX is a memory resident version of TbScan that automatically scans files that are being executed, copied, de-archived, or downloaded. Ex. 1006, p. 2; Ex. 1002 ¶ 77. TbScanX is "virtually identical to TbScan" with the only difference being that it is memory-resident. Ex. 1006, p. 84; Ex. 1002 ¶ 77. TbScanX tracks all file operations and automatically scans executed files and executable files that are copied, created, downloaded, modified, or unarchived. Ex. 1006, p. 84; Ex. 1002 ¶ 77.

Ji: Ji is directed to a system for detecting and eliminating viruses on a computer network. An FTP proxy server scans all incoming and outgoing files for viruses and transfers the files if they do not contain viruses. Ex. 1009, p. 1 (57); Ex. 1002 ¶ 78.

Ji teaches a gateway node 33 comprising a CPU 42, memory 44, and other computer components. Ex. 1009, col. 3, l. 64 – col. 4, l. 16; Ex. 1002 ¶ 79. The memory 44 includes an FTP proxy server 60, which controls file transfers to and from the gateway node 33, thereby controlling file transfers to and from a network. Ex. 1009, col. 4, l. 56 – col. 5, l. 2; Ex. 1002 ¶ 79. When a client node sends a connection request 600, an instance of the FTP proxy server 60 is created 602. The FTP proxy server 60 receives the data transfer request and file name associated with the request 606. Ex. 1009, col. 6, l. 55 – col. 7, l. 28; Ex. 1002 ¶ 79.

FIG. 6C of Ji shows what happens when the request is an inbound request (e.g., a download). FTP proxy server 60 receives an inbound file from FTP daemon 78 and analyzes it. Ex. 1009, col. 8, l. 57 – col. 9, l. 2; Ex. 1002 ¶ 80. Specifically, a connection is established over the network 642, and the file is sent from a server to the FTP daemon and from the FTP daemon to the FTP proxy server 644. Ex. 1009, col. 8, ll. 40-58; Ex. 1002 ¶ 80. If the file is a type that can contain viruses (e.g., an executable), the file is analyzed. Ex. 1009, col. 7, ll. 33-40; col. 8, l. 58 – col. 9, l. 2; Ex. 1002 ¶ 80.

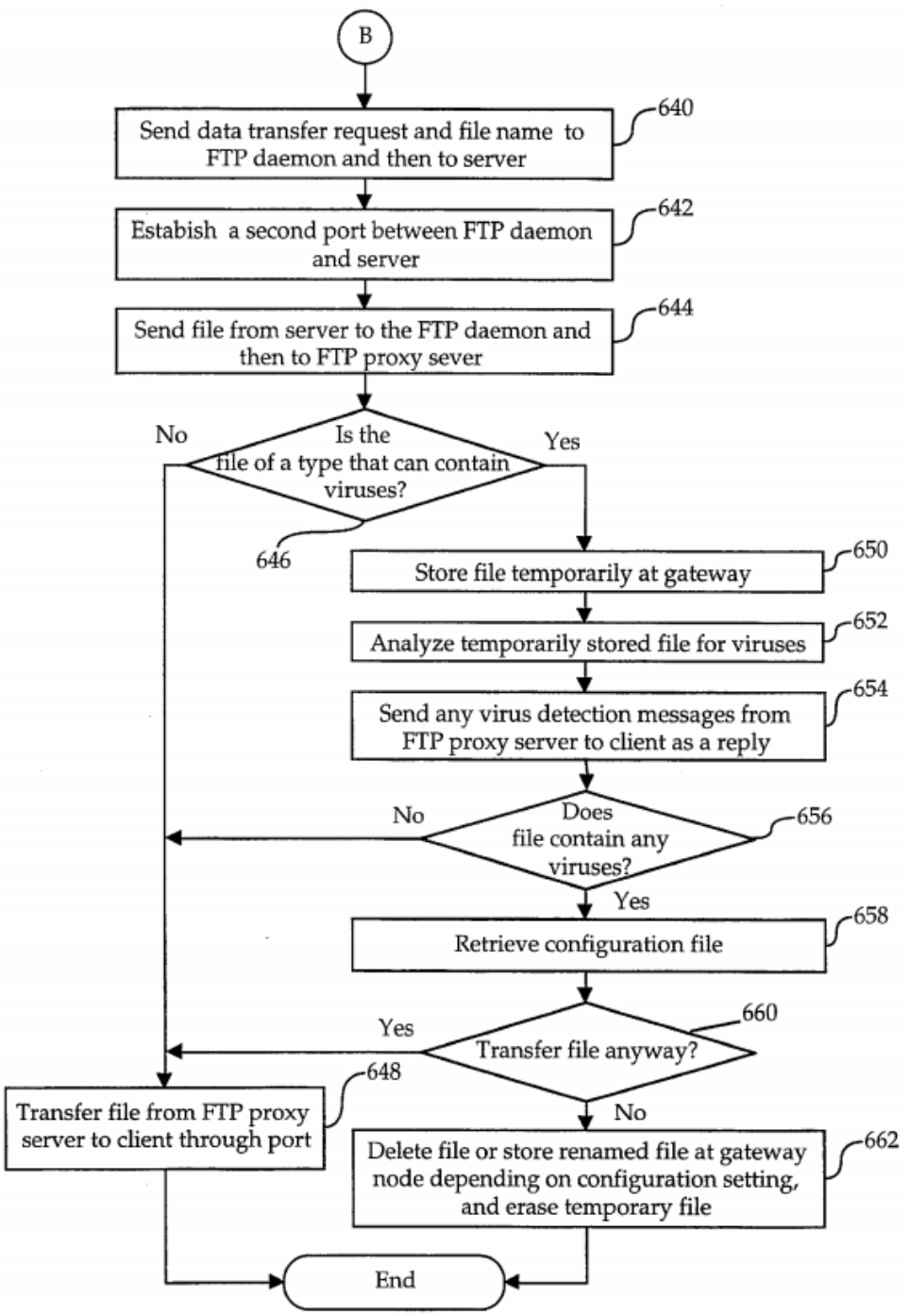

FIG. 6C

In light of the above, TBAV, either alone or in combination with Ji, would have rendered systems and methods that receive Downloadables, derive security

profile data for the Downloadables (including suspicious computer operations), and store the security profile data in a database obvious to one of ordinary skill in the art. Ex. 1002 ¶ 81. This combination renders each of claims 1, 10, 14, and 18 obvious as shown in the following charts. See Ex. 1002 ¶¶ 82-103.

Claim 1(pre)	**TBAV and Ji**
A computer-based method, comprising the steps of:	TBAV discloses a computer-based method of running software utilities on a computer system to provide virus protection. Ex. 1006, pp. 1-5; Ex. 1002 ¶ 82.
Claim 1(a)	**TBAV and Ji**
receiving an incoming Downloadable;	TBAV includes TbScan, which scans the files on a computer system. Ex. 1006, pp.47-48; Ex. 1002 ¶ 83. TBAV includes TbScanX, which automatically scans a file when it is downloaded. Ex. 1006, p. 2, 84; Ex. 1002 ¶ 84.

To the extent claim 1 requires immediate scanning after downloading, TBAV includes TbScanX. Ex. 1006, p. 2, 84; Ex. 1002 ¶ 84.

Claim 1(b)	**TBAV and Ji**
deriving security profile data for the Downloadable, including a list of suspicious computer operations that may be attempted by the Downloadable; and	TBAV includes TbScan, which performs heuristic analysis of files. Heuristic analysis includes detecting suspicious instruction sequences within a file and applying heuristic flags to the file. The heuristic flags indicate the suspicious instructions. Ex. 1006, pp. 153-155, 180-185. TbScan performs scans of files, including files that have been downloaded, whenever TbScan is run. Ex. 1006, pp.47-48. Additionally, TbScanX (a memory resident version of TbScan) automatically scans a file when it is downloaded. Ex. 1006, pp. 1-2, 76-77, 84, 153-155, 180-185; Ex. 1002 ¶ 85.

Because the heuristic flags indicate the suspicious instructions, the list of heuristic flags in the file is a list of suspicious instructions. Additionally, it would

have been obvious to one of ordinary skill in the art to provide more details about the suspicious instructions in the file. TBAV teaches that if a detailed log level is selected for the log, TbScan adds a description to the log file. Ex. 1006, pp. 76-77; Ex. 1002 ¶ 86. Also, TBAV teaches specific suspicious instructions in a signature definition, and it would have been obvious to one of ordinary skill in the art that this level of detail could be included in the log file as well. Ex. 1006, pp. 173-174; Ex. 1002 ¶ 86.

To the extent TBAV does not explicitly describe TbScanX performing heuristic analysis, TbScanX is described as "virtually identical to TbScan" except that it is memory-resident. Ex. 1006, p. 84; Ex. 1002 ¶ 87. Thus, at minimum, it would have been obvious to one of ordinary skill in the art to provide any features of TbScan in the memory-resident version of TbScanX, including heuristic analysis. Ex. 1002 ¶ 87.

Claim 1(c)	**TBAV and Ji**
storing the Downloadable security profile data in a database.	TBAV teaches that heuristic analysis results, including the heuristic flags describing the suspicious instructions, can be output to a file. Ex. 1006, p. 60; Ex. 1002 ¶ 88. TBAV also teaches using TbScan to extract instructions from files that are believed to be infected with viruses. The instructions form a signature for the virus infecting the file. TbGenSig adds the signature to the TBSCAN.SIG file. Ex. 1006, p.165-168; Ex. 1002 ¶ 89.

TBAV teaches outputting heuristic analysis results to a file and storing signatures in a file. Ex. 1002 ¶ 90. A person of ordinary skill in the art would

understand either or both of these files could be is a database containing that contains one or more data entries. In either case, the data in the file is made available for review by a user or use by TbScan at a later time. Ex. 1006, pp. 60, 165; Ex. 1002 ¶ 90. Thus, at minimum, it would have been obvious to one of ordinary skill in the art in view of TBAV to store suspicious computer operations in a database. Ex. 1002 ¶ 90.

Claim 10(pre)	**TBAV and Ji**
A system for managing Downloadables, comprising:	TBAV discloses software utilities running on a computer system that provide virus protection against downloaded viruses. The utilities manage infected downloaded files (e.g., the TbDel utility of TBAV deletes infected files). Ex. 1006, pp. 1-5, 84; Ex. 1002 ¶ 91.
Claim 10(a)	**TBAV and Ji**
a receiver for receiving an incoming Downloadable;	TBAV includes TbScan, which scans the files on a computer system. Ex. 1006, pp.47-48; Ex. 1002 ¶ 92. TBAV includes TbScanX, which automatically scans a file when it is downloaded. Ex. 1006, p. 2, 84; Ex. 1002 ¶ 93. Ji teaches an FTP proxy server 60 that receives an inbound file from FTP daemon 78 and analyzes it. Ex. 1009,col. 8, l. 57 – col. 9, l. 2; Ex. 1002 ¶ 94.

To the extent claim 1 requires immediate scanning after downloading, TBAV includes TbScanX. Ex. 1006, p. 2, 84; Ex. 1002 ¶ 93.

TBAV teaches receiving downloaded files, which one of ordinary skill in the art would understand can be done by a receiver. It was well known when TBAV was published that in order to receive data from a network, a system requires hardware and software that interfaces with the network. Such hardware and software constitutes a receiver. Ex. 1002 ¶ 95. Therefore, the receiver is at least

implicitly taught by TBAV, since a person of ordinary skill in the art would understand the computer receiving downloaded files would have the necessary hardware and software to receive them. Ex. 1002 ¶ 95. Indeed, TBAV explicitly teaches use in a networked environment. Ex. 1009, pp. 8, 11, 21-24, 33, 42, 51, 84-85, 150; Ex. 1002 ¶ 95.

To the extent TBAV does not explicitly describe a receiver, Ji teaches an FTP proxy server 60 that receives an inbound file that a client is trying to download so the file can be scanned for viruses before being passed on to the client. Ex. 1009,col. 8, l. 57 – col. 9, l. 2; Ex. 1002 ¶ 96. TBAV and Ji are both directed to virus scanning software that can scan incoming downloaded files. Thus, at minimum, it would have been obvious to one of ordinary skill in the art to apply the teachings of a receiver in Ji to the system of TBAV. Ex. 1002 ¶ 96.

Claim 10(b)	**TBAV and Ji**
a Downloadable scanner coupled with said receiver, for deriving security profile data for the Downloadable, including a list of suspicious computer operations that may be attempted by the Downloadable; and	TBAV includes TbScan, which performs heuristic analysis of files. Heuristic analysis includes detecting suspicious instruction sequences within a file and applying heuristic flags to the file. The heuristic flags indicate the suspicious instructions. performs scans of files, including files that have been downloaded, whenever TbScan is run. Ex. 1006, pp.47-48. Additionally, TbScanX (a memory resident version of TbScan) automatically scans a file when it is downloaded. Ex. 1006, pp. 1-2, 76-77, 84, 153-155, 180-185; Ex. 1002 ¶ 97.

Because the heuristic flags indicate the suspicious instructions, the list of heuristic flags in the file is a list of suspicious instructions. Additionally, it would

have been obvious to one of ordinary skill in the art to provide more details about the suspicious instructions in the file. TBAV teaches that if a detailed log level is selected for the log, TbScan adds a description to the log file. Ex. 1006, pp. 76-77; Ex. 1002 ¶ 98. Also, TBAV teaches specific suspicious instructions in a signature definition, and it would have been obvious to one of ordinary skill in the art that this level of detail could be included in the log file as well. Ex. 1006, pp. 173-174; Ex. 1002 ¶ 98.

To the extent TBAV does not explicitly describe TbScanX performing heuristic analysis, TbScanX is described as "virtually identical to TbScan" except that it is memory-resident. Ex. 1006, p. 84; Ex. 1002 ¶ 99. Thus, at minimum, it would have been obvious to one of ordinary skill in the art to provide any features of TbScan in the memory-resident TbScanX variant, including heuristic analysis. Ex. 1002 ¶ 99.

Claim 10(c)	**TBAV and Ji**
a database manager coupled with said Downloadable scanner, for storing the Downloadable security profile data in a database.	TBAV teaches that heuristic analysis results, including the heuristic flags describing the suspicious instructions, can be output to a file. Ex. 1006, p. 60; Ex. 1002 ¶ 100. TBAV also teaches using TbScan to extract instructions from files that are believed to be infected with viruses. The instructions form a signature for the virus infecting the file. TbGenSig adds the signature to the TBSCAN.SIG file. Ex. 1006, p.165-168; Ex. 1002 ¶ 101.

TBAV teaches outputting heuristic analysis results to a file and storing signatures in a file. A person of ordinary skill in the art would understand either or

both of these files could be a flat file or database containing one or more data entries. In either case, the data in the file is made available for review by a user or use by TbScan at a later time. Ex. 1006, pp. 60, 165; Ex. 1002 ¶ 102. Thus, at minimum, it would have been obvious to one of ordinary skill in the art in view of TBAV to store suspicious computer operations in a database. Ex. 1002 ¶ 1020

Claim 18	**TBAV and Ji**
The system of Claim 10 wherein said Downloadable scanner comprises a disassembler for disassembling the incoming Downloadable.	TBAV's TbScan includes a disassembler that disassembles files so they can be scanned for suspicious instructions. Ex. 1006, pp. 2, 153-155; Ex. 1002 ¶ 103.

B. Challenge to Claims 1, 10, 14, and 18 based on Arnold in view of Chen and Ji

Chen qualifies as prior art to the '494 patent under 35 U.S.C. § 102(e) because Chen was granted on an application filed in the U.S. on October 2, 1996, which is prior to the earliest possible '494 patent effective filing date.

Chen is directed to a system for detecting and removing viruses from macros that determines whether a targeted file contains a macro and, if so, decodes the macro and scans it for viruses. Ex. 1010, p. 1 (57); Ex. 1002 ¶ 105.

Macros are combinations of instructions that are recorded and made available for future use, e.g. through a user pressing an assigned key. Macros may be embedded within application data files. These macros can be executed automatically and can contain viruses. Ex. 1010, col. 1, ll. 37-52; Ex. 1002 ¶ 106.

A person of ordinary skill in the art would understand a macro to be a sequence of instructions executed through an interpreter (e.g., at the application level by a program configured to accept the combinations of instructions) and not directly executable by the operating system. Ex. 1002 ¶ 106.

Chen teaches a computer system 100 comprising a CPU 104, a memory 106, a communications unit 112 for communicating with other computers, and other computer components. Ex. 1010, col. 4, ll. 42-59; Ex. 1002 ¶ 107. The CPU 104 executes instructions received from the memory 106 to access computer files, determine whether they include macros, locate the macros, scan the macros to determine whether viruses are present in the macros (including unknown viruses), and take action against the viruses. Ex. 1010, col. 5, ll. 3-9; Ex. 1002 ¶ 107. Specifically, the memory 106 includes a macro virus detection module 206. Ex. 1010, col. 5, ll. 10-14; Ex. 1002 ¶ 107. The macro virus detection module 206 includes a macro locating and decoding module 302, a macro virus scanning module 304, a macro treating module 306, a virus information module 308, a file correcting module 310, and a data buffer 312. Ex. 1010, col. 5, l. 64 – col. 6, l. 9; Ex. 1002 ¶ 107.

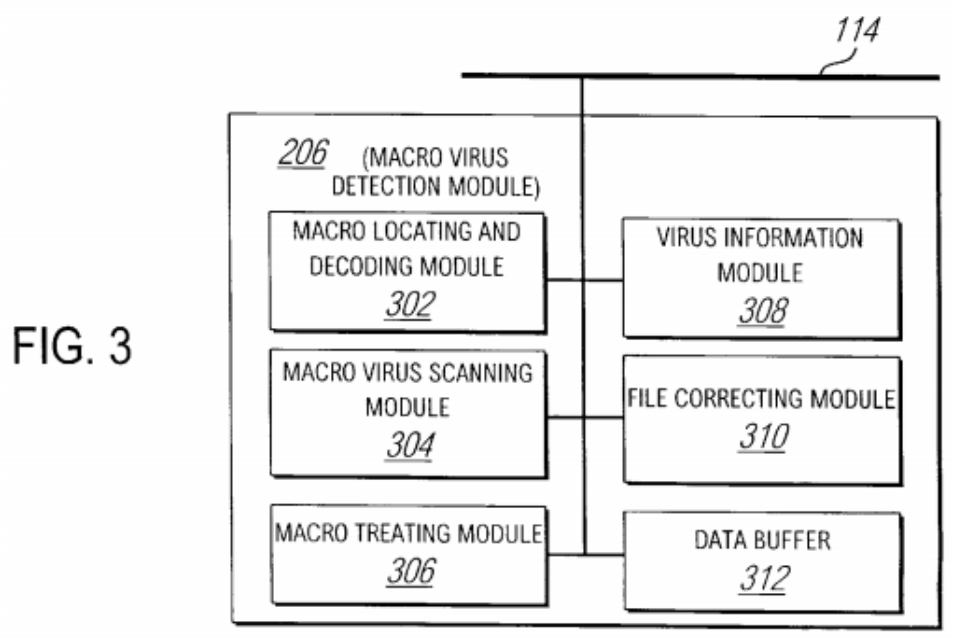

FIG. 3

Files are targeted for access by user selection or upon events taking place (e.g., on system boot, whenever certain files are opened, at periodic intervals), preferably prior to launch of an application program that may run macros in the files. Ex. 1010, col. 6, ll. 10-30; Ex. 1002 ¶ 108. The macro locating and decoding module 302 selects targeted files to be scanned for viruses by first determining whether a file in question is a type of tile that may include a macro. If the file may include a macro, the macro locating and decoding module 302 examines the file to determine whether a macro is actually present. If a macro is present, the macro is located and decoded into binary code and stored so it can be scanned for viruses. Ex. 1010, col. 12, ll. 4-65; Ex. 1002 ¶ 108.

The macro virus scanning module 304 detects combinations of suspicious instructions, which are macro instruction combinations that are likely to be used by

macro viruses. Example suspicious instructions include macro enablement instructions, which allow formatting of a file to indicate that the file includes a macro for execution, and macro reproduction instructions, which allow a macro virus to be replicated. The macro virus scanning module 304 identifies suspicious instructions based on instruction identifiers (e.g., binary strings) by scanning decoded macros for the instruction identifiers. If suspicious instructions are found, the macro virus scanning module 304 determines that the file is infected by an unknown virus, flags the decoded macro as infected, and stores information associating the decoded macro to the instruction identifiers that triggered the virus detection. Ex. 1010, col. 8, l. 40 – col. 9, l. 15; Ex. 1002 ¶ 109.

Specifically, as shown in FIG. 6, a decoded macro is scanned for known viruses using signature scanning and, if no known viruses are present, is then scanned for unknown viruses. To identify unknown viruses, the macro virus scanning module 304 obtains a set of instruction identifiers useful to detect suspect instructions that are likely to be used by macro viruses. Ex. 1010, col. 13, ll. 7-32; Ex. 1002 ¶ 110.

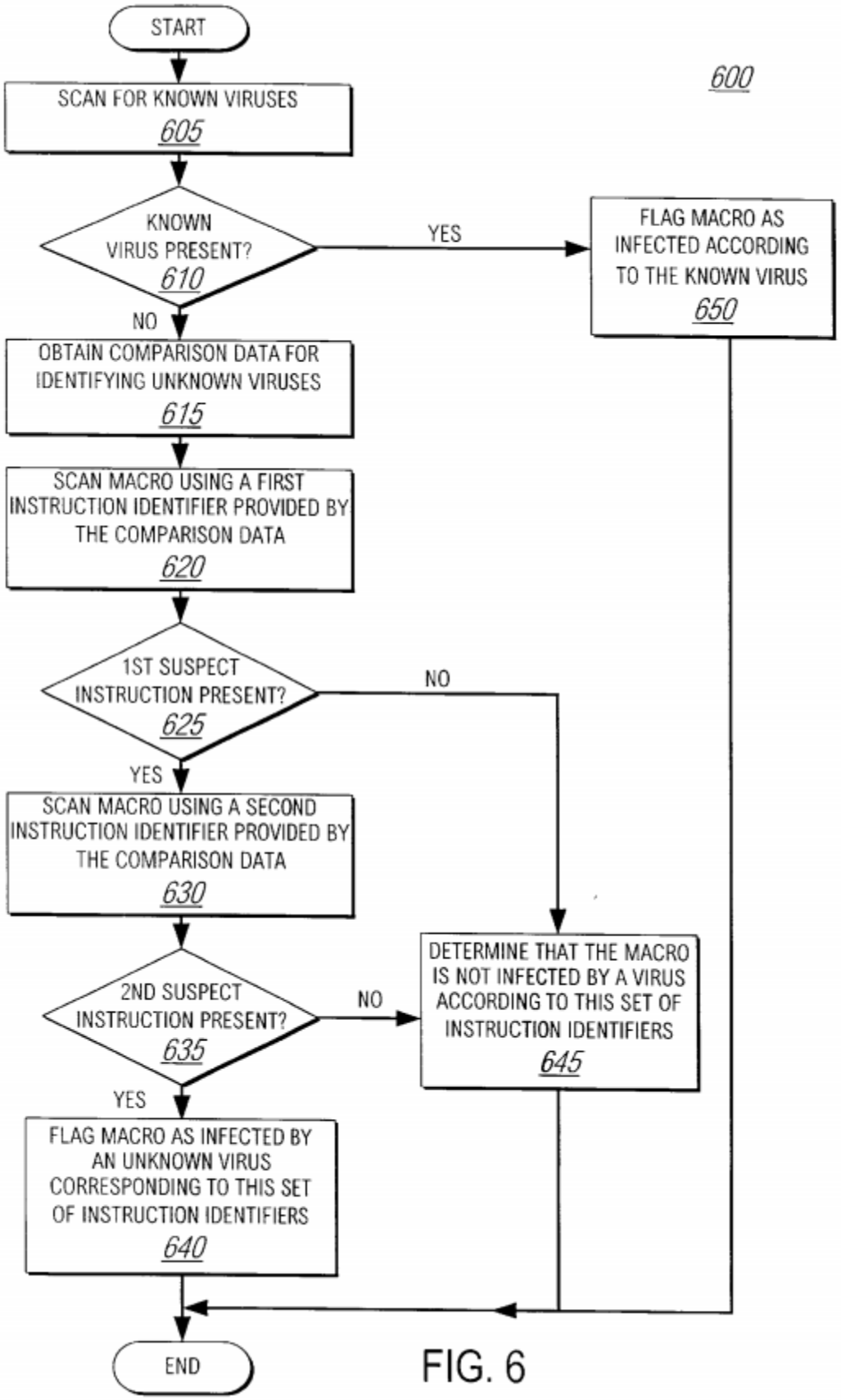
START
600
SCAN FOR KNOWN VIRUSES
605
KNOWN VIRUS PRESENT?
610
YES
FLAG MACRO AS INFECTED ACCORDING TO THE KNOWN VIRUS
650
NO
OBTAIN COMPARISON DATA FOR IDENTIFYING UNKNOWN VIRUSES
615
SCAN MACRO USING A FIRST INSTRUCTION IDENTIFIER PROVIDED BY THE COMPARISON DATA
620
1ST SUSPECT INSTRUCTION PRESENT?
625
NO
YES
SCAN MACRO USING A SECOND INSTRUCTION IDENTIFIER PROVIDED BY THE COMPARISON DATA
630
2ND SUSPECT INSTRUCTION PRESENT?
635
NO
DETERMINE THAT THE MACRO IS NOT INFECTED BY A VIRUS ACCORDING TO THIS SET OF INSTRUCTION IDENTIFIERS
645
YES
FLAG MACRO AS INFECTED BY AN UNKNOWN VIRUS CORRESPONDING TO THIS SET OF INSTRUCTION IDENTIFIERS
640
END

FIG. 6

The macro instructions, which have been decoded into binary code, are analyzed to identify the suspect instructions. For example, unique binary code portions are known to correspond to suspect instructions, and the instruction identifiers include the unique binary code portions. Ex. 1010, col. 14, ll. 24-64; Ex. 1002 ¶ 111. The macro virus scanning module 304 compares the decoded binary code with the instruction identifiers to reveal the presence of the unique binary code portions, and thus the suspicious instructions, within the macro. Ex. 1010, col. 14, ll. 13-31; Ex. 1002 ¶ 111. The macro virus scanning module 304 can scan for multiple suspicious instructions to confirm the likely presence of a macro virus. If multiple suspicious instructions are found, the macro is flagged as infected by an unknown virus. Information associating the decoded macro to the instruction identifiers that resulted in positive detection is stored in the data buffer 312 for use by other modules such as the macro treating module 306. Ex. 1010, col. 15, ll. 43-54; Ex. 1002 ¶ 111.

In light of the above, TBAV, either alone or in combination with Ji and/or Chen, would have rendered systems and methods that receive Downloadables, derive security profile data for the Downloadables (including suspicious computer operations), and store the security profile data in a database, wherein the Downloadables include program script, obvious to one of ordinary skill in the art.

Ex. 1002 ¶ 112. This combination renders claim 14 obvious as described in the following chart. See Ex. 1002 ¶¶ 113-115.

Claim 14	**TBAV, Ji, and Chen**
The system of Claim 10 wherein the Downloadable includes program script.	TBAV teaches scanning files that do not have filename extensions that indicate the files are program files, including .DLLs. Ex. 1006, p. 57; Ex. 1002 ¶ 113. Chen teaches files containing macros that are scanned by a macro virus detection module 206. Ex. 1010, col. 1, ll. 37-52, col. 5, ll. 48-52; Ex. 1002 ¶ 114.

TBAV describes Downloadables including .DLLs and other files that can contain executable code but are not program files. The code in a .DLL can be executed by another program. Ex. 1002 ¶ 115. To the extent TBAV does not describe Downloadables including program script, Chen teaches scanning macros for viruses. Ex. 1010, col. 1, ll. 37-52, col. 5, ll. 48-52; Ex. 1002 ¶ 115. TBAV and Chen are both directed to virus scanning software. As discussed above, a person of ordinary skill in the art would understand program script to be a set of instructions that can be executed through an interpreter or some other program with a computer. Ex. 1002 ¶ 115.A person of ordinary skill in the art would further understand that higher level script languages, like any other computer language, can contain malicious instructions. A macro is an example of program script often executed automatically without the user's knowledge and thus poses high potential threat. Thus, at minimum, it would have been obvious to one of ordinary skill in

the art to apply the teachings of Chen to the system of TBAV to allow the system of TBAV to scan for viruses in both executable files and files containing program script. Ex. 1002 ¶ 115.

C. Challenge to Claims 1, 10, 14, and 18 based on Arnold in view of Chen and Ji

Arnold: Arnold qualifies as prior art to the '494 patent under 35 U.S.C. § 102(b) because Arnold was granted on August 8, 1995, which is more than one year prior to the earliest possible '494 patent effective filing date.

Arnold is directed to a method wherein an undesired software entity is detected within a data processing system. An identifying signature is extracted for the undesired software entity and added to a database so it may be used by a scanner to prevent subsequent infections of the system and a network of systems. Ex. 1008, p. 1 (57); Ex. 1002 ¶ 117.

The method of Arnold is performed by a computer system 10 that can be connected to a network via network adapter 31. Ex. 1008, col. 3, l. 49 – col. 4, l. 28; Ex. 1002 ¶ 118. The method includes the steps of (A) monitoring for anomalous behavior that may be caused by a virus, (B) scanning for and removal of known viruses, (C) deploying decoy programs to capture virus samples, (D) segregating a captured virus sample into portions that likely vary among instances and portions that are likely invariant, (E) extracting a viral signature from the

invariant portions and storing the signature in a database, (F) informing other network computers of the viral infection, and (G) generating a distress signal. Ex. 1008, col. 4, ll. 29-56; Ex. 1002 ¶ 118.

When an anomaly is detected, and no known viruses are found, the anomaly may be due to an unknown virus. A decoy program is used to obtain a sample of the unknown virus. Ex. 1008, col. 6, ll. 3-7; Ex. 1002 ¶ 119. The sample is filtered to generate invariant portions of code that are unlikely to vary between instances of the virus. Ex. 1008, col. 7, ll. 11-21; Ex. 1002 ¶ 119. The invariant portions are machine language "code" portions of a program, which are less likely to vary than non-executable "data" portions of the program. Code-data segregation is performed to separate the code from the data, for example by using static disassembly techniques. Ex. 1008, col. 7, l. 59 – col. 8, l. 27; Ex. 1002 ¶ 119.

A viral signature is extracted from the likely invariant code portions. Ex. 1008, col. 9, ll. 13-16; Ex. 1002 ¶ 120. Specifically, input files (IFs 32) containing invariant portions of viral code are supplied. Ex. 1008, col. 10, ll. 13-21; Ex. 1002 ¶ 120. A list of n-grams (instances of n consecutive bytes) contained in an IF 32 containing previously identified invariant portions of virus machine code is formed. Probabilities for n-grams are estimated by comparing the n-grams to a corpus of programs, and probabilities for exact and fuzzy matches for candidate signatures are generated and combined to obtain an overall evaluation of each

candidate signature. Finally, the outcome for at least some of the candidate signatures is reported and stored in a file. Ex. 1008, col. 10, l. 63 – col. 11, l. 20; Ex. 1002 ¶ 120. For each virus, one or more candidate signatures having the best score(s) is selected to represent the virus. Ex. 1008, col. 19, ll. 15-20; Ex. 1002 ¶ 120. As these signatures are generated from previously identified invariant portions of code that caused anomalous behavior, they represent known suspicious instructions. Ex. 1002 ¶ 120.

A specific embodiment of a computer system 10 is shown in the block diagram of FIG. 8. An anomaly detector 72 detectes anomalous behavior of the CPU 14. Upon detection of anomalous behavior, a scanner 74 compares valid signatures from a signature database (SDB 74a) to programs stored in the memory to identify known viruses. Ex. 1008, col. 28, ll. 43-62; Ex. 1002 ¶ 121.

If no known virus is found, a decoy program unit 76 deploys a decoy program and, if a modification to the decoy program is detected, the undesirable software entity is isolated and supplied to the invariant code identifier 38. The invariant code identifier 38 identifies candidate signatures and provides them to the n-gram processor 40. The n-gram processor 40 processes the candidate signatures and, if a valid signature for the unknown undesirable software identity is found, the valid signature is stored in the SDB 74a. Ex. 1008, col. 29, ll. 1-23; Ex. 1002 ¶ 122.

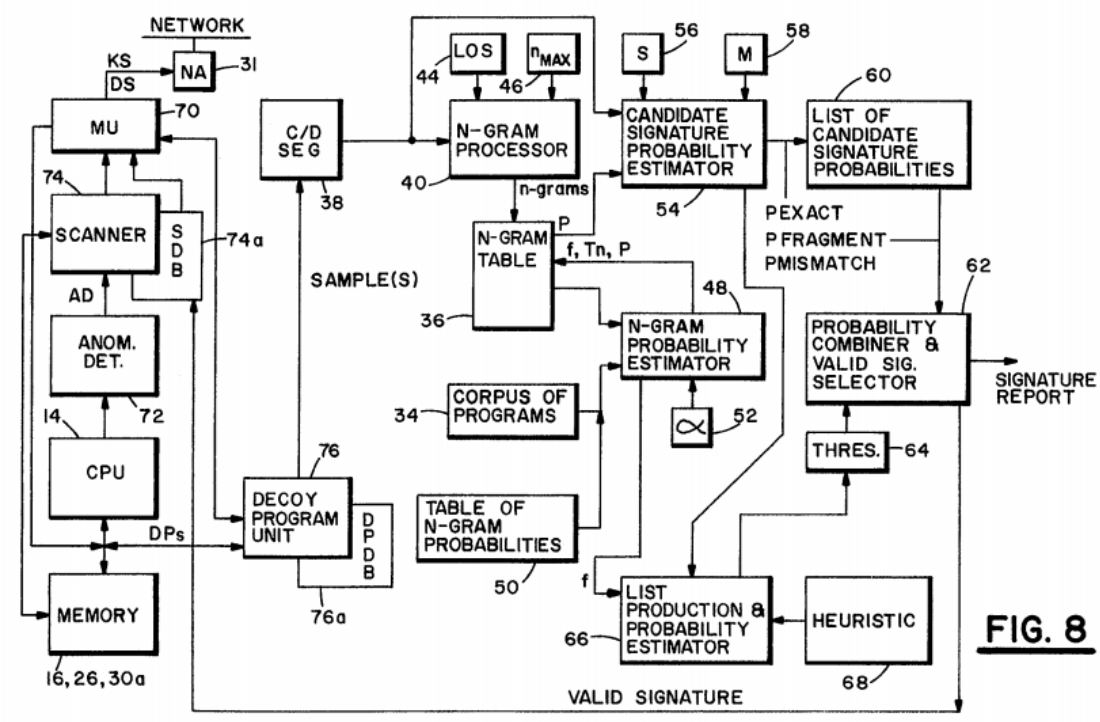

FIG. 8

Chen and Ji are described above in section IX(A) and qualify as prior art to the ’494 patent as explained therein. In light of the above, Arnold, either alone or in combination with Chen and Ji, would have rendered systems and methods that receive Downloadables, derive security profile data for the Downloadables (including suspicious computer operations), and store the security profile data in a database obvious to one of ordinary skill in the art. Ex. 1002 ¶ 123. This combination renders each of claims 1, 10, 14, and 18 obvious as shown in the following charts. See Ex. 1002 ¶¶ 124-141.

Claim 1(pre)	**Arnold, Chen, and Ji**
A computer-based method, comprising the steps of:	Arnold teaches a data processing system 10 that is suitable for practicing the teaching of the invention. Ex. 1008, col. 3, l. 49 – col. 4, l. 28; col. 27, ll. 16-20; Ex. 1002 ¶ 124.
Claim 1(a)	**Arnold, Chen, and Ji**
receiving an	Arnold teaches a data processing system 10 that is

incoming Downloadable;	connected to a network via a network adapter 31. Ex. 1008, col. 4, ll. 1-2 and 23-28; Ex. 1002 ¶ 125. Ji teaches an FTP proxy server 60 that receives an inbound file from FTP daemon 78 and analyzes it. Ex. 1009,col. 8, l. 57 – col. 9, l. 2; Ex. 1002 ¶ 126.

Arnold teaches scanning files on a network-connected computer, and one of ordinary skill in the art would recognize that these files can be incoming downloaded files. To the extent Arnold does not explicitly describe receiving an incoming Downloadable, Ji teaches an FTP proxy server 60 that receives an inbound file that a client is trying to download so the file can be scanned for viruses before being passed on to the client. Ex. 1009,col. 8, l. 57 – col. 9, l. 2; Ex. 1002 ¶ 127. Arnold and Ji are both directed to virus scanning software that can scan files on a network-connected computer. Thus, at minimum, it would have been obvious to one of ordinary skill in the art to apply the teachings of Ji of receiving a Downloadable to the system of Arnold. Ex. 1002 ¶ 127.

Claim 1(b)	**Arnold, Chen, and Ji**
deriving security profile data for the Downloadable, including a list of suspicious computer operations that may be attempted by the Downloadable; and	Arnold teaches deriving a signature for a virus by analyzing an invariant section of code from a program. Ex. 1008, col. 10, l. 63 – col. 11, l. 20; col. 19, ll. 15-20; Ex. 1002 ¶ 128. Chen teaches comparing decoded binary code with instruction identifiers to reveal the presence of code portions corresponding to the instructions of the instruction identifiers. Ex. 1010, col. 14, ll. 13-64; Ex. 1002 ¶ 129.

To the extent Arnold does not explicitly describe deriving a list of suspicious computer operations, it would have been obvious to one of ordinary skill in the art

that the signature generated by Arnold is indicative of a sequence of suspicious computer operations because it represents an invariant portion of code that caused observed anomalous behavior. Ex. 1008, col. 6, ll. 3-7; col. 7, ll. 11-21; Ex. 1002 ¶ 130. Furthermore, Chen teaches comparing instruction identifers to decoded binary code. The instruction identifers include unique binary code portions known to correspond to suspect instructions. The presence of the unique binary code portions within the decoded binary code reveals the presence of the suspicious instructions within the file. Ex. 1010, col. 14, ll. 13-31; Ex. 1002 ¶ 130. Arnold and Chen are both directed to virus scanning software that can detect unidentified viruses within a file. Thus, at minimum, it would have been obvious to one of ordinary skill in the art to apply the teachings of Chen of identifying suspicious computer operations in a file to the system of Arnold. Ex. 1002 ¶ 130.

Claim 1(c)	**Arnold, Chen, and Ji**
storing the Downloadable security profile data in a database.	Arnold teaches identifying valid signatures from among candidate signatures using an invariant code identifier 38 and storing them in a database using an n-gram processor 40. Ex. 1008, col. 29, ll. 14-23; Ex. 1002 ¶ 131.
Claim 10(pre)	**Arnold, Chen, and Ji**
A system for managing Downloadables, comprising:	Arnold teaches a data processing system 10 that is suitable for practicing the teaching of the invention. Ex. 1008, col. 3, l. 49 – col. 4, l. 28; col. 27, ll. 16-20; Ex. 1002 ¶ 132.
Claim 10(a)	**Arnold, Chen, and Ji**
a receiver for receiving an incoming Downloadable;	Arnold teaches a data processing system 10 that is connected to a network via a network adapter 31. Ex. 1008, col. 4, ll. 1-2 and 23-28; Ex. 1002 ¶ 132. Ji teaches an FTP proxy server 60 that receives an

	inbound file from FTP daemon 78 and analyzes it. Ex. 1009,col. 8, l. 57 – col. 9, l. 2; Ex. 1002 ¶ 133.

Arnold teaches scanning files on a network-connected computer comprising a network adapter, and one of ordinary skill in the art would recognize that these files can be incoming downloaded files received via the network adapter. To the extent Arnold does not explicitly describe receiving an incoming Downloadable, Ji teaches an FTP proxy server 60 that receives an inbound file that a client is trying to download so the file can be scanned for viruses before being passed on to the client. Ex. 1009,col. 8, l. 57 – col. 9, l. 2; Ex. 1002 ¶ 134. Arnold and Ji are both directed to virus scanning software that can scan files on a network-connected computer. Thus, at minimum, it would have been obvious to one of ordinary skill in the art to apply the teachings of Ji to the system of Arnold to use the network adapter to receive a Downloadable. Ex. 1002 ¶ 134.

Claim 10(b)	**Arnold, Chen, and Ji**
a Downloadable scanner coupled with said receiver, for deriving security profile data for the Downloadable, including a list of suspicious computer operations that may be attempted by the Downloadable; and	Arnold teaches deriving a signature for a virus by analyzing an invariant section of code from a program. Ex. 1008, col. 10, l. 63 – col. 11, l. 20; col. 19, ll. 15-20; Ex. 1002 ¶ 135. Chen teaches a macro virus scanning module 304 for comparing decoded binary code with instruction identifiers to reveal the presence of code portions corresponding to the instructions of the instruction identifiers. Ex. 1010, col. 14, ll. 13-64; Ex. 1002 ¶ 136.

To the extent Arnold does not explicitly describe deriving a list of suspicious computer operations, it would have been obvious to one of ordinary skill in the art

that the signature generated by Arnold is indicative of suspicious computer operations because it represents an invariant portion of code that caused observed anomalous behavior. Ex. 1008, col. 6, ll. 3-7; col. 7, ll. 11-21; Ex. 1002 ¶ 137. Furthermore, Chen teaches a macro virus scanning module 304 for comparing instruction identifers to decoded binary code. The instruction identifers include unique binary code portions known to correspond to suspect instructions. The presence of the unique binary code portions within the decoded binary code reveals the presence of the suspicious instructions within the file. Ex. 1010, col. 14, ll. 13-31; Ex. 1002 ¶ 137. Arnold and Chen are both directed to virus scanning software that can detect unidentified viruses within a file. Thus, at minimum, it would have been obvious to one of ordinary skill in the art to apply the teachings of Chen of identifying suspicious computer operations in a file to the system of Arnold. Ex. 1002 ¶ 137.

Claim 10(c)	**Arnold, Chen, and Ji**
a database manager coupled with said Downloadable scanner, for storing the Downloadable security profile data in a database.	Arnold teaches identifying valid signatures from among candidate signatures using an invariant code identifier 38 and storing them in a database using an n-gram processor 40. Ex. 1008, col. 29, ll. 14-23; Ex. 1002 ¶ 138.
Claim 14	**Arnold, Chen, and Ji**
The system of Claim 10 wherein the Downloadable includes program script.	Chen teaches files containing macros that are scanned by a macro virus detection module 206. Ex. 1010, col. 1, ll. 37-52, col. 5, ll. 48-52; Ex. 1002 ¶ 139.

To the extent Arnold does not describe Downloadables including program script, Chen teaches scanning macros for viruses. Ex. 1010, col. 1, ll. 37-52, col. 5, ll. 48-52; Ex. 1002 ¶ 140. Arnold and Chen are both directed to virus scanning software. A person of ordinary skill in the art would understand "program script" to be a sequence of instructions executed through an interpreter and not directly executable by the operating system. A macro is one example of program script. Ex. 1002 ¶ 140. Thus, at minimum, it would have been obvious to one of ordinary skill in the art to apply the teachings of Chen to the system of Arnold to allow the system of Arnold to scan for viruses in both executable files and files containing program script. Ex. 1002 ¶ 140.

Claim 18	**Arnold, Chen, and Ji**
The system of Claim 10 wherein said Downloadable scanner comprises a disassembler for disassembling the incoming Downloadable.	Arnold teaches an invariant code identifier 38 that separates machine language "code" portions of a program from non-executable "data" portions of the program using static disassembly techniques. Ex. 1008, col. 7, l. 59 – col. 8, l. 27; col. 29, ll. 10-17; Ex. 1002 ¶ 141.

D. Challenge to Claims 1, 10, 14, and 18 based on Chen in view of Arnold and Ji

Chen, Arnold, and Ji are described above in section IX(A) and qualify as prior art to the '494 patent as explained therein. In light of the above, Chen, either alone or in combination with Arnold and Ji, would have rendered systems and methods that receive Downloadables, derive security profile data for the

Downloadables (including suspicious computer operations), and store the security profile data in a database obvious to one of ordinary skill in the art. Ex. 1002 ¶ 142. This combination renders each of claims 1, 10, 14, and 18 obvious as shown in the following charts. See Ex. 1002 ¶¶ 143-165.

Claim 1(pre)	**Chen, Arnold, and Ji**
A computer-based method, comprising the steps of:	Chen teaches a computer system 100 comprising a CPU 104, a memory 106, and other computer components. The memory 106 includes a macro virus detection module 206. Ex. 1010, col. 4, ll. 42-59; col. 5, ll. 10-14; Ex. 1002 ¶ 143.
Claim 1(a)	**Chen, Arnold, and Ji**
receiving an incoming Downloadable;	Chen teaches a computer system 100 comprising a communications unit for communicating with other computers. Ex. 1010, col. 4, ll. 42-59; Ex. 1002 ¶ 144. Chen teaches targeting files for virus scanning. Ex. 1010, col. 6, ll. 10-30; Ex. 1002 ¶ 145. Ji teaches an FTP proxy server 60 that receives an inbound file from FTP daemon 78 and analyzes it. Ex. 1009,col. 8, l. 57 – col. 9, l. 2; Ex. 1002 ¶ 146.

Chen teaches scanning files on a computer that can communicate with other computers, and one of ordinary skill in the art would recognize that these files can be incoming downloaded files. To the extent Chen does not explicitly describe receiving an incoming Downloadable, Ji teaches an FTP proxy server 60 that receives an inbound file that a client is trying to download so the file can be scanned for viruses before being passed on to the client. Ex. 1009,col. 8, l. 57 – col. 9, l. 2; Ex. 1002 ¶ 147. Chen and Ji are both directed to virus scanning software that can scan files on a computer that communicates with other computers. Thus, at minimum, it would have been obvious to one of ordinary skill

in the art to apply the teachings of Ji of receiving a Downloadable to the system of Chen. Ex. 1002 ¶ 147.

Claim 1(b)	**Chen, Arnold, and Ji**
deriving security profile data for the Downloadable, including a list of suspicious computer operations that may be attempted by the Downloadable; and	Chen teaches comparing decoded binary code with instruction identifiers to reveal the presence of code portions corresponding to the instructions of the instruction identifiers. Ex. 1010, col. 14, ll. 13-64; Ex. 1002 ¶ 148.
Claim 1(c)	**Chen, Arnold, and Ji**
storing the Downloadable security profile data in a database.	Chen teaches that information associating the decoded macro to the instruction identifiers that resulted in positive detection is stored in the data buffer 312. Ex. 1010, col. 15, ll. 43-54; Ex. 1002 ¶ 149. Arnold teaches identifying valid signatures from among candidate signatures and storing them in a database. Ex. 1008, col. 29, ll. 14-23; Ex. 1002 ¶ 150.

Chen teaches storing information associating a decoded macro to instruction identifiers in a data buffer, and a person of ordinary skill in the art would understand that the same data may also be stored in a database. To the extent Chen does not explicitly describe storing the information in a database, Arnold teaches storing candidate signatures in a database. Ex. 1008, col. 29, ll. 14-23; Ex. 1002 ¶ 151. Chen and Arnold are both directed to virus scanning software wherein data about detected malicious files is stored. Thus, at minimum, it would have been obvious to one of ordinary skill in the art to apply the teachings of Arnold of a

database manager (n-gram processor 40) storing virus scan analysis data in a database to the system of Chen. Ex. 1002 ¶ 151.

Claim 10(pre)	**Chen, Arnold, and Ji**
A system for managing Downloadables, comprising:	Chen teaches a computer system 100 comprising a CPU 104, a memory 106, and other computer components. The memory 106 includes a macro virus detection module 206. Ex. 1010, col. 4, ll. 42-59; col. 5, ll. 10-14; Ex. 1002 ¶ 152.
Claim 10(a)	**Chen, Arnold, and Ji**
a receiver for receiving an incoming Downloadable;	Chen teaches a computer system 100 comprising a communications unit for communicating with other computers. Ex. 1010, col. 4, ll. 42-59; Ex. 1002 ¶ 153. Chen teaches targeting files for virus scanning. Ex. 1010, col. 6, ll. 10-30; Ex. 1002 ¶ 154. Ji teaches an FTP proxy server 60 that receives an inbound file from FTP daemon 78 and analyzes it. Ex. 1009,col. 8, l. 57 – col. 9, l. 2; Ex. 1002 ¶ 155.

Chen teaches scanning files on a computer comprising a communications unit for communicating with other computers, and one of ordinary skill in the art would recognize that these files can be incoming downloaded files received via the communications unit. To the extent Chen does not explicitly describe receiving an incoming Downloadable, Ji teaches an FTP proxy server 60 that receives an inbound file that a client is trying to download so the file can be scanned for viruses before being passed on to the client. Ex. 1009,col. 8, l. 57 – col. 9, l. 2; Ex. 1002 ¶ 156. Chen and Ji are both directed to virus scanning software that can scan files on a computer that communicates with other computers. Thus, at minimum, it would have been obvious to one of ordinary skill in the art to apply the teachings

of Ji to the system of Chen to use the communications unit to receive a Downloadable. Ex. 1002 ¶ 156.

Claim 10(b)	**Chen, Arnold, and Ji**
a Downloadable scanner coupled with said receiver, for deriving security profile data for the Downloadable, including a list of suspicious computer operations that may be attempted by the Downloadable; and	Chen teaches a macro virus scanning module 304 that compares decoded binary code with instruction identifiers to reveal the presence of code portions corresponding to the instructions of the instruction identifiers. Ex. 1010, col. 14, ll. 13-64; Ex. 1002 ¶ 157.
Claim 10(c)	**Chen, Arnold, and Ji**
a database manager coupled with said Downloadable scanner, for storing the Downloadable security profile data in a database.	Chen teaches that information associating the decoded macro to the instruction identifiers that resulted in positive detection is stored in the data buffer 312. Ex. 1010, col. 15, ll. 43-54; Ex. 1002 ¶ 158. Arnold teaches identifying valid signatures from among candidate signatures using an invariant code identifier 38 and storing them in a database using an n-gram processor 40. Ex. 1008, col. 29, ll. 14-23; Ex. 1002 ¶ 159.

Chen teaches storing information associating a decoded macro to instruction identifiers in a data buffer, and a person of ordinary skill in the art would understand that the same data may also be stored in a database by a database manager. To the extent Chen does not explicitly describe a database manager for storing the log file in a database, Arnold teaches storing candidate signatures in a database using an n-gram processor 40. Ex. 1008, col. 29, ll. 14-23; Ex. 1002 ¶ 160. Chen and Arnold are both directed to virus scanning software wherein data

about detected malicious files is stored. Thus, at minimum, it would have been obvious to one of ordinary skill in the art to apply the teachings of Arnold of a database manager (n-gram processor 40) storing virus scan analysis data in a database to the system of Chen. Ex. 1002 ¶ 160.

Claim 14	**Chen, Arnold, and Ji**
The system of Claim 10 wherein the Downloadable includes program script.	Chen teaches files containing macros that are scanned by a macro virus detection module 206. Ex. 1010, col. 1, ll. 37-52, col. 5, ll. 48-52; Ex. 1002 ¶ 161.

To the extent Chen does not explicitly use the term "program script", a person of ordinary skill in the art would understand "program script" to be a sequence of instructions executed through an interpreter and not directly executable by the operating system. A macro is one example of program script. Ex. 1002 ¶ 162.

Claim 18	**Chen, Arnold, and Ji**
The system of Claim 10 wherein said Downloadable scanner comprises a disassembler for disassembling the incoming Downloadable.	Chen teaches decoding macros into binary code. Ex. 1010, col. 14, ll. 24-64; Ex. 1002 ¶ 163. Arnold teaches an invariant code identifier 38 that separates machine language "code" portions of a program from non-executable "data" portions of the program using static disassembly techniques. Ex. 1008, col. 7, l. 59 – col. 8, l. 27; col. 29, ll. 10-17; Ex. 1002 ¶ 164.

Chen teaches decoding macros into binary code. Additionally, Arnold teaches an invariant code identifier 38 that separates machine language "code" portions of a program from non-executable "data" portions of the program using

static disassembly techniques. Ex. 1008, col. 7, l. 59 – col. 8, l. 27; col. 29, ll. 10-17; Ex. 1002 ¶ 165. Chen and Arnold are both directed to identifying suspicious code within code strings, with Chen examining macros and Arnold examining executable files. Thus, at minimum, it would have been obvious to one of ordinary skill in the art to apply the teachings of Arnold of an invariant code identifier using static disassembly techniques to the system of Chen to allow the system of Chen to work with both macros and executable files. Ex. 1002 ¶ 165.

XII. THE CHALLENGES ARE NOT REDUNDANT

Challenge A is stronger than Challenges B, C, and D in that Challenge A features a single primary reference TBAV that largely teaches the entire method of claim 1 and system of claims 10 and 18, with secondary references largely providing clarification or secondary features. Challenge B is stronger than Challenges A, C, and D in that challenge B combines the primary reference of TBAV with Chen to teach the system of claim 14. Challenge C is stronger than challenges A, B, and D in that the primary reference Arnold provides great detail about how security profile data can be derived and stored. Challenge D is stronger than Challenges A, B, and C in that the primary reference Chen strongly correlates suspicious computer operations with the results of its scan and deals largely with program script. For at least the foregoing reasons, none of the challenges are redundant.

XIII. CONCLUSION

For the foregoing reasons, Petitioner requests that Trial be instituted and claims 1, 10, 14, and 18 be cancelled.

Respectfully Submitted,

/James M. Heintz/

James M. Heintz
Registration Number 41,828
DLA Piper LLP (US)
11911 Freedom Drive, Suite 300
Reston, VA 20190
(703) 773-4148

Nicholas J. Panno
Registration Number 68,513
DLA Piper LLP (US)
11911 Freedom Drive, Suite 300
Reston, VA 20190
(703) 773-4149
Attorneys for Petitioner

CERTIFICATE OF SERVICE

The undersigned hereby certifies that a copy of the foregoing petition for *inter partes* review and all Exhibits and other documents filed together with the petition were served on April 8, 2015, via courier, directed to patent owner and patent owner correspondent at the following addresses:

Bey & Cotropia Pllc Attn: Dawn-Marie Bey 213 Bayly Ct Richmond, VA 23229-7343 804.404.2637 dbey@beycotropia.com Paul J. Andre KRAMER LEVIN NAFTALIS & FRANKEL LLP 990 Marsh Road Menlo Park, CA 94025 Telephone: (650) 752-1700 Facsimile: (650) 752-1800 pandre@kramerlevin.com	FINJAN, INC. 1313 N. Market Street Suite 5100 Wilmington, DE 19801 FINJAN, INC. 2000 University Ave Suite 600 East Palo Alto, CA 94303

By: /James M. Heintz/
James M. Heintz
Reg. No. 41828
Counsel for Petitioner