IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

_____

BEFORE THE PATENT TRIAL AND APPEAL BOARD

_____

TREND MICRO, INC.
Petitioner,

v.

TAASERA LICENSING LLC
Patent Owner.

_____

IPR2023-00798

U.S. Patent No. 8,850,517

DECLARATION OF SETH JAMES NIELSON

TABLE OF CONTENTS

I. INTRODUCTION 1

II. QUALIFICATIONS 1

III. MATERIALS CONSIDERED 12

IV. LEGAL STANDARDS 13

V. Technical Background that Would Have Been Known to A POSITA 15

A. System boot and login processes 16

B. System files, registry, and system calls 18

C. User metrics 22

VI. OVERVIEW OF THE CHALLENGED PATENT 22

A. THE ’517 PATENT SPECIFICATION AND CLAIMS 22

B. PROSECUTION HISTORY 24

VII. LEVEL OF SKILL IN THE ART 25

VIII. Overview of the Prior art 26

A. OVERVIEW OF US20110023115A1 PUBLISHED APPLICATION TO WRIGHT (“WRIGHT”) 26

B. OVERVIEW OF US 8,806,647 TO DASWANI (“DASWANI”) 29

IX. GROUND 1: Claims 1-24 are rendered obvious by WRIGHT IN VIEW OF DASWANI 30

A. Motivation to Combine Wright and Daswani 30

B. Claim 1 is rendered obvious by Wright in view of Daswani 35

1. Limitation 1[a] 35

2. Limitation 1[b] 37

3. Limitation 1[c] 44

4. Limitation 1[d] 46

5. Limitation 1[e] 48

6. Limitation 1[f] 52

7. Limitation 1[g] 53

C. Claim 2 is rendered obvious by Wright in view of Daswani 55

D. Claim 3 is rendered obvious by Wright in view of Daswani 57

E. Claim 4 is rendered obvious by Wright in view of Daswani 58

F. Claim 5 is rendered obvious by Wright in view of Daswani 59

G. Claim 6 is rendered obvious by Wright in view of Daswani 60

H. Claim 7 is rendered obvious by Wright in view of Daswani 60

I. Claim 8 is rendered obvious by Wright in view of Daswani 62

J. Claim 9 is rendered obvious by Wright in view of Daswani 63

K. Claim 10 is rendered obvious by Wright in view of Daswani 64

L. Claim 11 is rendered obvious by Wright in view of Daswani 67

M. Claim 12 is rendered obvious by Wright in view of Daswani 68

1. Limitation 12[a] 68

2. Limitation 12[b] 69

N. Claim 13 is rendered obvious by Wright in view of Daswani 70

1. Limitation 13[a] 70

2. Limitation 13[b] 71

3. Limitation 13[c] 71

4. Limitation 13[d] 71

5. Limitation 13[e] 72

6. Limitation 13[f] 72

7. Limitation 13[g] 72

8. Limitation 13[h] 73

O. Claim 14 is rendered obvious by Wright in view of Daswani 73

P. Claim 15 is rendered obvious by Wright in view of Daswani 73

Q. Claim 16 is rendered obvious by Wright in view of Daswani 74

R. Claim 17 is rendered obvious by Wright in view of Daswani 74

S. Claim 18 are rendered obvious by Wright in view of Daswani 75

T. Claim 19 is rendered obvious by Wright in view of Daswani 75

U. Claim 20 is rendered obvious by Wright in view of Daswani 75

V. Claim 21 is rendered obvious by Wright in view of Daswani 76

W. Claim 22 is rendered obvious by Wright in view of Daswani 76

X. Claim 23 is rendered obvious by Wright in view of Daswani 77

Y. Claim 24 is rendered obvious by Wright in view of Daswani 77

Trend Micro, Inc.
EX1003-3

1. Limitation 24[a] 77

2. Limitation 24[b] 78

X. There are no objective indicia of nonobviousness 78

XI. Conclusion 78

1. I, Seth James Nielson, submit the following declaration (the "Declaration") in connection with the proceeding identified above.

I. INTRODUCTION

2. I have been retained by counsel for Trend Micro, Inc. ("Trend Micro" or "Petitioner") as a technical expert in connection with the proceeding identified above. I submit this Declaration in support of Petitioner Trend Micro's Petition for Inter Partes Review (the "Petition") of United States Patent No. 8,850,517 ("the '517 patent") against Taasera Licensing LLC ("Taasera"). All "Ex. 10XX" cites herein are to the Exhibits to the Petitions. All citations to "Section XX" are internal citations to the sections of this Declaration. All citations to "Paragraph XX" are internal citations to the paragraphs of this Declaration.

3. I have been asked to consider whether the claims of the '517 patent are obvious in light of the prior art discussed below. The '517 patent is directed to general application of computer security techniques to identify malware through behavioral analysis.

4. In summary, my conclusion is that the claims of the '517 patent are obvious in light of the prior art discussed below. As set forth in detail in this declaration, I have reached the following conclusions:

a. Claims 1-24 are rendered obvious to a POSITA by the combination of U.S. Patent Publication No. 2011/0023115 A1 to Clifford C. Wright ("Wright") in view of U.S. Patent No. 8,806,647 to Neilkumar Daswani ("Daswani").

II. QUALIFICATIONS

5. I am a subject matter expert in cybersecurity, computer networks, and software engineering. I am the Founder and Chief Scientist of Crimson Vista, a computer security research and engineering company. Furthermore, I hold appointments at the University of Texas at Austin

as an Adjunct Associate Professor in the department of Computer Science and as a Cybersecurity Fellow in the Robert Strauss Center for International Security and Law.

6. I have been working with computer networks and associated technology, including parallel processing, since January 2000. My experience includes graduate-level teaching, academic research, industry employment, and consulting practice.

7. Exhibit 1004 is a current copy of my C.V. and includes all of my relevant education and work experience, a list of all publications that I have authored in the previous ten years, as well as a list of all cases in which, during the previous four years, I have testified as an expert at trial or by deposition. My experience most relevant to the opinions offered in this report is summarized below.

8. I received my B.S. in Computer Science in April of 2000. During my final undergraduate semester, I worked both as a teaching assistant for a Computer Networking course and as a researcher in the Networked Computing Lab. In these capacities, I assisted students in debugging and designing TCP/IP protocol stacks, Address Resolution Protocol implementations, and Remote Procedure Call projects. In the research lab, I collaborated on work related to resource reservations in high performance networking and MPLS analyses. Our investigations into statistical traffic engineering for bandwidth allocation resulted in a published paper entitled, "Effective Bandwidth for Traffic Engineering."

9. During the Fall of 2000, I studied for one semester at Brandeis University in the Boston area and, as part of my research, I developed a parallel-processing system for DNA string matching. The system that I created had a central node that managed the search space and solution aggregation. A cluster of available "worker" machines registered with the central node to indicate their availability for parallel processing. The central node would identify subdivisions of the search

space that could be farmed out to the worker nodes for processing. I also studied artificial intelligence and machine learning.

10. From 2001 through 2003, I worked as a software engineer at Metrowerks (formerly Lineo, Inc.), where I had substantial responsibilities relating to software architecture, computer networking, and technical project management. In particular, I developed and maintained the GUI for the Embedix SDK (Software Development Kit), created a network communication system between a Windows front-end and Linux back-end, developed an automated system to forward Linux python scripts to a Windows GUI over the network, and developed a network-deployed packaging and automated updating system for client software. To complete these assignments, I wrote tens of thousands of lines of computer code in C++, C, Python, and Perl.

11. As part of my technical training and development at Lineo/Metrowerks, I also gained and employed an increased exposure to computer networking and network security. I tested and evaluated a prototype firewall product, built a custom VPN solution, and trained in the use of web server and mail server administration (including security). I ran my own personal MTA with an IMAP and POP3 server based on these tools under development. I also deployed a Squid web proxy for both caching and content filtering.

12. Another major networking project involved porting the Embedix SDK from Linux to Windows. For this project I used a virtual-machine/networked solution. The underlying SDK engine remained running in a Linux VM while the GUI operated in native Windows. I developed all of the code for this remote communication system.

13. While working at Lineo/Metrowerks, I also returned to BYU to pursue my Master's degree in Computer Science. In addition to the graduate level course work in wireless computer networks and compilers, I pursued graduate research in software engineering topics, with a special

emphasis on how programmers think while creating and modifying code. During my course work, I took a special topics class called "Programmer Cognition" as well as a graduate-level neuroscience class from the Psychology department.

14. My research included a study of computer architectural patterns and how those patterns might need to change as programming languages change and evolve. Based on my research, I proposed a concept called "Design Dysphasia," wherein a programmer or software developer becomes trapped in their approach to solving problems based on the paradigms and design approaches of the programming language. My research was published as "Design dysphasia and the pattern maintenance cycle," in the Journal Information and Software Technology August 2006. This work also was a major component of my Master's thesis.

15. Another part of my Master's thesis was the identification of how certain programming language concepts can be "mixed" together. I investigated practical mechanisms whereby the Python programming language could be extended to support features known as "functional programming" and "logic (or declarative) programming." Languages with this mix of features are known as "multi-paradigm" programming languages.

16. After finishing my Master's degree, I moved to Houston TX in 2004 to begin a PhD program at Rice University. I chose to focus my research into topics of network security including the security of peer-to-peer cooperative storage and computation technologies.

17. During the 2004 fall semester of my Ph.D. program at Rice University, I identified a security vulnerability in the Google Desktop Search that could have allowed hackers to compromise users' computers and obtain private information. After contacting Google and assisting them in closing the vulnerability, we published the details of our investigation.

18. In 2005, I completed an internship at Google, where I designed and implemented a solution to privacy loss in Google Web Accelerator. The Google Web Accelerator was designed to increase the speed of browsing the Internet. Once installed on a user's computer, the browser would request all content through a Google Proxy. The proxy performed pre-fetching and extensive caching in order to provide fast and responsive service to the user. At the time of my internship, news reports had identified odd problems in which users of the Accelerator were accessing other individual's private pages. During my internship, I designed and implemented a prototype solution for this issue in C++.

19. In 2005, I published a paper entitled, "A Taxonomy of Rational Attacks." This paper categorized and described the various types of attacks that one might see in a decentralized, peer-to-peer network. When there is no centralized authority, users have to cooperate to obtain service. The term "rational attacks" refers to the economic incentives to not cooperate while still exploiting the system for service.

20. My Ph.D. Thesis was entitled "Designing Incentives for Peer-to-Peer Systems," and it built on this concept. Given a network where participants cannot be forced to cooperate, the operation of said network must induce cooperation by design of the outcomes. In other words, it must be in each participant's best interest to contribute to the cooperative operation. I conducted experiments including simulated extensions to the BitTorrent peer-to-peer protocol for long-term identities and mechanisms for cooperative anonymity. These systems were predecessors to modern cryptocurrency and distributed ledger technologies.

21. From 2005 through 2008, with the approval of my PhD adviser, I worked as a Security Analyst for Independent Security Evaluators (ISE). Much of my early work was spent developing a software encryption library, including the necessary tests and procedures for FIPS-

certification. The encryption library provided advanced operations such as secure data splitting and recovery.

22. In 2009, I went to work full time for ISE as a Security Analyst and later as a Senior Security Analyst. I built a number of advanced projects including a parallel-program for massive code coverage analysis, GPU hardware-accelerated AES encryption, and distributed file-system prototypes.

23. In addition to the software development, I also performed security evaluation services that included port-scanning analyses, security protocol analysis using formal and exploratory methods, and investigated security breaches.

24. I also designed and managed the implementation of a secure communication technology that splits trust between multiple SSL Certificate Authorities (CA), so that if one CA is compromised, the communication stream can still be safely authenticated. My work on the secure communications technology project led to the issuance of multiple patents. In total, I wrote hundreds of thousands of lines of code in C, C++, and Python, including projects where I had to implement the same functionality in two separate languages.

25. In 2011, I began work as a Research Scientist at Harbor Labs and continued with that consulting firm until fall 2015. I worked with a wide range of clients, specializing in network security, network communications, software architecture, and programming languages. I analyzed an extensive collection of commercial software, including software related to secure email, cloud-based multimedia delivery, document signing, anti-virus and anti-intrusion, high-performance routing, networking protocol stacks in mobile devices, PBX telecommunications software, VoIP, and peer-to-peer communications. I also analyzed security considerations for potential technology acquisitions.

26. Also at Harbor Labs, I reviewed technology and source code for multiple clients related to accusations of theft and/or misappropriation of trade secrets. These engagements included an analysis of C, C++, Java, Python, and other source code languages in high-frequency trading, e-commerce, and other similar systems.

27. I also assessed the security and privacy technologies and policies provided by a third-party vendor to the Center for Copyright Infringement (CCI). CCI represents content owners, such as the RIAA and the MPAA, in finding and reducing piracy online. Because this process necessarily involves collecting information about private individuals by scanning a network for illegal activity, I was asked to investigate and determine that the information collected from online computing devices was adequately safeguarded and protected.

28. For other clients, I have "resurrected" or re-created legacy software systems. For example, I helped one client make remote desktop-sharing/viewing systems from the mid 90's operational. I helped them identify the most compatible components from an old CVS repository, obtain the necessary legacy hardware and software to rebuild the source code, and diagnose why the separate components weren't completely compatible with each other. Using tools from the era (i.e., the mid-90's), I identified and fixed these issues in C++ and Java code, and successfully demonstrated the operational system across a small cluster of networked virtual machines.

29. During my final year at Harbor Labs, I was engaged as the principal consultant with a large biomedical device firm in a twelve-month analysis of the security of their products. Notably, medical devices were for some time not considered significant threats in terms of computer security. However, recent demonstrations by security researchers of the various ways in which a malicious individual might harm a person using a medical device has shifted the thinking in the industry. Accordingly, I was engaged to assist this company in the analysis of their products,

their process, and their future roadmap in order to ensure that patients are not harmed. I and my team analyzed design documents, hardware, and a broad range of additional resources in order to expose potential problems. The security of these systems depends, in part, on the architecture and deployment of the networks in which they operate.

30. In December 2015, I left Harbor Labs to assist Ironwood Experts, LLC., as the transitional managing partner. In three months, I helped to establish a new direction, streamline operations, and wrap up difficult negotiations.

31. After handing off management responsibilities at Ironwood, I founded Crimson Vista, Inc. as a boutique computer security engineering company. Similar to the work that I did at Harbor Labs, I continue to provide technical expertise to a wide range of clients in areas of programming languages, computer networks, and network security. My expertise in the area of "security engineering" provides comprehensive analysis, design, and insight into cybersecurity concerns before, during, and after development.

32. For example, I have been retained by a start-up in telecommunications security to provide cryptography expertise and evaluations of their protocols and architectures. My team and I have prototyped new protocols, written up analyses, and presented to potential partners and investors.

33. Another start-up company retained me for guidance in matters relating to Blockchain and Smart Contracts. This technology is very much dealing with a "fad" phase where there is a lot of misinformation and hype. I guided the start-up company through analyzing where these kinds of technologies would help and where they would not. I have also provided training on Blockchain at the Data Architecture Summit and Enterprise Data World conferences.

34. I have also provided technical guidance to an antitrust team in the United States Department of Justice. Although the technologies and parties are confidential, I can disclose that I provided in-person training on technical topics and analyses of competing security products.

35. More recently, I been retained by clients, including a Fortune 100 financial institution, to provide them with post-data-breach analyses of what went wrong, the impact of the lost data, and guidance on resolution. In these engagements, I provided reverse engineering of the data to demonstrate how an attacker can or would use the compromised information, analyzed software development to determine when the system became vulnerable, and helped identify impacted customers that had been missed in the investigations.

36. I have also been retained as a vCISO (virtual Chief Information Security Officer) for a small company that needs security expertise and guidance in protecting their newly developed intellectual property. I advise the company on security policy, operations, implementations, and training.

37. Through Crimson Vista, I also invest in research and development. Recent projects include engaging with a partner to implement prototypes of communications security protocols for next-generation automobiles. I also gave a talk on "Detecting Malicious Sandboxes" at the Workshop on Defensive Deception and Trust in Autonomy, in association with the 2018 Naval Applications of Machine Learning Workshop.

38. I was the primary investigator on a government funded research project. The project dealt with automatic recovery from ransomware attacks and was funded by the United States Army. I led a team of student researchers from Johns Hopkins and Brigham Young University in developing my design and implementing a prototype.

39. I am also a contributor to another small-business research project begun by a former student and funded by the National Science Foundation. We are researching the development of a new system for secure data storage that is protected even against "insider" threats, such as a rogue IT administrator. As part of this work, I am managing several interns for Summer 2022 and directing their work on performance, analysis, and vulnerability detection.

40. I also continue to perform a wide range of code reviews for diverse technologies including CAD software, video game systems, digital mobile radios (DMRs), video streaming, and digital rights management (DRM). I am often retained for my expertise in software forensics and have been instrumental in multiple cases to discovering and identifying stolen source code or misappropriated trade secrets.

41. Moreover, I maintain ties to academia. I held adjunct appointments at Johns Hopkins University from 2014 to 2019. From July 2016 to July 2019, I also held an appointment as the Director of Advanced Research Projects in the Johns Hopkins University Information Security Institute.

42. At Johns Hopkins University I taught Network security and Advanced Network Security. I created a custom curriculum and lab experience wherein students developed their own security protocols as a class and then attempted to break their own creations. Students learned how hard it is to get security right, and how easy it is to find something wrong. I published a paper about the labwork in the Journal of Computer Science Education entitled, "PLAYGROUND: Preparing Students for the Cyber Battleground."

43. Beyond course instruction, I also mentored Masters students at Johns Hopkins in their capstone projects. These projects included networking security and privacy concerns across a wide range of technologies including cryptography, drone security, iOS security, BitCoin, SSL

vulnerabilities, and Twitter "botnets." My students and I have published multiple papers from these capstone projects.

44. During my tenure as the Director of Advanced Research Projects, I was tasked with developing collaborative research opportunities. Through my efforts, a wide range of student capstones have been executed with partners from the Johns Hopkins Applied Physics Lab or outside corporate partners.

45. For example, we coordinated with the company OnBoard Security to develop better security for anti-collision protocols for air traffic. Students demonstrated the potential issues related to leaving the protocol unsecured and built a working prototype of a secured variant. OnBoard and Johns Hopkins published press releases which were picked up by aviation-focused news sources.

46. I am now an adjunct professor at the University of Texas at Austin. I have taught the undergraduate Network Security and Privacy class in the Computer Science department. I also teach the Introduction to Cybersecurity Technology class in the Law School

47. I am also the co-founder and current director of the Crypto Done Right project. This project is currently hosted by Johns Hopkins University and funded by a grant from Cisco. I am transitioning the project to be hosted by my company Crimson Vista or spun out into a completely independent non-profit entity. Crypto Done Right is designed to bridge the gap between cryptography SME's and the IT professionals that use it. It provides authoritative guidance on deployment, lifecycle, and management of cryptography in IT systems, software development, and technical management.

48. Finally, I am the author of "Cryptography in Python: Learning Correct Cryptography by Example." I am authoring a second book on computer security that will be published later this year.

III. MATERIALS CONSIDERED

49. I have considered information from various sources in forming my opinions. Besides drawing from approximately two decades of experience in the computer industry, I also have reviewed the '517 patent and its file history, the other documents and references as cited herein, and general technical references.

50. I understand that Counsel for Petitioner have submitted a Petition that references by exhibit number. For simplicity and convenience, I have adopted the same exhibit numbering. The exhibits are:

- Ex. 1001: U.S. Patent No. 8,850,517 to Kumar, et al. ('517 patent)
- Ex. 1002: Prosecution history of '517 patent
- Ex. 1003: Expert Declaration of Seth James Nielson, Ph.D.
- Ex. 1004: Curriculum Vitae of Seth James Nielson, Ph.D.
- Ex. 1005: U.S. Patent Publication No. 2011/0023115 A1 to Wright ("Wright")
- Ex. 1006: U.S. Patent No. 8,806,647 to Daswani, et al. ("Daswani")
- Ex. 1007: Andrew S. Tanenbaum, Modern Operating Systems 3, Pearson Prentice Hall, (2008).
- Ex. 1008: Brian Clifton, Advanced Web Metrics with Google Analytics 3, John Wiley & Sons, (2012).

IV. LEGAL STANDARDS

51. I am not an attorney, and I have relied on instructions from counsel as to the applicable legal standards to use in arriving at my opinions in this Declaration.

52. I have been informed and understand that patent claims are construed from the perspective of one of ordinary skill in the art at the time the claimed invention was made and that, during this proceeding, claims are construed using the same claim construction standard that would be used to construe the claim in a civil action. Under that standard, claim terms are to be given their plain and ordinary meaning, which is the meaning understood by a POSITA in light of the claim language, patent specification, and prosecution history.

53. I understand that a patent may include independent claims and dependent claims. An independent claim stands by itself and only includes the limitations it recites. A dependent claim can depend on (a) an independent claim, or (2) another dependent claim. I understand that a dependent claim includes all of the limitations it recites plus the limitations recited in the claim from which it depends.

54. I have been informed and understand that the subject matter of a patent claim is obvious if the differences between the subject matter of the claim and the prior art are such that the subject matter of the claim as a whole would have been obvious at the effective filing date to a person having ordinary skill in the art to which the subject matter pertains. I have also been informed that the framework for determining obviousness involves considering the following factors: (i) the scope and content of the prior art; (ii) the differences between the prior art and the claimed subject matter; (iii) the level of ordinary skill in the art; and (iv) any objective evidence of non-obviousness.

55. I further understand that the claimed subject matter would have been obvious to one of ordinary skill in the art if, for example, it results from the combination of known elements

according to known methods to yield predictable results, the simple substitution of one known element for another to obtain predictable results, the use of a known technique to improve similar devices in the same way, the application of a known technique to a known device ready for improvement to yield predictable results, or choosing from a finite number of identified, predictable solutions with a reasonable expectation of success. I have also been informed that the analysis of obviousness may include recourse to logic, judgment, and common sense available to the person of ordinary skill in the art that does not necessarily require explication in any reference. I understand that so-called "secondary considerations of non-obviousness," must be considered in an obviousness analysis. I understand that an analysis cannot use hindsight in determining whether a patent claim is obvious. I understand that secondary considerations of non-obviousness include: (a) a long-felt but unresolved need for the invention; (b) commercial success of the invention; (c) copying of the invention; (d) praise and recognition of the invention by others; (e) licensing of the rights to the invention; and (f) unexpected results.

56. I understand that obviousness must be determined with respect to the challenged claim as a whole.

57. I understand that a challenged claim can be invalid for obviousness over a combination of prior art references if a reason existed (at the effective filing date of the Patent in question) that would have prompted a person of ordinary skill in the art to combine prior art elements in the way required by the claim in question. I understand that this requirement is sometimes called "motivation to combine." I further understand that the prior art references themselves might provide a motivation to combine, and that sometimes commonsense as would have been known to a POSITA can provide a reason to combine the disclosures of two or more prior art references. I also understand that market demand, rather than scientific literature, often

drives innovation, so that market forces can give rise to a motivation to combine. I also understand that known work in a field might prompt variations of it to use in the same field or a different one based on design incentives or market forces if the variations would have been predictable to a POSITA. I also understand that a need or problem known in the field can provide a reason for combining elements in the manner claimed.

58. I understand that in order for a claim to be found invalid based upon a modification or combination of prior art references, a POSITA must have had a reasonable expectation of success in modifying or combining the prior art.

59. I further understand that claims are invalid as obvious if they merely recite combinations of, or obvious variations of, prior art teachings employing the understanding and knowledge of a POSITA and motivated by the general problem that the inventor faced at the time. The prior art references themselves, or any need or problem known in the field at the time of the invention can provide a motivation to combine the elements from prior art references or to attempt obvious variations on them in the claimed manner.

60. I understand that something is "inherent in" the prior art—and thus taught by it—it if necessarily follows from the prior art's explicit disclosures.

61. I understand that the filing date of a U.S. patent is generally the filing date of the U.S. application that issued as a patent. I also understand, though, that a U.S. patent may be granted an earlier effective filing date if the patent owner properly claimed priority to an earlier patent application.

V. TECHNICAL BACKGROUND THAT WOULD HAVE BEEN KNOWN TO A POSITA

62. In this section of my Declaration, I discuss some of the background and context for the alleged invention in the '517 patent that would have been known to a POSITA.

A. System boot and login processes

63. Starting up a computer system involves many steps, some of which are performed in kernel space and some of which are performed in user space. What follows is a general overview of the boot and login processes of computer systems at the time of the alleged invention.

64. In Linux systems, the computer's BIOS reads the disk, locates the Master Boot Record, and causes the bootloader to locate the operating system kernel. "Then boot reads in the operating system kernel and jumps to it. At this point, it has finished its job and the kernel is running." (Ex. 1007 at p. 5). Next, the kernel identifies various hardware characteristics of the machine, like CPU type and RAM size, and allocates the memory and data structures necessary to operate. (*Id.* at pp. 5-6). Following that, the kernel will identify connected input/output devices and loads their associated device drivers. (*Id.* at p. 6).

65. At this point, the kernel is ready to start creating processes. It starts with "process 0", which "continues initialization, doing things like programming the real-time clock, mounting the root file system, and creating init (process 1) and the page daemon (process 2)." (*Id.* at p. 6). Init is now responsible for launching the user interface of the operating system. In multi-user mode, it first "forks off a process that executes the system initialization shell script, /etc/rc, which can do file system consistency checks, mount additional file systems, start daemon processes, and so on. Then it reads /etc/ttys, which lists the terminals and some of their properties. For each enabled terminal, it forks off a copy of itself, which does some housekeeping and then executess [sic] a program called getty." (*Id.* at p. 6).

66. Now the system is almost ready to have the user log in. After some additional configuration, getty prints "login:" to the terminal and waits for someone at the keyboard to type in a username. If that occurs, "getty terminates by executing /bin/login, the login program. Login then asks for a password, encrypts it, and verifies it against the encrypted password stored in the

password file, /etc/passwd. If it is correct, login replaces itself with the user's shell, which then waits for the first command." (*Id*. at p. 7).

67. The following figure illustrates an exemplary process tree of a pending login ("Terminal 0"), pending authentication ("Terminal 1"), and successful login with subsequent user program execution ("Terminal 2"):

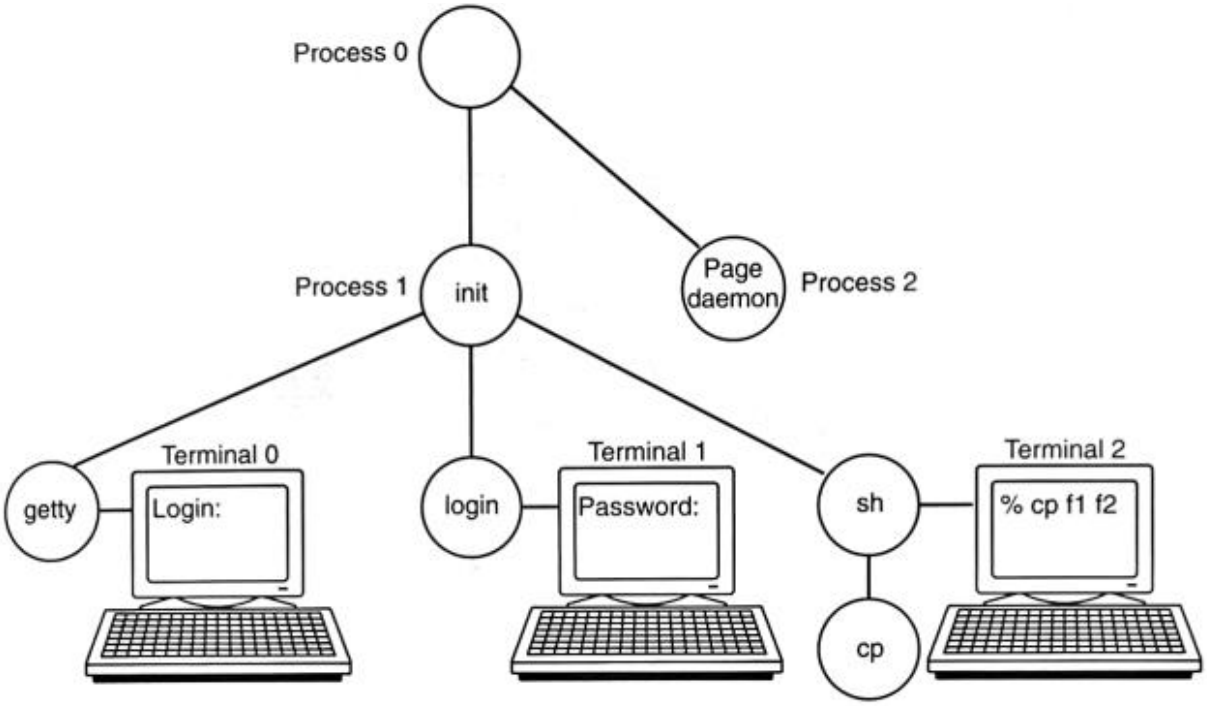


Figure 10-11. The sequence of processes used to boot some Linux systems.

(*Id.* at Figure 10-11, p. 7).

68. Thus, the process of system startup on Linux is a long sequence of operations that include processes creating other processes (e.g., by "forking"). Although a POSITA might refer to system startup, or boot, as a single concept, they would nonetheless understand it to consist of many individual steps, some of which executed by the bootloader, some by the operating system kernel, and some in user space.

69. Microsoft Windows has a different process for system startup, but it shares many of the same principles. The process similarly begins with the BIOS, which is code available "in some form of nonvolatile CMOS memory that is initialized by the computer manufacturer". (*Id.* at p. 11). Windows Vista in particular works this way: "The BIOS brings up Windows Vista by first loading small bootstrap programs found at the beginning of the disk drive partitions. The bootstrap programs know how to read enough information off a file system volume to find the standalone Windows BootMgr program in the root directory." (*Id.*). BootMgr then loads either WinResume.exe (if the system was hibernated or in standby) or WinLoad.exe if performing a fresh boot. "WinLoad loads the boot components of the system into memory: the kernel/executive (normally ntoskrnl.exe), the HAL (hall.dll), the file containing the SYSTEM hive, the Win32k.sys driver containing the kernel-mode parts of the Win32 subsystem, as well as images of any other drivers that are listed in the SYSTEM hive as boot drivers—meaning they are needed when the system first boots." (*Id.*). Once the lower-level components have been successfully initialized, "the first user-mode process is created using for running the smss.exe program (which is like /etc/init in UNIX systems)." (*Id.*). Only after this point can a user interface be launched for the user to interact with the computer.

B. System files, registry, and system calls

70. In Windows, much critical system information is maintained in the registry. This includes information about boot drivers, without which the system is unable to talk to the hardware devices necessary to start up the machine:

> The root of the NT namespace is maintained in the kernel. Storage, such as file system volumes, is attached to the NT namespace. Since the NT namespace is constructed afresh every time the system boots, how does the system know about any specific details of the system configuration? The answer is that ***Windows attaches a special kind of file system (optimized for small files) to the NT namespace. This file system is called the registry***. The registry is organized into separate volumes called hives. ***Each hive is***

> ***kept in a separate file (in the directory C:\Windows\system32\config\ of the boot volume)***. When a Windows system boots, one particular hive named SYSTEM is loaded into memory by the same boot program that loads the kernel and other boot files, such as boot drivers, from the boot volume.
>
> Windows keeps a great deal of crucial information in the SYSTEM hive, including information about what drivers to use with what devices, what software to run initially, and many parameters governing the operation of the system. This information is used even by the bot program itself to determine which drivers are boot drivers, being needed immediately upon boot. Such drivers include those that understand the file system and disk drivers for the volume containing the operating system itself.
> (Ex. 1007 at pp. 8-9).

71. A POSITA would have recognized the registry to comprise system files, in that they are crucial to the functioning of the system, are maintained in a particular place in memory unique to the operating system, and must be meticulously managed to maintain their integrity. For example,

> When the system is turned off, most of the registry information is stored on the disk in the hives. ***Because their integrity is so critical to correct system functioning, backups are made automatically and metadata writes are flushed to disk to prevent corruption in the event of a system crash.*** Loss of the registry requires reinstalling all software on the system.
> (*Id.* at p. 10).

72. A POSITA would have understood from this that changes made to the registry (e.g., "metadata writes") are persisted to disk automatically by the system.

73. Besides being a repository for critical system configuration, the registry also allows user programs to store information in its hives. As such, manipulation of the registry is made available to user programs via Windows API calls. This table outlines some of the common operations:

Win32 API function	Description
RegCreateKeyEx	Create a new registry key
RegDeleteKey	Delete a registry key
RegOpenKeyEx	Open a key to get a handle to it
RegEnumKeyEx	Enumerate the subkeys subordinate to the key of the handle
RegQueryValueEx	Look up the data for a value within a key

Figure 11-12. Some of the Win32 API calls for using the registry

(*Id.* at Figure 11-12, p. 10).

74. These Windows API calls allow user programs to access protected Windows system resources (including but not limited to the registry) in a controlled manner. Windows does not directly expose such kernel-level operations (also known as "system calls") to user programs, but instead makes them available via user-level Windows API calls. "To start with, the library calls and the actual system calls are highly decoupled. Microsoft has defined a set of procedures called the Win32 API (Application Program Interface) that programmers are expected to use to get operating system services." (*Id.* at p. 3).

75. Windows additionally has a wide array of processes that invoke system-level actions, such as managing user authentication, installing drivers, or managing network activity:

> Windows Vista makes significant use of user-mode service processes to extend the functionality of the system. Some of these services are strongly tied to the operation of kernel-mode components, such as lsass.exe which is the local security authentication service which manages the token objects that represent user-identity, as well as managing encryption keys used by the file system. The user-mode plug-and-play manager is responsible for determining the correct driver to use when a new hardware device is encountered, installing it, and telling the kernel to load it. Many facilities provided by third parties, such as antivirus and digital rights management, are implemented as a combination of kernel-mode drivers and user-mode services.
> (*Id.* at p. 12).

> Devices in Windows are represented by device objects. Device objects are also used to represent hardware, such as buses, as well as software abstractions like file systems, network protocol engines, and kernel extensions, like antivirus filter drivers.
> (*Id.* at pp. 13-14).

76. Linux has a similar technique of exposing system calls to user programs (e.g., via the "standard library") to safely provide the necessary facilities while controlling access to hardware devices and critical system components. "Running on the bare hardware is the operating system. Its function is to control the hardware and provide a system call interface to all the programs. These system calls allow user programs to create and manage processes, files, and other resources." (*Id.* at p. 4). This figure illustrates the role of the "standard library" in interfacing between user programs and the kernel, which in turn functions atop the hardware:

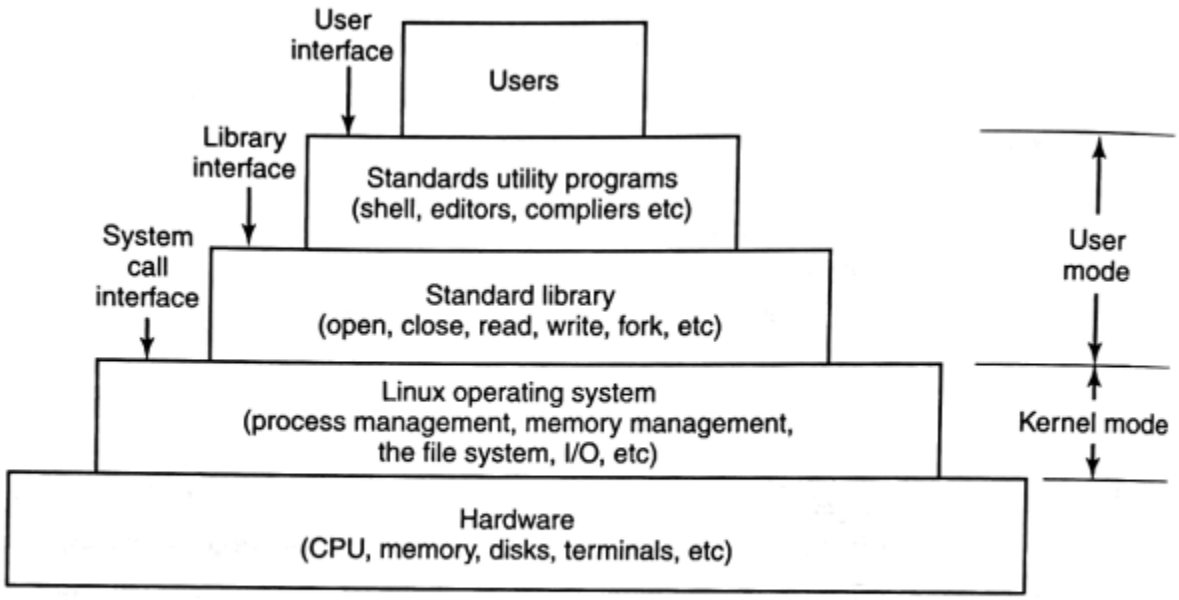


Figure 10-1. The layers in a Linux system.

(*Id.* at Figure 10-1, p. 4).

C. User metrics

77. Website creators and owners commonly collect engagement metrics to measure the effectiveness of their content, marketing, or advertising endeavors. One such metric, used widely at the time of the alleged invention and supported in widely-available analytics tools such as Google Analytics, is called "time on site".

> The average time on site is ***the length of time visitors spend interacting with your website***, and it is a good base metric to help you understand whether your visitors are engaging with your site. All content creators want to increase this KPI—assuming, of course, the visitor experience is a good one.
>
> The calculation is straightforward, though it is worth mentioning how it is determined. To calculate the time on site, Google Analytics uses ***the difference in time between the last and first pageview a visitor requests (or event if you are also tracking these).***
> (Ex. 1008 at p. 5).

78. For "time on site" in particular, the goal is quantifying how engaging the website content is by calculating a temporal measure (i.e., based on the difference between two timestamps).

79. Other metrics can be calculated based on user actions (defined, for example, as "any non-pageview action that can be tracked by Google Analytics—such as an event (file download, Flash movie interaction, and so on) or e-commerce transaction." (*Id.* at p. 4)). These events may each have an associated timestamp that can be correlated as desired.

VI. OVERVIEW OF THE CHALLENGED PATENT

A. THE '517 PATENT SPECIFICATION AND CLAIMS

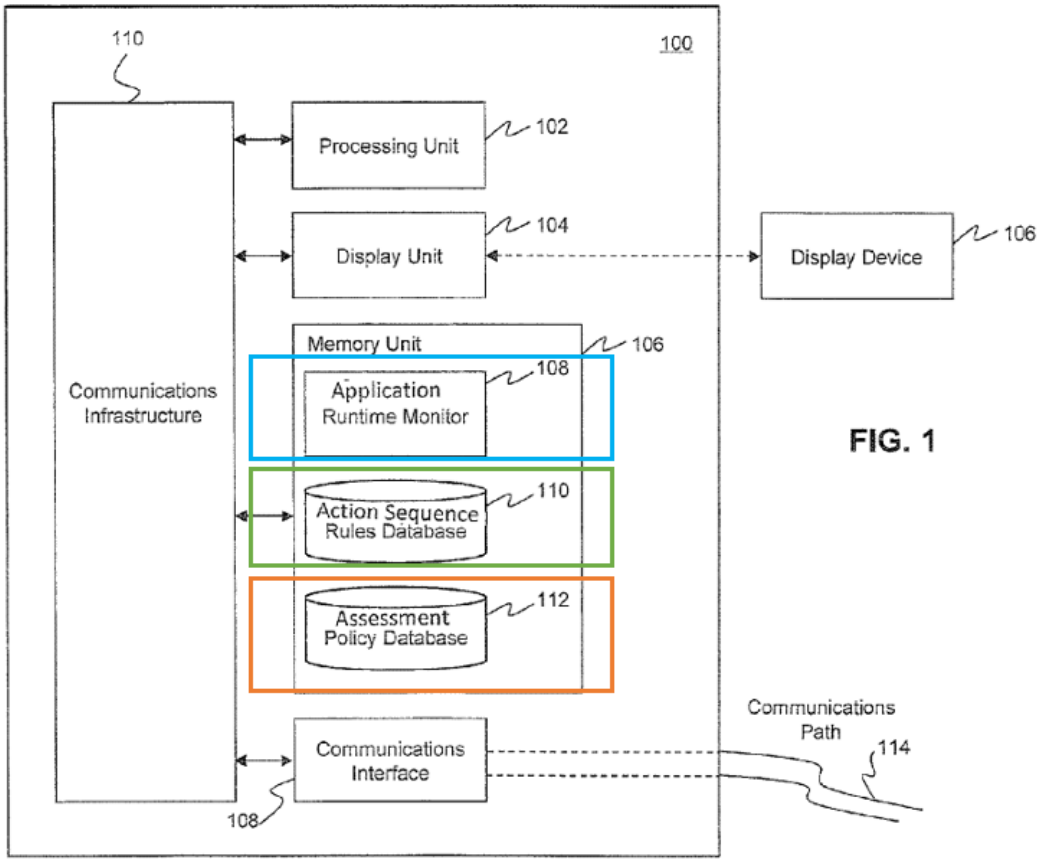
80. U.S. Patent No. 8,850,517 to Srinivas Kumar, titled "Runtime risk detection based on user, application, and system action sequence correlation" issued on September 30, 2014. The '517 patent is directed to detecting risks to computer security, including advanced persistent threats, through the use of behavioral analysis techniques. (*See* Ex. 1001 at 1:7-39).

81. To do this, the ’517 patent discloses “a system and method for the assessing of a runtime risk of an application or device.” (*Id.* at 1:43-45). The first building block of this technique is an action, which may be of one of three types: user action, application action, and system action:

> The user action 302 may be any form of explicit input from a user of the computing system 100. The application action 304 may be any activity performed by the application initiated programmatically by a task in execution of the computing system 100. The system action 306 may be any operation performed by the computing system 100 on behalf or, or as a consequence of, a user action 302 or application action 304 that changes the state of the computing system 100.
> (*Id.* at 9:17-24).

82. The alleged invention uses these actions to document rules, “wherein each rule identifies an action sequence” of two or more actions. (*Id.* at 9:67, 10:13-14). These rules are stored in a “rules database” and used as a component of “assessment policies” (stored in a “policy database”). (*Id.* at 9:66-10:3). Using those assessment policies, the ’517 patent discloses “identifying, using at least one assessment policy, a runtime risk for an application program that executes on a device, wherein the identified runtime risk indicates a risk or threat of the identified action sequence of the application.” (*Id.* at 10:4-8). In addition, a “runtime monitor” produces a “behavior score for the application” based on the identified risk. (*Id.* at 10:9-12). Finally, the patent discloses various ways these assessments may be made, such as action sequences which “describe a forensic chain of events” that may include “ordered sequences, unordered sequences, proximity analysis, outliers, probabilistic weights and scores, and probable action inferences based on an application’s competency.” (*Id.* at 10:65-11:5).

83. These components are illustrated in Figure 1 of the ’517 patent. Annotated hereinbelow are the “runtime monitor”, the “rules database”, and the “policy database”.



(*Id.* at Figure 1, annotated).

B. PROSECUTION HISTORY

84. Prosecution began with a non-final Office Action on May 7, 2013, that rejected claims 1-5, 10-17, and 22-24 under 35 U.S.C. §102(e) as anticipated by U.S. Pub. No. 2013/0096980 to Basavapatna, and rejected claims 6-9 and 18-21 under 35 U.S.C. §103(a) as unpatentable in view of Basavapatna.[1] (Ex. 1002 at pp. 164-170). Applicant responded by amending independent claims 1 and 13 to recite, in part "application program that executes on a device." (*Id.* at pp. 135-141).

[1] The Office Actions referred to "Basavapatna-McAfee" but the §103 rejections are based solely on Basavapatna.

85. A final Office Action dated August 15, 2013, reasserted the Examiner's prior position, rejecting claims 1-5, 10-17, and 22-24 under 35 U.S.C. §102(e) as anticipated by Basavapatna, and rejecting claims 6-9 and 18-21 under 35 U.S.C. §103(a) as being unpatentable in view of Basavapatna. (*Id.* at pp. 117-122). Applicant responded by amending claims 1 and 13 to recite "wherein the identified runtime risk <u>indicates a risk or threat of the identified action sequence of the application</u>", and "a <u>runtime monitor including a</u> processing device". (*Id.* at pp. 93-99).

86. In an Applicant-initiated Examiner interview conducted November 8, 2013, Applicant argued that amendments to claims 1 and 13 distinguished over Basavapatna. (*Id.* at p. 89). Thereafter, a Notice of Allowance issued on February 12, 2014, with no Examiner's reasons for allowance. (*Id.* at pp. 60-64).

VII. LEVEL OF SKILL IN THE ART

87. Based on my knowledge and experience, I understand that a person of ordinary skill in the art ("POSITA") at the time of the earliest claimed priority date for the '517 patent would have had a bachelor's degree in computer science, or a related field, or an equivalent technical degree or equivalent work experience, and an additional one to two years of education or experience in the field of computer security. More education can supplement practical experience and vice versa.

88. I meet that definition of a person of ordinary skill in the art, and I did so on, in, and before January 2013. I also have worked with persons of ordinary skill in the art as part of my professional and academic experience, and I have an understanding of their skill level in and before January 2013.

VIII. OVERVIEW OF THE PRIOR ART

A. OVERVIEW OF US20110023115A1 PUBLISHED APPLICATION TO WRIGHT ("WRIGHT")

89. U.S. Patent Publication No. 2011/0023115 A1 to Clifford C. Wright ("Wright", Ex. 1005) is entitled "Host intrusion prevention system using software and user behavior analysis." Wright discloses a method to detect malicious system behavior and perform an appropriate remedial action if needed. Wright proposes using a behavioral based method to identify the runtime legitimacy of an application for the purpose of malware prevention. The behavioral identification technique subverts anti-hash malware mutations and obfuscation techniques on which many other tools rely by comparing common behaviors and tendencies of malware rather than direct comparisons of code or hashed values.

90. Wright is in the field of computer science. However, it uses terms that originate in the life sciences, such as "gene" and "phenotype," to describe behaviors of users and code, and to describe sequences or collections of such behaviors.

91. Wright models a computing phenotype as a combination of user behaviors and code behaviors, all three of which are stored in their own databases, shown in the figure below.

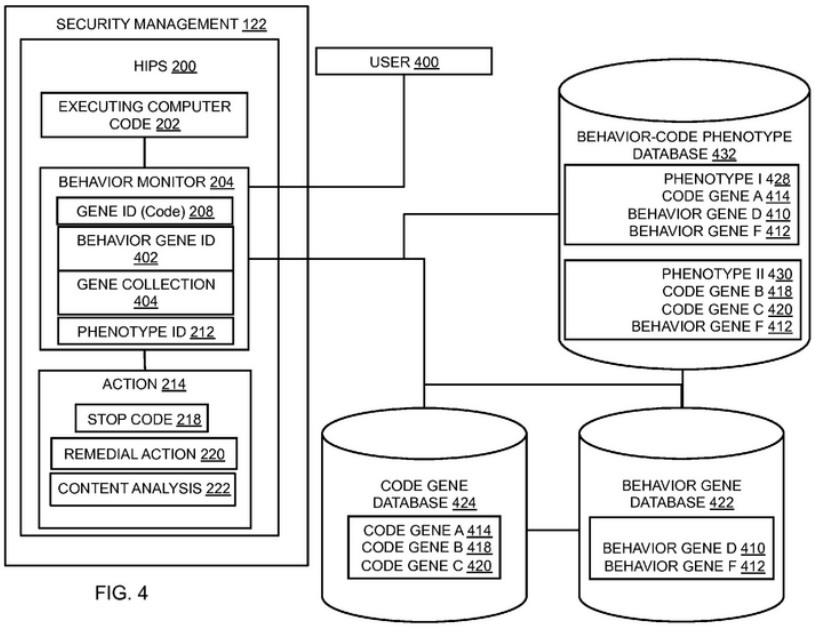


FIG. 4

(Ex. 1005 at Figure 4).

92. Wright refers to these as behavioral genes (referring to user behavior) and code genes (recognized code patterns). In addition to the above Figure 4 labeling, Wright expressly states that each of these are stored in a database:

> a behavioral gene may ***be stored in a database*** that is associated with the computer network in which the data relating to the user behavior is recorded and stored.
> (Ex. 1005 at [0007], emphasis added).

> monitoring a computer code process executing during the usage session for an indication of a code operation, wherein the indication of the code operation is a result of comparing an operation with a predetermined code behavior, referred to as a code gene, where the code gene is stored for reference **in a database**.
> (Ex. 1005 at Claim 1.B, emphasis added).

> Matching this runtime genotype data with known combinations **stored in the phenotype database** 224, with or without an additional content analysis 222, may enable the identification and interruption of malware while it is executing.
> (Ex. 1005 at [0066], emphasis added).

93. A behavioral gene is a recognized pattern of a user interaction, which is any action performed by a user. A behavioral gene may include clicking on a link or opening a file, or any explicit input caused by a user. Each behavioral gene may include one or more of these actions.

> For example, a behavioral gene called "Internet download" may be created and stored in association with the computer network. This behavioral gene ***may summarize the user behaviors*** of 1) executing a hyperlink on the Internet, and 2) performing a "save as action in which a file is saved to a location in the computer network, such as a hard drive. A single user behavior (e.g., closing an application) may be recorded and stored as a behavioral gene. ***Alternatively, a plurality of user behaviors may be recorded within a single behavioral gene***.
> (Ex. 1005 at [0007], emphasis added).

94. A code gene is a pattern of code processing that is recognized. Wright describes this code gene recognition as "the result of comparing an operation with a predetermined code behavior… For example, a code gene may be a virus definition, a malware definition, an adware definition, or some other code behavior." (Ex. 1005 at [0008]).

95. The behavioral genes and code genes are combined to produce a "behavior-code phenotype", which can include any number of behavioral genes and code genes that can occur at different times. (Ex. 1005 at [0008]). These phenotypes model the recognized actions that are performed by malicious code. (*See* Ex. 1005 at [0014]). Furthermore, different phenotypes may contain similar behavior-code gene portions. A single behavior gene can be used in multiple behavior-code gene combinations:

> ***A phenotype may comprise one behavioral gene and a plurality of code genes***, a plurality of behavioral genes and one code gene, or some other combination of behavioral genes and code genes. In some embodiments, a phenotype may comprise a combination of one or more behavioral genes, one or more code genes, ***including embodiments in which there is a temporal disparity between the identification of the behavioral gene and the identification of the code gene***.
> (Ex. 1005 at [0008], emphasis added).

96. Based on the phenotype, action or remedial action may be initiated to neutralize the threat or to bring the system back to a safer state:

> Based on the results of the behavior monitor facility 204, if the executing code is deemed to be suspicious or malicious, ***an action 214 may be caused to occur based on a prediction that the executing code is the type of malicious code as indicated by the phenotype***. For example, a user input may be ignored, a user input may be overridden, code execution may be halted 218, a remedial action 220 may be taken, a content analysis 222 may be performed on the executing code or on a file produced by the executing code, the code execution may be paused while content analysis occurs, and the like. A remedial action 220 may be specific to the kind of malicious code, such as the patching of specific exploits that are indicated by the chain of behavior observed. A remedial action 220 may be specific to the user behavior associated with the malicious code. For example, the remedial action for a user emailing a file infected with a virus may include prompting the user to clean the infected file.
> (Ex. 1005 at [0091], emphasis added).

B. OVERVIEW OF US 8,806,647 TO DASWANI ("DASWANI")

97. U.S. Patent No. 8,806,647 by Neilkumar Daswani, et al. ("Daswani") is entitled "Behavioral scanning of mobile applications." Daswani teaches a method wherein behavioral analysis of an application is performed to determine whether the application is malicious. If the application accepts user input, different user actions are simulated to determine possible behaviors that an application can express. Both static and behavioral analysis is performed in order to determine the likelihood an application contains malware.

98. Static analysis is performed on the application primarily to enable behavioral analysis. All events, including user triggered event inputs triggered through "user action or system hardware" (Ex. 1006 at 5:37-38), may be logged in order to place the application into various states through simulated actions during behavioral analysis. Each of the states contains a

99. Behavioral analysis is performed on the application to use the above events and states to map the corresponding behaviors to one of three danger levels: an unsuspicious behavior, a soft signal indication of malicious behavior, or a hard signal indication of malicious behavior.

Soft signals are those which "have an associated probability of the application containing malware" (*Id.* at 6:50-51), whereas hard signals are those which "are by definition malicious… or associated with a probability of the application containing malware, the probability exceeding a hard signal threshold" (*Id.* at 6:52-55).

100. After the behaviors that correspond with the events and states analyzed during the static analysis phase have completed, the behaviors are probabilistically combined in order to make a determination regarding the overall likelihood that the application is malicious. Daswani teaches that a hard signal is a strong indicator that the determination for the application will be malicious, but more than one soft signal may be required to make a determination of maliciousness if no hard signal is found.

101. After analysis has been performed, an action is taken based on the results of the analysis. The notification may inform the application distributor, developers, or consumers with threat information.

IX. GROUND 1: CLAIMS 1-24 ARE RENDERED OBVIOUS BY WRIGHT IN VIEW OF DASWANI

102. Claims 1-24 of the '517 patent are rendered obvious to a POSITA by Wright in view of Daswani, as explained in the sections below. I also explain that it would have been obvious for a POSITA to combine Wright and Daswani.

A. Motivation to Combine Wright and Daswani

103. A POSITA would have been motivated to combine Wright and Daswani. Both Wright and Daswani are directed toward identifying unknown malware and adversarial operations using behavioral analysis.

104. For example, Wright discloses: "In an embodiment, the security management facility 122 may provide for host intrusion prevention through behavioral based protection, which

may guard against unknown threats by analyzing behavior before software code executes. Behavioral based protection may monitor code when it runs and intervene if the code is deemed to be suspicious or malicious." (Ex. 1005 at [0031]). Notably, Wright identifies new threats but then creates definitions that can be used subsequently. "As threats are identified and characterized, the threat management facility 100 may create definition updates that may be used to allow the threat management facility 100 to detect and remediate the latest malicious software, unwanted applications, configuration and policy changes, and the like." (Ex. 1005 at [0041]).

105. Daswani also discloses dealing with previously unseen malware. Daswani explains, "[m]alware detection is commonly accomplished by looking for certain signatures within binaries. However, this method of detection can be negated by a malware generator generating variants of a piece of malware. These variants can be scanned with known antivirus engines to ensure that no malware signature match is made." (Ex. 1006 at 1:18-23). To solve this problem, Daswani, like Wright, proposes behavioral detection using mechanisms like emulation to determine the software's operations. "In one example embodiment, the behavioral detection is performed on a software emulation platform. In this configuration, the system is structured to customize the emulated environment, record the observed output, analyze the practical significance of the observed output, and optionally provide suggested course of action, e.g., avoid download, articulate impact on mobile device, etc." (Ex. 1006 at 2:29-36).

106. A POSITA would have been motivated to combine Daswani with Wright because, as I explained in Section VIII.A., Wright uses ranking of phenotypes based on ***confidence*** to identify the unknown software as belonging to a certain family of malware. As Wright explains:

> **The behavior monitoring facility 204 may be able to identify a phenotype 212** of user and code behaviors in relation to an executing code by comparing a collection of observed user and code behaviors with the predetermined collections of known malicious user and code **behaviors**

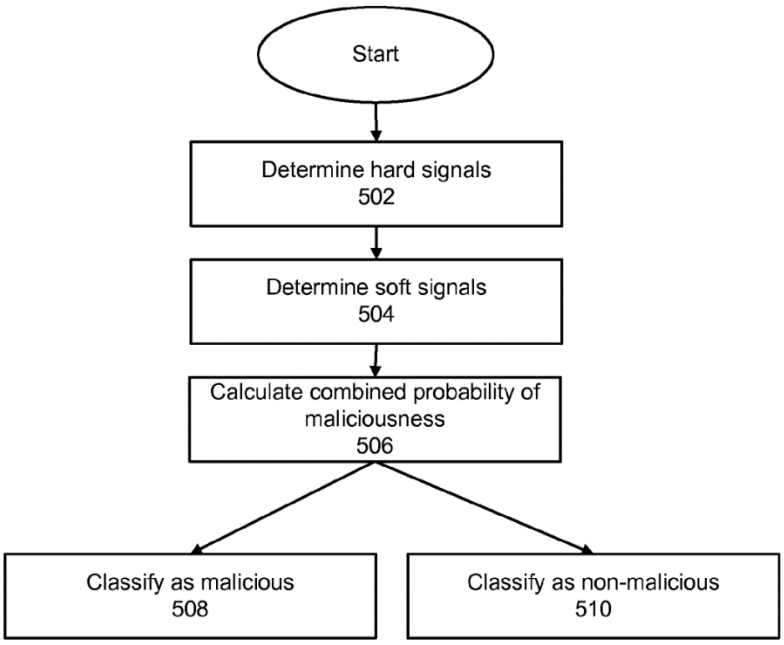
> **stored as phenotypes** in a behavior-code phenotype database 432, either temporarily or for a longer term. **Behavior-code phenotypes may capture a combination or *a series of user and code behaviors that may be ranked to create increasing levels of confidence that the runtime object being monitored is executing a behavior pattern comparable to a known family of malware***. The predetermined collections of behaviors, or phenotypes, may be created based on ***suspicious or risky user behavior*** (e.g. downloading a file from a website, running an executable, opening a signed file with an unverifiable digital signature, or other potentially suspicious or risky behavior) and scaled analysis of families of malware in runtime, such as in a threat research facility 132 of a threat management facility 100.
> (Ex. 1005 at [0089], emphasis added).

107. A POSITA would have been motivated to accurately compute these confidence metrics in order to effectively rank the phenotypes in order to correctly identify the malware.

108. A POSITA would have been motivated to use Daswani to create these confidence levels. Daswani discloses hard and soft signals that represent indications of how likely a system call or action is to contain malware, a significant indicator of risk. According to Daswani, hard signals are behaviors that are almost certainly malicious, while soft signals are only malicious in context. Daswani explains:

> Behavioral analysis of a mobile application is performed to determine whether the application is malicious. During analysis, various user interactions are simulated in an emulated environment to activate many possible resulting behaviors of an application. The behaviors are classified as hard or soft signals.
> (Ex. 1006 at Abstract).
>
> In one embodiment, some system calls or actions simulated in the emulated environment may be classified as soft signals or hard signals. In one embodiment, soft signals comprise system calls or actions which have an associated probability of the application containing malware. Hard signals comprise system calls or actions that are by definition malicious actions, or associated with a probability of the application containing malware, the probability exceeding a hard signal threshold.
> (Ex. 1006 at 6:47-55).

109. Daswani uses both hard and soft signals as inputs to a probabilistic signal combination module that computes the likelihood that an application is malicious. This process is shown in the following figure:



(Ex. 1006 at Figure 5).

> Signal combination module 306 combines multiple behaviors signals to determine an overall likelihood that an application is malicious.
> (Ex. 1006 at 6:45-47).

110. A POSITA would have recognized that Daswani is compatible with Wright. Specifically, the probabilistic computations disclosed by Daswani are applicable to the phenotypes disclosed in Wright. As I explained in Section VIII.A., Wright discloses a "phenotype" which identifies potential "malicious usage" based on at least one behavioral gene combined with other data called code genes.

> A combination of one or more behavioral genes may be combined with one or more code genes and the combination compared to a predetermined

> collection of user behavior-code operation indications ("phenotypes"). The phenotypes may comprise a grouping of specific behavioral and code gene combinations that are typically present in a type of malicious usage session or operation with a computer.
> (Ex. 1005 at [0008]).

111. A POSITA would have recognized that Daswani's hard and soft signals can be applied to the behavioral genes as they are arbitrarily applied to behaviors. "In one embodiment, soft signal and hard signal classifications are assigned to behaviors to indicate a likelihood that a certain behavior is due to malware on the application being tested. Additionally, in one embodiment detected soft signals and hard signals can be combined to determine an overall likelihood of the tested application containing malware." (Ex. 1006 at 4:2-8).

112. A POSITA also would have been motivated to use Daswani's approach because there are multiple configurations of hard and soft signal calculations in order to compute a final probability, and the calculations can be adjusted according to the false positive rate.

> FIG. 5 is a flow diagram illustrating an example method for combining hard signals and soft signals to determine a likelihood of malware according to one embodiment of the present disclosure. ***After the analysis module 204 observes behaviors and classifies hard and soft signals, the analysis module 204 proceeds to assess the likelihood of an application being malicious based on hard and soft signal infection rates and a false positive rate.*** Analysis module 204 determines 502 observed hard signal behaviors and determines a probability of maliciousness for each of the hard signals. Similarly, analysis module 204 determines 504 observed soft signal behaviors and determines a probability of maliciousness for each of the soft signals. In one embodiment, as previously discussed, individual hard signals have a probability of maliciousness high enough to classify the application as malicious. On the other hand, the probability of maliciousness for each of the observed soft signals is combined to calculate 506 a combined probability of maliciousness. If hard signal is observed, or if the combined probability of maliciousness exceeds a maliciousness threshold, the application is classified 508 as malicious. If no hard signal is observed, and if the combined probability of maliciousness fails to exceed the maliciousness threshold, the application is classified as non-malicious.
> (Ex. 1006 at 8:3-28).

113. Based on these disclosures, a POSITA would have understood that the phenotypes disclosed by Wright could be ranked by hard signals, soft signals, or a combination based on infection rates and false positive rates. This method would maximize both flexibility in classification of phenotypes and permit minimization of false positives.

114. Because the arbitrary definitions of hard and soft signal can be applied directly to the behavioral genes, as discussed above, there are no difficult or complex steps to integrate the two systems. A POSITA would have recognized that the integration of the soft and hard signals to behavioral genes are most, if not all, of the combination process. Because Wright already teaches a ranking, once the signals are in place, Wright proceeds without modification. Thus, a POSITA would have recognized that this combination would require minimal experimentation while giving the POSITA high confidence of success.

115. Thus, a POSITA would have been motivated to combine Wright and Daswani.

B. Claim 1 is rendered obvious by Wright in view of Daswani

116. Each of the limitations of Claim 1 of the ’517 patent is rendered obvious to a POSITA by Wright in view of Daswani as described below.

1. Limitation 1[a]

117. Limitation 1[a] of the ’517 patent reads:

> *A method for assessing runtime risk for an application program that executes on a device, comprising:*

118. I understand that the preamble of a patent claim is not always considered limiting. However, to the extent that the preamble of Claim 1 of the ’517 is considered limiting, Wright in view of Daswani discloses a method comprising the limitations of the claim as shown in the following analysis for the individual limitations of the claim.

119. The ’517 patent method and system assess the “runtime risk” for an application, described as follows: “a runtime risk may identify a physical damage threat, accidental threat, deliberate threat, a threat of technical failure, a compromise of function or information, a combination thereof, etc.” (Ex. 1001 at 6:9-12).

120. Wright teaches a method of assessing the application during its runtime through an analysis of the application’s behaviors.

> Referring to FIG. 2, the security management facility 122 may comprise a host intrusion prevention system 200 which may provide behavioral-based protection through a behavior monitor facility 204. ***The behavior monitoring facility 204 may guard against unknown threats by analyzing behavior at runtime for executing software code 202.*** HIPS 200 may enable the assignment and temporary storage of Behavioral Genotype data against processes currently running, in order to construct a sequence of runtime behaviors. Matching this runtime genotype data with known combinations stored in the phenotype database 224, with or without an additional content analysis 222, may enable the identification and interruption of malware while it is executing. By matching combinations of behaviors in this way, detection of malware may be improved over solutions where only singular behaviors and a static content analysis is utilized.
> (Ex. 1005 at [0066], emphasis added).

121. Wright further teaches identifying the risk of the runtime qualities. After detecting sequences of behaviors that correspond to risky or dangerous processes, Wright’s system uses groupings of behaviors to identify malware, which a POSITA would have understood to be any software running on a system that contains (or causes to run) malicious and harmful code. In this way, Wright’s system assesses the runtime risk of an application.

> HIPS 200 may enable the assignment and temporary storage of Behavioral Genotype data against processes currently running, in order to construct a sequence of runtime behaviors. Matching this runtime genotype data with known combinations stored in the phenotype database 224, with or without an additional content analysis 222, ***may enable the identification and interruption of malware while it is executing***. By matching combinations of behaviors in this way, detection of malware may be improved over solutions where only singular behaviors and a static content analysis is utilized.

(Ex. 1005 at [0066], emphasis added).

122. Wright discloses "assessing the runtime risk of an application program that runs on a device." For example, Wright discloses monitoring a user's interaction with a computer, software application, operating system, graphical user interface, and so forth.

> In embodiments of the present invention improved capabilities are described for threat detection using a behavioral based host-intrusion prevention method and system ***for monitoring a user interaction with a computer, software application, operating system, graphic user interface, or some other component or client of a computer network***, and performing an action to protect the computer network based at least in part on the user interaction and a computer code process executing during or in association with a computer usage session.
> (Ex. 1005 at Abstract, emphasis added).

123. A POSITA would have understood a software application, an operating system, and a graphical user interface to be applications that can only be interacted with while running on a computer, which a POSITA would have understood to teach the claimed "application program that executes on a device."

124. Thus, I conclude that Wright in view of Daswani discloses limitation 1[a].

2. Limitation 1[b]

125. Limitation 1[b] of the '517 patent reads:

storing, in a rules database, a plurality of rules, wherein each rule identifies an action sequence;

126. Wright discloses limitation 1[b]. For clarity, I have divided my analysis into two sections below.

storing, in a rules database, a plurality of rules.

127. As explained in more detail below, Wright discloses storing behavioral genes and code genes, each of which a POSITA would have understood to teach the claimed "rules", in a database. A POSITA would have understood a database storing rules to be a "rules database."

128. Wright explains that a behavioral gene describes one or more user behaviors.

> For example, a behavioral gene called "Internet download" may be created and stored in association with the computer network. This behavioral gene ***may summarize the user behaviors*** of 1) executing a hyperlink on the Internet, and 2) performing a "save as" action in which a file is saved to a location in the computer network, such as a hard drive. A single user behavior (e.g., closing an application) may be recorded and stored as a behavioral gene. ***Alternatively, a plurality of user behaviors may be recorded within a single behavioral gene***.
> (Ex. 1005 at [0007], emphasis added).

129. On the other hand, a code gene describes the behavior of running software (code). As contrasted to a behavioral gene that tracks a user's behavior, a code gene tracks code behavior:

> In embodiments, the indication of the code operation may be a result of comparing ***an operation with a predetermined code behavior*** (hereinafter, a "code gene"). For example, a code gene may be a virus definition, a malware definition, an adware definition, or some other code behavior.
> (Ex. 1005 at [0008], emphasis added).

130. Wright describes some examples of the code operations associated with code genes, including "system startup, login, scheduled job process, [and] system service…" (Ex. 1005 at [0081]). A POSITA would have understood that although described as a single "operation," performing system startup, login, job processes, or a system service is a combination of multiple component operations. (*See* Section V.A.).

131. Wright discloses that these genes will later be used to match against observed behavior at runtime:

> In an example, the behavior monitoring facility 204 may monitor behaviors associated with the W32/Sality executable file virus. The behaviors monitored may be: disables OS tools in the registry, disables the MS windows firewall in the registry, adds itself to the firewall authorised

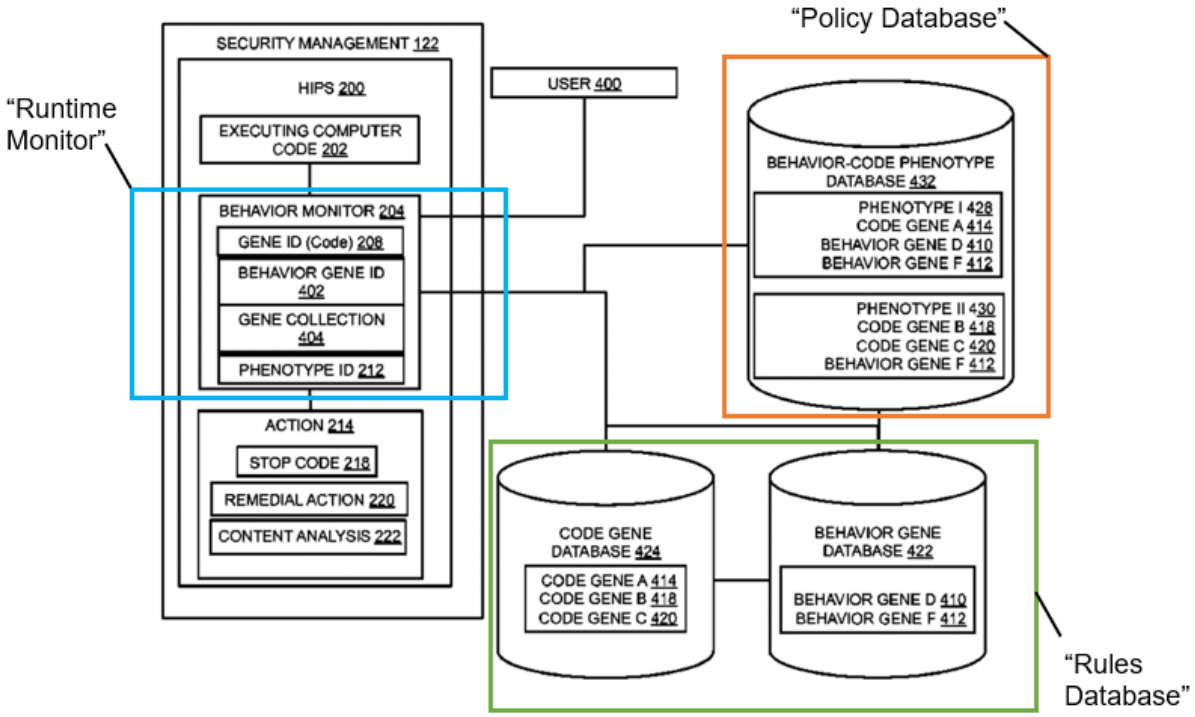
> applications list in the registry, and injects code into explorer.exe. ***The behavioral monitoring facility 204 would set runtime genes associated with these behaviors*** and the phenotype would be the combination of those runtime genes, not necessarily in order.
> (Ex. 1005 at [0072], emphasis added).

132. Additionally, Wright discloses that both behavioral genes and code genes are stored in a database, which discloses "storing, in a rules database, a plurality of rules."

> For each of the user behaviors occurring within the computer network, or derived from behaviors recorded from interaction with another computer network, a ***behavioral gene may be stored in a database*** that is associated with the computer network in which the data relating to the user behavior is recorded and stored.
> (Ex. 1005 at [0007], emphasis added).

> The behavioral gene ***and the code gene*** may be stored for reference in a database.
> (Ex. 1005 at [0008], emphasis added).

133. In Figure 4, Wright visualizes both types of genes as being stored in a database, annotated hereinbelow. A POSITA would have understood that a cylinder in a computer diagram represents a database.



(Ex. 1005 at Figure 4, annotated).

wherein each rule identifies an action sequence

134. As explained previously Wright discloses a "behavioral gene" that indicates one or more user behaviors. Each user behavior may be determined based on matching one or more "user interactions."

> In embodiments of the present invention, a behavioral based host-intrusion prevention method and system may be used for monitoring a user interaction with a computer, software application, operating system, graphic user interface, or some other component of a computer (e.g. a client) within a computer network, and performing an action to protect the computer and/or computer network based at least in part on the user interaction. ***Such protection may be provided based at least in part by monitoring a user interaction with a computer, and/or computer network client device***, during a usage session for an indication of a user behavior and monitoring a computer code process executing during the usage session for an indication of a code operation. ***The indication of the user behavior may be a result of comparing the user interaction, or plurality of user***

> ***interactions, with a predetermined behavior (hereinafter, a "behavioral gene")***. For example, markers of user behaviors with a computer or client device of a computer network may be monitored, recorded, stored, classified, and/ or analyzed based at least in part on the indicia of behavior, and data relating to the behavior, that are created by the user behavior.
> (Ex. 1005 at [0007], emphasis added).

135. Wright further explains that a "user interaction" is an action performed by the user that causes some sort of effect, or intentional non-effect (such as a "indicator of inactivity, such as activation of a screen saver or hibernation process" (*Id*. at [0011])). User interactions include a wide variety of actions a user can perform. A POSITA would understand the following examples as actions that correspond with the "action" mentioned in 1[a].

> In embodiments, the ***user interaction*** with a computer may include, but is not limited to, executing a link, executing an Internet link, executing an intranet link, downloading a content item from a server, forwarding an URL, forwarding an email, opening an application, closing an application, updating an application, patching an application, bookmarking a website, placing an VoIP call, adding an item to a shopping cart, entering a financial datum in a field, making a purchase, using a webcam, creating an application macro, an application shortcut, entering a novel keystroke combination, remotely accessing a computer, submitting a password, attaching a peripheral device, a temporal measure (e.g., time spent on a page, or time between a first and second user action), a change in a network behavior, a change in a firewall setting, a bridging of two previously unconnected networks, or some other type of user interaction.
> (Ex. 1005 at [0010]).

136. A POSITA would have understood Wright's "user interaction" to be an "action" as recited by the claim language. This is consistent with the '517 patent specification which describes "user actions" in a similar fashion:

> The user action 302 may be any form of explicit input from a user of the computing system 100. For example, the user action may include a program launch, key board input, click of a mouse, touch input on a touch screen, moving of a mouse, scrolling or clicking on a click wheel, voice command into a microphone, gestures into a device configured to read human gestures, etc.
> (Ex. 1001 at 6:56-62).

137. As explained previously, Wright discloses that a single behavioral gene may identify more than one user behavior and each behavior may identify more than one user interaction:

> For example, a behavioral gene called "Internet download" may be created and stored in association with the computer network. This behavioral gene may summarize the user behaviors of 1) executing a hyperlink on the Internet, and 2) performing a "save as action in which a file is saved to a location in the computer network, such as a hard drive. A single user behavior (e.g., closing an application) may be recorded and stored as a behavioral gene. ***Alternatively, a plurality of user behaviors may be recorded within a single behavioral gene.***
> (Ex. 1005 at [0007], emphasis added).
>
> ***The indication of the user behavior may be a result of comparing the user interaction, or plurality of user interactions, with a predetermined behavior (hereinafter, a "behavioral gene")***. For example, markers of user behaviors with a computer or client device of a computer network may be monitored, recorded, stored, classified, and/ or analyzed based at least in part on the indicia of behavior, and data relating to the behavior, that are created by the user behavior.
> (Ex. 1005 at [0007], emphasis added).

138. Thus, Wright discloses that a behavioral gene may indicate more than one user interaction. Therefore, it indicates a sequence of actions.

139. Moreover, Wright also discloses that code genes can be predetermined code operations where the "operation" is composed of component operations. As explained above, the code operations disclosed by Wright include "system startup, login, scheduled job process, [and] system service…" (Ex. 1005 at [0081]). A POSITA would have understood that although described as a single "operation," performing system startup, login, job processes, or a system service is a combination of multiple component operations. For example, system startup would involve at least booting a kernel and launching a user interface. Logging in would involve at least comparing a user's login credentials to a database as well as initiating a controlling process (e.g., a terminal, or GUI session) under the user's identity.

140. The ’517 Patent illustrates the granularity of an “action” in the specification. It explains that:

> The application action 304 may include, for example, ***a start type (e.g., console, service, parent process, script), first time execution, fork process (e.g., child process creation), code injection methods, keyboard capture methods, screen capture methods, dynamic code blocks, open data documents, automatic or self-termination, abnormal terminations (e.g., exception conditions), multiple restarts, opening of device(s) (e.g., the device for which the runtime risk is identified), process enumeration, cryptography, communication protocols, performance metrics,*** etc. as will be apparent to persons having skill in the relevant art.
> (Ex. 1001 at 7:4-14).

141. A POSITA, for example, would have noted that a system start-up would involve many first-time executions and forked processes. It would likely include at least opening data documents, automatic termination, opening of devices, communication protocols, and so forth. (*See* Section V.A. and V.B.). Moreover, Wright explains that changing OS tools in a registry and making system modifications are examples of code operations that may be stored as code genes.

> In embodiments, indication of a code operation may include, but is not limited to, an automatic code behavior, such as a system startup, login, scheduled job process, system service, or some other type of code operation. The gene may be at least one of: disables OS tools in a registry, disables a firewall in a registry, adds itself to a fire wall authorized applications list in a registry***, injects code into an application***, copies itself to a system folder, sets a run key to itself in a registry, sets a second run key to itself in a registry in a different location, and opens a hidden file with write access. The gene may be at least one of a system modification and a behavior of process. The phenotype may be a combination of more than one gene.
> (Ex. 1005 at [0013]-[0014], emphasis added).

142. A POSITA would further recognize that injecting code into an application as a “code injection method” as mentioned in the 517.

143. The ’517 Patent also discloses that, in addition to “application” actions, there are also “system” actions.

> The system action 306 may include, for example, ***system file creation, system folder modification, registry modifications or additions, application setting modifications, driver install or uninstall (e.g., system files), network activity, execution from temporary system folders, alternate data streams (ADS), environmental aspects (e.g., debugger presence, virtual environments, system boot times/sequences)***, etc.
> (Ex. 1001 at 7:19-24).

144. A POSITA would have known that system startup, logging in, starting a scheduled process, and system services would include multiple of these "system actions" including, at least, system file creation, system folder modification, registry modification or additions, application setting modifications, driver install or uninstall, network activity, execution from temporary system files, and so forth. (*See* Section V.A. and V.B.).

145. Thus, Wright discloses that a code gene contains many application and/or system actions. Therefore, the code gene indicates an action sequence.

146. Thus, I conclude that Wright discloses limitation 1[b].

3. Limitation 1[c]

147. Limitation 1[c] of the '517 patent reads:

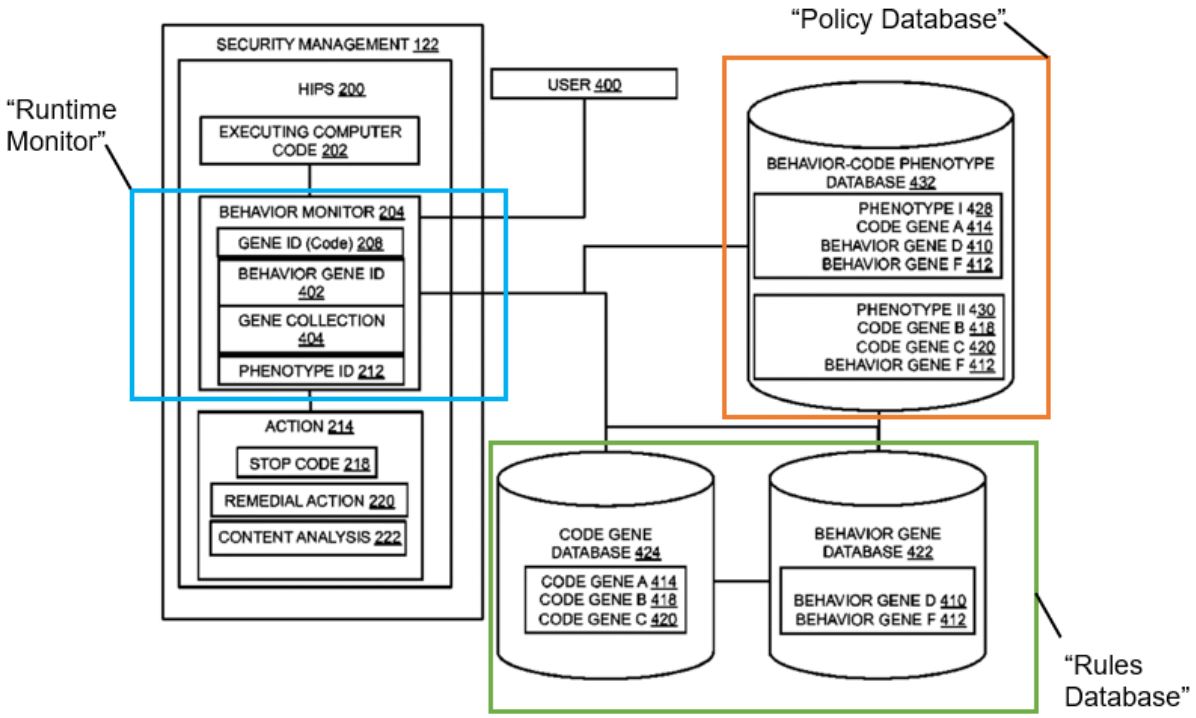
> *storing, in a policy database, a plurality of assessment policies, wherein each assessment policy includes at least one rule of the plurality of rules;*

148. Wright discloses limitation 1[c].

149. I have already explained with reference to limitation 1[b] that Wright discloses rules in the form of behavioral genes and code genes. Wright further discloses a "phenotype" which identify potential "malicious usage" based on behavioral genes combined with code genes.

> A combination of ***one or more behavioral genes may be combined with one or more code genes*** and the combination compared to a predetermined collection of user behavior-code operation indications ("phenotypes). The phenotypes may comprise a grouping of specific behavioral and code gene combinations that are typically present in a type of malicious usage session or operation with a computer.

(Ex. 1005 at [0008], emphasis added).



(Ex. 1005 at Figure 4).

150. Wright explains that "[p]henotypes may capture a combination or a series of behaviors that may be ranked to create increasing levels of confidence that the runtime object being monitored is executing a behavior pattern comparable to a known family of malware." (Ex. 1005 at [0069]). In other words, different phenotypes, each of which is associated with suspicious behaviors and code operations, may be compared against the runtime object's behavior and a confidence level associated with each one. A POSITA would have understood evaluations of known behavior against actual behavior to be an "assessment policy". Notably, the '517 patent specification does not describe an "assessment policy" except to describe using an assessment policy to identify a runtime risk. I address the requirement to identify a runtime risk in my analysis of limitation 1[d].

151. And, as I explained with respect to limitation 1[b], the phenotype includes at least one behavioral gene which I explained teaches a "rule" as recited by the claim language.

152. Thus, I conclude that Wright discloses limitation 1[c].

4. Limitation 1[d]

153. Limitation 1[d] of the '517 patent reads:

> *identifying, using at least one assessment policy, a runtime risk for an application program that executes on a device, wherein the identified runtime risk indicates a risk or threat of the identified action sequence of the application; and*

154. Wright discloses limitation 1[d]. For clarity, I have divided my analysis into two sections. The first portion of this claim recites:

> *identifying, using at least one assessment policy, a runtime risk for an application program that executes on a device*

155. The '517 patent defines a "runtime risk" as one that "may identify a physical damage threat, accidental threat, deliberate threat, a threat of technical failure, a compromise of function or information, a combination thereof, etc." (Ex. 1001 at 6:9-12). Wright discloses this risk by identifying the connection between a recognized phenotype, and a known family of malware. This connection represents an identification of a "deliberate threat" or a "compromise of function or information", as denoted in the specification of the '517 patent.

156. This connection is further elaborated on by Wright by ranking the phenotypes according to their propensity to be related to known families of malware:

> **The behavior monitoring facility 204 may be able to identify a phenotype 212** of user and code behaviors in relation to an executing code by comparing a collection of observed user and code behaviors with the predetermined collections of known malicious user and code **behaviors**

> **stored as phenotypes** in a behavior-code phenotype database 432, either temporarily or for a longer term. **Behavior-code phenotypes may capture a combination or *a series of user and code behaviors that may be ranked to create increasing levels of confidence that the runtime object being monitored is executing a behavior pattern comparable to a known family of malware***. The predetermined collections of behaviors, or phenotypes, may be created based on ***suspicious or risky user behavior*** (e.g. downloading a file from a website, running an executable, opening a signed file with an unverifiable digital signature, or other potentially suspicious or risky behavior) and scaled analysis of families of malware in runtime, such as in a threat research facility 132 of a threat management facility 100.
> (Ex. 1005 at [0089], emphasis added).

157. Wright further explains that a phenotype may identify specific families of malware. For example:

> Including greater combinations of user and code behaviors in the phenotype database 432 may enable more detailed analysis of a suspicious user behavior or a suspicious code. For example, an analysis of a suspicious code downloaded from a suspicious site by a user may be moved from 'Accessing the registry in this location is bad' to 'This process is exhibiting the same behavior as the family of malware Conficker' based on a comparison of a detected phenotype with an existing phenotype.
> (Ex. 1005 at [0069]).

158. A POSITA would have understood the identification of malware, especially by name and/or family type, to be an identification of a runtime risk, as the goals of malware align with the definition in the '517 specification of "runtime risk," especially when the particular type of malware can be determined.

159. Furthermore, Wright is explicit that its risk identification is of executing code: "The behavior monitoring facility 204 may guard against unknown threats by ***analyzing behavior at runtime for executing software 202***." (Ex. 1005 at [0066], emphasis added). A POSITA would have understood that "executing software" teaches the same concept as "an application program that executes on a device."

160. The final portion of the claim limitation reads:

wherein the identified runtime risk indicates a risk or threat of the identified action sequence of the application

161. As explained above, the '517 patent defines a "runtime risk" as one that "may identify a physical damage threat, accidental threat, deliberate threat, a threat of technical failure, a compromise of function or information, a combination thereof, etc." (Ex. 1001 at 6:9-12). As the specification explains, a "risk" or "threat" includes determinations such as "physical damage threat", "accidental threat", "deliberate threat", and so forth.

162. The runtime risk disclosed by Wright, as discussed above, identifies known family of malware. A POSITA would recognize an identification of a known family of malware to be a "risk or threat of the identified action sequence." In terms of the language of the specification, an identification of a known family of malware is also an identification of a "deliberate threat" or a "compromise of function or information", as denoted in the specification of the '517 patent.

163. Thus, I conclude that Wright discloses limitation 1[d].

5. Limitation 1[e]

164. Limitation 1[e] of the '517 patent reads:

identifying, by a runtime monitor including a processing device, a behavior score for the application program that executes on the device based on the identified runtime risk, wherein

165. Wright in view of Daswani discloses limitation 1[e].

166. Wright discloses a ranking system that a POSITA would understand to have "a behavior score based on the identified runtime risk". This score would explain the basis for the ranking expressed in the. As a POSITA would have understood, a ranking cannot exist without a qualitative or quantitative score.

> Phenotypes may capture a combination or a series of behaviors that may be ranked to create increasing levels of confidence that the runtime object being monitored is executing a behavior pattern comparable to a known family of malware.
> (Ex. 1005 at [0089]).

167. Wright discloses the components of phenotypes: "A phenotype may comprise one behavioral gene and a plurality of code genes, a plurality of behavioral genes and one code gene, or some other combination of behavioral genes and code genes." (Ex. 1005 at [0008]). These phenotypes are shown to be associated with a known family of malware, as I explained with reference to limitation 1[d], thereby associating the behavior-code phenotypes—the objects shown to include rules as an assessment policy—with a corresponding identified runtime risk.

168. Wright further discloses ranking behaviors as an indication of confidence that observed combinations of behaviors are malicious:

> The behavior monitoring facility 204 may be able to identify a phenotype 212 of behaviors in an executing code by comparing a collection of observed behaviors with the predetermined collections of known malicious behaviors stored as phenotypes in a phenotype database 224, either temporarily or for a longer term. ***Phenotypes may capture a combination or a series of behaviors that may be ranked to create increasing levels of confidence that the runtime object being monitored is executing a behavior pattern comparable to a known family of malware.*** The predetermined collections of behaviors, or phenotypes, may be created based on scaled analysis of families of malware in runtime, such as in a threat research facility 132 of a threat management facility 100. Including greater combinations of behaviors in the phenotype database 224 may enable more detailed analysis of a suspicious code. For example, an analysis of a suspicious code may be moved from 'Accessing the registry in this location is bad' to 'This process is exhibiting the same behavior as the family of malware Conficker' based on a comparison of a detected phenotype with an existing phenotype. ***This more detailed analysis may eliminate many unwanted detections as the probability of a legitimate application attempting to execute a series of bad behaviors is significantly lower.***
> (Ex. 1005 at [0069], emphasis added).

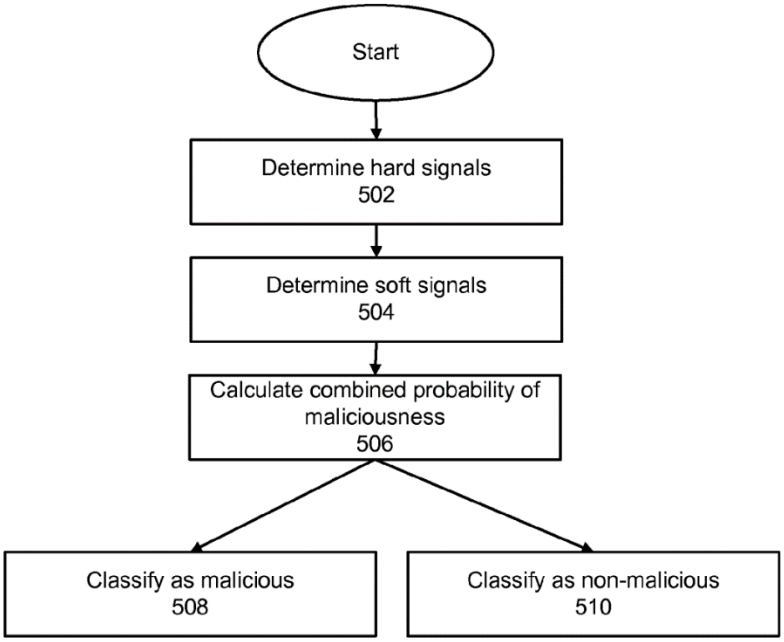
169. A POSITA would have recognized Wright's disclosures of ranking observed behaviors as equivalent to the claimed "behavior score," as both concepts are designed to measure risk posed by a monitored application.

170. To the extent that Wright's usage of user interactions to describe a ranked behavior score based on the user interaction's phenotype comparison with known families of malware does not constitute an identification of a behavior score, Wright in view of Daswani discloses limitation 1[e]. While Wright indicates that the combination or series of behaviors will be ranked to create increasing levels of confidence, Daswani directly indicates a probabilistic scoring to determine an overall likelihood that an application is malicious.

171. Daswani discloses hard and soft signals that represent indications of how much a system call or action is likely to contain malware, a significant indicator of risk. Daswani models behaviors as signals:

> Behavioral analysis of a mobile application is performed to determine whether the application is malicious. During analysis, various user interactions are simulated in an emulated environment to activate many possible resulting behaviors of an application. The behaviors are classified as hard or soft signals.
> (Ex. 1006 at Abstract).
>
> In one embodiment, some system calls or actions simulated in the emulated environment may be classified as soft signals or hard signals. In one embodiment, soft signals comprise system calls or actions which have an associated probability of the application containing malware. Hard signals comprise system calls or actions that are by definition malicious actions, or associated with a probability of the application containing malware, the probability exceeding a hard signal threshold.
> (Ex. 1006 at 6:47-55).

172. These hard and soft signals enter into a probabilistic signal combination module that computes the likelihood that an application is malicious. Shown in the following figure and excerpt are this process:



(Ex. 1006 at Figure 5).

> Signal combination module 306 combines multiple behaviors signals to determine an overall likelihood that an application is malicious.
> (Ex. 1006 at 6:45-47).

173. Daswani's signal combination module outputs a determination for an application, malicious or non-malicious, based on a computed score. Daswani refers to this score as the "probability of maliciousness". Daswani gives an example for how this score may be computed:

> Likewise if one or more independent soft signals are observed, each of the soft signals has their own probability of indicating that an application is malicious. According to one embodiment, the probability of maliciousness, X, that an application is malicious when a single independent soft signal is observed is shown in equation (1) below, where P is the overall probability that any application contains malware without specific information about the application, D is the detection rate or probability that the soft signal is observed for a malicious application, and F is the false positive rate or probability that the soft signal is observed for a legitimate application.
> (Ex. 1006 at 7:9-20).

$$X=(D*P)/(D*P+F*(1-P)) \qquad (1)$$

(Ex. 1006 at Equation 1).

174. A POSITA would recognize a score that classifies an application as malicious or non-malicious as a score that is quantifying a behavior, which would be understood as a "behavior score" as mentioned in the '517 patent.

175. A POSITA would have been motivated to combine Wright with Daswani. In addition, a POSITA using Wright would have sought other methods that assign values to the propensity a behavior expresses malicious functionality in order to create Wright's rankings. Daswani expresses a mathematical computation to attain such a numerical value, which would be of great value to a POSITA implementing Wright.

176. Thus, I conclude that Wright in view of Daswani discloses limitation 1[e].

6. Limitation 1[f]

177. Limitation 1[f] of the '517 patent reads:

the action sequence is a sequence of at least two performed actions, and

178. Wright in view of Daswani discloses limitation 1[f]. Specifically, Wright discloses multiple included user interactions for a given action sequence. As I explained with reference to limitation 1[b], Wright describes a behavioral gene as one that may be a result of a comparison between a plurality of user interactions with a predetermined behavior. A plurality of user interactions is encompassing of at least two performed actions.

> The indication of the user behavior may be a result of comparing the user interaction, **or plurality of user interactions**, with a predetermined behavior (hereinafter, a "behavioral gene"). For example, markers of user behaviors with a computer or client device of a computer network may be monitored, recorded, stored, classified, and/or analyzed based at least in part on the indicia of behavior, and data relating to the behavior, that are created by the user behavior. For example, a user clicking on a link may be detected, recorded, stored, classified, and/or analyzed. Other sample behaviors may include, but are not limited to, a user scrolling down a page, opening or closing a window, clicking on a link on a webpage, downloading a file,

> saving a file, running a file, typing a keyword, or some other user interaction.
> (Ex. 1005 at [0075], emphasis added).

179. As I also explained with respect to limitation 1[b], Wright also discloses that a code gene contains many application and/or system actions. Therefore, the code gene indicates an action sequence. I incorporate my analysis of limitation 1[b] into this section by reference.

180. Thus, I conclude that Wright discloses limitation 1[f].

7. Limitation 1[g]

181. Limitation 1[g] of the '517 patent reads:

> *each performed action is at least one of: a user action, an application action, and a system action.*

182. Wright discloses limitation 1[g]. Although the claim language only requires "at least one of," Wright discloses all three types of actions: "user actions", "application actions", and "system actions."

183. For example, Wright discloses actions which as "user interactions" compared against the predetermined behavior (genes):

> The indication of the user behavior may be a result of comparing the user interaction, ***or plurality of user interactions***, with a predetermined behavior (hereinafter, a "behavioral gene"). For example, markers of user behaviors with a computer or client device of a computer network may be monitored, recorded, stored, classified, and/or analyzed based at least in part on the indicia of behavior, and data relating to the behavior, that are created by the user behavior. For example, a user clicking on a link may be detected, recorded, stored, classified, and/or analyzed. Other sample behaviors may include, but are not limited to, a user scrolling down a page, opening or closing a window, clicking on a link on a webpage, downloading a file, saving a file, running a file, typing a keyword, or some other user interaction.
> (Ex. 1005 at [0075], emphasis added).

184. These user interactions are the actions performed by the user that cause an effect on the computer system. A POSITA would have recognized that these user interactions are like the user actions described in the specification of the ’517 patent:

> The user action 302 may be any form of explicit input from a user of the computing system 100. For example, the user action may include a program launch, key board input, click of a mouse, touch input on a touch screen, moving of a mouse, scrolling or clicking on a click wheel, voice command into a microphone, gestures into a device configured to read human gestures, etc.
> (Ex. 1001 at 6:56-62).

185. As noted in both Wright and the ’517 patent specification, user actions include using a mouse keyboard input, and so forth.

186. I also explained with respect to limitation 1[b] that code genes, which are also rules identified in the phenotypes, include application and system actions. As explained above, the code operations disclosed by Wright include “system startup, login, scheduled job process, [and] system service…” (Ex. 1005 at [0081]). A POSITA would have understood that although described as a single “operation,” performing system startup, login, job processes, or a system service is a combination of multiple component operations. For example, system startup would involve at least booting a kernel and launching a user interface. Logging in would involve at least comparing a user’s login credentials to a database as well as initiating a controlling process (e.g., a terminal, or GUI session) under the user’s identity. (*See* Section V.A.).

187. The ’517 patent illustrates the granularity of an “action” in the specification. It explains that:

> The application action 304 may include, for example, ***a start type (e.g., console, service, parent process, script), first time execution, fork process (e.g., child process creation), code injection methods, keyboard capture methods, screen capture methods, dynamic code blocks, open data documents, automatic or self-termination, abnormal terminations (e.g., exception conditions), multiple restarts, opening of device(s) (e.g., the***

> ***device for which the runtime risk is identified), process enumeration, cryptography, communication protocols, performance metrics,*** etc. as will be apparent to persons having skill in the relevant art.
> (Ex. 1001 at 7:4-14).

188. A POSITA, for example, would have noted that a system start-up would involve many first-time executions and forked processes. It would likely include at least opening data documents, automatic termination, opening of devices, communication protocols, and so forth. A POSITA would have identified all of these actions as application actions. (*See* Section V.A.).

189. The '517 Patent also discloses that, in addition to "application" actions, there are also "system" actions.

> The system action 306 may include, for example, ***system file creation, system folder modification, registry modifications or additions, application setting modifications, driver install or uninstall (e.g., system files), network activity, execution from temporary system folders, alternate data streams (ADS), environmental aspects (e.g., debugger presence, virtual environments, system boot times/sequences)***, etc.
> (Ex. 1001 at 7:19-24).

190. A POSITA would have known that system startup, logging in, starting a scheduled process, and system services would include multiple of these "system actions" including, at least, system file creation, system folder modification, registry modification or additions, application setting modifications, driver install or uninstall, network activity, execution from temporary system files, and so forth. (*See* Section V.A. and V.B.).

191. Thus, I conclude that Wright discloses limitation 1[g].

192. Thus, Wright in view of Daswani renders Claim 1 obvious to a POSITA.

C. Claim 2 is rendered obvious by Wright in view of Daswani

193. Claim 2 of the '517 patent is rendered obvious to a POSITA by Wright in view of Daswani. This claim, which depends from Claim 1, reads:

The method of claim 1, wherein the user action is any form of explicit input from a user of a computing system.

194. Wright discloses the additional requirements of Claim 2.

195. As mentioned previously in limitation 1[b], Wright discloses the action of executing a link as a "user interaction".

> In embodiments, the **user interaction** with a computer may include, but is not limited to, executing a link, executing an Internet link, executing an intranet link, downloading a content item from a server, forwarding an URL, forwarding an email, opening an application, closing an application, updating an application, patching an application, bookmarking a website, placing an VoIP call, adding an item to a shopping cart, entering a financial datum in a field, making a purchase, using a webcam, creating an application macro, an application shortcut, entering a novel keystroke combination, remotely accessing a computer, Submitting a password, attaching a peripheral device, a temporal measure (e.g., time spent on a page, or time between a first and second user action), a change in a network behavior, a change in a firewall setting, a bridging of two previously unconnected networks, or some other type of user interaction.
> (Ex. 1005 at [0010], emphasis added).

196. The ’517 patent specification recites the following regarding a “user action”:

> The user action 302 may be any form of explicit input from a user of the computing system 100. For example, the user action may include a program launch, keyboard input, click of a mouse, touch input on a touch screen, moving of a mouse, scrolling or clicking on a click wheel, voice command into a microphone, gestures into a device configured to read human gestures, etc.
> (Ex. 1001 at 6:56-62).

197. This form of input is consistent with the definition for a user action as an “explicit input by the user”. A POSITA would understand the action of executing a link to include either a keyboard input, or the clicking of a mouse. As shown above, both the clicking of a mouse and the pressing of a button are included in the ’517 patent specification’s definition for a “user action”.

198. Thus, Wright in view of Daswani renders Claim 2 obvious to a POSITA.

D. Claim 3 is rendered obvious by Wright in view of Daswani

199. Claim 3 of the ’517 patent is rendered obvious to a POSITA by Wright in view of Daswani. This claim, which depends on Claim 1, reads:

> *The method of claim 1, wherein the application action is any activity performed by the application initiated programmatically by a task in execution of a computing system.*

200. Wright discloses the additional requirements of Claim 3.

201. I also explained with respect to limitation 1[g], Wright discloses application actions that include “system startup, login, scheduled job process, [and] system service…” (Ex. 1005 at [0081]). A POSITA would have known that, at the very least, “system startup”, “scheduled job process”, and “system service” are initiated programmatically by a task in execution of a computing system. Each of these three examples includes launching one program by another program.

202. For example, system startup would involve at least booting a kernel and launching a user interface. As I explained in technical background Section V.A., these processes are launched programmatically in a particular order as a result of system configuration and system actions. This includes loading the necessary components and processes to start the operating system kernel. After the kernel has launched, it can launch user interface processes (e.g., a terminal or a Graphical User Interface) to collect and respond to user input. (*See* Section V.A.).

203. As a further example, a scheduled job process is a program that is launched programmatically on a specific schedule or timer. In addition, a system service is a background application that can be launched automatically on system start-up, login, or in response to another program. (*See* Sections V.A. and V.B.).

204. These examples represent various forms of "activity performed by the application initiated programmatically by a task in execution of a computing system."

205. Thus, Wright in view of Daswani renders Claim 3 obvious to a POSITA.

E. Claim 4 is rendered obvious by Wright in view of Daswani

206. Claim 4 of the '517 patent is rendered obvious to a POSITA by Wright in view of Daswani. This claim, which depends on Claim 1, reads:

> *The method of claim 1, wherein the system action is any operation performed by a computing system on behalf of, or as a consequence of, a user action or application action that changes the state of the computing system.*

207. Wright discloses the additional requirements of Claim 4.

208. Wright teaches monitoring certain user actions that a POSITA would have understood to modify the state of the system, including the following: "updating an application, patching an application" (i.e., modifying the contents of the filesystem), "attaching a peripheral device" (i.e., modifying the collection of devices available to the system), "a change in network behavior, a change in a firewall setting, a bridging of two previously unconnected networks" (i.e., changing the network configuration of the device). (Ex. 1005 at [0010]).

209. As I explained with respect to limitation 1[g] and Claim 3, a POSITA would have known that system startup, logging in, starting a scheduled process, and system services would include multiple of these "system actions" including, at least, system file creation, system folder modification, registry modification or additions, application setting modifications, driver install or uninstall, network activity, execution from temporary system files, and so forth. (*See* Sections V.A. and V.B.). A POSITA would have understood such "system actions" to modify the system

(e.g., the filesystem, the network configuration of the system, the configuration of applications on the system).

210. A POSITA would have understood that these system actions are either performed on behalf of or as a consequence of other user actions or application actions. For example, system file creation, system folder modification, and registry changes are restricted to system privileges (either the OS, a "System" user, or a "root" user). Users and applications cannot perform these operations directly. (*See* Section V.B.). Thus, a POSITA would have known that system files created are created on behalf of users or applications. Similarly, driver install or uninstall, network activity, and execution from temporary system files are triggered in response to a user action or an application action and would change the state of the system. (*See* Section V.B.).

211. Thus, Wright in view of Daswani renders Claim 4 obvious to a POSITA.

F. Claim 5 is rendered obvious by Wright in view of Daswani

212. Each of the limitations of Claim 5 of the '517 patent, which depends from Claim 4, is rendered obvious to a POSITA by Wright in view of Daswani as described below.

213. Claim 5 reads:

> *The method of claim 1, wherein a first action of the at least two performed actions is performed at a first time, and the second action of the at least two performed actions is performed at a second time.*

214. Wright discloses the additional requirements of Claim 5.

215. Wright explains that the various behavior, which are actions, are tracked "progressively" over time.

> A plurality of malicious user and code behavior indications may be collected in this fashion by comparing a user behavior or a behavior of an executing code with a predetermined behavior. Behavior and code genes may continue to be identified progressively over the entire course of a code execution so that at any time during execution, the behavior monitor facility

> 204 may match an observed phenotype with one stored in a behavior-code phenotype database 432.
> (Ex. 1005 at [0087]).

216. A POSITA would have understood tracking behavior over time means that one action is performed before another.

217. Thus, I conclude that Wright in view of Daswani renders Claim 5 obvious to a POSITA.

G. Claim 6 is rendered obvious by Wright in view of Daswani

218. Each of the limitations of Claim 6 of the ’517 patent, which depends from Claim 5, is rendered obvious to a POSITA by Wright in view of Daswani as described below.

219. Claim 6 reads:

> *The method of claim 5, wherein the first action is a user action, the second action is a system action, and the second time occurs after the first time.*

220. I have already explained with respect to Claim 4 that a POSITA would have known that system startup, logging in, starting a scheduled process, and system services would include multiple of these “system actions” including, at least, system file creation, system folder modification, registry modification or additions, application setting modifications, driver install or uninstall, network activity, execution from temporary system files, and so forth. A POSITA would have known that, at the very least, “logging in” is a user action and that this triggers (as a subsequent, second action), system actions. (*See* Section V.A.).

221. Thus, I conclude that Wright in view of Daswani renders Claim 6 obvious to a POSITA.

H. Claim 7 is rendered obvious by Wright in view of Daswani

222. Claim 7 of the ’517 patent is rendered obvious to a POSITA by Wright in view of Daswani. This claim, which depends on Claim 4, reads:

The method of claim 5, wherein the first action is a user action, the second action is an application action, and the second time occurs after the first time.

223. Wright discloses the additional requirements of Claim 7.

224. Wright explains that user interactions, such as clicking on a link, also cause code actions, such as downloading content from a server or forwarding a URL:

> In embodiments, the user interaction with a computer may include, but is not limited to, ***executing a link, executing an Internet link***, executing an intranet link, ***downloading a content item from a server***, ***forwarding an URL***, forwarding an email, opening an application, closing an application, updating an application, patching an application, bookmarking a website, placing an VoIP call, adding an item to a shopping cart, entering a financial datum in a field, making a purchase, using a webcam, creating an application macro, an application shortcut, entering a novel keystroke combination, remotely accessing a computer, Submitting a password, attaching a peripheral device, a temporal measure (e.g., time spent on a page, or time between a first and second user action), a change in a network behavior, a change in a firewall setting, a bridging of two previously unconnected networks, or some other type of user interaction.
> (Ex. 1005 at [0010], emphasis added).

225. The user and application actions emphasized above are non-limiting examples. A POSITA would have recognized that almost all of the user interactions above trigger one or more code actions. Wright also discloses various code actions that are triggered by user actions.

> In embodiments, indication of a code operation may include, but is not limited to, an automatic code behavior, such as a system startup, login, scheduled job process, system service, or some other type of code operation. The gene may be at least one of: disables OS tools in a registry, disables a firewall in a registry, adds itself to a fire wall authorized applications list in a registry**,** injects code into an application, copies itself to a system folder, sets a run key to itself in a registry, sets a second run key to itself in a registry in a different location, and opens a hidden file with write access. The gene may be at least one of a system modification and a behavior of process. The phenotype may be a combination of more than one gene.
> (Ex. 1005 at [0013]-[0014], emphasis added).

226. In the examples above, logging in involves a user action. This is a non-limiting example. A POSITA would have understood that many of these operations may be preceded by a user action.

227. Thus, Wright in view of Daswani renders Claim 7 obvious to a POSITA.

I. Claim 8 is rendered obvious by Wright in view of Daswani

228. Wright in view of Daswani renders Claim 8 obvious to a POSITA.

229. Claim 8 reads:

> *The method of claim 5, wherein the first action is an application action, the second action is a system action, and the second time occurs after the first time.*

230. Wright discloses the additional requirements of Claim 8.

231. I have already explained with respect to limitation 1[b] that Wright identifies system startup, login, and so forth to be ***code operations***, which are application actions.

> In embodiments, indication of a code operation may include, but is not limited to, an automatic code behavior, such as a system startup, login, scheduled job process, system service, or some other type of code operation. The gene may be at least one of: disables OS tools in a registry, disables a firewall in a registry, adds itself to a fire wall authorized applications list in a registry**,** injects code into an application, copies itself to a system folder, sets a run key to itself in a registry, sets a second run key to itself in a registry in a different location, and opens a hidden file with write access. The gene may be at least one of a system modification and a behavior of process. The phenotype may be a combination of more than one gene.
> (Ex. 1005 at [0013]-[0014], emphasis added).

232. I also explained with respect to 1[g] that a POSITA would have known that system startup, logging in, starting a scheduled process, and system services (which are "application actions") would include trigger "system actions" including, at least, system file creation, system folder modification, registry modification or additions, application setting modifications, driver install or uninstall, network activity, execution from temporary system files, and so forth.

233. Thus, I conclude that Wright in view of Daswani renders Claim 8 obvious to a POSITA.

J. Claim 9 is rendered obvious by Wright in view of Daswani

234. Claim 9 of the '517 patent is rendered obvious to a POSITA by Wright. This claim, which depends from Claim 1, recites:

> *The method of claim 1, wherein the action sequence includes three performed actions, a first action of the three performed actions is a user action, a second action of the three performed actions is an application action, a third action of the three performed actions is a system action, the second action is performed temporally after the first action, and the third action is performed temporally after the second action.*

235. Wright discloses the additional requirements of Claim 8.

236. I have already explained with respect to limitation 1[b] that Wright identifies system startup, login, and so forth to be ***code operations***, which are application actions.

> In embodiments, indication of a code operation may include, but is not limited to, an automatic code behavior, such as a system startup, login, scheduled job process, system service, or some other type of code operation. The gene may be at least one of: disables OS tools in a registry, disables a firewall in a registry, adds itself to a fire wall authorized applications list in a registry**,** injects code into an application, copies itself to a system folder, sets a run key to itself in a registry, sets a second run key to itself in a registry in a different location, and opens a hidden file with write access. The gene may be at least one of a system modification and a behavior of process. The phenotype may be a combination of more than one gene.
> (Ex. 1005 at [0013]-[0014]).

237. A POSITA would have understood that these code operations, such as system startup or logging in, are associated with a user action. For example, logging in requires a user to trigger the login, after entering credentials. Then, the code operation ("application actions") take place as I have explained above. (*See* Section V.A.).

238. I also explained with respect to 1[g] that a POSITA would have known that system startup, logging in, starting a scheduled process, and system services (which are "application actions") would include trigger "system actions" including, at least, system file creation, system folder modification, registry modification or additions, application setting modifications, driver install or uninstall, network activity, execution from temporary system files, and so forth. (*See* Section V.A. and V.B.).

239. Thus, I conclude that Wright in view of Daswani renders Claim 9 obvious to a POSITA.

K. Claim 10 is rendered obvious by Wright in view of Daswani

240. Each of the limitations of Claim 10 of the '517 patent, which depends from Claim 8, is rendered obvious to a POSITA by Wright in view of Daswani as described below.

241. Claim 10 reads:

> *The method of claim 1, wherein the runtime risk is identified based on matching rules and probabilistic weights and scores of performed actions on the device.*

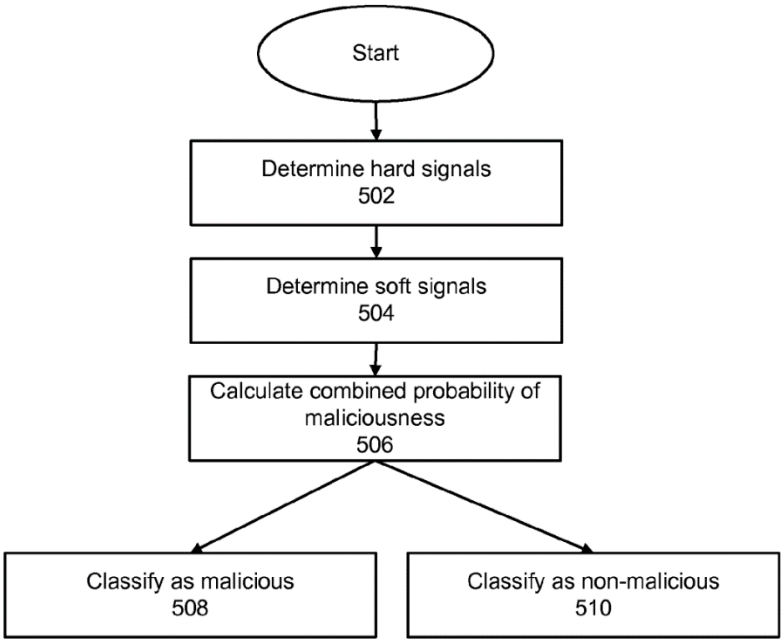
242. Wright in view of Daswani discloses the additional requirements of Claim 10.

243. I already explained with respect to limitation 1[e] that while Wright indicates that the combination or series of behaviors will be ranked to create increasing levels of confidence, Daswani directly indicates a probabilistic scoring to determine an overall likelihood that an application is malicious. Daswani discloses identifying the runtime risk "based on matching rules and probabilistic weights and scores of performed actions on the device."

244. Daswani discloses hard and soft signals that represent indications of how much a system call or action indicates the likelihood that the application contains malware, a significant indicator of risk. Each "signal" models behavior, and represents rules and probabilistic weights.

> Behavioral analysis of a mobile application is performed to determine whether the application is malicious. During analysis, various user interactions are simulated in an emulated environment to activate many possible resulting behaviors of an application. The behaviors are classified as hard or soft signals.
> (Ex. 1006 at Abstract).
>
> In one embodiment, some system calls or actions simulated in the emulated environment may be classified as soft signals or hard signals. ***In one embodiment, soft signals comprise system calls or actions which have an associated probability of the application containing malware***. Hard signals comprise system calls or actions that are by definition malicious actions***, or associated with a probability of the application containing malware, the probability exceeding a hard signal threshold***.
> (Ex. 1006 at 6:47-55, emphasis added).

245. These hard and soft signals enter into a probabilistic signal combination module that computes the likelihood that an application is malicious. This is shown in the following figure (Ex. 1006 at Figure 5) and the following excerpt:



> Signal combination module 306 combines multiple behaviors signals to determine an overall likelihood that an application is malicious.
> (Ex. 1006 at 6:45-47).

246. Daswani's signal combination module outputs a determination for an application, malicious or non-malicious, based on a computed score. Daswani refers to this score as the "probability of maliciousness". Daswani gives an example for how this score may be computed:

> Likewise if one or more independent soft signals are observed, each of the soft signals has their own probability of indicating that an application is malicious. According to one embodiment, the probability of maliciousness, X, that an application is malicious when a single independent soft signal is observed is shown in equation (1) below, where P is the overall probability that any application contains malware without specific information about the application, D is the detection rate or probability that the soft signal is observed for a malicious application, and F is the false positive rate or probability that the soft signal is observed for a legitimate application.
> (Ex. 1006 at 7:9-20).

$$X=(D*P)/(D*P+F*(1-P)) \qquad (1)$$

(Ex. 1006 at Equation 1).

247. In Daswani's disclosure, the "probability of maliciousness, X" represents the claimed "probabilistic weight." (Ex. 1006 at 7:12). As Daswani explains, this calculation is used to inform the malware determination when particular signals are observed: "A signal with a high detection rate and a low false positive rate increases the probability of maliciousness when the signal is observed." (Ex. 1006 at 7:35-28). An example illustrates the utility of having such a probability when detecting actions that may be either benign or malicious:

> For example, a soft signal may be accessing an address book on the mobile device. This action may be used for malicious purposes, but is often benign. In one embodiment, a soft signal has a higher probability of maliciousness if similar applications are unlikely to perform a similar action. A soft signal of accessing address book information may also be assigned an increased probability of maliciousness based on the amount of the address book entries accessed or the pattern in which entries are accessed. For example, iteratively retrieving every address from an address book may be assigned a higher probability of maliciousness.
> (Ex. 1006 at 6:55-66).

248. I have already explained with respect to limitation 1[e] why a POSITA would have been motivated to combine Daswani's soft and hard signals analysis with Wright. I incorporate my analysis into this section by reference.

249. Thus, Wright in view of Daswani renders Claim 10 obvious to a POSITA.

L. Claim 11 is rendered obvious by Wright in view of Daswani

250. Claim 11 of the '517 patent is rendered obvious to a POSITA by Wright in view of Daswani. This claim, which depends on Claim 8, reads:

> *The method of claim 1, wherein the runtime risk is identified based on time proximity of the at least two performed actions of the action sequence and natural affinity inferred between the at least two performed actions.*

251. Wright discloses the additional requirements of Claim 11.

252. Wright discloses that user interactions may include a temporal indicator. The temporal indicator may identify the time between two user actions.

> In embodiments, the user interaction with a computer may include, but is not limited to, executing a link, executing an Internet link, executing an intranet link, downloading a content item from a server, forwarding an URL, forwarding an email, opening an application, closing an application, updating an application, patching an application, bookmarking a website, placing an VoIP call, adding an item to a shopping cart, entering a financial datum in a field, making a purchase, using a webcam, creating an application macro, an application shortcut, entering a novel keystroke combination, remotely accessing a computer, submitting a password, attaching a peripheral device, ***a temporal measure (e.g., time spent on a page, or time between a first and second user action)***, a change in a network behavior, a change in a firewall setting, a bridging of two previously unconnected networks, or some other type of user interaction.
> (Ex. 1005 at [0010], emphasis added).

253. A POSITA would have understood this temporal measure based on a time between a first and second user action to be a "time proximity of the at least two performed actions of the action sequence." I have already explained that user interactions are the "user actions" required

by the claim. Thus, the user actions in the temporal measure are actions of the action sequence. Moreover, I have already explained that the runtime risk is calculated based on the user actions. Therefore, the runtime risk is based on the time indicator as well.

254. Moreover, Wright discloses that the two actions that define the "temporal measure" may have an "affinity." For example, in the citation above, Wright provides as an example "time spent on a page." A POSITA would have understood that the time spent on a webpage is commonly measured by the time between two user actions[2]. A first action indicates that the user is "on" the page, such as executing a link associated with the page. A second action indicates that the user is "off" the page, such as executing a link associated with another page. (*See* Section V.C.). A POSITA would have recognized, therefore, that measuring the time spent on a page is a measurement of two actions wherein the two actions a natural affinity inferred between them (e.g., the time spent on the page). (*See* Section V.C.).

255. Thus, I conclude that Claim 11 is rendered obvious to a POSITA by Wright in view of Daswani.

M. Claim 12 is rendered obvious by Wright in view of Daswani

256. Claim 12 of the '517 patent, which depends from Claim 1, is rendered obvious to a POSITA by Wright in view of Daswani.

1. Limitation 12[a]

257. Limitation 12[a] of the '517 patent reads:

[2] This metric was referred to, at the time of the alleged invention, as "time on site" and is calculated as "the difference in time between the last and first pageview a visitor requests (or event if you are also tracking these)." (Ex. 1008 at p. 5; *see also* Section V.C.).

The method of claim 1, further comprising: assessing, by the processing device, that the at least two performed actions of the action sequence describe a forensic chain of events based on a plurality of factors,

258. Wright discloses the requirements of limitation 12[a].

259. For example, Wright explains that malware is identified by phenotypes, which are specific combinations of actions.

> A combination of one or more behavioral genes may be combined with one or more code genes and the combination compared to a predetermined collection of user behavior-code operation indications ("phenotypes"). The phenotypes may comprise a grouping of specific behavioral and code gene combinations that are typically present in a type of malicious usage session or operation with a computer.
> (Ex. 1005 at [0008]).

260. A POSITA would have recognized "phenotypes", which are used to identify malicious usage, are a "forensic chain". And, as explicitly disclosed by Wright, a phenotype has at least two performed actions where one of the actions is a user "gene" and another action is a code "gene."

261. Thus, Wright discloses limitation 12[a].

2.Limitation 12[b]

262. Limitation 12[b] of the '517 patent reads:

wherein the plurality of factors includes at least one of: ordered sequences, unordered sequences, proximity analysis, outliers, probabilistic weights and scores, and probable action inferences based on an application's competency.

263. Wright discloses the additional requirements of limitation 12[b].

264. As I explained with respect to limitation 12[a], Wright discloses phenotypes, which are a forensic chain of actions. A phenotype may also include an "ordered sequence," an "unordered sequence", or "proximity analysis." For example, as explained with respect to limitation 12[a], the phenotype includes at least one user behavior gene, and one code gene. A POSITA would have known that this is either an ordered sequence of actions or an unordered sequence of actions.

265. As I also explained with respect to Claim 11, Wright also disclosed that user actions could include a measurement of time between two actions. A POSITA would have recognized this as "proximity analysis." I incorporate my analysis of Claim 11 into this section by reference.

266. Thus, Wright discloses limitation 12[b].

267. Thus, Wright in view of Daswani renders Claim 12 obvious to a POSITA.

N. Claim 13 is rendered obvious by Wright in view of Daswani

268. Claim 13 of the '517 patent is rendered obvious to a POSITA by Wright in view of Daswani.

1. Limitation 13[a]

269. Limitation 13[a] of the '517 patent reads:

> *A system for assessing runtime risk for an application program that executes on a device, comprising*

270. This limitation differs from limitation 1[a] only in the replacement of the word "method" with "system". Wright discloses that its techniques may be executed on a computer, which a POSITA would have understood to teach the claimed "system". For example:

> A computer program product embodied in a computer readable medium that, when executing on one or more computers, performs the steps of: … (Ex. 1005 at Claim 1).

271. Specifically, Wright discloses the following:

> The methods and/or processes described above, and steps thereof, may be realized in hardware, software or any combination of hardware and software suitable for a particular application. The hardware may include a general purpose computer and/or dedicated computing device or specific computing device or particular aspect or component of a specific computing device. (Ex. 1005 at [0107]).

272. The remainder of this limitation matches my analysis of limitation 1[a], which I incorporate here by reference.

273. Thus, Wright discloses limitation 13[a].

2. Limitation 13[b]

274. Limitation 13[b] of the '517 patent reads:

> *a rules database storing a plurality of rules, wherein each rule identifies an action sequence;*

275. This limitation is identical to limitation 1[b], except for the phrasing "a rules database ***storing***" instead of "***storing, in*** a rules database", which does not change the meaning of the limitation. Therefore, I incorporate my analysis of limitation 1[b] here by reference and conclude that Wright discloses limitation 13[b].

3.Limitation 13[c]

276. Limitation 13[c] of the '517 patent reads:

> *a policy database storing a plurality of assessment policies, wherein each assessment policy includes at least one rule of the plurality of rules; and*

277. This limitation is identical to limitation 1[c], except for the phrasing "a policy database ***storing***" instead of "***storing, in*** a policy database", which does not change the meaning of the limitation. Therefore, I incorporate my analysis of limitation 1[c] here by reference and conclude that Wright discloses limitation 13[c].

4. Limitation 13[d]

278. Limitation 13[d] of the ’517 patent reads:

a runtime monitor including a processing device

279. This limitation is identical to a portion of limitation 1[e], which recites “a runtime monitor including a processing device”. Therefore, I incorporate my analysis of limitation 1[e] here by reference and conclude that Wright in view of Daswani discloses limitation 13[d].

5. Limitation 13[e]

280. Limitation 13[e] of the ’517 patent reads:

identifying, using at least one assessment policy, a runtime risk for an application program that executes on a device, wherein the identified runtime risk indicates a risk or threat of the identified action sequence of the application, and

281. This limitation is identical to limitation 1[d]. I incorporate my analysis of limitation 1[d] here by reference and conclude that Wright discloses limitation 13[e].

6. Limitation 13[f]

282. Limitation 13[f] of the ’517 patent reads:

identifying a behavior score for the application program that executes on the device based on the identified runtime risk, wherein

283. This limitation is identical to limitation 1[e], except for the omission of “by a runtime monitor including a processing device” (which is instead recited in limitation 13[d], analyzed hereinabove). Therefore, I incorporate my analysis of limitation 1[e] here by reference and conclude that Wright in view of Daswani discloses limitation 13[f].

7.Limitation 13[g]

284. Limitation 13[g] of the ’517 patent reads:

the action sequence is a sequence of at least two performed actions, and

285. This limitation is identical to limitation 1[f]. Therefore, I incorporate my analysis of limitation 1[f] here by reference and conclude that Wright in view of Daswani discloses limitation 13[g].

8.Limitation 13[h]

286. Limitation 13[h] of the ’517 patent reads:

each performed action is at least one of: a user action, an application action, and a system action.

287. This limitation is identical to limitation 1[g]. Therefore, I incorporate my analysis of limitation 1[g] here by reference and conclude that Wright in view of Daswani discloses limitation 13[h].

288. Thus, Wright in view of Daswani renders Claim 13 obvious to a POSITA.

O. Claim 14 is rendered obvious by Wright in view of Daswani

289. Claim 14 of the ’517 patent reads:

The system of claim 13, wherein the user action is any form of explicit input from a user of a computing system.

290. The language of this claim is identical to that of Claim 2, except that “The method of claim 1” is replaced with “The system of claim 13”. As the claim otherwise has the same meaning, I incorporate here by reference my analysis of Claim 2 and conclude that Claim 14 is rendered obvious to a POSITA by Wright in view of Daswani.

P. Claim 15 is rendered obvious by Wright in view of Daswani

291. Claim 15 of the ’517 patent reads:

The system of claim 13, wherein the system action is any operation performed by a computing system on behalf of, or as a consequence of, a user action or application

292. The language of this claim is identical to that of Claim 3, except that "The method of claim 1" is replaced with "The system of claim 13". As the claim otherwise has the same meaning, I incorporate here by reference my analysis of Claim 3 and conclude that Claim 15 is rendered obvious to a POSITA by Wright in view of Daswani.

Q. Claim 16 is rendered obvious by Wright in view of Daswani

293. Claim 16 of the '517 patent reads:

The system of claim 13, wherein the system action is any operation performed by a computing system on behalf of, or as a consequence of, a user action or application action that changes the state of the computing system.

294. The language of this claim is identical to that of Claim 4, except that "The method of claim 1" is replaced with "The system of claim 13". As the claim otherwise has the same meaning, I incorporate here by reference my analysis of Claim 4 and conclude that Claim 16 is rendered obvious to a POSITA by Wright in view of Daswani.

R. Claim 17 is rendered obvious by Wright in view of Daswani

295. Claim 17 of the '517 patent reads:

The system of claim 13, wherein a first action of the at least two performed actions is performed at a first time, and the second action of the at least two performed actions is performed at a second time.

296. The language of this claim is identical to that of Claim 5, except that "The method of claim 1" is replaced with "The system of claim 13". As the claim otherwise has the same

meaning, I incorporate here by reference my analysis of Claim 5 and conclude that Claim 17 is rendered obvious to a POSITA by Wright in view of Daswani.

S. Claim 18 are rendered obvious by Wright in view of Daswani

297. Claim 18 of the ’517 patent reads:

> *The system of claim 17, wherein the first action is a user action, the second action is a system action, and the second time occurs after the first time.*

298. The language of this claim is identical to that of Claim 6, except that “The method of claim 5” is replaced with “The system of claim 17”. As the claim otherwise has the same meaning, I incorporate here by reference my analysis of Claim 6 and conclude that Claim 18 is rendered obvious to a POSITA by Wright in view of Daswani.

T. Claim 19 is rendered obvious by Wright in view of Daswani

299. Claim 19 of the ’517 patent reads:

> *The system of claim 17, wherein the first action is a user action, the second action is an application action, and the second time occurs after the first time.*

300. The language of this claim is identical to that of Claim 7, except that “The method of claim 5” is replaced with “The system of claim 17”. As the claim otherwise has the same meaning, I incorporate here by reference my analysis of Claim 7 and conclude that Claim 19 is rendered obvious to a POSITA by Wright in view of Daswani.

U. Claim 20 is rendered obvious by Wright in view of Daswani

301. Claim 20 of the ’517 patent reads:

The system of claim 17, wherein the first action is an application action, the second action is a system action, and the second time occurs after the first time.

302. The language of this claim is identical to that of Claim 8, except that "The method of claim 5" is replaced with "The system of claim 17". As the claim otherwise has the same meaning, I incorporate here by reference my analysis of Claim 8 and conclude that Claim 20 is rendered obvious to a POSITA by Wright in view of Daswani.

V. Claim 21 is rendered obvious by Wright in view of Daswani

303. Claim 21 of the '517 patent reads:

The system of claim 13, wherein the action sequence includes three performed actions, a first action of the three performed actions is a user action, a second action of the three performed actions is an application action, a third action of the three performed actions is a system action, the second action is performed temporally after the first action, and the third action is performed temporally after the second action.

304. The language of this claim is identical to that of Claim 9, except that "The method of claim 1" is replaced with "The system of claim 13". As the claim otherwise has the same meaning, I incorporate here by reference my analysis of Claim 9 and conclude that Claim 21 is rendered obvious to a POSITA by Wright in view of Daswani.

W. Claim 22 is rendered obvious by Wright in view of Daswani

305. Claim 22 of the '517 patent reads:

The system of claim 13, wherein the runtime risk is identified based on matching rules and probabilistic weights and scores of performed actions on the device.

306. The language of this claim is identical to that of Claim 10, except that "The method of claim 1" is replaced with "The system of claim 13". As the claim otherwise has the same meaning, I incorporate here by reference my analysis of Claim 10 and conclude that Claim 22 is rendered obvious to a POSITA by Wright in view of Daswani.

X. Claim 23 is rendered obvious by Wright in view of Daswani

307. Claim 23 of the '517 patent recites:

> *The system of claim 13, wherein the runtime risk is identified based on time proximity of the at least two performed actions of the action sequence and natural affinity inferred between the at least two performed actions.*

308. The language of this claim is identical to that of Claim 11, except that "The method of claim 1" is replaced with "The system of claim 13". As the claim otherwise has the same meaning, I incorporate here by reference my analysis of Claim 11 and conclude that Claim 23 is rendered obvious to a POSITA by Wright in view of Daswani.

Y. Claim 24 is rendered obvious by Wright in view of Daswani

309. Claim 24 is rendered obvious to a POSITA by Wright in view of Daswani.

1. Limitation 24[a]

310. Limitation 24[a] of the '517 patent reads:

> *The system of claim 13, wherein the processing device assesses that the at least two performed actions of the action sequence describe a forensic chain of events based on a plurality of factors, and*

311. The language of this limitation is identical to that of limitation 12[a], with two exceptions. First, "The method of claim 1" is replaced with "The system of claim 13". As I explained with reference to limitation 13[a], Wright teaches both the claimed "method" of Claim 1 and the claimed "system" of Claim 13. Second, the language "***further comprising: assessing***,

by the processing device" is replaced with "***wherein*** the processing device ***assesses***". As these changes do not alter the meaning of the limitation, I incorporate herein by reference my analysis of limitation 12[a] and conclude that limitation 24[a] is disclosed by Wright.

2. Limitation 24[b]

312. Limitation 24[b] of the '517 patent reads:

> *the plurality of factors includes at least one of: ordered sequences, unordered sequences, proximity analysis, outliers, probabilistic weights and scores, and probable action inferences based on an application's competency.*

313. The language of this limitation is identical to that of limitation 12[b], except for the omission of the leading "wherein". As this change does not alter the meaning of the limitation, I incorporate herein by reference my analysis of limitation 12[b] and conclude that limitation 24[b] is disclosed by Wright.

314. Thus, Wright in view of Daswani renders Claim 24 obvious to a POSITA.

X. THERE ARE NO OBJECTIVE INDICIA OF NONOBVIOUSNESS

315. I am not aware of any objective indicia of non-obviousness related specifically to the '517 patent. Indeed, I am not aware of any evidence of commercial success, long felt but unsolved need, failure of others, industry praise, or unexpected results specifically related to the '517 patent.

XI. CONCLUSION

316. I declare that all statements made herein are true to the best of my knowledge and that all statements made on information and belief are believed to be true, and that this Declaration is made with 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.

Executed on 4 April, 2023

By: _____
Seth James Nielson