

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

Declaration of Dr. Paul C. Clark Regarding U.S. Patent No. 8,677,494

I, Paul C. Clark, do hereby declare and state, that all statements made herein of my own knowledge are true, and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Dated: April 8, 2015



Dr. Paul C. Clark

TABLE OF CONTENTS

I.	INTRODUCTION	1
A.	Engagement	1
B.	Background And Qualifications.....	1
C.	Compensation And Prior Testimony.....	3
D.	Information Considered.....	4
II.	LEGAL STANDARDS FOR PATENTABILITY	4
A.	Anticipation	5
B.	Obviousness.....	6
III.	STATE OF THE ART IN NOVEMBER 1996	12
IV.	THE '494 PATENT	13
A.	Technical Overview of the '494 Patent.....	13
B.	Prosecution History of the '494 Patent	18
C.	The Person of Ordinary Skill In The Art	20
D.	Claim Construction	20
V.	PATENTABILITY ANALYSIS OF THE '494 PATENT	24
A.	Challenge to Claims 1, 10, and 18 based on ThunderBYTE Anti-Virus Utilities User Manual in view of Ji	24
B.	Challenge to Claim 14 based on ThunderBYTE Anti-Virus Utilities User Manual in view of Ji and Chen.....	37
C.	Challenge to Claims 1, 10, 14, and 18 based on Arnold in view of Chen and Ji	43
D.	Challenge to Claims 1, 10, 14, and 18 based on Chen in view of Arnold and Ji	53
V.	CONCLUSION.....	59

I. INTRODUCTION

A. Engagement

1. I have been retained by counsel for Petitioner as an expert witness in the above-captioned proceeding. I have been asked to render an opinion regarding the validity of claims 1 and 44 of U.S. Patent No. 8,677,494 (“the ’494 patent”), which is submitted herewith as Exhibit 1001. The following contains my written report on that topic.

B. Background And Qualifications

2. In 1986, I received a Bachelor of Science degree in Mathematics from the University of California, Irvine. In 1988, I received a Master of Science degree in Electrical Engineering and Computer Science from the University of Southern California. In 1994, I received a Doctor of Science degree in Computer Science from George Washington University specializing in computer security. A copy of my current curriculum vitae is attached to the above-captioned proceeding as Exhibit 1003.

3. Since 1999, I have been President and Chief Technology Officer of SecureMethods, Inc. and Paul C. Clark LLC. SecureMethods specializes in the design, development, and deployment of advanced secure network applications for commercial and government clients, including the United States Department of Defense (“DoD”). SecureMethods provides a comprehensive, scalable, COTS-based secure architecture, implemented through

the use of the SM Gateway. The SM Gateway is a proxy-based network appliance developed by SecureMethods that is available on UNIX-based platforms. The SM Gateway proxy supports content based filtering as well as the parsing and verification of security services. In my capacity as President and Chief Technology Officer of SecureMethods, I have technical and operational oversight of all projects and corporate technical operations. I provide guidance to senior technical personnel relating to design, implementation, and troubleshooting for a wide range of systems both internal and external. My work includes network systems development and integration as well as large scale secure data access and storage. My firm specializes in complex software and hardware systems for commercial and DoD clients.

4. Over the past 25 years, as more fully set forth in my curriculum vitae, I have worked extensively in the area of secure computer and network systems development, including complex data processing, secure database, and other technologies. These activities include the design and development of: large-scale simulation and network-based systems for the DoD; a high-speed database server used for real-time intelligence collection by the National Security Agency (“NSA”); network and load generators to measure network performance load metrics for the Central Intelligence Agency (“CIA”); X Windows interfaces for a large-scale event-driven network system for the NSA; and high assurance security

systems, such as databases and applications for the NSA and the Defense Advanced Research Projects Agency (“DARPA”).

5. In addition, I invented, designed and developed the Boot Integrity Token System (BITS) upon which my thesis work was based. BITS relied upon embedded software and hardware circuitry to secure the computer startup or boot process and enable trusted computing applications both locally and over the network.

6. My Curriculum Vitae, including my publications and patents, is submitted herewith as Exhibit 1003.

C. Compensation And Prior Testimony

7. I am being compensated at a rate of \$(590) for time expended working on this matter. I also am being reimbursed for reasonable and customary expenses associated with my work and testimony in this investigation. My compensation is not contingent on the outcome of this matter or the specifics of my testimony, nor do I have a financial interest in either of the parties.

8. As detailed in my CV I have served as a testifying expert in approximately (20) matters dating back to 1996.

D. Information Considered

9. My opinions are based on my years of education, research, and experience, as well as my investigation and study of relevant materials. In forming my opinions, I have considered the materials referred to herein.

10. I may rely upon these materials and/or additional materials to rebut arguments raised by the patentee. Further, I also may consider additional documents and information in forming any necessary opinions – including documents that may not yet have been provided to me.

11. My analysis of the materials produced in this investigation is ongoing and I will continue to review any new material as it is provided. This report represents only those opinions I have formed to date. I reserve the right to revise, supplement, and/or amend my opinions stated herein based on new information and on my continuing analysis of the materials already provided.

II. LEGAL STANDARDS FOR PATENTABILITY

12. In expressing my opinions and considering the subject matter of the claims of the '494 patent, I am relying upon certain basic legal principles that counsel has explained to me.

13. First, I understand that for an invention claimed in a patent to be found patentable, it must be, among other things, new and not obvious in light

of what came before it. That which came before is generally referred to as “prior art.”

14. I understand that in this context the burden is on the party asserting unpatentability to prove it by a preponderance of the evidence. I understand that “a preponderance of the evidence” is evidence sufficient to show that a fact is more likely than not.

15. I understand that in this proceeding, the claims must be given their broadest reasonable interpretation (BRI) consistent with the specification. The claims after being construed in this manner are then to be compared to the information in the prior art.

16. I understand that in this proceeding, the information that may be evaluated is limited to patents and printed publications. My analysis below compares the claims to patents and printed publications that are prior art to the claims. I understand that there are two ways in which prior art may render a patent claim unpatentable. First, the prior art can be shown to “anticipate” the claim. Second, the prior art can be shown to “render obvious” the claim. My understanding of the two legal standards is set forth below.

A. Anticipation

17. I understand that the following standards govern the determination of whether a patent claim is “anticipated” by the prior art. I have

applied these standards in my evaluation of whether the claims asserted in this investigation are anticipated.

18. I understand that, for a patent claim to be “anticipated” by the prior art, each and every requirement of the claim must be found, expressly or inherently, in a single prior art reference as recited in the claim. I understand that claim limitations that are not expressly found in a prior art reference are inherent if the prior art necessarily functions in accordance with, or includes, the claim limitations.

19. I understand that it is acceptable to examine evidence outside the prior art reference (extrinsic evidence) in determining whether a feature, while not expressly discussed in the reference, is necessarily present within that reference.

B. Obviousness

20. I understand that a claimed invention is not patentable if it would have been obvious to a person of ordinary skill in the field of the invention at the time the invention was made.

21. I understand that the '494 patent was granted from an application that was filed on November 7, 2011. Ex. 1001. I further understand that during prosecution of the application that issued as the '494 patent, the applicants petitioned the USPTO to accept a delayed claim of priority to U.S.

Provisional Patent No. 08/790,639 which was filed on November 8, 1996. I have used November 8, 1996, as the “date the invention was made” in my analysis.

22. I understand that the obviousness standard is defined in the patent statute (35 U.S.C. § 103(a)) as follows:

A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

23. I understand that the following standards govern the determination of whether a claim in a patent is obvious. I have applied these standards in my evaluation of whether claims 1, 10, 14, and 18 of the '494 patent would have been considered obvious at the time of the invention.

24. I understand that obviousness may be shown by considering more than one item of prior art. I also understand that the relevant inquiry into obviousness requires consideration of four factors (although not necessarily in the following order):

- The scope and content of the prior art;
- The differences between the prior art and the claims at issue;
- The knowledge of a person of ordinary skill in the pertinent art; and

- Whatever objective factors indicating obviousness or non-obviousness may be present in any particular case.

25. In addition, I understand that the obviousness inquiry should not be done in hindsight, but should be done through the eyes of a person of ordinary skill in the relevant art at the time the subject patent was filed.

26. I understand the objective factors indicating obviousness or non-obviousness may include: commercial success of products covered by the patent claims; a long-felt need for the invention; failed attempts by others to make the invention; copying of the invention by others in the field; unexpected results achieved by the invention; praise of the invention by the infringer or others in the field; the taking of licenses under the patent by others; expressions of surprise by experts and those skilled in the art at the making of the invention; and the patentee's approach being contrary to the accepted wisdom of the prior art.

27. I understand that the combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results. I understand that when a work is available in one field of endeavor, design incentives and other market forces can prompt variations of that work, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, that variation would have been considered obvious. I understand that for similar reasons, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it

would improve similar devices in the same way, using that technique to improve the other device would have been obvious unless its actual application yields unexpected results or challenges in implementation.

28. I understand that the obviousness analysis need not seek out precise teachings directed to the specific subject matter of the challenged claim, but instead can take account of the “ordinary innovation” that does no more than yield predictable results, which are inferences and creative steps that a person of ordinary skill in the art would employ.

29. I understand that sometimes it will be necessary to look to interrelated teachings of multiple patents; the effects of demands known to the design community or present in the marketplace; and the background knowledge possessed by a person having ordinary skill in the art. I understand that all these issues may be considered to determine whether there was an apparent reason to combine the known elements in the fashion claimed by the patent at issue.

30. I understand that the obviousness analysis cannot be confined by a formalistic conception of the words “teaching, suggestion, and motivation.” I understand that in 2007, the Supreme Court issued its decision in *KSR Int’l Co. v. Teleflex, Inc.* where the Court rejected the previous requirement of a “teaching, suggestion, or motivation to combine” known elements of prior art for purposes of an obviousness analysis as a precondition for finding obviousness. It is my

understanding that *KSR* confirms that any motivation that would have been known to a person of skill in the art, including, common sense, or derived from the nature of the problem to be solved, is sufficient to explain why references would have been combined.

31. A person of ordinary skill attempting to solve a problem will not be led only to those elements of prior art designed to solve the same problem. I understand that under the *KSR* standard, common sense is important and should be considered. Common sense teaches that familiar items may have obvious uses beyond the particular application being described in a reference, that if something can be done once it is obvious to do it multiple times, and in many cases a person of ordinary skill will be able to fit the teachings of multiple patents together like pieces of a puzzle. As such, the prior art considered can be directed to any need or problem known in the field of endeavor at the time of the invention and can provide a reason for combining the elements of the prior art in the manner claimed. In other words, the prior art does not need to be directed towards solving the same problem that is addressed in the patent. Further, the individual prior art references themselves need not all be directed towards solving the same problem.

32. I understand that an invention that might be considered an obvious variation on, or modification of, the prior art, may nonetheless be considered non-obvious if one or more prior art references discourage or lead away

from the line of inquiry disclosed in the reference(s). A reference does not “teach away” from an invention simply because the reference suggests that another embodiment of the invention is optimal or preferred. My understanding of the doctrine of teaching away requires some clear discouragement of that combination in the prior art – such as expressly stated reasons why one should not make the claimed combination or invention.

33. I understand that a person of ordinary skill is also a person of ordinary creativity.

34. I further understand that in many fields, it may be that there is little discussion of obvious techniques or combination, and it often may be the case that market demand, rather than scientific literature or knowledge, will drive design trends. When there is such a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within their technical grasp. If this leads to the anticipated success, it is likely to be the product not of innovation but of ordinary skill and common sense. In that instance the fact that a combination was obvious to try might show that it was obvious. The fact that a particular combination of prior art elements was “obvious to try” may indicate that the combination was obvious even if no one attempted the combination. If the combination was obvious to try (regardless of whether it was

actually tried) or leads to anticipated success, then it is likely the result of ordinary skill and common sense rather than innovation.

III. STATE OF THE ART IN NOVEMBER 1996

35. 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.

36. 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.

37. 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 in 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.

38. 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. Several examples of these tools are presented in the following discussion.

IV. THE '494 PATENT

A. Technical Overview of the '494 Patent

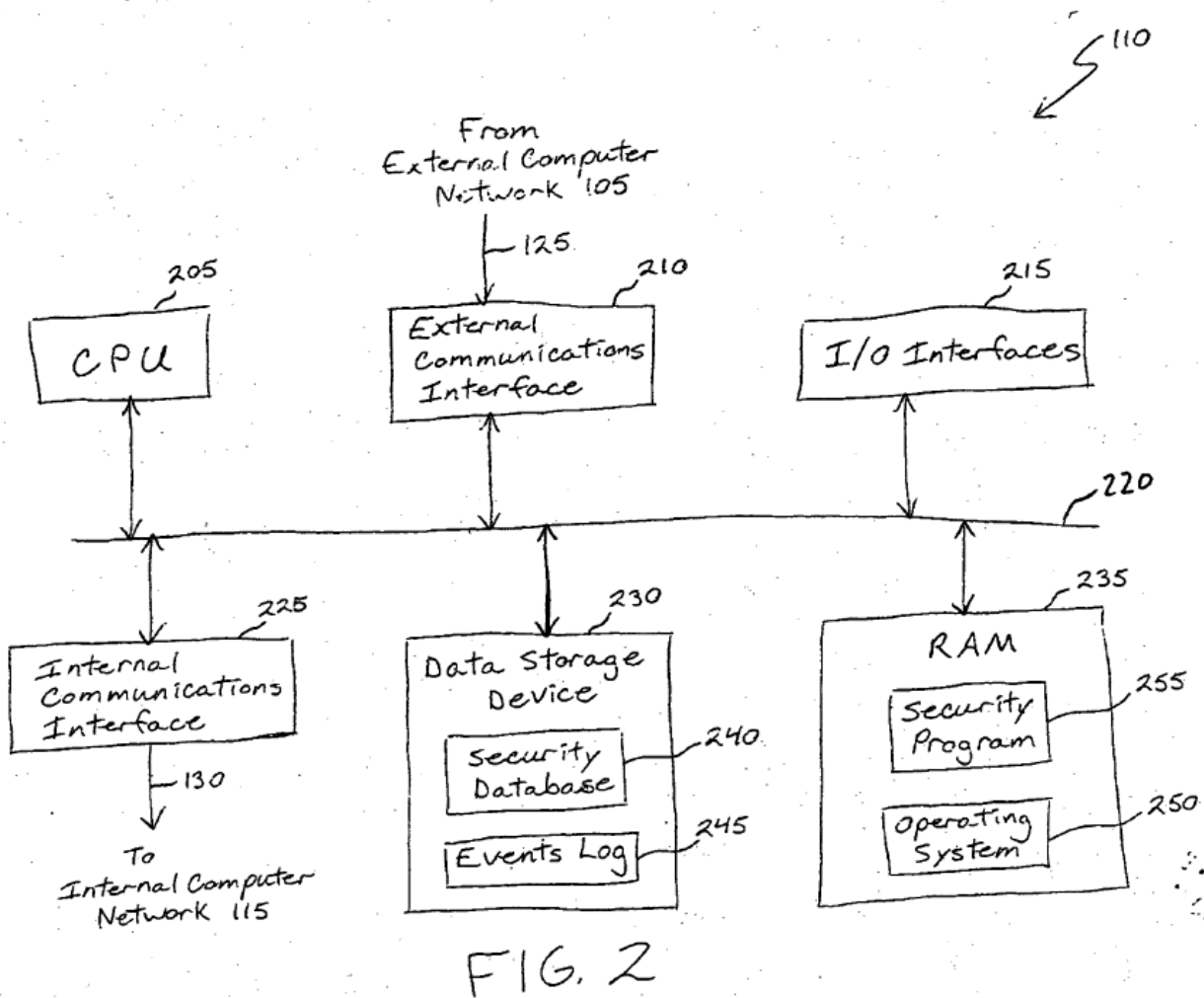
39. The '494 patent relates generally to protecting personal computers (PCs) or other devices from malicious software obtained over a network. The '494 patent refers to executable files and software obtained over a network as "Downloadables" and focuses on identifying malicious Downloadables. The specification 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.

40. The '494 patent claims the benefit of U.S. Provisional Patent Application No. 60/030,639 to Touboul et al. (hereinafter "the '639 application"). It is this disclosure, of the '639 application, that 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. A Downloadable is an executable application program which is downloaded from a source computer and run on the destination computer. Ex. 1005, p. 2, ll. 1-4. Examples of Downloadables include executables as well as applets for Java and Active X. Ex. 1005, p. 2, ll. 4-7. According to the '639 application, viruses may be attached to Downloadables. Ex. 1005, p. 1, ll. 20-22; p. 2, ll. 7-9.

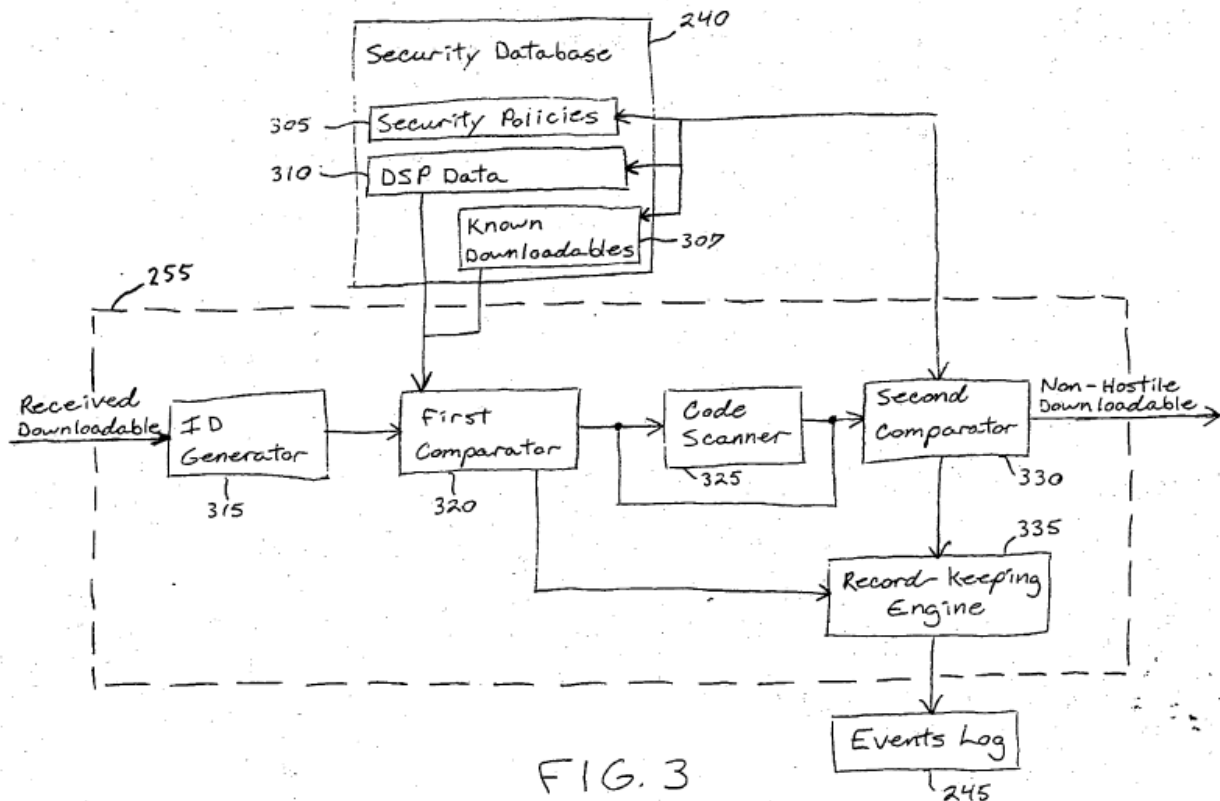
41. The method taught by the '639 application includes receiving a Downloadable, and, when the Downloadable does not get identified as a previously identified hostile Downloadable, deriving Downloadable security profile data from the received Downloadable. Ex. 1005, p. 3, ll. 13-21.

42. The specification describes an internal network security system 110 that 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. 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. The security program 255 in memory 235 controls the operations of the internal network security system 110. Ex. 1005, p. 8, ll. 1-4. The security database 240 in the data storage device 230 stores Downloadable security profiles (DSPs). Ex. 1005, p. 8, ll. 14-19.



43. 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.



44. 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. 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. 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.

45. 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. 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. 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. 1005, p. 15, l. 19 – p. 16, l. 8.

46. 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. 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.

B. Prosecution History of the '494 Patent

47. I understand that the '494 patent names Yigal Mordechai Edery, Nimrod Itzhak Vered, David R. Kroll, and Shlomo Touboul as inventors.

48. I understand the application from which the '494 patent issued, U.S. patent application No. 13/290,708 ("the '708 application"), was filed on November 7, 2011.

49. I understand 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. I understand 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. I understand this petition was dismissed on November 27, 2012 because it was not accompanied by a copy of the '639 application. Ex. 1004, pp. 719-720.

50. I understand 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. I understand 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.

51. I understand a notice of allowance was mailed on August 29, 2013. Ex. 1004, pp. 118-124. I understand 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. I understand 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. The Person of Ordinary Skill In The Art

52. 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.

D. Claim Construction

53. I understand that, at the Patent Office, claims are to be given their broadest reasonable interpretation in light of the specification as would be read by a person of ordinary skill in the relevant art.

54. I also understand that the words in the claims are to be evaluated using the perspective of a person of ordinary skill in the art at the time the application was filed.

55. I understand that where a patent applicant provides an explicit definition of a claim term in the specification, that definition may control the interpretation of that term in the claim.

56. I understand that if no explicit definition is given to a term in the patent specification, the claim terms must be evaluated using the ordinary meaning of the words being used in those claims.

57. I have considered the '494 patent in light of the specification, reading the claim limitations as one of ordinary skill in the art, and provide the following analysis. As I explain below, I believe claims 1, 10, 14, and 18 of the '494 patent are unpatentable at least for obviousness over the prior art.

58. I understand that claims 1 and 10 are independent claims, and claims 14 and 18 depend from claim 10.

1. “Downloadable”

59. The term “Downloadable” appears in claims 1, 10, 14, and 18.

60. Under broadest reasonable interpretation (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. 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. This construction is also consistent with what was agreed upon by Petitioner and Patent Owner in related proceedings cited above. Ex. 1011, p. 4.

2. “suspicious computer operations”

61. The term “suspicious computer operations” appears in claims 1 and 10.

62. 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. 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.

3. “database”

63. The term “database” appears in claims 1 and 10.

64. 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. 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.

4. “program script”

65. The term “program script” appears in claim 14.

66. 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.” 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. 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. 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.

V. PATENTABILITY ANALYSIS OF THE ’494 PATENT

67. I have been asked to review certain claims of the ’494 patent, and compare those to the prior art known to me.

68. As detailed below, I believe claims 1, 10, 14, and 18 of the ’494 patent would have been obvious to one of ordinary skill in the art as of the earliest

possible effective filing date of the '494 patent, November 8, 1996, on the grounds set forth below.

A. Challenge to Claims 1, 10, and 18 Based on ThunderBYTE Anti-Virus Utilities User Manual ("TBAV") in view of U.S. Patent No. 5,623,600 to Ji et al. ("Ji")

69. 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 to the PTO 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 in 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. Likewise, 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.

TBAV

70. 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.

71. 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. 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. 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.

72. 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. 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.

73. 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.

74. 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. 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.

75. 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. 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.

76. TBAV also includes a TbGenSig program that can be called by TbScan to define signature records for suspicious files. Ex. 1006, pp. 4, 57. 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 of the potential virus infecting the file. Ex. 1006, pp. 166-168. A name,

one or more keywords, and the instructions are added to a file (e.g., USERSIG.DAT). Ex. 1006, p. 168. Note that TBAV explicitly uses *.DAT files as databases (see also, ANTI-VIR.DAT). Ex. 1006, pp. 4, 36. TbGenSig further adds the virus name, the keywords, and the suspicious instructions to the TBSCAN.SIG file. Ex. 1006, p.165. 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.

77. 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. TbScanX is “virtually identical to TbScan” with the only difference being that it is memory-resident. Ex. 1006, p. 84. 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.

Ji

78. 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).

79. 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. 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. 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.

80. 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. 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. 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.

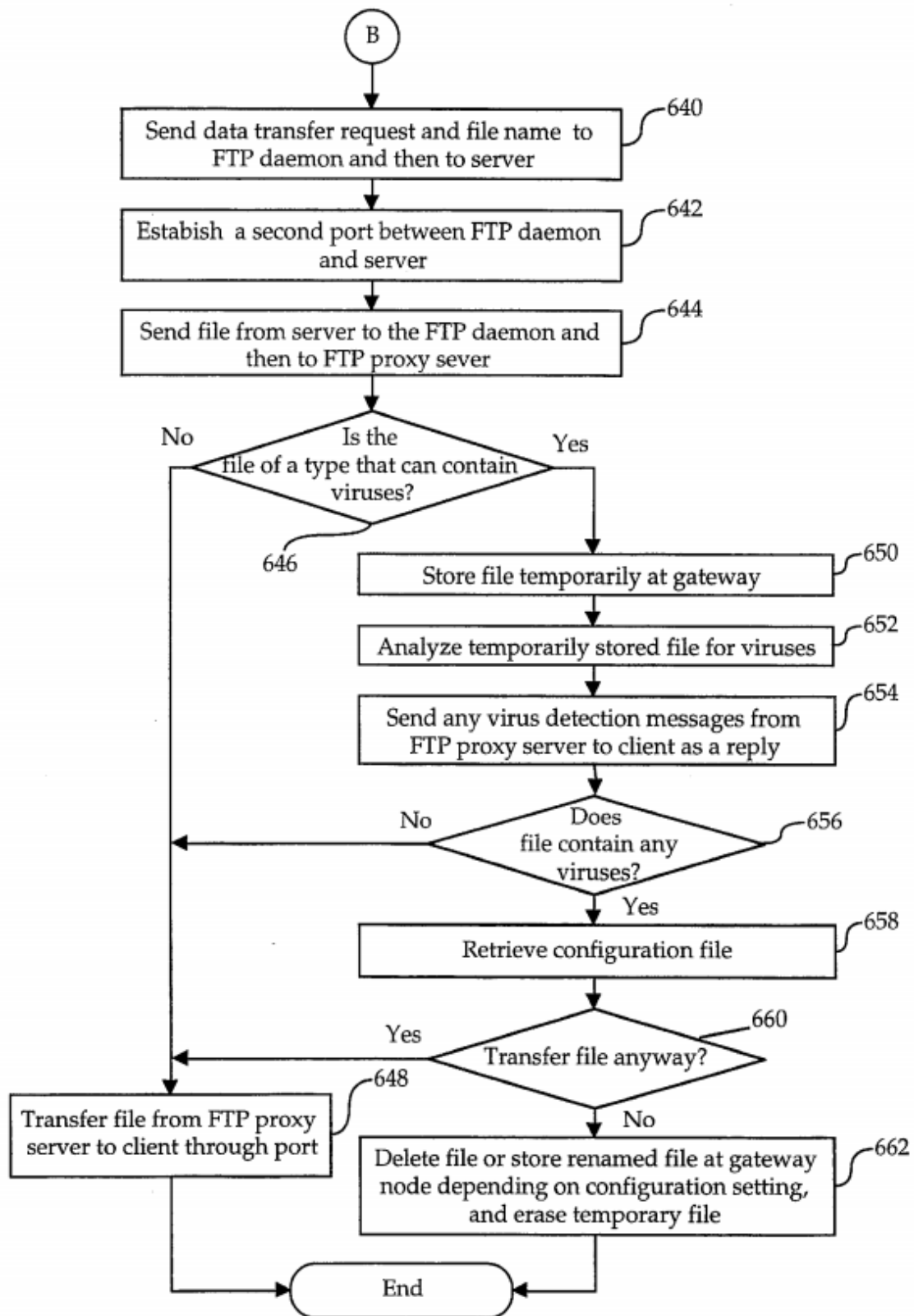


FIG. 6C

81. 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. This combination renders each of claims 1, 10, and 18 obvious as described below.

1. Claim 1

82. TBAV discloses a computer-based method of running software utilities on a computer system to provide virus protection. Ex. 1006, pp. 1-5.

83. TBAV includes TbScan, which scans the files on a computer system. Ex. 1006, pp.47-48.

84. To the extent claim 1 requires immediate scanning after downloading, TBAV includes TbScanX, which automatically scans a file when it is downloaded. Ex. 1006, p. 2, 84.

85. 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.

86. 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. 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.

87. 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. 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.

88. TBAV teaches that heuristic analysis results, including the heuristic flags describing the suspicious instructions, can be output to a file. Ex. 1006, p. 60.

89. 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. Ex. 1006, pp. 166-168. TbGenSig adds the signature to the TBSCAN.SIG file. Ex. 1006, p.165.

90. 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 is a database 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. 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.

2. Claim 10

91. TBAV discloses software running on a computer system that provides virus protection against downloaded viruses. TBAV's software utilities manage infected downloaded files (e.g., the TbDel utility of TBAV deletes infected files). Ex. 1006, pp. 1-5, 84.

92. TBAV includes TbScan, which scans the files on a computer system. Ex. 1006, pp.47-48.

93. To the extent claim 1 requires immediate scanning after downloading, TBAV includes TbScanX, which automatically scans a file when it is downloaded. Ex. 1006, p. 2, 84.

94. 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.

95. 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. 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. Indeed, TBAV explicitly teaches use in a networked environment. Ex. 1009, pp. 8, 11, 21-24, 33, 42, 51, 84-85, 150.

96. 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. 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.

97. 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.

98. 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. 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.

99. 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. 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.

100. TBAV teaches that heuristic analysis results, including the heuristic flags describing the suspicious instructions, can be output to a file. Ex. 1006, p. 60.

101. 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. Ex. 1006, pp. 166-168. TbGenSig adds the signature to the TBSCAN.SIG file. Ex. 1006, p.165.

102. 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. 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.

3. Claim 18

103. TBAV’s TbScan includes a disassembler that disassembles files so they can be scanned for suspicious instructions. Ex. 1006, pp. 2, 153-155.

B. Challenge to Claim 14 Based on ThunderBYTE Anti-Virus Utilities User Manual (“TBAV”) in view of U.S. Patent No. 5,623,600 to Ji et al. (“Ji”) and U.S. Patent No. 5,951,698 to Chen et al. (“Chen”)

104. 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.

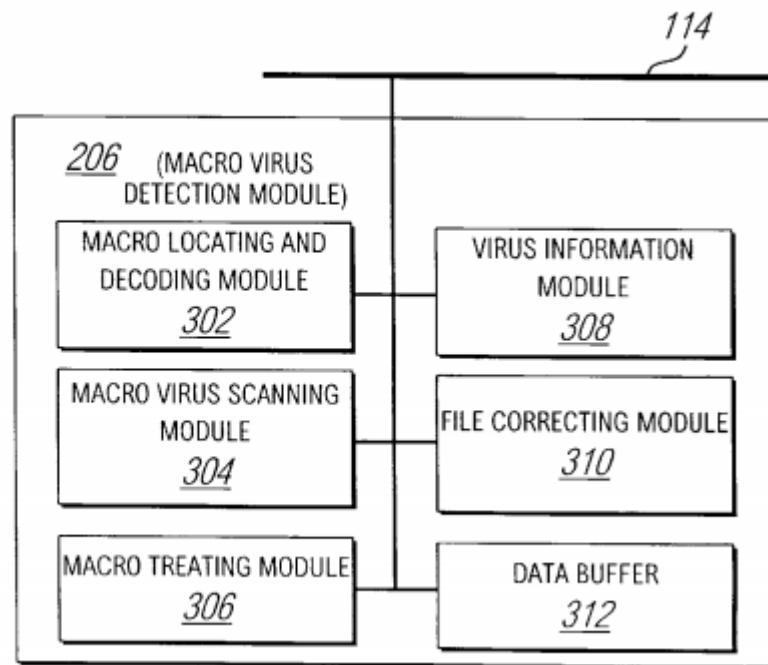
105. Chen is directed to a system for detecting and removing viruses from macros that determines whether a targeted file contains a macro and, if so, decodes the macro and scans it for viruses. Ex. 1010, p. 1 (57).

106. 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. 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.

107. 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. 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. Specifically, the memory 106 includes a macro virus detection module 206. Ex. 1010, col. 5, ll. 10-14. 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.

FIG. 3



108. 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. 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 file 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.

109. 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.

110. 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.

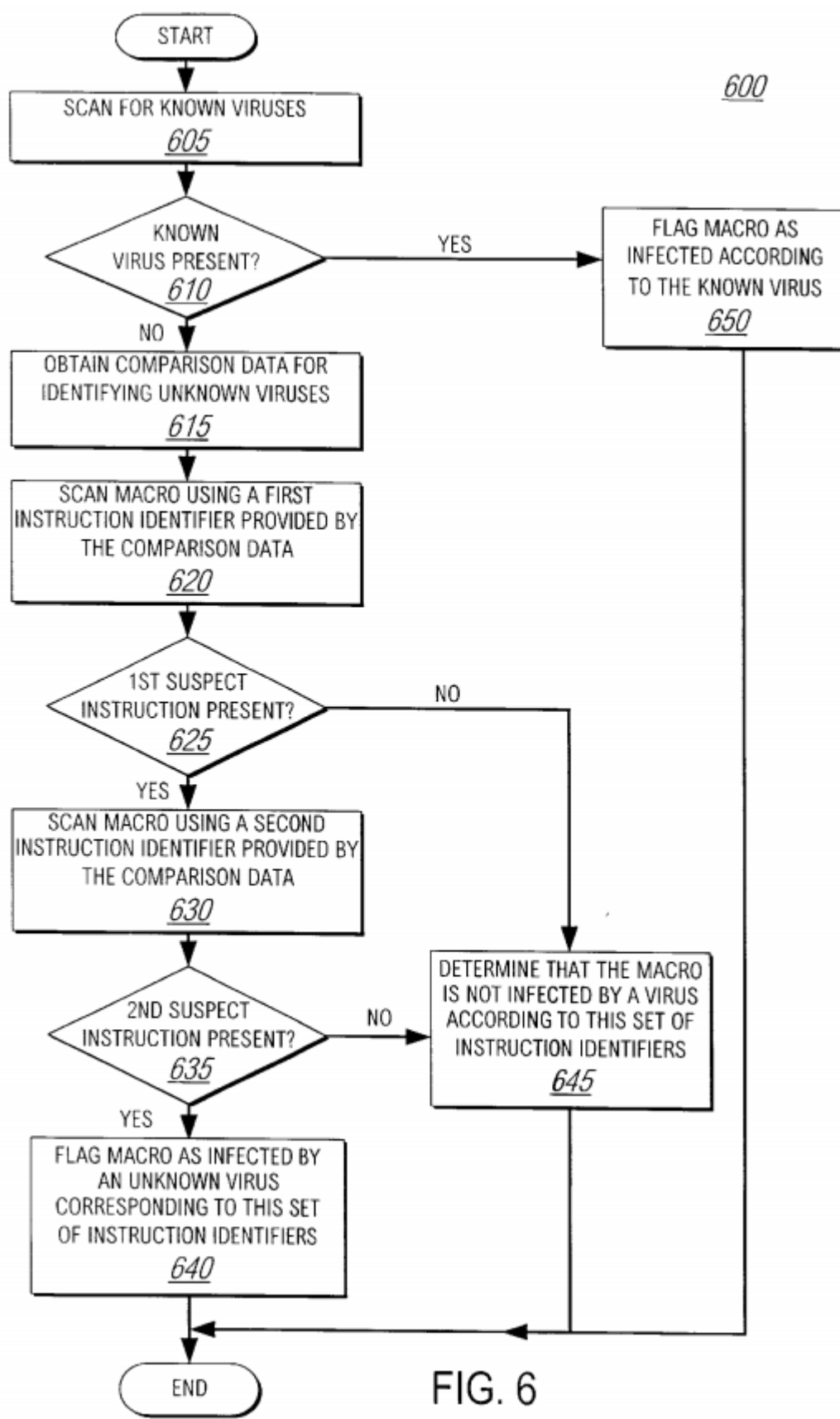


FIG. 6

111. 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.

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.

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.

112. 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. This combination renders claim 14 obvious as described below.

Claim 14

113. TBAV teaches scanning files that do not have filename extensions that indicate the files are program files, including .DLLs. Ex. 1006, p. 57.

114. 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.

115. 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. 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. 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. 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.

C. Challenge to Claims 1, 10, 14, and 18 Based on U.S. Patent No. 5,440,723 to Arnold et al. (“Arnold”) in view of Chen and Ji

116. 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.

117. 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).

118. 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. 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.

119. 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

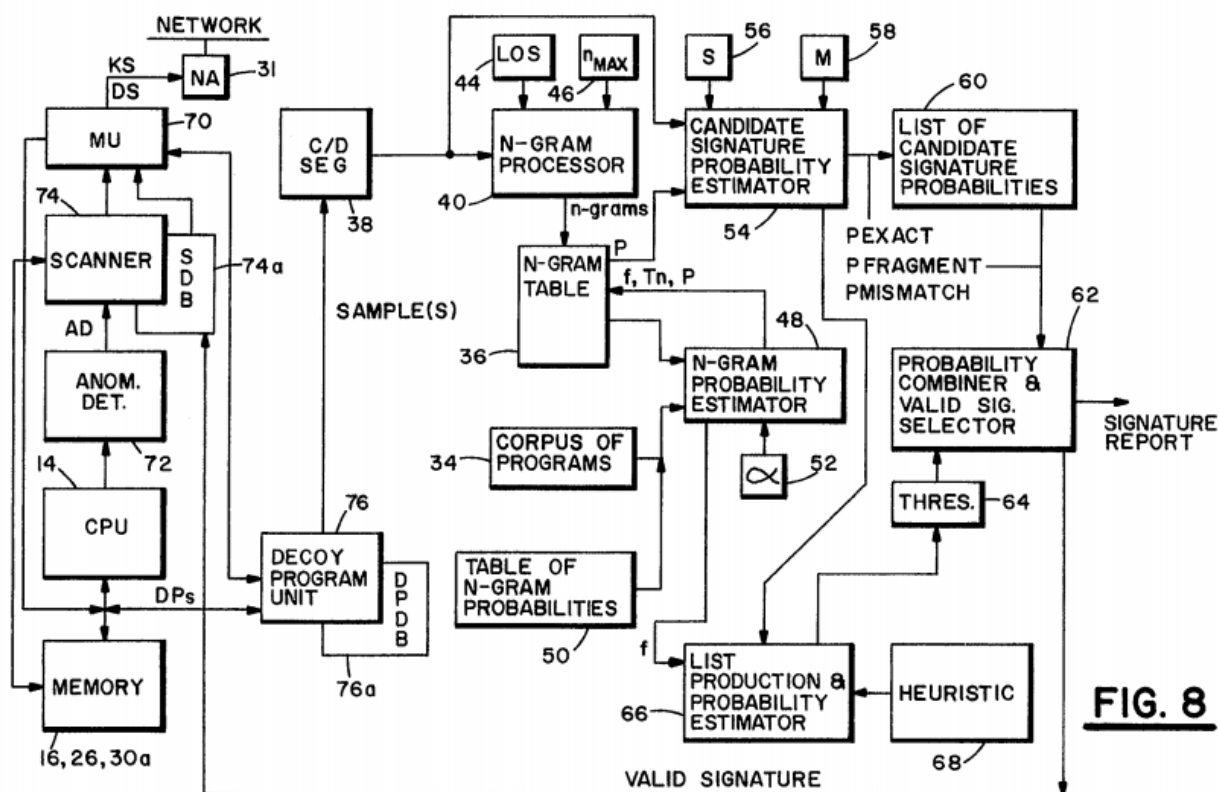
sample of the unknown virus. Ex. 1008, col. 6, ll. 3-7. The sample is filtered to generate invariant portions of code that are unlikely to vary between instances of the virus. Ex. 1008, col. 7, ll. 11-21. The invariant portions are machine language “code” portions of a program, which are less likely 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.

120. A viral signature is extracted from the likely invariant code portions. Ex. 1008, col. 9, ll. 13-16. Specifically, input files (IFs 32) containing invariant portions of viral code are supplied. Ex. 1008, col. 10, ll. 13-21. 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. 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. As these signatures are generated from

previously identified invariant portions of code that caused anomalous behavior, they represent known suspicious instructions.

121. A specific embodiment of a computer system 10 is shown in the block diagram of FIG. 8. An anomaly detector 72 detects 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.

122. 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.



123. 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. This combination renders each of claims 1, 10, 14, and 18 obvious as shown as described below.

1. Claim 1

124. 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.

125. 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.

126. 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.

127. 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. 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.

128. 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.

129. 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.

130. 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.

Furthermore, Chen teaches comparing instruction identifiers to decoded binary code. The instruction identifiers 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. 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.

131. 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.

2. Claim 10

132. 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.

133. 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.

134. 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. 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.

135. 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.

136. 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.

137. 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. Furthermore, Chen teaches a macro virus scanning module 304 for comparing instruction identifiers to decoded binary code. The instruction identifiers 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. 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.

138. 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.

3. Claim 14

139. 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.

140. 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. Arnold and Chen are both directed to virus scanning software. As discussed above, amacro is one example of program script. 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.

4. Claim 18

141. 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.

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

142. 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. This combination renders each of claims 1, 10, 14, and 18 obvious as shown as described below.

1. Claim 1

143. 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.

144. Chen teaches a computer system 100 comprising a communications unit for communicating with other computers. Ex. 1010, col. 4, ll. 42-59.

145. Chen teaches targeting files for virus scanning. Ex. 1010, col. 6, ll. 10-30.

146. 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.

147. 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 ¶ 55. 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.

148. 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.

149. 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.

150. Arnold teaches identifying valid signatures from among candidate signatures and storing them in a database. Ex. 1008, col. 29, ll. 14-23.

151. 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. 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.

2. Claim 10

152. 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.

153. Chen teaches a computer system 100 comprising a communications unit for communicating with other computers. Ex. 1010, col. 4, ll. 42-59.

154. Chen teaches targeting files for virus scanning. Ex. 1010, col. 6, ll. 10-30.

155. 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.

156. 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. 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.

157. 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.

158. 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.

159. 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.

160. 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. 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.

3. Claim 14

161. 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.

162. To the extent Chen does not explicitly use the term “program script”, a person of ordinary skill in the art would understand a macro to be one example of program script, as discussed above.

4. Claim 18

163. Chen teaches decoding macros into binary code. Ex. 1010, col. 14, ll. 24-64.

164. 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.

165. 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 ¶ 55. 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.

VI. CONCLUSION

166. For the reasons set forth above, it is my opinion that the subject matter recited in claims 1, 10, 14, and 18 of the '494 patent was obvious to a person of ordinary skill in the art prior to the earliest effective filing date of the '494 patent.