UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CROWDSTRIKE, INC.
Petitioner

v.

WEBROOT, INC.,
Patent Owner

Case IPR2023-01159
Patent No. 8,856,505

DECLARATION OF DR. MARKUS JAKOBSSON

TABLE OF CONTENTS

I. Introduction 7

II. Background and Qualifications 8

III. Level of Ordinary Skill in the Art 17

IV. Materials Considered and Relied Upon 17

V. Legal Standards 19

VI. Overview of the '505 Patent 26

A. Brief Description of the '505 Patent 26

B. File History of the '505 Patent 27

C. Interpretation of the '505 Patent Claims at Issue 28

VII. Overview of the Cited References 29

A. Field (EX1004) 29

B. Glick (EX1007) 33

C. Shukla (EX1005, EX1006) 34

D. An Obvious Combination of Field, Glick, and Shukla 37

1. A POSITA Would Have Found Incorporating Glick's and Shukla's User-Mode Components Into Field Obvious 38

2. A POSITA Would Have Found Incorporating Glick's and Shukla's Record Generation Into Field Obvious 41

3. A POSITA Would Have Found Incorporating Shukla's Use of Behavior Rules Into Field Obvious 43

4. A POSITA Would Have Found Incorporating Shukla's Registry Scanning Into Field Obvious 45

VIII. Overview of Unpatentability Grounds 46

IX. Ground 1: Field, Glick, and Shukla REnder Obvious Claims 1-20 46

A. Claim 1 46

1. [1pre] A method for managing pestware on a computer comprising:.....47
2. [1a] monitoring events during a boot sequence of the computer; 48
3. [1b] managing pestware-related events during a first period in a boot sequence of the computer, the first period in the boot sequence occurring before the computer becomes configured to run native applications, before a sub-system of an operating system is loaded, and after a kernel is loaded; 51
4. [1c] managing pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occurring when the computer is configured to run native applications; 55
5. [1d] generating, in response to the monitoring, a record of events, the record of events including the pestware-related events; 62
6. [1e] analyzing the record of events so as to identify the pestware-related events 66
7. [1f] modifying the set of behavior rules so as to prevent the pestware related events; and 68
8. 1[g] scanning data in a registry of the computer for pestware during the second period in the boot sequence.....69

B. Claim 2 71

1. 2[a] The method of claim 1, further comprising launching, after an operating system is initiated, a pestware management engine; and 71
2. 2[b] wherein the set of behavior rules includes behavior rules compiled by the pestware management engine. 74

C. Claim 3 76

D. Claim 4 77

E. Claim 5 81

F. Claim 6 83

1. 6[pre] A system for managing pestware on a computer comprising:....84
2. 6[a] a processor; 84
3. 6[b] a memory comprising:....85
4. 6[b.i] a first instruction set configured to monitor events during a boot sequence of the computer; 87
5. 6[b.ii] a second instruction set configured to manage pestware-related events during first period in a boot sequence of the computer, the first period in the boot sequence occurring before the computer becomes configured to run native applications, before a WIN32 subsystem is loaded, and after a kernel is loaded 88
6. 6[b.iii] a third instruction set configured to manage pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occurring when the computer is configured to run native applications; 92
7. 6[b.iv] a fourth instruction set configured to generate, in response to the monitoring, a record of events, the record of events including the pestware-related events; 92
8. 6[b.v] a fifth instruction set configured to analyze the record of events so as to identify the pestware-related events; and....92
9. 6[b.vi] a sixth instruction set configured to modify the set of behavior rules so as to prevent the pestware related events; and....92
10. 6[b.vii] a seventh instruction set configured to scan data in a registry of the computer for pestware during the second period in the boot sequence. 93

G. Claim 7 93

H. Claim 8 93

I. Claim 9 93

J. Claim 10 94

K. Claim 11 94

1. 11[pre] A non-transitory tangible computer-readable storage medium comprising program instructions for:.....95
2. 11[a] monitoring events during a boot sequence of a computer; 97
3. 11[b] managing pestware-related events during a first period in a boot sequence of the computer, the first period in the boot sequence occurring before the computer becomes configured to run native applications, before a WIN32 subsystem is loaded, and after a kernel is loaded; 98
4. 11[c] managing pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occurring when the computer is configured to run native applications; 98
5. 11[d] generating, in response to the monitoring, a record of events, the record of events including the pestware-related events; 98
6. 11[e] analyzing the record of events so as to identify the pestware-related events; 98
7. 11[f] modifying the set of behavior rules so as to prevent the pestware related events; and 98
8. 11[g] scanning data in a registry of the computer for pestware during the second period in the boot sequence.....99

L. Claim12 99

M. Claim 13 99

N. Claim 14 99

O. Claim 15 100

P. Claim 16 100

Q. Claim 17 103

R. Claim 18 104

S. Claim 19 105

T. Claim 20 105

X. Additional Remarks 107

I. INTRODUCTION

1. My name is Dr. Markus Jakobsson, and I have been retained by counsel for Petitioner CrowdStrike, Inc. ("CrowdStrike" or "Petitioner") as an expert witness to provide assistance regarding U.S. Patent No. 8,856,505 ("the '505 Patent"). Specifically, I have been asked to consider the patentability of claims 1-20 of the '505 Patent ("Challenged Claims") in view of prior art, anticipation and obviousness considerations, and the understanding of a person of ordinary skill in the art at the time of the invention ("POSITA") as it relates to the '505 Patent.

2. I am being compensated for my time at my standard consulting rate. I am also being reimbursed for expenses that I incur during the course of this work. My compensation is not contingent upon the results of my study, the substance of my opinions, or the outcome of any proceeding involving the Challenged Claims. I have no financial interest in the outcome of this matter or on the pending litigation between Petitioner and Patent Owner.

3. My analysis here is based on my years of education, research and experience, as well as my investigation and study of relevant materials, including those cited herein.

4. I may rely upon these materials, my knowledge and experience, and/or additional materials to rebut arguments raised by the Patent Owner. Further, I may

also consider additional documents and information in forming any necessary opinions, including documents that may not yet have been provided to me.

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

II. BACKGROUND AND QUALIFICATIONS

6. I have summarized in this section my educational background, career history, and other qualifications relevant to this matter. I have also included a current version of my curriculum vitae, which is attached as Appendix A.

7. I received a Master of Science degree in Computer Engineering from the Lund Institute of Technology in Sweden in 1993, a Master of Science degree in Computer Science from the University of California at San Diego in 1994, and a Ph.D. in Computer Science from the University of California at San Diego in 1997, specializing in Cryptography. During and after my Ph.D. studies, I was also a researcher at the San Diego Supercomputer Center and General Atomics, where I did research on authentication, privacy, and security. Much of my work was on distributed protocols to enable security and privacy, and my Ph.D. thesis described a distributed electronic payment system. Whereas the term “cloud computing” had

not been popularized at this point, many of the principles I studied later became critical aspects of cloud computing and virtualization.

8. Since earning my doctorate degree over twenty-six years ago, I have been an employee, entrepreneur, and consultant in the computer systems, security, and cloud computing industry. I have worked extensively with technologies related to computer security, mobile security, malware detection, quantitative and qualitative fraud analysis, and disruptive security. I have worked as an engineer or technical advisor at a number of leading companies in the computer industry, including PayPal, Qualcomm, Bell Labs, and LifeLock. I have also taught and performed research at several prestigious universities, including New York University (NYU) and Indiana University (IU). Further, I have been materially involved in designing, developing, testing, and assessing technologies for computer and network security, digital rights management, trust establishment, randomness, cryptography, socio-technical fraud, malware detection, detection of malicious emails, user interfaces, authentication, and fraud detection for over twenty years.

9. From 1997 to 2001, I was a Member of Technical Staff at Bell Labs, where I did research on authentication, privacy, multi-party computation, contract exchange, digital commerce including crypto payments, and fraud detection and prevention. Example publications of mine from this time period include “Secure distributed computation in cryptographic applications” (U.S. Patent No. 6,950,937,

2001 priority date) and "Secure server-aided signature generation" (International Workshop on Public Key Cryptography PKC 2001: Public Key Cryptography 383-401), both of which relate to virtualization. At the end of my tenure at Bell Labs, I had started to research identity theft, years before the banking industry became aware of the existence of phishing attacks.

10. From 2001 to 2004, I was a Principal Research Scientist at RSA Labs, where I worked on predicting future fraud scenarios in commerce and authentication and developed solutions to those problems. Much of my work related to the development of anti-fraud mechanisms, addressing authentication abuse and identity theft. During that time, I predicted the rise of what later became known as phishing. I also laid the groundwork for what I termed "proof of work," and described how this could be applied in the context of distributed computations such as mining of crypto payments; this work later came to influence the development of BitCoin. I was also an Adjunct Associate Professor in the Computer Science department at New York University from 2002 to 2004, where I taught cryptographic protocols, with an emphasis on authentication.

11. From 2004 to 2016, I held a faculty position at Indiana University at Bloomington, first as an Associate Professor of Computer Science, Associate Professor of Informatics, Associate Professor of Cognitive Science, and Associate Director of the Center for Applied Cybersecurity Research (CACR) from 2004 to

2008; and then as an Adjunct Associate Professor from 2008 to 2016. I was the most senior security researcher at Indiana University, where I built a research group focused on online fraud and countermeasures, resulting in over 50 publications and two books. One of these books, "Crimeware: Understanding New Attacks and Defenses" (Wiley, 2008), described man-in-the-middle attacks, online abuse in general, and countermeasures to attacks and abuses of these types.

12. While a professor at Indiana University, I was also employed by Xerox PARC, PayPal, and Qualcomm to provide thought leadership to their security groups. I was a Principal Scientist at Xerox PARC from 2008 to 2010, a Director and Principal Scientist of Consumer Security at PayPal from 2010 to 2013, a Senior Director at Qualcomm from 2013 to 2015, Chief Scientist at Agari from 2016 to 2018, and Chief of Security and Data Analytics at Amber Solutions from 2018 to 2020.

13. Agari is a cybersecurity company that develops and commercializes technology to protect enterprises, their partners and customers from advanced email phishing attacks. At Agari, my research studied and addressed trends in online fraud, especially as related to email, including problems such as Business Email Compromise, Ransomware, and other abuses based on social engineering and identity deception. My work primarily involved identifying trends in fraud and computing before they affected the market, and developing and testing

countermeasures, including technological countermeasures, user interaction and education. Example publications include "Detecting computer security risk based on previously observed communications" (U.S. Patent No. 10,715,543 B2, 2016 priority date) and "Using message context to evaluate security of requested data" (U.S. Patent No. 10,805,314 B2, 2017 priority date).

14. In 2020, after leaving Agari, I was the Chief of Security and Data Analytics at Amber Solutions, an IoT startup in the San Francisco Bay Area. Amber Solutions is a cybersecurity company that develops home and office automation technologies. At Amber Solutions, my research addressed privacy, user interfaces and authentication techniques in the context of ubiquitous and wearable computing. My work often related to authentication and cloud computing, as exemplified in "Configuration and management of smart nodes with limited user interfaces" (U.S. Patent No. 10,887,447).

15. I left Amber Solutions to take the role of Chief Scientist at ByteDance in 2019, where I guided the security research both for ByteDance and TikTok until I left to co-found Artema Labs, where I am the Chief Scientist. My work at ByteDance involved identifying new threats, developing new solutions, and guiding research in areas relating to cryptography and Internet security.

16. I am also the Chief Technology Officer at ZapFraud, a cybersecurity company that develops techniques to detect deceptive emails, such as Business

Email Compromise ("BEC") emails. At ZapFraud, my research studies and addresses computer abuse, including social engineering, malware and privacy intrusions, and has also involved studying security aspects related to cloud computing. My work primarily involves identifying risks, developing protocols and user experiences, and evaluating the security of proposed approaches. Using virtual machines is an important tool for security researchers, as it allows them to study the impact of a given executable, website, or other context on a computational environment. My students, direct reports and I have often relied on this approach in the context of interacting with online criminals, their websites and emails they send, as virtual machines enable us to identify their motives and report them to the authorities without making us easy for them to track or identify. Doing this is important to identify new and trending attacks.

17. I also work as an independent security researcher and consultant in the fields of computer/mobile security, fraud detection/prevention, digital rights management, malware, phishing, crimeware, mobile malware, and cryptographic protocols. In addition, I am a visiting research fellow of the Anti-Phishing Working Group, an international consortium focused on unifying the global response to cybercrime in the public and private sectors.

18. I have also served on the advisory board of several companies, including Metaforic, a company that developed technology for watermarking

software and provided security software with a built-in framework that resists attacks from malware, bots, hackers and other attempts. Metaforic was acquired by Inside Secure in 2014.

19. I have founded or co-founded several successful computer security companies. In 2005 I co-founded RavenWhite Security, a provider of authentication solutions, and I am currently its Chief Technical Officer. RavenWhite owns intellectual property related to authentication, such as "Performing authentication" (U.S. Patent No. 10,079,823 2007 priority date). In 2007 I co-founded Extricatus, one of the first companies to address consumer security education. In 2009 I founded FatSkunk, a provider of mobile malware detection software; I served as Chief Technical Officer of FatSkunk from 2009 to 2013, when FatSkunk was acquired by Qualcomm and I became a Qualcomm employee. FatSkunk developed anti-virus technology. In 2013 I founded ZapFraud—the same company described above, where I currently serve as Chief Technical Officer. In 2014 I co-founded RightQuestion, a security consulting company.

20. I have additionally served as a member of the fraud advisory board at LifeLock (an identity theft protection company); a member of the technical advisory board at CellFony (a mobile security company); a member of the technical advisory board at PopGiro (a user reputation company); a member of the technical advisory board at MobiSocial dba Omlet (a social networking company); and a member of

the technical advisory board at Stealth Security (an anti-fraud company). I have provided anti-fraud consulting to KommuneData (a Danish government entity), J.P. Morgan Chase, PayPal, Boku, and Western Union.

21. I have authored seven books and over 100 peer-reviewed publications, and have been a named inventor on over 200 patents and patent applications. A large number of these relate to Internet security, authentication, malware, spoofing and related topics.

22. Examples of textbooks I have authored or co-authored on the topic of Internet security include:

- "Crimeware: Understanding New Attacks and Defenses" (Symantec Press, 2008)
- "Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft" (Wiley, 2006);
- "The Death of the Internet" (Wiley, 2012);
- "Understanding Social Engineering Based Scams" (Springer 2016); and
- "Towards Trustworthy Elections: New Directions in Electronic Voting" (Springer, 2010)

23. I have authored numerous publications regarding Internet security, including Internet security as it pertains to web browsers. Relevant examples include:

- "Web Camouflage: Protecting Your Clients from Browser-Sniffing Attacks," 5 IEEE Security & Privacy (Nov. 2007)
- "Invasive Browser Sniffing and Countermeasures," Conference: 15th International Conference on World Wide Web (May 2006)
- "Remote Harm-Diagnostics", (PDF) Privacy-preserving history mining for web browsers (researchgate.net)
- "Privacy-Preserving History Mining for Web Browsers", (PDF) Privacy preserving history mining for web browsers (researchgate.net)
- "Server-Side Detection of Malware Infection", NSPW Proceedings on New Security Paradigms Workshop (2009), Server-side detection of malware infection | Proceedings of the 2009 workshop on New Security Paradigms Workshop (acm.org)

24. Related patent publications include U.S. Patent Nos. 10,498,735, 9,203,835, 8,578,174. These patents were assigned to PayPal, Inc. and Palo Alto Research Center Incorporated arising out of my employment.

25. In sum, I have extensive experience both as a developer and a researcher relating to system monitoring, software security, attack detection, and network security.

III. LEVEL OF ORDINARY SKILL IN THE ART

26. In my opinion, a person of ordinary skill in the art ("POSITA") relating to, and at the time of, the '505 Patent's Critical Date (i.e., August 7, 2006) would have had a bachelor's degree in computer science, computer engineering, or an equivalent, as well as two years of industry experience and would have had a working knowledge of endpoint monitoring systems, software security analysis, and dynamic malware analysis. Additional education in a relevant field, or relevant industry experience may compensate for a deficit in one of the other aspects of the requirements stated above.

27. Based on my knowledge, skill, and experience, I have a good understanding of the capabilities of the POSITA. Indeed, I have taught, participated in organizations with, hired, and worked closely with many such persons over the course of my career. For example, from my industry experience, I am familiar with what an engineer would have known and found predictable in the art. From teaching and supervising post-graduate students, I also have an understanding of the knowledge that a person with this academic experience possesses. Furthermore, I possess those capabilities myself.

IV. MATERIALS CONSIDERED AND RELIED UPON

28. In reaching the conclusions described in this declaration, I have relied on the documents and materials cited herein as well as those identified in this

declaration, including the '505 Patent, the prosecution history of the '505 Patent, and prior art references cited herein. These materials comprise patents, related documents, and printed publications. Each of these materials is a type of document that experts in my field would have reasonably relied upon when forming their opinions.

29. I have also relied on my education, training, research, knowledge, and personal and professional experience in the relevant technologies and systems that were already in use prior to and within the timeframe of the earliest asserted priority date of the claimed subject matter in the '505 Patent, which I have been informed by counsel is August 7, 2006.

- EX1001: U.S. Patent No. 8,856,505 ("the '505 Patent)
- EX1002: Excerpts from the Prosecution History of the '505 Patent ("the Prosecution History")
- EX1004: U.S. Patent Pub. No. 2008/0005797 to Scott A. Field *et al.* ("Field")
- EX1005: U.S. Patent Pub. No. 2008/0016339 to Jayant Shukla ("Shukla")
- EX1006: U.S. Provisional Application No. 60/806,143 ("the '143 Application")
- EX1007: U.S. Patent No. 8,239,947 to Adam Glick *et al.* ("Glick")
- EX1009: U.S. Patent No. 7,784,098 to Paul Fan *et al.* ("Fan")
- EX1010: U.S. Patent No. 7,673,341 to Michael Kramer *et al.* ("Kramer")
- EX1011: Silberschatz, Operating System Concepts (2004)

V. LEGAL STANDARDS

30. I am a technical expert and do not offer any legal opinions. However, counsel has informed me as to certain legal principles regarding patentability and related matters under United States patent law, which I have applied in performing my analysis and arriving at my technical opinions in this matter.

31. I have been informed that claim terms are to be given the meaning they would have to a person having ordinary skill in the art at the time of the invention, taking into consideration the patent, its file history, and, secondarily, any applicable extrinsic evidence (e.g., dictionary definitions).

32. I have also been informed that the implicit or inherent disclosures of a prior art reference may anticipate the claimed invention. Specifically, if a person having ordinary skill in the art at the time of the invention would have known that the claimed subject matter is necessarily present in a prior art reference, then the prior art reference may "anticipate" the claim. Therefore, a claim is "anticipated" by the prior art if each and every limitation of the claim is found, either expressly or inherently, in a single item of prior art.

33. I have also been informed that a person cannot obtain a patent on an invention if the differences between the invention and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art. I have been informed that a

conclusion of obviousness may be founded upon more than a single item of prior art. I have been further informed that obviousness is determined by evaluating the following factors: (1) the scope and content of the prior art, (2) the differences between the prior art and the claim at issue, (3) the level of ordinary skill in the pertinent art, and (4) secondary considerations of non-obviousness. In addition, the obviousness inquiry should not be done in hindsight. Instead, the obviousness inquiry should be done through the eyes of a person having ordinary skill in the relevant art at the time the patent was filed.

34. In considering whether certain prior art renders a particular patent claim obvious, counsel has informed me that I can consider the scope and content of the prior art, including the fact that one of skill in the art would regularly look to the disclosures in patents, trade publications, journal articles, industry standards, product literature and documentation, texts describing competitive technologies, requests for comment published by standard setting organizations, and materials from industry conferences, as examples. I have been informed that for a prior art reference to be proper for use in an obviousness analysis, the reference must be "analogous art" to the claimed invention. I have been informed that a reference is analogous art to the claimed invention if: (1) the reference is from the same field of endeavor as the claimed invention (even if it addresses a different problem); or (2) the reference is reasonably pertinent to the problem faced by the inventor (even if it

is not in the same field of endeavor as the claimed invention). In order for a reference to be "reasonably pertinent" to the problem, it must logically have commended itself to an inventor's attention in considering his problem. In determining whether a reference is reasonably pertinent, one should consider the problem faced by the inventor, as reflected either explicitly or implicitly, in the specification. I believe that all of the references that my opinions in this IPR are based upon are well within the range of references a person having ordinary skill in the art would consult to address the type of problems described in the Challenged Claims.

35. I have been informed that, in order to establish that a claimed invention was obvious based on a combination of prior art elements, a clear articulation of the reason(s) why a claimed invention would have been obvious must be provided. Specifically, I am informed that a combination of multiple items of prior art renders a patent claim obvious when there was an apparent reason for one of ordinary skill in the art, at the time of the invention, to combine the prior art, which can include, but is not limited to, any of the following rationales: (A) combining prior art methods according to known methods to yield predictable results; (B) substituting one known element for another to obtain predictable results; (C) using a known technique to improve a similar device in the same way; (D) applying a known technique to a known device ready for improvement to yield predictable results; (E) trying a finite number of identified, predictable potential solutions, with a reasonable expectation

of success; (F) identifying that known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations are predictable to one of ordinary skill in the art; or (G) identifying an explicit teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine the prior art references to arrive at the claimed invention.

36. I am informed that the existence of an explicit teaching, suggestion, or motivation to combine known elements of the prior art is a sufficient, but not a necessary, condition to a finding of obviousness. This so-called "teaching suggestion-motivation" test is not the exclusive test and is not to be applied rigidly in an obviousness analysis. In determining whether the subject matter of a patent claim is obvious, neither the particular motivation nor the avowed purpose of the patentee controls. Instead, the important consideration is the objective reach of the claim. In other words, if the claim extends to what is obvious, then the claim is invalid. I am further informed that the obviousness analysis often necessitates consideration of the interrelated teachings of multiple patents, the effects of demands known to the technological community or present in the marketplace, and the background knowledge possessed by a person having ordinary skill in the art. All of these issues may be considered to determine whether there was an apparent reason to combine the known elements in the fashion claimed by the patent.

37. I also am informed that in conducting an obviousness analysis, a precise teaching directed to the specific subject matter of the challenged claim need not be sought out because it is appropriate to take account of the inferences and creative steps that a person of ordinary skill in the art would employ. The prior art considered can be directed to any need or problem known in the field of endeavor at the time of invention and can provide a reason for combining the elements of the prior art in the manner claimed. In other words, the prior art need not be directed towards solving the same specific problem as the problem addressed by the patent. Further, the individual prior art references themselves need not all be directed towards solving the same problem. I am informed that common sense is important and should be considered. Common sense teaches that familiar items may have obvious uses beyond their primary purposes.

38. I also am informed that the fact that a particular combination of prior art elements was "obvious to try" may indicate that the combination was obvious even if no one attempted the combination. If the combination was obvious to try (regardless of whether it was actually tried) or leads to anticipated success, then it is likely the result of ordinary skill and common sense rather than innovation. I am further informed that in many fields it may be that there is little discussion of obvious techniques or combinations, and it often may be the case that market demand, rather than scientific literature or knowledge, will drive the design of an invention. I am

informed that an invention that is a combination of prior art must do more than yield predictable results to be non-obvious.

39. I am informed that for a patent claim to be obvious, the claim must be obvious to a person of ordinary skill in the art at the time of the invention. I am informed that the factors to consider in determining the level of ordinary skill in the art include (1) the educational level and experience of people working in the field at the time the invention was made, (2) the types of problems faced in the art and the solutions found to those problems, and (3) the sophistication of the technology in the field.

40. I am informed that it is improper to combine references where the references teach away from their combination. I am informed that a reference may be said to teach away when a person of ordinary skill in the relevant art, upon reading the reference, would be discouraged from following the path set out in the reference, or would be led in a direction divergent from the path that was taken by the patent applicant. In general, a reference will teach away if it suggests that the line of development flowing from the reference's disclosure is unlikely to be productive of the result sought by the patentee. I am informed that a reference teaches away, for example, if (1) the combination would produce a seemingly inoperative device, or (2) the references leave the impression that the product would not have the property sought by the patentee. I also am informed, however, that a reference does not teach

away if it merely expresses a general preference for an alternative invention but does not criticize, discredit, or otherwise discourage investigation into the invention claimed.

41. I am informed that the final determination of obviousness must also consider "secondary considerations" if presented. In most instances, the patentee raises these secondary considerations of non-obviousness. In that context, the patentee argues an invention would not have been obvious in view of these considerations, which include: (a) commercial success of a product due to the merits of the claimed invention; (b) a long-felt, but unsatisfied need for the invention; (c) failure of others to find the solution provided by the claimed invention; (d) deliberate copying of the invention by others; (e) unexpected results achieved by the invention; (f) praise of the invention by others skilled in the art; (g) lack of independent simultaneous invention within a comparatively short space of time; (h) teaching away from the invention in the prior art.

42. I am further informed that secondary considerations evidence is only relevant if the offering party establishes a connection, or nexus, between the evidence and the claimed invention. The nexus cannot be based on prior art features. The establishment of a nexus is a question of fact. While I understand that the Patent Owner here has not offered any secondary considerations at this time, I will

supplement my opinions if the Patent Owner raises secondary considerations during this proceeding.

VI. OVERVIEW OF THE '505 PATENT

A. Brief Description of the '505 Patent

43. The '505 Patent describes "a system and method for managing pestware on a protected computer" including "monitoring events during a boot sequence of the computer; managing pestware-related events before native applications can run and after a kernel is loaded; managing pestware-related events when native applications can run; and scanning a registry of the computer for pestware when native applications can run." EX1001, Abstract. The pestware-related events are monitored by comparing log activity to a set of behavioral rules. *Id*.

44. With reference to FIG. 3, the '505 Patent describes that previous pestware detection and prevention methods operated as "user-mode services or applications," which are loaded relatively late in the boot sequence. *Id.*, 1:61-65. This allows pestware to begin execution earlier in the boot sequence and employ "cloaking techniques" to avoid detection by prevention methods that only employ user-mode services or applications. *See id.*, 1:46-67. Thus, the '505 Patent proposes the implementation of a "kernel level monitor" that "begins execution early in the boot and operating system loading process." *Id.*, 3:24-31.

45. However, as I will discuss in greater detail in this declaration, this method of loading pestware monitoring processes early in the boot sequence was not new at the time of the alleged invention. Several prior art references, including those cited in this petition, identified the same problem and proposed the same or similar solutions to those claimed the '505 Patent.

B. File History of the '505 Patent

46. The '505 Patent issued from U.S. Patent App. No. 13/460,655 (the '655 application). The '655 application is an April 30, 2012 continuation of U.S. Patent App. No. 11/462,827, filed on Aug. 7, 2006, now Pat. No. 8,190,868. The independent claims of the '655 application were initially rejected over Zimmer (U.S. Pub. No. 2005/0114687) in view of Rothwell (WO 2006/077443) and Rothman (U.S. Pub. No. 2005/0216759); and were subject to a double patenting rejection over claims of U.S. Patent No. 8,190,868. EX1002, 159-173.

47. After a telephonic examiner interview, the applicant filed a terminal disclaimer to overcome the double patenting rejection. *Id.*, 134. The applicant also argued that the examiner's cited references failed "to teach the combination of features as claimed". *Id*., 147-148. In particular, the applicant alleged that none of the cited prior art taught "managing pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occurring when the computer is configured to run native applications; and scanning data in a registry of

the computer for pestware during the second period in the boot sequence." *Id*., 150. A Notice of Allowance was issued on Apr 29, 2014 stating that applicant's arguments were persuasive. *Id.*, 94.

C. Interpretation of the '505 Patent Claims at Issue

48. I understand that the terms that appear in the claims of the '505 Patent should be interpreted according to their ordinary and customary meaning in light of the specification (EX1001) and prosecution file history (EX1002). In determining the ordinary and customary meaning, I understand that the words of a claim are first given their plain meaning that those words would have had to one of ordinary skill at the time of the alleged invention. I also understand that the structure of the claims, the specification and file history also may be used to better construe a claim insofar as the plain meaning of the claims cannot be understood. Moreover, I understand that even treatises and dictionaries may be used, albeit under limited circumstances, to determine the meaning attributed by one of ordinary skill to a claim term at the time of filing.

49. I understand that in parallel litigation, Patent Owner contends that no claim language requires express construction. For the purposes of my analysis, I have been instructed to apply the plain and ordinary meaning to the claim terms at issue herein unless noted otherwise. For these terms, based on the prior art's description of the claimed elements being similar to that of the '505 Patent

specification, I understand that no formal claim constructions are necessary for these terms in this proceeding because claim terms only need to be construed to the extent necessary to resolve a controversy.

50. I also understand that the words of the claims should be interpreted as they would have been interpreted by one of ordinary skill at the time the invention was made (not today). For purposes of my analysis here, I have used the August 7, 2006 purported priority date of the '505 Patent as the date of invention (i.e., the Critical Date). Without exception, however, my analysis of the proper meaning of the recited claim elements in this declaration would be correct if the date of invention was earlier or later than 2006.

VII. OVERVIEW OF THE CITED REFERENCES

51. The general descriptions that I have provided for the following references and obvious modifications thereof are incorporated into each subsection and mapping of the claims that includes citations to these references.

A. Field (EX1004)

52. Field describes a method "that performs a search for malware during the boot process" and "handles the malware infection." EX1004, Abstract, 1:7-9. Specifically, the method "causes a software module configured to scan for malware to be initialized at startup. Then, in response to identifying the occurrence of a scanning event, the method causes the software module to search computer memory

for data that is characteristic of malware." *Id*. at [0007]. If the scan does find data characteristic of malware, "the functionality is implemented to prevent malware from executing on the computer." *Id*.

53. Like the '505 Patent, Field recognized that "malware is being distributed with one or more programs specifically designed to 'hide' the malware from software designed to protect a computer (e.g., antivirus software, anti-spyware software, and the like)." *Id.*, [0004]. To do so, Field similarly recognized that malware may hook into the operating system and operate "in kernel mode," and may thus disguise itself from monitoring applications operating in user mode. *See id.*, [0005]. Thus, like the '505 Patent, Field proposes loading the monitoring program early in the boot sequence to identify and prevent these types of attacks. *See id.*, [0007].

54. Figure 3 in Field shows a timeline of the events that are performed at computer startup.

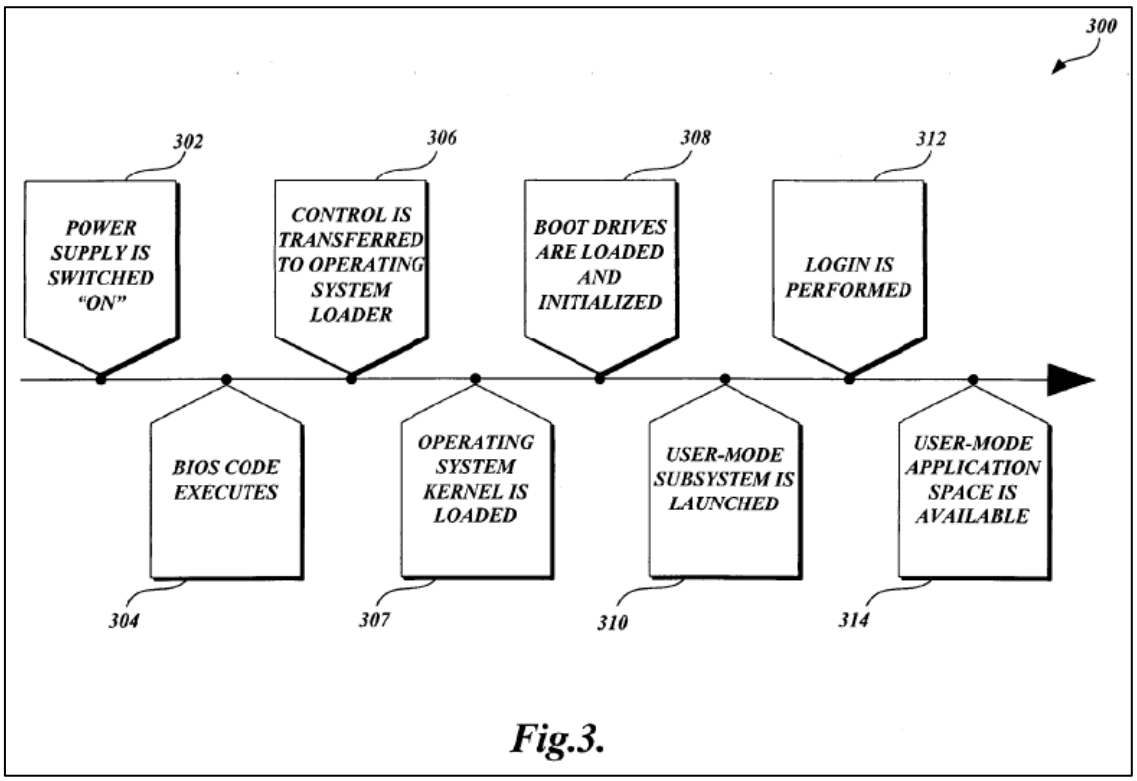


Fig.3.

55. Field explains that the "software module may be loaded into memory" at multiple points of the timeline, and may begin executing at events 304, 306, or 310 of the timeline. *Id*. at [0023]. As the captions on Figure 3 show, events 304 and 306 take place before a subsystem of the operating system is loaded (which takes place in event 307). At or after event 310, the computer is configured to run native applications. *Id.* It would not be meaningful to launch at 304 or 306 if the process only observes application-level calls, as application-level calls only happen after event 310. Instead, the reason to launch at 304 or 306 is that Field's process observes kernel-level calls, and aims to detect pestware as the computer boots by having the "antivirus software" "execut[ing] once the computer boots." To identify pestware during these periods, the scan engine may search for, for example, "***activities*** such

as jump instructions in unexpected locations, hidden processes, references to memory addresses that are outside of the range allocated to the operating system, and the like." EX1004, [0017][1].

56. If malware is identified, "the malware may be removed from the computer or the malware may be 'quarantined'". EX1004, [0027]. Quarantining may mean using a "stub" module that is "configured to sustain a malware process, accept and/or return valid data in response to being called, or perform any action to prevent triggering of a malware self-preservation technique." *Id.* at [0034]. In this way, Field teaches managing or preventing the pestware related event. Other ways of handling the malware "includes attempting to remove the malware from the computer by killing processes, deleting files, removing entries in configuration files that are associated with the malware, and the like." *Id*. at [0034].

57. Field describes that "the scan engine may employ traditional signature-based techniques in a boot environment to detect malware" and that "[o]nce a signature for the malware is available, the scan engine may search data in the memory for a match." *Id*. at [0017]. In other words, Field suggests the analysis of stored records to identify pestware. However, Field leaves the details of this process to a POSITA by noting that its approach utilizes "traditional . . . techniques." *Id.*

[1] Emphasis added throughout unless noted otherwise.

Further, Field does not appear to contain any suggestion that would limit the ways a POSITA might determine to best implement or supplement its system to further its goal of "identifying malware that uses program code activated in a boot environment". EX1004, [0007]. To the contrary, Field notes that "it will be appreciated that various changes can be made" to the operation of its system "without departing from the spirit and scope of the invention." EX1004, [0036]. From these teachings, a POSITA would have been motivated to turn to known techniques (demonstrated, *e.g.*, by Glick and Shukla) to inform implementation details of Field's system and method.

B. Glick (EX1007)

58. Glick teaches a method "for the detection and removal of threats" which uses a combination of user-mode and kernel-mode detection to identify malware. EX1007 at Title, Abstract; *see also id.*, 3:32-39, 4:11-18. Glick teaches code that "is installed on [a] host computer system and includes [a] user mode application that operates from user mode and [a] kernel mode driver that operates from kernel mode." *Id.* at 4:11-18. Like Field, Glick describes that the kernel mode portion of its program "is loaded prior to user mode code in the boot process of host computer system 102 to protect privileges being removed by a threat." *Id.*, 6:23-32.

59. Glick provides various implementation details that facilitate communication and coordination between a user mode application and a kernel

mode driver to allow for a more capable antivirus protection method. *See id.*, 3:21-52. As part of the capabilities of a kernel mode driver, Glick recognizes the importance of monitoring the registry, teaching that the "kernel mode driver locates the system privileges in the key descriptor table of the registry and resets the privilege flag to allow [the] user mode application to use the debug privilege of the administrator account." *Id*. at 7:14-19. Moreover, "in one embodiment, [the] kernel mode driver monitors the registry on the host computer system and prevents changes to the registry that prevent a listed file from being deleted on reboot." *Id*. at 12:37-41.

60. Furthermore, Glick teaches the importance of reporting significant events. Specifically, when malicious code is detected, "a record or error message is generated" which, "in one embodiment… identifies the particular malicious code and/or any protective action taken." *Id*. at 8:13-20.

C. Shukla (EX1005, EX1006)

61. Shukla was filed on June 27, 2007, published on Jan 17, 2008, and claims priority to U.S. Provisional Application 60/806,143 (the '143 provisional application) which was filed on Jun 29, 2006. The '143 provisional application provides a nearly identical specification to Shukla's specification. *Compare* EX1005 *to* EX1006. To illustrate, some paragraphs from the "summary of the invention" section of both Shukla and the '143 application are recreated below.

'143 Application p. 4	**Shukla pp. 14-15**
SUMMARY OF THE INVENTION The present invention provides a new system and method for removing malware and protecting applications against attacks that exploit application vulnerabilities or loopholes to gain unauthorized access into a network or computing device. This approach is significantly different from traditional approaches for intrusion prevention and is "application centric." In one embodiment of the present invention secures application or any other module in memory with a sandbox that allows that module or application to function properly while preventing attacks from succeeding and causing any harm. For purposes of clarity the term sandbox as used herein means a method to restrict actions taken by an application or a module. The sandbox for any application is defined by API function calls that determine its interaction with the operating system and are needed by application to function and by other parameters that can restrict any action by the application that can cause irreversible harm or even temporary malfunction. Applications protected by sandbox are able to thwart attacks that exploit vulnerabilities in that application to gain unauthorized access. The sandbox itself is elastic that is elastic in nature and morphs based on external parameters such as the website or IP address visited by the user, external components installed inside the application, and interaction of the application with other applications on the system.	SUMMARY OF THE INVENTION **[0011]** The present invention provides a new system and method for removing malware and protecting applications against attacks that exploit application vulnerabilities or loopholes to gain unauthorized access into a network or computing device. This approach is significantly different from traditional approaches for intrusion prevention and is "application centric." **[0012]** In one embodiment of the present invention secures application or any other module in memory with a sandbox that allows that module or application to function properly while preventing attacks from succeeding and causing any harm. For purposes of clarity the term sandbox as used herein means a method to restrict actions taken by an application or a module. **[0013]** The sandbox for any application is defined by API function calls that determine its interaction with the operating system and are needed by application to function and by other parameters that can restrict any action by the application that can cause irreversible harm or even temporary malfunction. Applications protected by sandbox are able to thwart attacks that exploit vulnerabilities in that application to gain unauthorized access. **[0014]** The sandbox itself is elastic that is elastic in nature and morphs based on external parameters such as the website or IP address visited by the user, external components installed inside the application, and interaction of the application with other applications on the system.

62. Because the specifications in Shukla and the '143 applications are nearly identical, the disclosure in the '143 application supports at least one of Shukla's claims. In fact, claim 1 of the '143 application (EX1006, p. 31, claim 1) is nearly identical to claim 1 of Shukla (EX1005, pp. 22-23, claim 1), meaning there is literal, word-for-word support for Shukla's claim 1 in the '143 application. Nonetheless, the following table further identifies exemplary support in the '143 Provisional application for each limitation of Shukla's independent claim 1:

Claim Limitation (Shukla)	Exemplary support ('143 Provisional)
1. A method for monitoring behavior of plurality of applications or modules in a computing device, comprising the steps of:	EX1006, 9
injecting a module into the memory space of the said applications;	EX1006, 9-10, Figure

	4
the injected module monitoring said applications' file system accesses by intercepting API function calls via imported or exported functions table patching and inline hooking of functions at the application layer;	EX1006, 9-10, 12
the injected module monitoring said applications' network accesses by intercepting API function calls via imported or exported functions table patching and inline hooking of functions at the application layer;	EX1006, 9-10, 12
the injected module monitoring said applications' executable content loading by intercepting API function calls via imported or exported functions table patching and inline hooking of functions at the application layer;	EX1006, 10-11
the injected module monitoring the memory access by the applications via inline hooks in API function call and the application programming interface functions provided;	EX1006, 10-11
and the injected module monitoring the registry access by the applications via inline hooks in API function call and the application programming interface functions provided.	EX1006, 10-11

63. More generally, Shukla teaches a sandbox for detecting, removing, and preventing malware. EX1005, Title, Abstract; EX1006, 1. The sandbox is constructed "by loading a module or executable code into the application" and, if necessary, "another component may be loaded into the kernel." EX1005, [0035]; EX1006, 9. The sandbox is constructed by hooking into application programing interface (API) function calls that "can be either user mode or kernel mode functions and the hooking can be done anywhere along the entire chain for the API function call." EX1005, [0039]; EX1006, 10. Thus, like Field and Glick, Shukla describes the use of both kernel mode and user mode code and techniques to identify and prevent malware.

64. Shukla teaches the importance of using behavioral rules, in addition to scanning for signature patterns, describing that "[t]he malware is detected based on

its behavior, signature patterns, and code authentication." EX1005, [0015]; EX1006, 4. "The behavior classification is based on actions taken by applications such as network access, file and registry modifications, API function interception, creation of executable content, stealth behavior, modification of system calls, etc." EX1005, [0015]; EX1006, 4-5. The behavioral rules can be modified to suit the situation: "[t]he sandbox boundaries themselves are not static and can be tightened or relaxed based on operational parameters…" and "based on which component or module inside the application is executing." EX1005, [0033]; EX1006, 9. Thus, Shukla provides a more comprehensive method of identifying malware that adds techniques beyond scanning for known signatures.

65. Shukla teaches that "[o]nce an application or a module is classified as malicious, its one disk, in registry, and in memory attributes are recorded and used to neutralized [sic] and remove it." EX1005, [0063]; EX1006, 14.

D. An Obvious Combination of Field, Glick, and Shukla

66. In the Field, Glick, and Shukla combination, a POSITA would have found it obvious that Glick's user-mode component(s) and record generation and Shukla's behavioral rules and registry scanning would have been incorporated into Field's method of monitoring and managing malware during the boot sequence. A POSITA would have been motivated to implement the Field, Glick, and Shukla combination for several reasons.

1. A POSITA Would Have Found Incorporating Glick's and Shukla's User-Mode Components Into Field Obvious

67. A POSITA would have been motivated to implement Glick's and Shukla's user-mode/application-level operation into the invention disclosed in Field. Field teaches that "resources available in a boot environment are limited," and therefore "successfully removing all components of a malware . . . may be difficult or impossible." EX1004, [0034]. Thus, Field teaches that its boot detection module may quarantine the malware, and data about the malware "will typically be communicated to antivirus software that executes once the computer boots so that all of the components of the malware may be removed." *Id.* The fact that Field describes the "antivirus software" as "execut[ing] once the computer boots" suggests that the "antivirus software" described by Field is a user-mode application that communicates with the kernel-mode components provided by Field's boot detection module.[2] Yet, even though Field does not explicitly describe the "antivirus software" as a user-mode application, Glick also renders this obvious and provides

[2] Notably, it is not uncommon for traditional antivirus and anti-malware services to include both kernel-mode and user-mode components that communicate with one another, consistent, for example, with the system described in Glick. However, it appears that Field refers to the "antivirus software" as the user-mode component of the system.

the implementation details for performing the beneficial functions identified by Field.

68. Glick describes that "user mode application 106A includes code that detects malicious code, such as spyware and computer viruses, from user mode and when appropriate takes protective action when malicious code is detected." EX1007, 3:32-35. Thus, Glick's user mode application 106A operates like the antivirus software[3] described by Field (i.e., user-mode protection software that loads up toward the end of the boot sequence and is capable of communicating with the kernel-mode software). *See* EX1004, [0034]. "[W]hen user mode application 106A is unable to perform a selected operation in attempting to detect and/or take protective action, user mode application 106A invokes kernel mode driver 106B

[3] Notably, the '505 Patent illustrates in FIGS. 3, 4, and 6, that "Anti-Virus & Anti-Spyware as Services" are started during the period where the Win32 subsystem is loaded and executed. Though the '505 Patent does not explicitly refer to these services in the specification, a POSITA would have understood in the context of the '505 Patent that these are or are at least akin to the user-mode services that make up the pestware management engine 110, which communicates with the kernel-level monitor 105 "so as to retrieve the event logs for examination and analysis." EX1001, 5:29-37.

assistance." EX1007, 3:36-39. Thus, based on the combined teachings of Field and Glick, a POSITA would have understood that the kernel-mode component(s) (e.g., the scan engine and boot detection module of Field and the kernel mode driver 106B of Glick) and the user-mode component(s) (e.g., the antivirus software of Field and the user mode application 106A of Glick) are capable of extending each other's capabilities. The kernel-mode component(s) are capable of operating and monitoring events during the boot sequence (*see* EX1004, [0022]-[0023]) to prevent a RootKit or other malware from concealing itself (*see* EX1004, [0016]), and also capable of extending the access of the user mode component(s) to certain restricted kernel functionalities (*see* EX1007, 2:30-42, 3:32-52). Conversely, the user-mode component(s) are capable of extending the access of the kernel-mode component(s) to certain remediation functions, such as full malware removal (*see* EX1004, [0034]).

69. Similar to Glick, Shukla describes malware detection and removal system that includes both an application-layer component (i.e., the application sandbox) and a "component . . . loaded into the kernel." EX1005, [0032]-[0036]; EX1006, 8-9. This kernel portion of the malware detection system "hooks into kernel-mode API function calls or even unexported API calls and monitors various components of the kernel." EX1005, [0041]; EX1006, 10. Shukla's kernel-level component serves an "important function of being able to detect any rootkit or

malware that may compromise the application layer component of the sandbox." EX1005, [0041]; EX1006, 10-11. Thus, Shukla's kernel-level component operates in a similar manner to and for a similar purpose of the kernel-level component in each of Field and Glick.

70. Shukla describes coordination between its kernel-level and application-layer components to develop and administer flexible behavior rules (described in greater detail below) that allow for better detection of malware. *See* EX1005, [0035], [0039]-[0045], [0056]-[0063]; EX1006, 9, 10-12, 13-15. Thus, a POSITA would have found it obvious to implement in Field's system the coordination between the kernel-level and application-layer components described by Shukla, which would allow for the creation of behavior rules that permit Field's boot detection module 210 and scan engine 208 to better "search for suspicious activity." EX1004, [0017].

2. A POSITA Would Have Found Incorporating Glick's and Shukla's Record Generation Into Field Obvious

71. A POSITA would have been motivated to implement Glick's record generation in the invention disclosed in Field. Field implies record generation, though it does not specifically provide the implementation details for doing so. For example, as noted above, Field describes communicating data about malware detected during the boot sequence to "antivirus software that executes once the computer boots." EX1004, [0034]. This implies that a record of signatures is created

for pestware-related events. However, Field does not limit the type or amount of data collected or the manner in which it is collected or communicated. *See id.* A POSITA would have been motivated to create a record of the events that the program monitors because it would have allowed for notification of various stakeholders (e.g., a user, system administrator, or security center), which a POSITA would have understood would have permitted better tracking of system security and performance. *See* EX1007, 5:52-67; EX1011, 223-224 (teaching different types of logging used to record information).

72. For example, Glick teaches the creation of "a record or error message" that "is generated and provided to the user and/or system administrator" when malicious code is detected. EX1007 at 8:13-20. Glick also teaches an "identifier of the malicious code file" being stored. *Id.* at 13:12-24. As noted above, a POSITA would have recognized that these notifications would have permitted better tracking of system security and performance.

73. Similarly, Shukla describes a kernel-component of a malware detection system that hooks into kernel API calls. *See* EX1005, [0043], [0056]-[0057]; EX1006, 11, 13. Shukla describes a "Learn mode," in which "kernel functions are hooked and the behavior of the application is monitored." EX1005, [0110]; EX1006, 24. This "Learn mode" is "used to silently watch the access made by the application without enforcing any rules." *Id.* Shukla teaches that this monitoring is

useful in determining the "risk" of various applications and their behavior, and to develop the behavior rules that are used by the system to detect and remove malware. *See* EX1005, [0110]-[0118]; EX1006, 24-25. The claims of Shukla refer to these steps as "recording" the events (*e.g.*, "writing executable code to disk or memory, creating startup items, starting new services or drivers or application") and "generating a report." *See* EX1005, claim 52; EX1006, claim 52.

74. From these teachings of Field, Glick, and Shukla, a POSITA would have recognized various benefits (noted above) to configure Field's scan engine 208 and boot detection module 210 to create a record of malware events during the boot process. For example, because Field's scan engine 208 and boot detection module 210 operate before user-mode/application-level components are loaded and execute, employing record generation during the boot sequence would help combat against rootkits and other threats that attempt to take actions during the boot sequence. *See, e.g.*, EX1004, [0014]; EX1007, 6:27-32; EX1005, [0041], [0072]-[0081]; EX1006, 10-11, 16-19.

3. A POSITA Would Have Found Incorporating Shukla's Use of Behavior Rules Into Field Obvious

75. A POSITA would have been motivated to implement behavior rules into the invention disclosed in Field because Field already implies the use of behavior rules by stating that "the scan engine may ***search for suspicious activities*** such as jump instructions in unexpected locations, hidden processes, references to

memory addresses that are outside of a range allocated to the operating system, and the like" and that "[a]ntivirus software ***may identify activity*** that could be characteristic of malware." EX1004, [0017], [0027]. In one example, Field describes that "[r]emoving an entry from a process table or a similar data structure may be ***a strong heuristic indicator*** that a RootKit is present." EX1004, [0017]. While Field does not specifically describe this functionality as the use of behavior rules to managing pestware-related events, a POSITA would nonetheless have understood or at least found it obvious that Field's search for suspicious activities would have included the use of behavior rules based on the teachings of Shukla.

76. Specifically, Shukla teaches that "a search" can be conducted using "behavior signatures" based on "runtime behavior," such as "network access" or "file creation/deletion." EX1005, [0062]; EX1006, 14. In other words, Shukla describes that the search criteria described in Field may include the search for "behavior signatures" within runtime data. Shukla describes that by monitoring "how [an application] interacts with other applications," it is possible to "detect malware content inside an application," which improves upon simple signature-based scanning techniques. EX1005, [0015]-[0016]; EX1006, 4-5. While one of Shukla's embodiments relates to employing these behavior scanning techniques in the context of a sandbox, Field's "invention is also able to remove or neutralized malware on disk and in memory." EX1005, [0017]; EX1006, 5. Thus, when Field

describes that ***"[a]ny currently existing or yet to be developed techniques may be used*** by the scan engine 208 to identify malware that starts executing before services provided by the operating system 204 are available," a POSITA would have understood or at least found it obvious that the use of behavior rules as implied in Field and explicitly described in Shukla was one of these "currently existing . . . techniques" employed by Field's scan engine 208. *See* EX1004, [0017].

4. A POSITA Would Have Found Incorporating Shukla's Registry Scanning Into Field Obvious

77. A POSITA would have been motivated to implement registry scanning into the invention disclosed in Field, because Shukla describes that the content of registry entries is a known indication of malware and therefore scanning a registry would have helped Field's scan engine 208 better identify malware. *See, e.g.*, EX1005, [0008], [0015], [0032], [0062]; EX1006, 2-3, 4-5, 8, 14. Field already describes the importance of certain data structures maintained by the operating system (e.g., "a process table or a similar data structure") in identifying the presence of malware such as a RootKit. EX1004, [0017]. For example, to scan the registry, Shukla describes "checking for inconsistencies between the startup list for drivers in the registry compared to the one in the memory." EX1005, [0084]; EX1006, 20. A POSITA would have recognized or at least found it obvious that one known way to accomplish the comparison described by Shukla would be to scan data in the registry (e.g., the startup list for drivers) and compare the results of that scan to data in the

memory that represents what data should be in the registry (sometimes referred to as a clean list). Thus, a POSITA would have been motivated to implement registry scanning as part of the scan performed by Field's scan engine 208 to better detect malware capable of hiding itself. *Id.*

VIII. OVERVIEW OF UNPATENTABILITY GROUNDS

Ground	Claim(s)	Prior Art Basis
1	1-20	§103 – Field, Glick, and Shukla

IX. GROUND 1: FIELD, GLICK, AND SHUKLA RENDER OBVIOUS CLAIMS 1-20

A. Claim 1

78. Claim 1 of the '505 Patent recites:

> **1**. A method for managing pestware on a computer comprising:
> monitoring events during a boot sequence of the computer;
> managing pestware-related events during a first period in a boot sequence of the computer, the first period in the boot sequence occurring before the computer becomes configured to run native applications, before a subsystem of an operating system is loaded, and after a kernel is loaded;
> managing pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occurring when the computer is configured to run native applications;
> generating, in response to the monitoring, a record of events, the record of events including the pestware-related events;
> analyzing the record of events so as to identify the pestware-related events
> modifying the set of behavior rules so as to prevent the pestware related events; and
> scanning data in a registry of the computer for pestware during the second period in the boot sequence.

79. As I explain in more detail below through my feature-by-feature analysis, Field, Glick, and Shukla disclose or render obvious claim 1.

1. [1pre] A method for managing pestware on a computer comprising:

80. Field describes a "method [that] causes a software module configured to ***scan for malware*** to be initialized at ***computer*** startup..." and "***handle[] the malware*** infection" if one is found. EX1004, Abstract. The '505 Patent describes that "[p]ersonal computers and business computers are continually attacked by trojans, spyware, and adware, collectively referred to as 'malware' or 'pestware.'" EX1001, 1:32-34. Thus, the '505 Patent uses the terms "malware" and "pestware" interchangeably.

81. Similarly, Glick describes an "invention [which] relates to the protection of computer systems from malicious code." EX1007 at 1:11-12. And Shukla describes a method and apparatus that "detects and removes malware" by "monitoring and controlling the behavior and access privileges of the application and only selectively granting access." EX1005, Abstract; EX1006, Abstract. Thus, each of Field, Glick, and Shukla—and the combination of the three—describes a "method for managing pestware on a computer," as recited in [1pre].

2. *[1a] monitoring events during a boot sequence of the computer;*

82. Field, which is titled "identifying malware in a boot environment," describes "a method… that provides a search for malware ***during the boot process***." EX1004 at Abstract. Figures 3 and 4 of Field also illustrate a malware scan taking place during the boot process, before it is complete. *Id*., FIGS. 3-4, [0023]-[0026]. "Generally stated, it is desirable to start ***scanning for malware as early as possible in the boot process*** to prevent malware that infects a previous component from having the opportunity to implement obfuscation techniques." *Id.*, [0023].

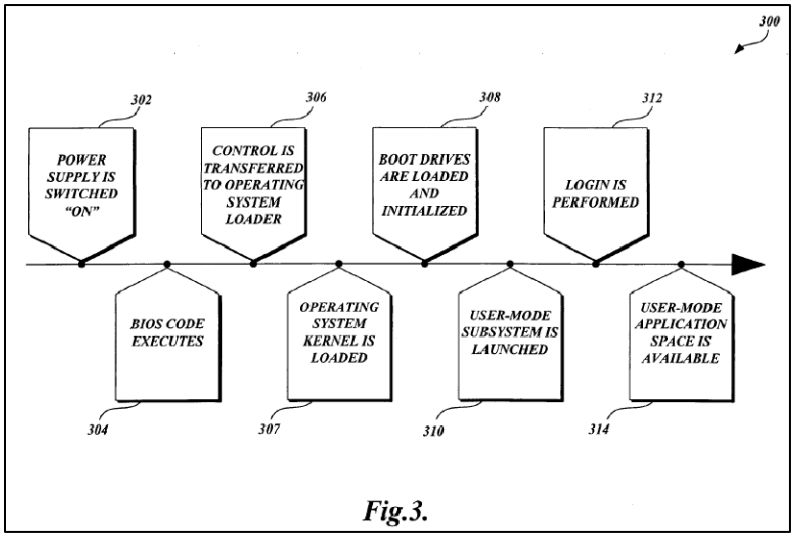


Fig.3.

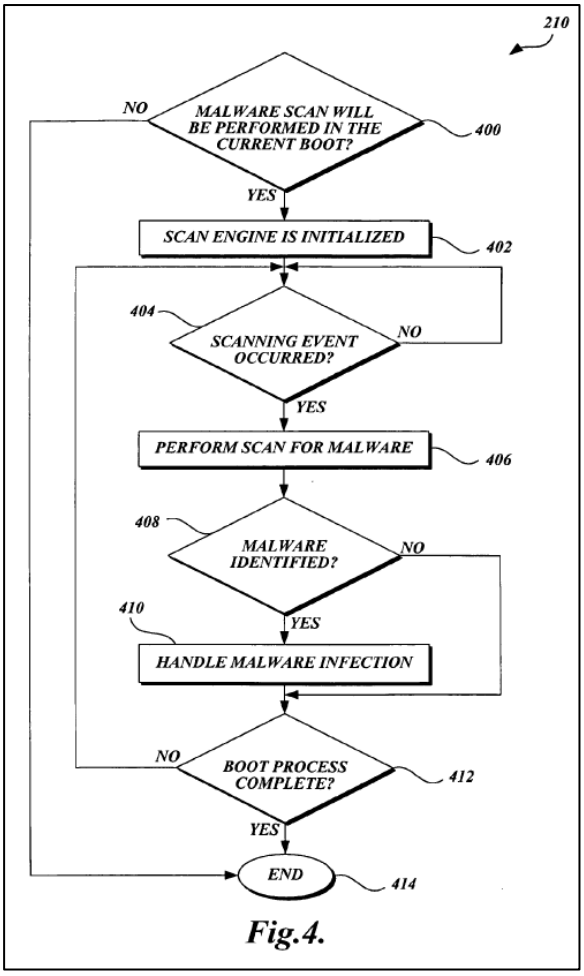


Fig.4.

83. Field also teaches ***monitoring events*** during its operations. Field teaches that, "in response to identifying the occurrence of a scanning event[4], the method causes the software module to search computer memory for data that is characteristic of malware." *Id*. at [0007]. For example, Field describes that "the scan engine 208 may ***search for suspicious activities*** such as jump instructions in

[4] Field describes a "scanning event" as an event (e.g., the loading of a driver into memory) that triggers a scan for malware. *See* EX1004, [0031].

unexpected locations, hidden processes, references to memory addresses that are outside of a range allocated to the operating system 204, and the like." *Id.* at [0017]. A POSITA would have understood or at least found it obvious that these "suspicious activities" described in Field are examples of "events" for which Field's scan engine 208 monitors.

84. Similarly, Shukla describes "a method for ***monitoring behavior*** of a number of applications or modules in a computing device." EX1005, [0037]; EX1006, 9. For example, Shukla describes that its system "may monitor the applications file system access or network access by intercepting its API function calls through imported or exported functions table patching and inline hooking of functions at the application layer." EX1005, [0037]; EX1006, 9-10. In another example, Shukla describes that its system "may monitor the applications executable content, memory access, and registry access in a like manner." *Id.* Shukla also describes a "Learn mode," in which "kernel functions are hooked and the behavior of the application is monitored." EX1005, [0110]; EX1006, 24. And Shukla's own claims refer to these monitored behaviors as "events." *See* EX1005, claims 65, 69; EX1006, claims 65, 69. Thus, in the Field-Glick-Shukla combination in which the scan engine 208 of Field utilizes the behavior rules described by Shukla (*see* Section IV.A.4.iii, *supra*), it would have at least been obvious to monitor events during a boot sequence of the computer, as recited in [1a].

3. [1b] managing pestware-related events during a first period in a boot sequence of the computer, the first period in the boot sequence occurring before the computer becomes configured to run native applications, before a sub-system of an operating system is loaded, and after a kernel is loaded;

85. Field teaches ***managing pestware-related events***. For example, Field describes identifying "activity that could be characteristic of malware" and removing or quarantining identified malware "from the computer". EX1004, [0016], [0027]. In one specific example implementation, Field describes a computer 200 that includes a scan engine 208 and a boot detection module 210, which "collectively provide ***a way to identify malware***, such as a RootKit, that starts executing before the services provided by traditional antivirus software are available." EX1004, [0014]. As I described with respect to [1a] (the discussion of which I incorporate herein), Field describes that "the scan engine 208 may ***search for suspicious activities*** such as jump instructions in unexpected locations, hidden processes, references to memory addresses that are outside of a range allocated to the operating system 204, and the like." *Id.*, [0017].

> For example, some operating systems maintain a data structure, sometimes referred to as a process table, with a list of programs that are currently executing. Removing an entry from a process table or a similar data structure may be a strong heuristic indicator that a RootKit is present.

Id. If such malware is identified, "the boot detection module 210 causes the malware infection to be handled," which may include, for example, "attempting to remove

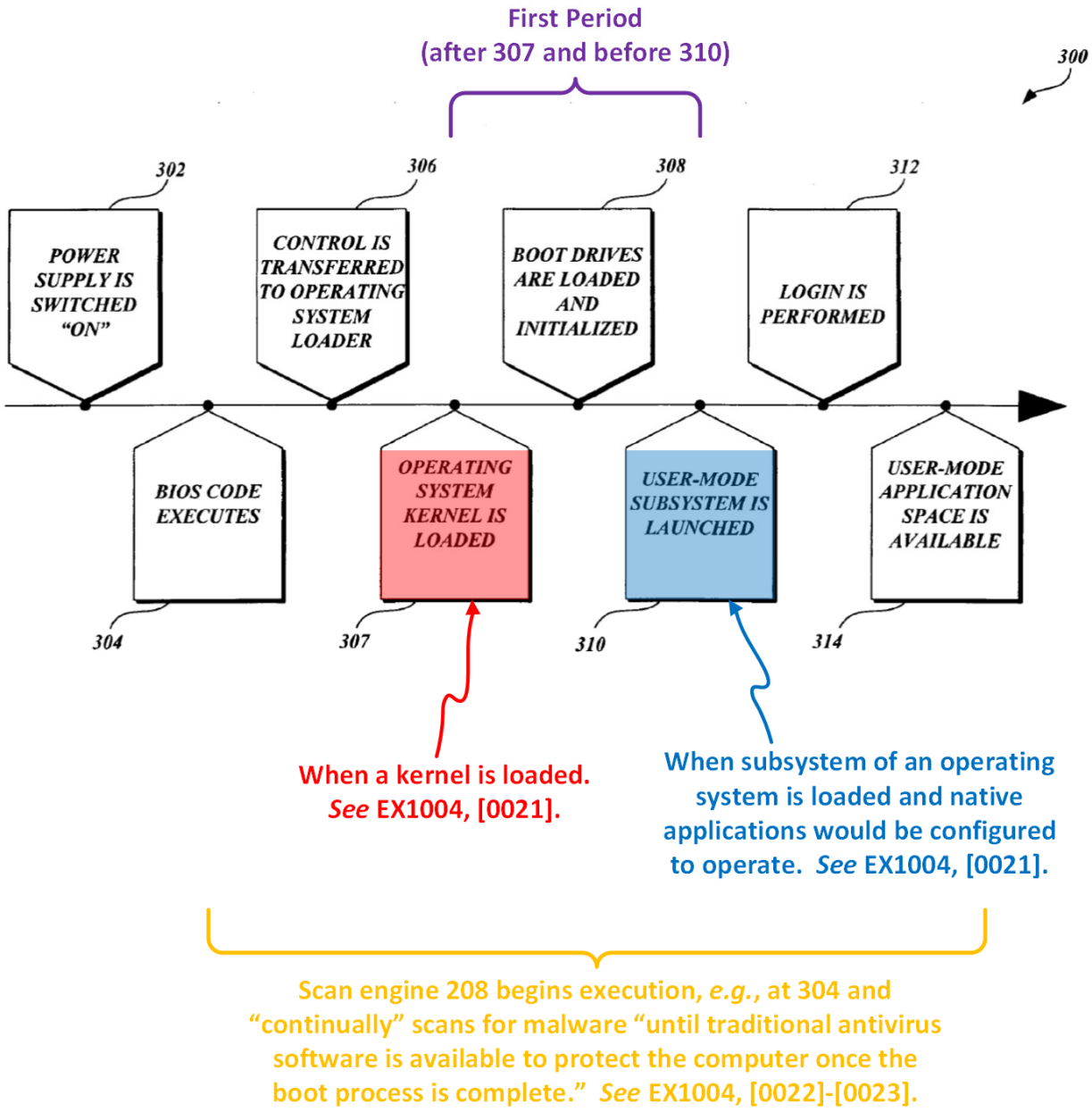
the malware from the computer by killing processes, deleting files, removing entries in configuration files that are associated with the malware, and the like." *Id.*, [0034].

86. A POSITA would have recognized or at least found it obvious that the search for malware and handling of the malware identified in the search is a process of "managing pestware-related events," as recited in [1b]. For example, a POSITA would have recognized or at least found it obvious that identifying the removal of "an entry from a process table or a similar data structure" as a pestware-related event, and that then removing the associated pestware to be at least one way to "manage" that event.

87. Moreover, for the reasons I set forth in my description of the Field, Glick, and Shukla combination above, a POSITA would have found it obvious to utilize the behavior rules described by Shukla in Field's scan engine 208. Shukla describes "a method for ***monitoring behavior*** of a number of applications or modules in a computing device." EX1005, [0037]; EX1006, 9. For example, Shukla describes that its system "may monitor the applications file system access or network access by intercepting its API function calls through imported or exported functions table patching and inline hooking of functions at the application layer." EX1005, [0037]; EX1006, 9-10. In another example, Shukla describes that its system "may monitor the applications executable content, memory access, and registry access in a like manner." *Id.* Shukla's own claims refer to these monitored behaviors as

"events." *See* EX1005, EX1006, claims 65, 69. Thus, in the Field-Glick-Shukla combination, in which the scan engine 208 of Field utilizes the behavior rules described by Shukla, it would have at least been obvious to ***manage pestware-related events*** by utilizing the behavior rules of Shukla to identify the pestware-related events, and then "handle" the malware associated with those events using Field's boot detection module 210.

88. Field teaches managing pestware-related events during the claimed ***first period***. For example, Field describes that "a boot detection module 210 is provided that causes malware to be identified ***before the services provided by an operating system are available***." EX1004, [0016]. Generally, "the boot detection module 210 causes the scan engine 208 to be loaded into memory and executed ***during the boot process***." *Id.* "More specifically, at any number of different locations in the timeline 300, a software module (e.g., the scan engine 208) may be loaded into memory. EX1004, [0022]. An annotated version of timeline 300 in FIG. 3 is provided below.



89. Generally, Field discloses that it "is desirable to start scanning for malware as early as possible in the boot process to prevent malware that infects a previous component from having the opportunity to implement obfuscation technique." EX1004, [0023]. More specifically, Field's specification states that its scan engine "may be loaded into memory" at "any number of different locations in

the timeline" and may "begin executing at events ***304, 306***, or 310." *Id.*, [0022]-[0023]. "In this regard, components that execute during the boot process may be continually scanned until traditional antivirus software is available to protect the computer once the boot process is complete." *Id.*, [0022]. Thus, as shown in the above annotation of FIG. 3, where a POSITA configured the scan engine 208 to begin executing at events 304 or 306 per the explicit teachings of Field, the scan engine 208 would be executing (e.g., managing pestware-related events by scanning for malware and removing or quarantining identified malware) during a **first period** that is **before the computer becomes configured to run native applications**, **before a subsystem of an operating system is loaded (event 310)**, and **after a kernel is loaded (event 307)**.

4. [1c] managing pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occurring when the computer is configured to run native applications;

90. Field teaches ***managing pestware-related events***. For example, Field describes identifying "activity that could be characteristic of malware" and removing or quarantining identified malware "from the computer". EX1004, [0016], [0027]. In one specific example implementation, Field describes a computer 200 that includes a scan engine 208 and a boot detection module 210, which "collectively provide ***a way to identify malware***, such as a RootKit, that starts executing before the services provided by traditional antivirus software are available." EX1004,

[0014]. As described with respect to [1a], Field describes that "the scan engine 208 may ***search for suspicious activities*** such as jump instructions in unexpected locations, hidden processes, references to memory addresses that are outside of a range allocated to the operating system 204, and the like." *Id.*, [0017].

> For example, some operating systems maintain a data structure, sometimes referred to as a process table, with a list of programs that are currently executing. Removing an entry from a process table or a similar data structure may be a strong heuristic indicator that a RootKit is present.

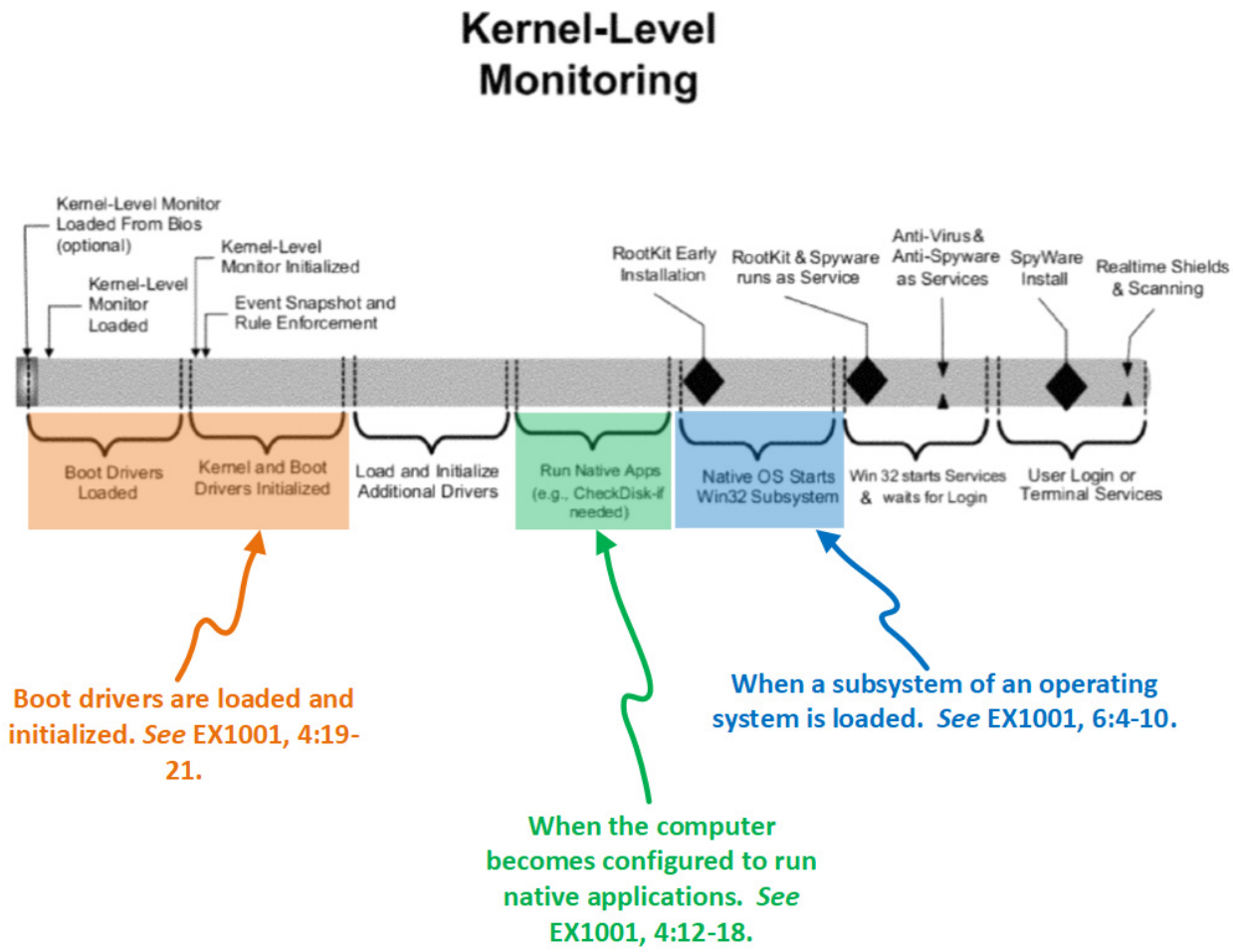
Id. If such malware is identified, "the boot detection module 210 causes the malware infection to be handled," which may include, for example, "attempting to remove the malware from the computer by killing processes, deleting files, removing entries in configuration files that are associated with the malware, and the like." *Id.*, [0034].

91. A POSITA would have recognized or at least found it obvious that the search for malware and handling of the malware identified in the search is a process of "managing pestware-related events," as recited in [1b]. For example, a POSITA would have recognized or at least found it obvious that identifying the removal of "an entry from a process table or a similar data structure" as a pestware-related event, and that then removing the associated malware to be at least one way to "manage" that event.

92. Moreover, for the reasons I set forth in my description of the Field, Glick, and Shukla combination above, a POSITA would have found it obvious to utilize the behavior rules described by Shukla in Field's scan engine 208. Shukla describes "a method for ***monitoring behavior*** of a number of applications or modules in a computing device." EX1005, [0037]; EX1006, 9. For example, Shukla describes that its system "may monitor the applications file system access or network access by intercepting its API function calls through imported or exported functions table patching and inline hooking of functions at the application layer." EX1005, [0037]; EX1006, 9-10. In another example, Shukla describes that its system "may monitor the applications executable content, memory access, and registry access in a like manner." *Id.* Shukla's own claims refer to these monitored behaviors as "events." *See* EX1005, EX1006, claims 65, 69. Thus, in the Field-Glick-Shukla combination in which the scan engine 208 of Field utilizes the behavior rules described by Shukla, it would have at least been obvious to ***manage pestware-related events in accordance with a set of behavior rules*** by utilizing the behavior rules of Shukla to identify the pestware-related events, and then "handle" the malware associated with those events using Field's boot detection module 210.

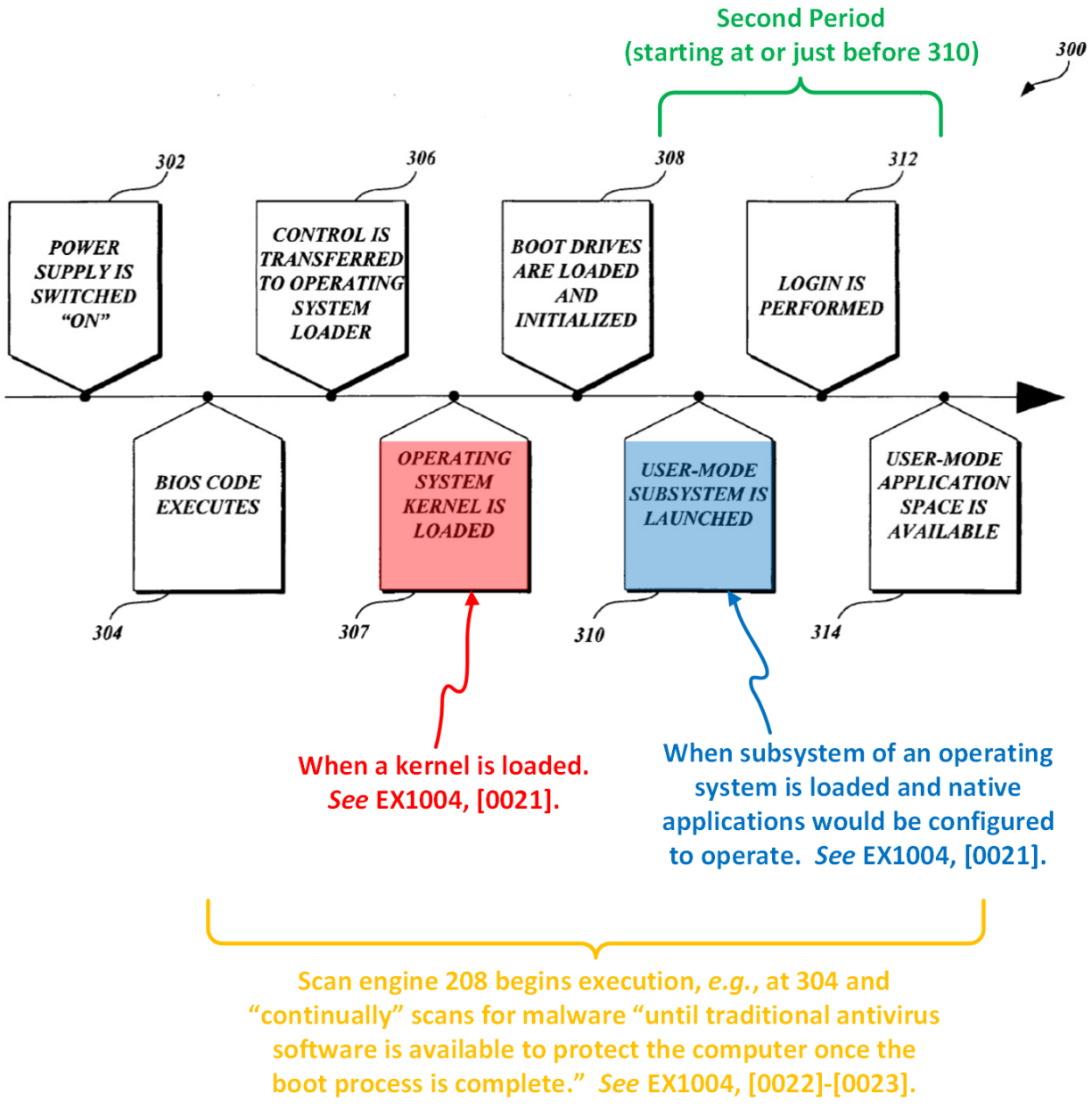
93. With respect to the claimed ***second period***, the '505 Patent describes that "a boot sequence of the protected computer is initiated (e.g., by a user of the protected computer) (Block 202), and in advance of ***a period when native***

applications (e.g., checkdisk) of the protected computer beginning execution, the kernel-level monitor 105 is initialized." EX1001, 4:12-18. Thus, FIG. 4 of the '505 patent shows that an operating system is able to run native applications after drivers are loaded and before the native operating system starts its subsystem, annotated below. EX1001, FIG. 4.



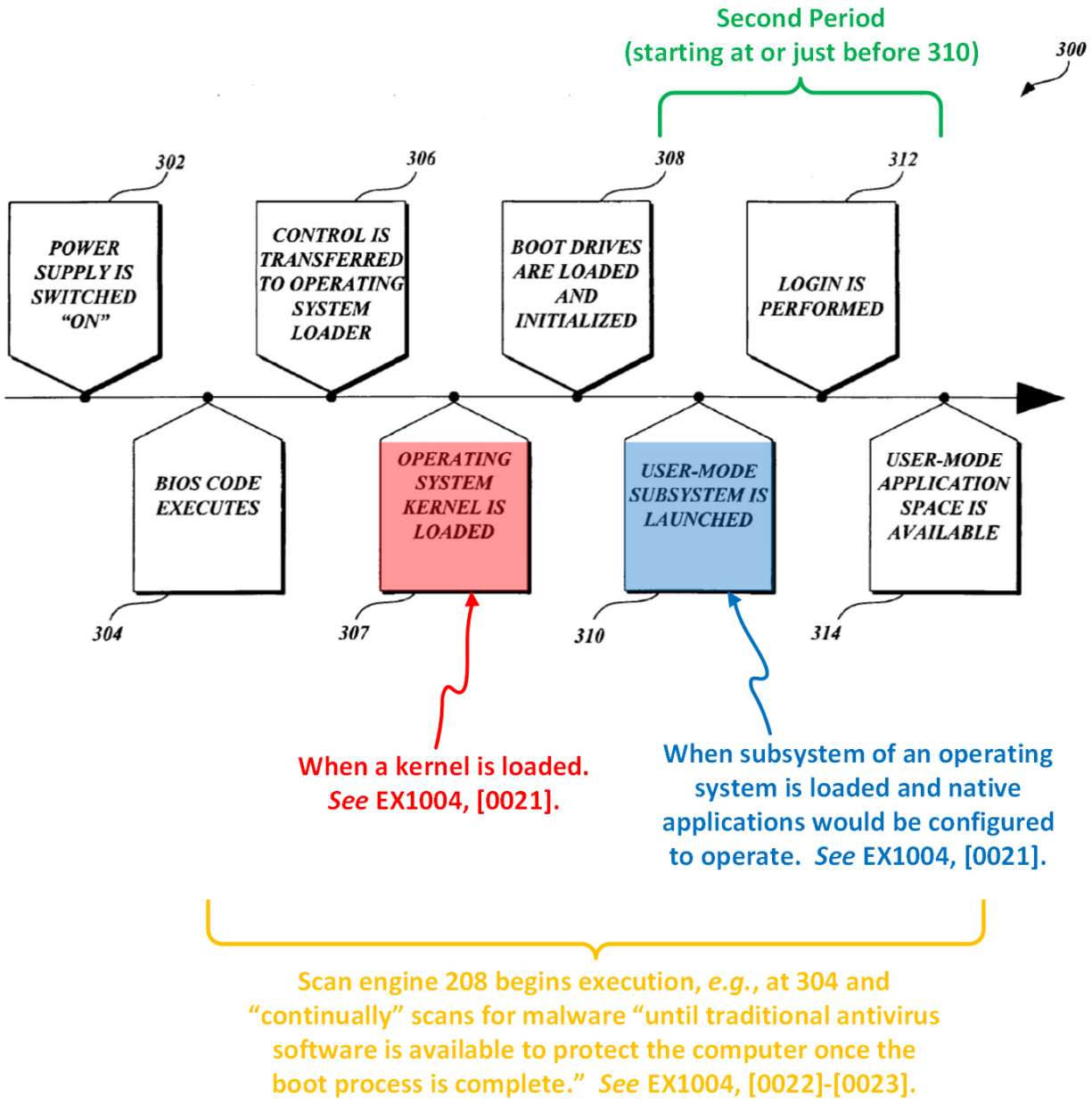
94. A POSITA would have understood or at least found it obvious that Field's computer becomes configured to run native applications around **event 310**. For example, Field describes that "the user mode subsystem provides support

services to the ***user mode application space***." EX1004, [0021]. "Once the user mode services are available, at event 314, the user-mode application space may be accessed to execute programs. Once event 314 is reached, programs designed to perform specific tasks on a general purpose computer may be executed." *Id.*



95. A POSITA would have understood or at least found it obvious that "native applications" would have been configured to run at approximately the same time and certainly no later than when the "user-mode applications" described by Field would have been configured to run. The timeline in FIG. 4 of the '505 Patent describes effectively the same sequence of events in a standard boot process and the timeline in FIG. 3 of Field. The methods and systems in each of the '505 Patent and Field are described as being operable on general purpose computers that run a variety of operating systems, including Microsoft Windows. EX1001, 3:5-8 ("And if implemented in software, the modules can be designed to operate on any type of computer system including WINDOWS and Linux-based systems"); EX004, [0015] ("The operating system 204 illustrated in FIG.2 may be a general-purpose operating system such as a Microsoft® operating system, UNIX® operating system, or Linux® operating system."). Thus, while the exact events indicated on each of the timelines are not identical, a POSITA would nevertheless have understood the two timelines as depicting the same general boot process flow, which would have been specific to a given operating system and computer configuration and independent of the monitoring system/method. Accordingly, because the '505 Patent describes that **the computer is configured to run native applications** between when **the boot drivers are loaded and initialized** and when **a subsystem of an operating system is loaded**, for the same reasons, a POSITA would have understood or at least found

it obvious that Field's computer is configured to run native applications around or just before Field's event 310 where Field describes that the "user mode subsystem is launched." EX1004, [0021].



96. Generally, Field discloses that "is desirable to start scanning for malware as early as possible in the boot process to prevent malware that infects a

previous component from having the opportunity to implement obfuscation technique." EX1004, [0023]. More specifically, Field's specification states that its scan engine "may be loaded into memory" at "any number of different locations in the timeline" and may "begin executing at events ***304, 306***, or 310." *Id.*, [0022]-[0023]. "In this regard, components that execute during the boot process may be continually scanned until traditional antivirus software is available to protect the computer once the boot process is complete." *Id.*, [0022]. Thus, as shown in the above annotation of FIG. 3, where a POSITA configured the scan engine 208 to begin executing at events 304 or 306 per the explicit teachings of Field, the scan engine 208 would be executing (e.g., managing pestware-related events by scanning for malware and removing or quarantining identified malware) during **a second period when the computer is configured to run native applications**.

5. [1d] generating, in response to the monitoring, a record of events, the record of events including the pestware-related events;

97. As I described with respect to [1a]-[1c], Field generally describes that the boot detection module 210 and scan engine 208 "search[es] for suspicious activity" and, if malware is detected, "causes the malware infection to be handled." *See* EX1004, [0017], [0027], [0032], [0034]. When the boot detection module 210 and scan engine 208 detect potential malware, Field describes that it "may be difficult or impossible" to "successfully remov[e] all components of a malware"

using the limited "resources available in a boot environment." EX1004, [0034]. Thus, Field describes that, "when the malware is quarantined, data will typically be communicated to antivirus software that executes once the computer boots so that all of the components of the malware may be removed." EX1004, [0034]. Thus, Field suggests that certain records are generated and stored by the boot detection module 210 and scan engine 208 (i.e., the kernel-mode components) and communicated to antivirus software (e.g., user-mode components) so that the detected malware can be further handled after the boot process is complete. *See* EX1004, [0034]. Field does not prescribe or otherwise limit the form of the records about the potential malware that are generated and stored during the boot process that is passed to the post-boot antivirus software. *Id.*

98. Based on the teachings of Glick and Shukla, as well as the general knowledge of a POSITA, it would have been obvious to a POSITA that the records generated by Field's boot detection module 210 and scan engine 208 and passed to the post-boot antivirus software would include a record of events that includes pestware-related events. A POSITA at the time of filing would have been motivated to create a record of the events that the program monitors. For example, Glick describes that, any time malicious code is detected, "***a record*** or error message ***is generated*** and provided to the user and/or system administrator," and the record may include information that identifies "the particular malicious code and/or any

protective action taken." *See* EX1007, 5:52-59. These records can be written to a file or otherwise logged. *See* EX1007, 5:60-64.

99. Similarly, Shukla describes malware detection and removal system that includes both an application-layer component (i.e., the application sandbox) and a "component . . . loaded into the kernel." EX1005, [0032]-[0036]; EX1006, 8-9. This kernel portion of the malware detection system "hooks into kernel-mode API function calls or even unexported API calls and monitors various components of the kernel." EX1005, [0041]; EX1006, 10. Shukla's kernel-level component serves an "important function of being able to detect any rootkit or malware that may compromise the application layer component of the sandbox." EX1005, [0041]; EX1006, 10-11. Thus, Shukla's kernel-level component operates in a similar manner to and for a similar purpose of the kernel-level component in each of Field and Glick.

100. Shukla describes that its kernel-level component works by hooking in various API calls within the kernel that allows Shukla's malware detection system to observe and control the execution of these API calls. *See* EX1005, [0043], [0056] ; EX1006, 11, 13. "Target functions are primarily related to file manipulation, process creations, network access, registry access, memory access etc." EX1005, [0056]; EX1006, 13. "After the target functions are hooked in kernel, the processing of intercepted API function calls may be done in conjunction with the application

sandbox ***or independently***." EX1005, [0057]; EX1006, 13. In other words, Shukla describes that its kernel-level component can work in conjunction with an application-layer component or independently of one. *Id.* A POSITA would have recognized that this ability to operate independently makes the methods employed by Shukla's kernel-level component ideal for implementing the "suspicious activity" monitoring performed by Field's scan engine 208, which operates prior to the loading and initialization of an application-level component with which it later communicates and coordinates. *Id.*; EX1004, [0017] ("Any currently existing or yet to be developed techniques may be used by the scan engine 208 to identify malware . . . . Moreover, the scan engine 208 may search for suspicious activities such as jump instructions in unexpected locations, hidden processes, references to memory addresses that are outside of a range allocated to the operating system 204, and the like."), [0034] ("when the malware is quarantined, data will typically be communicated to antivirus Software that executes once the computer boots so that all of the components of the malware may be removed.").

101. In order to create the rules upon which Shukla's malware detection system evaluates behavior, Shukla describes a "Learn mode," in which "kernel functions are hooked and the behavior of the application is monitored." EX1005, [0110]; EX1006, 24. This "Learn mode" is "used to silently watch the access made by the application without enforcing any rules." *Id.* Shukla teaches that this

monitoring is useful in determining the "risk" of various applications and their behavior, and to develop the behavior rules that used by the system to detect and remove malware. *See* EX1005, [0110]-[0118]; EX1006, 24-25. From these teachings, a POSITA would have been motivated to configure Field's scan engine 208 and detection module 210 to hook into certain kernel APIs (e.g., those "related to file manipulation, process creations, network access, registry access, memory access etc.," as taught by Shukla) and record the behavior of the drivers and applications that are loaded and begin execution during the boot cycle. *See* EX1005, [0110]-[0118] ; EX1006, 24-25. Through the creation of these records, the application-level component of the combined system would advantageously be able to statistically analyze the observed and recorded behavior from the boot process and better protect the system from threats that begin during the boot process (e.g., rootkits). *Id.* Configured in this manner, the Field-Glick-Shukla combination would "generat[e], in response to the monitoring, a record of events, the record of events including the pestware-related events," as recited in [1d].

6. [1e] analyzing the record of events so as to identify the pestware-related events

102. As I described above with respect to [1d], Field describes that, "when the malware is quarantined, data will typically be communicated to antivirus software that executes once the computer boots so that all of the components of the malware may be removed." EX1004, [0034]. Thus, Field suggests that certain

records are generated and stored by the boot detection module 210 and scan engine 208 (i.e., the kernel-mode components) and communicated to antivirus software (e.g., user-mode components) so that the detected malware can be further handled after the boot process is complete. *See* EX1004, [0034].

103. As I further described above with respect to [1d], Shukla describes a "Learn mode," in which "kernel functions are hooked and the behavior of the application is monitored." EX1005, [0110]; EX1006, 24. This "Learn mode" is "used to silently watch the access made by the application without enforcing any rules."[5] *Id.* Shukla teaches that this monitoring is useful in determining the "risk" of various applications and their behavior, and to develop the behavior rules that used by the system to detect and remove malware. *See* EX1005, [0110]-[0118]; EX1006, 24-25. From these teachings a POSITA would have found it obvious to

[5] Shukla describes using the "learn mode" for new applications for which rules have not been generated. *See* EX1005, [0110]-[0112]; EX1006, 24. A POSITA would have understood or at least found it obvious that the system of Shukla would still be able to enforce rules for those applications for which rules have already been created while in the described "learn mode," as this would have avoided a system in which the "learn mode" leaves the entire system completely vulnerable to attack.

configure the application-level component of the Field-Glick-Shukla combination to "analyz[e] the record of events so as to identify the pestware-related events" (as recited in [1e]), because this analysis would allow the application-level component of the combined system to statistically assess the risk factor of the behaviors observed during the boot cycle and develop rules to identify and remove malware. *See id.*

7. [1f] modifying the set of behavior rules so as to prevent the pestware related events; and

104. As I described above with respect to [1d] and [1e], as well as with respect to the combination of Field, Glick, and Shukla, a POSITA would have been motivated to configure Field's scan engine 208 and detection module 210 to hook into certain kernel APIs (e.g., those "related to file manipulation, process creations, network access, registry access, memory access etc.," as taught by Shukla) and record the behavior of the drivers and applications that are loaded and begin execution during the boot cycle. *See* EX1005, [0110]-[0118]; EX1006, 24-25. Shukla teaches that this "Learn mode" process is used to generate and test behavior rules that will be used to identify malware, and that the process ensures proper functioning and security. *See* EX1005, [0110]; EX1006, 24.

105. A POSITA would have understood that iteratively observing new behavior and assessing it would have allowed the system of the Field-Glick-Shukla combination to better adapt to new threats by allowing for the creation of new

behavior rules and the modification of old behavior rules. *See* EX1005, [0078]; EX1006, 18. As Shukla recognized, using rigid signature-based detection techniques was "ineffective" (*see* EX1005, [0078]; EX1006, 18) and that using rigid behavior rules leads to a malware detection system that is either overly burdensome on the user or overly permissive to malware (*see* EX1005, [0119]-[0123]; EX1006, 25-26). Thus, Shukla teaches the benefits of flexibility in the creation, maintenance, and application of behavior rules to adapt to new malware threats. From these teachings, a POSITA would have found it obvious for the Field-Glick-Shukla combination to "modify[] the set of behavior rules so as to prevent the pestware related events," as recited in 1[f].

8. 1[g] scanning data in a registry of the computer for pestware during the second period in the boot sequence.

106. During the original prosecution of the '505 Patent, Applicant distinguished a prior art reference by arguing that "[d]etecting suspicious reads of a registry is not the same as scanning data in a registry of the computer for pestware during the second period in the boot sequence." EX1002, 151 (emphasis in original). Thus, under the Applicant's own interpretation, the claims require "scanning data in a registry of the computer," and not just monitoring reads of data. *Id.* The Field-Glick-Shukla combination renders this limitation, under Applicant's own interpretation, obvious.

107. As I described in with respect to the combination of Field, Glick, and Shukla, Field describes the importance of certain data structures maintained by the operating system (e.g., "a process table or a similar data structure") in identifying the presence of malware such as a RootKit. EX1004, [0017]. A POSITA would have understood or at least found it obvious based on the teachings of Shukla that the registry is one example of an operating system data structure, and that scanning the registry "can either reveal malicious intent or hint at potentially malicious nature." EX1005, [0062], [0068]; EX1006, 14, 15. For example, to scan the registry, Shukla describes "checking for inconsistencies between the startup list for drivers in the registry compared to the one in the memory." EX1005, [0084]; EX1006, 20. A POSITA would have recognized or at least found it obvious that one known way to accomplish the comparison described by Shukla would be to scan data in the registry (e.g., the startup list for drivers) and compare the results of that scan to data in the memory that represents what data should be in the registry (sometimes referred to as a clean list). Indeed, such registry scanning was well known to POSITAs as a standard part of monitoring pestware-related events. *See e.g.* EX1010, 3:35-37 ("System configuration files such as ***a system registry are scanned for specific entries*** that are associated with active malware."). From these teachings, a POSITA would have found it obvious for the Field-Glick-Shukla

combination to **scan[] data in a registry of the computer for pestware during the second period in the boot sequence.**

B. Claim 2

108. Claim 2 of the '505 Patent recites:

> **2**. The method of claim **1**, further comprising launching, after an operating system is initiated, a pestware management engine; and
>
> wherein the set of behavior rules includes behavior rules compiled by the pestware management engine.

1. 2[a] The method of claim 1, further comprising launching, after an operating system is initiated, a pestware management engine; and

109. As I described above, a POSITA would have found it obvious to implement Glick's and Shukla's user-mode/application-level operation into the invention disclosed in Field. Field teaches that "resources available in a boot environment are limited," and therefore "successfully removing all components of a malware . . . may be difficult or impossible." EX1004, [0034]. Thus, Field teaches that its boot detection module may quarantine the malware, and data about the malware "will typically be communicated to antivirus software that executes once the computer boots so that all of the components of the malware may be removed." *Id.*

110. Field teaches that "traditional antivirus software" (i.e., a user-mode/application-level component) is not available until after the boot sequence.

See EX1004, [0014], [0022] ("components that execute during the boot process may be continually scanned until traditional antivirus software is available to protect the computer once the boot process is complete."); EX1003, [XX]. Indeed, this is consistent with the '505 Patent's own description of the prior art with respect to FIG. 3, which shows that "user-mode services or applications" such as "anti-virus" and "anti-spyware" services are initiated after the boot and operating system sequences. EX1001, FIG. 3, 1:61-65.

111. Glick and Shukla both describe implementations details for a user-mode/application-level component that a POSITA would have found it obvious to utilize in Field's malware detection and removal system as part of or in place of the "traditional antivirus software" that Field relies upon to fully remove malware identified during the boot sequence. *See* EX1004, [0034]; Section IV.A.4.i, *supra*. Based on the teachings of Field and Glick, a POSITA would have recognized or at least found it obvious that these user-mode/application-level components would have been launched at the end of or after the boot sequence, and thus **after an operating system is initiated**. *See* EX1004, [0022]; EX1007, 6:23-32.

112. Based on the teachings of Field, Glick, and Shukla, a POSITA would have further recognized or found it obvious that these user-mode/application-level components together constitute a **pestware management engine**. As noted above, Field teaches that its boot detection module may quarantine the malware, and data

about the malware "will typically be communicated to antivirus software that executes once the computer boots ***so that all of the components of the malware may be removed***." EX1004, [0034]. A POSITA would have recognized or at least found it obvious that the removal of malware is an integral part of pestware management. Additionally, the implementation details of the user-mode/application-level components provided by Glick and Shukla confirm that these components constitute a pestware management engine. For example, Glick describes that "user mode application 106A includes code that detects malicious code, such as spyware and computer viruses, from user mode and when appropriate takes protective action when malicious code is detected." EX1007, 3:32-39. Similarly, Shukla describes that its system includes both application-level and kernel-level components, and that the application-level component "prevents attacks, detects and removes malware based on an elastic or adaptive limits on application behavior" by "blocking, permitting, or monitoring the API function calls at the application layer." *See* EX1005, [0032]-[0035], [0040]-[0041]; EX1006, 8-9, 10-11. These example functions described by Glick and Shukla are consistent with the functions that the pestware management engine of the '505 Patent performs. *See, e.g.*, EX1001, 5:34-59. Based at least on these teachings, a POSITA would have further recognized or found it obvious that these user-mode/application-level components together constitute a **pestware management engine**.

2. 2[b] wherein the set of behavior rules includes behavior rules compiled by the pestware management engine.

113. As I described with respect [1d]-[1f], a POSITA would have found it obvious and been motivated to implement the creation, modification, and enforcement of behavior rules into the invention disclosed in Field, as described by Shukla.

114. Shukla describes a "Learn mode," in which "kernel functions are hooked and the behavior of the application is monitored." EX1005, [0110]; EX1006, 24. This "Learn mode" is "used to silently watch the access made by the application without enforcing any rules." *Id.* Shukla teaches that this monitoring is useful in determining the "risk" of various applications and their behavior, and to develop the behavior rules that used by the system to detect and remove malware. *See* EX1005, [0110]-[0118]; EX1006, 24-25. Because Shukla teaches that the "Learn mode" is based on hooking of both application and kernel functions, a POSITA would have understood or at least found it obvious that rule creation and compilation process is based on coordination between both the kernel-level and application-level components of the system. *See* EX1005, [0110]-[0118]; EX1006, 24-25. Given that the application-level components of these types of malware detection systems were known to have broader processing and remediation capabilities (*see, e.g.*, EX1004, [0034]), a POSITA would have understood or at least found it obvious that the user-mode/application-level components of the Field-

Glick-Shukla combination would have been configured to compile the behavior rules. Indeed, this is consistent with Glick's general descriptions of its user mode application 106A performing the primary analysis and coordination functions, and calling upon the kernel mode driver 106B for specific functional assistance. *See, e.g.*, EX1007, 12:50-13:44, 18:6-33. And while Shukla does not specifically describe where the risk scores and rules are generated based on the information collected in "Learn Mode" (*see* EX100 EX1005, [0110]-[0118]), a POSITA would have recognized that there were two particularly logical locations to perform these functions: (1) on the computer being monitored; and (2) at a central server that coordinates rules for a set of monitored computes. A POSITA would have understood that there were pros and cons to each of these approaches (e.g., the former is less complicated and may allow individual users a greater degree of control, as well as being less dependent on network connectivity, while the latter would have allowed for greater consistency for a group of related computers owned/operated by a common entity, as well as enabling centralized identification of threats, which in turn enables proactive generation and distribution of signatures and behavioral rules). Thus, in the Field-Glick-Shukla combination, a POSITA would have understood or at least found it obvious that **the set of behavior rules includes behavior rules compiled by the pestware management engine**.

C. Claim 3

115. Claim 3 of the '505 Patent recites:

> **3**. The method of claim **1**, wherein the record of events includes information selected from a group consisting of: process identification information, file identification information, and hook generation information.

116. As I described above with respect to [1d], Shukla describes that its kernel-level component works by hooking in various API calls within the kernel that allows Shukla's malware detection system to ***observe*** and control the execution of these API calls. *See* EX1005, [0043], [0056]; EX1006, 11, 13. Shukla describes a "Learn mode," in which "kernel functions are hooked and the behavior of the application is monitored." EX1005, [0110]; EX1006, 24. This "Learn mode" is "used to silently watch the access made by the application without enforcing any rules." *Id.* The claims of Shukla refer to these steps as "recording" the events (*e.g.*, events such as "writing executable code to disk or memory, creating startup items, starting new services or drivers or application") and "generating a report." *See* EX1005, claim 52; EX1006, claim 52.

117. According to Shukla, the "[t]arget functions" that the kernel-level component hooks "are primarily related to file manipulation, process creations, network access, registry access, memory access etc." EX1005, [0056]; EX1006, 13. "Once an application or a module is classified as malicious, ***its [] disk, in registry,***

and in memory attributes are recorded and used to neutralized and remove it." EX1005, [0063]; EX1006, 14-15. Similarly, Shukla describes that "[o]nce a hooked module is discovered, ***it is flagged*** and its execution path is traced to discover any other modules that may be hooked into that function." EX1005, [0073]-[0074]; EX1006, 16-17.

118. From these teachings, a POSITA have understood or at least found it obvious that "the record of events includes information selected from a group consisting of: process identification information, file identification information, and hook generation information" (as recited in claim 3), as these various information are consistent with the information described as being record in Shukla (e.g., the "disk, in registry, and in memory attributes") and would have allowed the Field-Glick-Shukla combination "to neutralize[] and remove" the malware that is detected, as is taught by Field and Shukla. *See* EX1004, [0022], [0034]; EX1005, [0043], [0056], [0110], claim 52; EX1006, 11, 13, 24, claim 52.

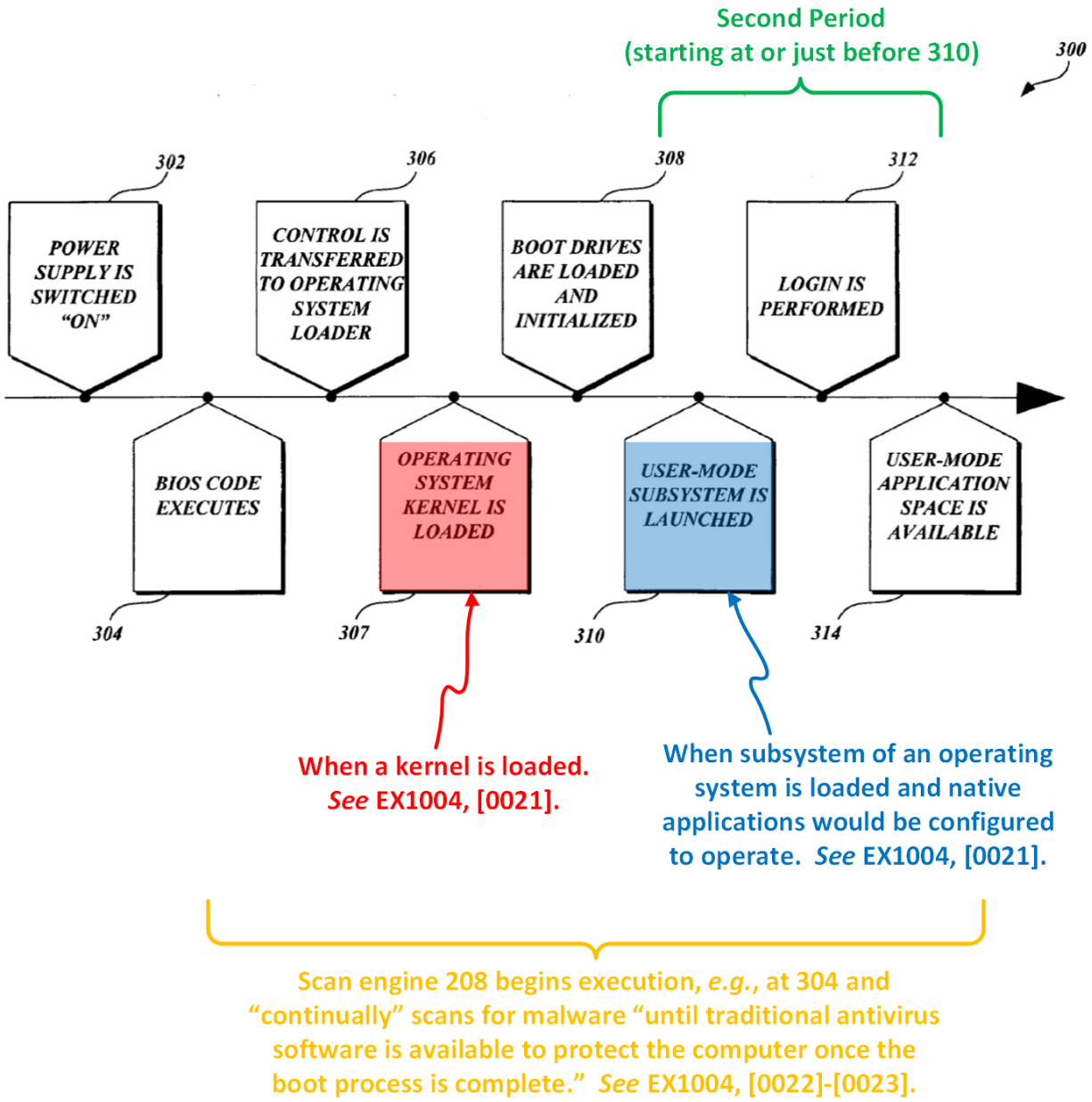
D. Claim 4

119. Claim 4 of the '505 Patent recites:

> **4**. The method of claim **1**, further comprising:
> managing pestware-related events after the computer becomes configured to run native applications.

120. As described above with respect to [1c], FIG. 4 of the '505 patent shows that an operating system is able to run native applications after drivers are loaded and before the native operating system starts its subsystem.

121. A POSITA would have understood or at least found it obvious that Field's computer becomes configured to run native applications around event 310. For example, Field describes that "the user mode subsystem provides support services to the ***user mode application space***." EX1004, [0021]. "Once the user mode services are available, at event 314, the user-mode application space may be accessed to execute programs. Once event 314 is reached, programs designed to perform specific tasks on a general purpose computer may be executed." *Id.* For the reasons explained with respect to [1c], a POSITA would have understood or at least found it obvious that "native applications" would have been configured to run at approximately the same time and certainly no later than when the "user-mode applications" described by Field would have been configured to run, consistent with the following annotation of Field's FIG. 3.



122. Field discloses that "is desirable to start scanning for malware as early as possible in the boot process," and that this scanning continues throughout the boot process "until traditional antivirus software is available to protect the computer once the boot process is complete." EX1004, [0022]-[0023]. From this a POSITA would have understood or at least found it obvious that Field's scan engine 208 and boot

detection module 210 would have been configured to **manage pestware-related events after the computer becomes configured to run native applications**.

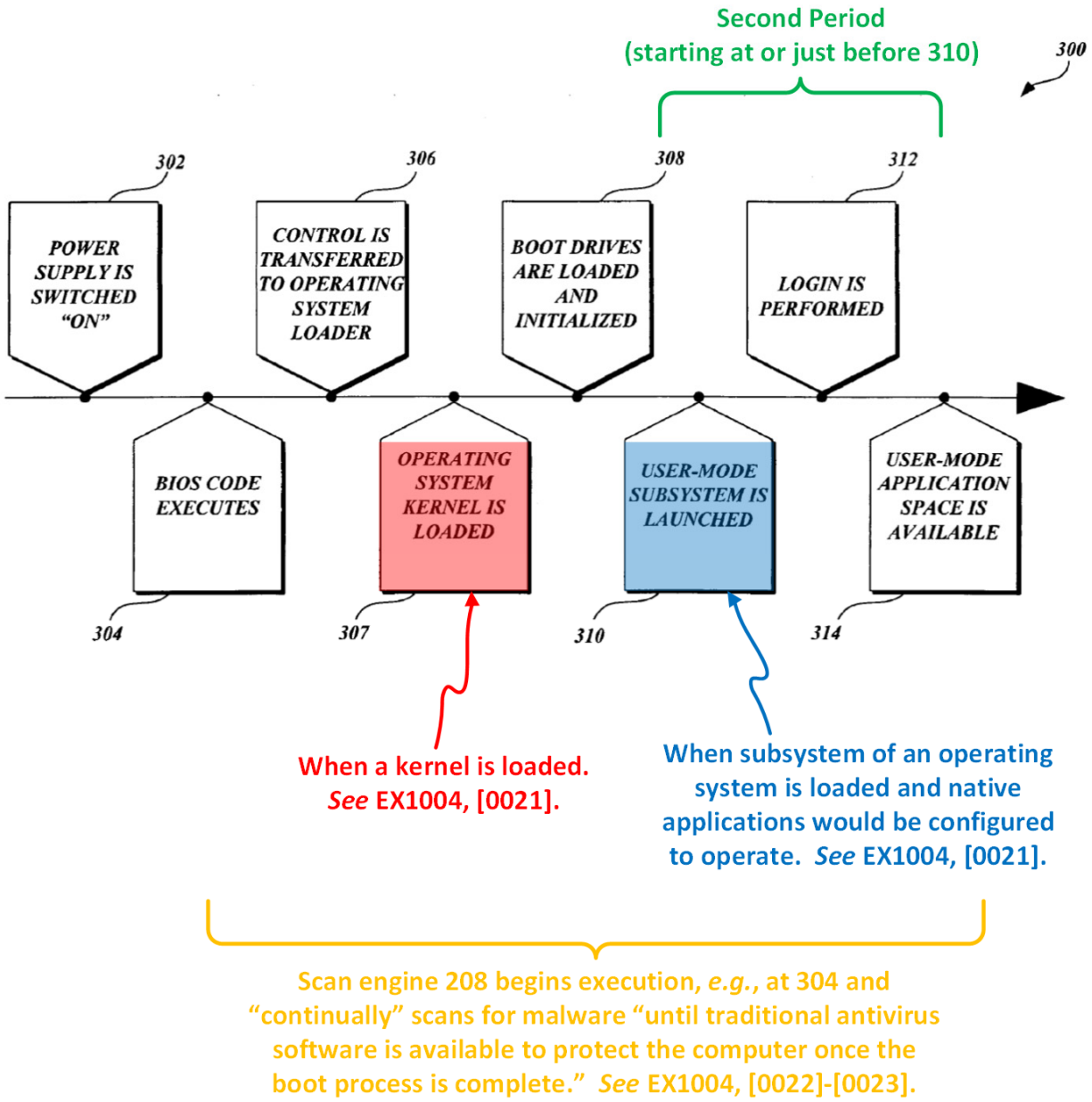
123. Furthermore, as described above with respect to the Field, Glick, and Shukla combination, a POSITA would have found it obvious that Field's scan engine 208 and boot detection module 210 would coordinate with user-mode/application-level components that perform the "traditional antivirus" functionality to which Field refers. *See* EX1004, [0022], [0034]. To this point, both Glick and Shukla describe that the kernel-level components of a malware detection system may continue to run after the boot sequence to coordinate with the application-level components of the system and extend the functionality of the application-level components during normal computer operation. For example, Glick describes that its kernel mode driver 106B is "loaded prior to all user mode software applications during boot up of host computer system 102" (*see* EX1007, 6:23-32) and that it continues to run after the boot sequence to assist the user mode application 106A to detect and mitigate malware (*see* EX1007, 12:50-13:44, 18:6-33 (describing various "Request Kernel Mode Assistance" operations)). From these teachings a POSITA would have found it obvious that Field's scan engine 208 and boot detection module 210 would have been configured to continue executing and managing pestware-related events after the boot sequence and during the normal operation of the computer.

E. Claim 5

124. Claim 5 of the '505 Patent recites:

> **5**. The method of claim **1**, wherein the monitoring includes monitoring the boot sequence while boot drivers are initiated.

125. As I discussed above with respect to 1[c], Field teaches operating ***while the boot drivers are initiated***. For example, Field teaches that its scan engine "may be loaded into memory" at "any number of different locations in the timeline" and may "begin executing at events 304, 306, or 310." EX1004, [0022]-[0023]. This is illustrated in the following annotation of FIG. 3.



126. As shown in the above annotation and described in Field, boot drivers are initiated during event 308, which is after event 304 when the scan engine 208 and boot detection module 210 would have begun executing. *See* EX1004, FIG. 3, [0021]-[0023]. Accordingly, in the Field-Glick-Shukla combination **the**

monitoring includes monitoring the boot sequence while boot drivers are initiated.

F. Claim 6

127. Claim 6 of the '505 Patent recites:

> **6**. A system for managing pestware on a computer comprising:
> a processor;
> a memory comprising:
> a first instruction set configured to monitor events during a boot sequence of the computer;
> a second instruction set configured to manage pestware-related events during first period in a boot sequence of the computer, the first period in the boot sequence occurring before the computer becomes configured to run native applications, before a WIN32 subsystem is loaded, and after a kernel is loaded;
> a third instruction set configured to manage pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occurring when the computer is configured to run native applications;
> a fourth instruction set configured to generate, in response to the monitoring, a record of events, the record of events including the pestware-related events;
> a fifth instruction set configured to analyze the record of events so as to identify the pestware-related events; and
> a sixth instruction set configured to modify the set of behavior rules so as to prevent the pestware related events; and
> a seventh instruction set configured to scan data in a registry of the computer for pestware during the second period in the boot sequence.

128. As I explain in more detail below through my feature-by-feature analysis, Field, Glick, and Shukla disclose or render obvious claim 6.

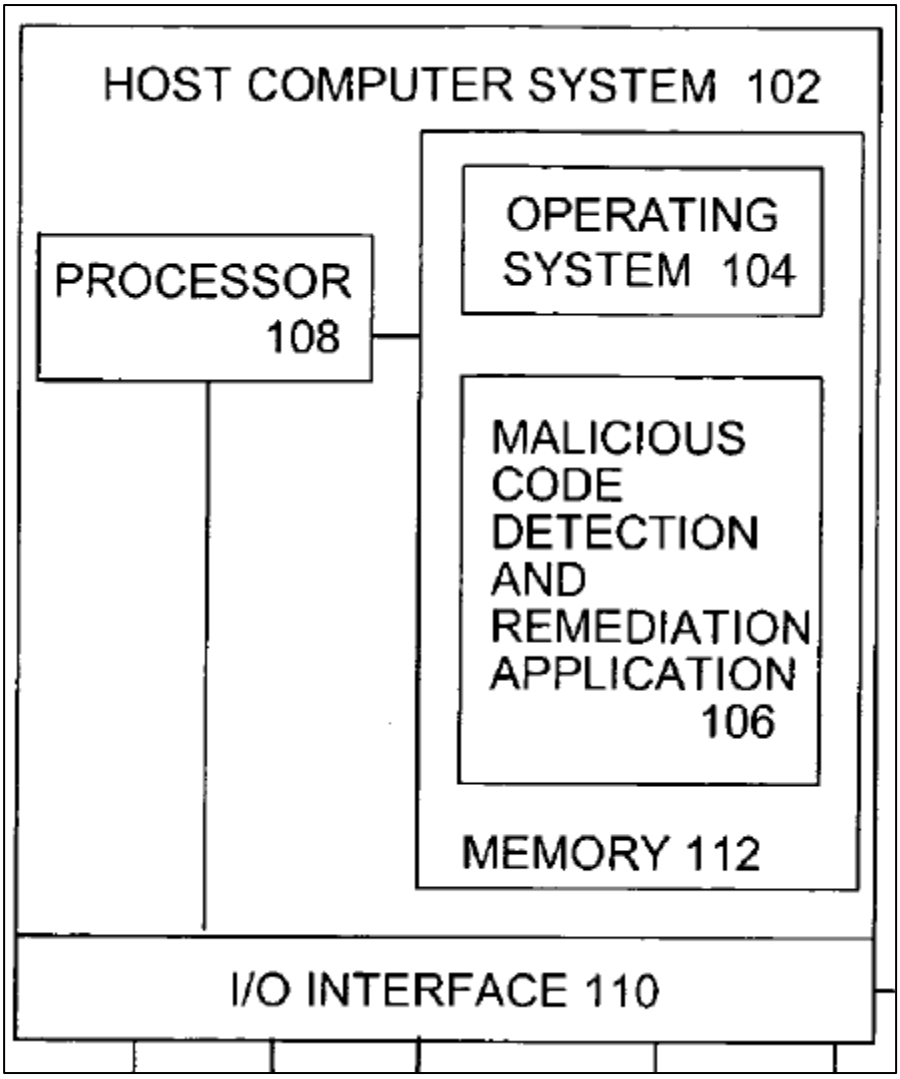
1. 6[pre] A system for managing pestware on a computer comprising:

129. This limitation is substantially that same as [1pre] and is obvious over the Field-Glick-Shukla combination for at least similar reasons.

2. 6[a] a processor;

130. Field describes that its system operates on a typical computer 200 that "may be any one of a variety of devices including, but not limited to, personal computing devices, server-based computing devices, mini- and main frame computers, laptops, personal digital assistants ('PDAs'), set-top boxes, entertainment and gaming systems, or other electronic devices having some type of memory." EX1004, [0014]. Field describes that these types of computers typically included "a keyboard, a mouse, a printer, a display, ***CPU***, memory, etc." *Id.* Thus, Field describes that the scan engine 208 and a boot detection module 210 are configured to run on any typical computer 200, and that computer would include a processor (e.g., a CPU). *Id*.

131. Glick's Figure 1 similarly shows a "malicious code detection and remediation application" on a host computer system with a central processing unit (CPU) 108. EX1007, FIG. 1, 2:3-13.

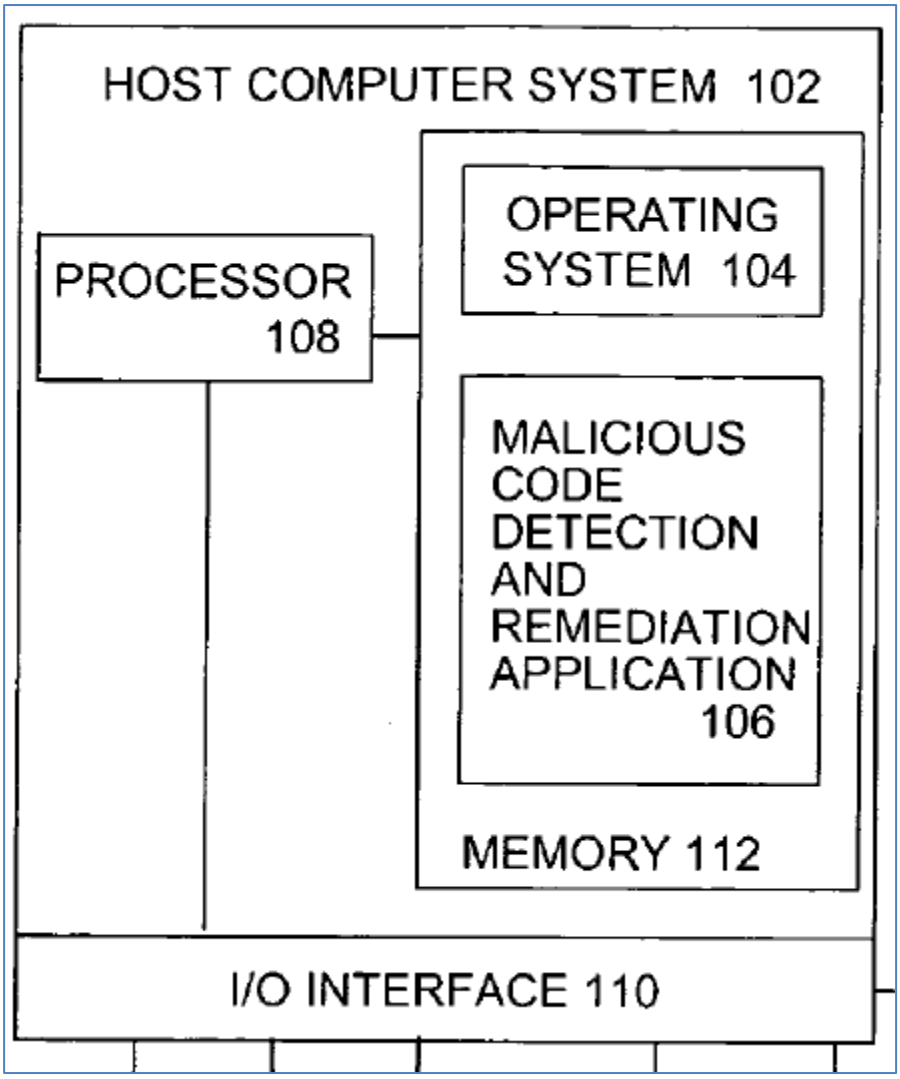


3. 6[b] a memory comprising:

132. As noted above, Field describes that its system operates on a typical computer 200 that "may be any one of a variety of devices including, but not limited to, personal computing devices, server-based computing devices, mini- and main frame computers, laptops, personal digital assistants ('PDAs'), set-top boxes, entertainment and gaming systems, or other electronic devices having some type of memory." EX1004, [0014]. Field describes that these types of computers typically

included "a keyboard, a mouse, a printer, a display, CPU, ***memory***, etc." *Id.* Indeed, Field teaches that "the boot detection module causes the scan engine 208 to be loaded into memory." EX1004, [0016].

133. Glick's Figure 1 similarly shows a "malicious code detection and remediation application" on a host computer system with a memory 112, and that the "malicious code detection and remediation application 106 is typically stored in memory 112 of host computer system 102 and executed on host computer system 102." EX1007, FIG. 1, 2:3-13, 3:8-11.

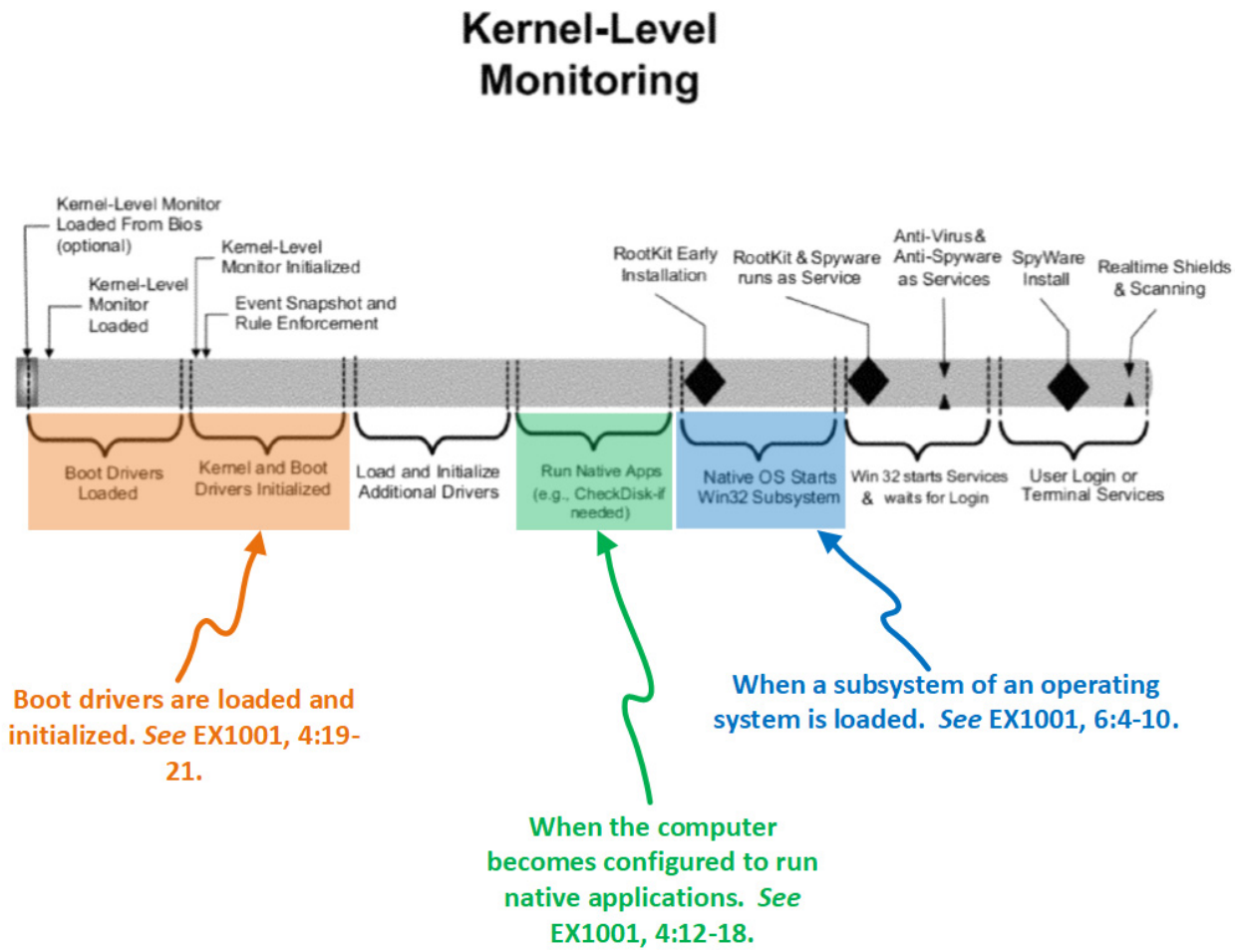


4. 6[b.i] a first instruction set configured to monitor events during a boot sequence of the computer;

134. This limitation is substantially that same as 1[a] and is obvious over the Field-Glick-Shukla combination for at least similar reasons.

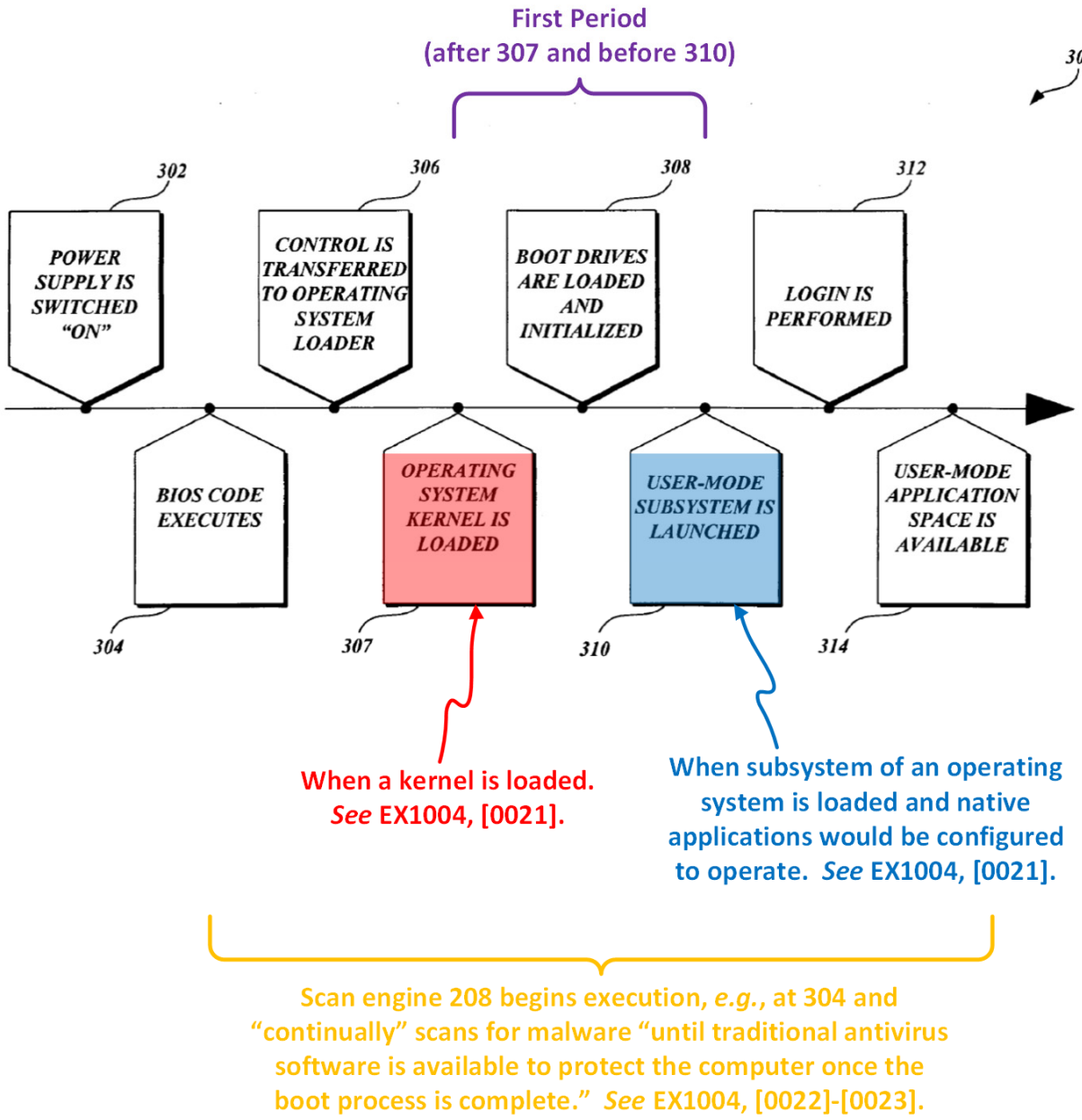
5. 6[b.ii] a second instruction set configured to manage pestware-related events during first period in a boot sequence of the computer, the first period in the boot sequence occurring before the computer becomes configured to run native applications, before a WIN32 subsystem is loaded, and after a kernel is loaded

135. This limitation is substantially that same as 1[b] and is obvious over the Field-Glick-Shukla combination for at least similar reasons. Notably, this limitation specifies that the first period is "before a WIN32 subsystem is loaded," where as 1[b] specifies that the first period is "before a sub-system of an operating system is loaded." As described with respect to 1[b], the sub-system of an operating system is loaded at event 310. *See* EX1004, [0021].



136. Field does not explicitly describe in its timeline when the WIN32 subsystem is loaded, but a POSITA would have understood it to be at or around the same time as event 310. As I described with respect to 1[c], the timeline in FIG. 4 of the '505 Patent describes effectively the same sequence of events in a standard boot process and the timeline in FIG. 3 of Field. The methods and systems in each of the '505 Patent and Field are described as being operable on general purpose computers that run a variety of operating systems, including Microsoft Windows. EX1001, 3:5-8 ("And if implemented in software, the modules can be designed to

operate on any type of computer system including WINDOWS and Linux-based systems"); EX004, [0015] ("The operating system 204 illustrated in FIG.2 may be a general-purpose operating system such as a Microsoft® operating system, UNIX® operating system, or Linux® operating system."). Thus, while the exact events indicated on each of the timelines are not identical, a POSITA would nevertheless have understood the two timelines as depicting the same general boot process flow, which would have been specific to a given operating system and computer configuration and independent of the monitoring system/method. Based on the '505 Patent description of when the WIN32 subsystem is loaded, a POSITA would have understood this to be at or around the same time as event 310 in Field's FIG. 3. *Compare* EX1001, 6:4-10 *to* EX1004, [0021].



137. Thus, as shown in the above annotation of FIG. 3, where a POSITA configured the scan engine 208 to begin executing at events 304 or 306 per the explicit teachings of Field, the scan engine 208 would be executing (e.g., managing pestware-related events by scanning for malware and removing or quarantining identified malware) during a **first period** that is **before the computer becomes**

configured to run native applications, before a WIN32 subsystem is loaded (event 310), and **after a kernel is loaded (event 307)**.

6. 6[b.iii] a third instruction set configured to manage pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occurring when the computer is configured to run native applications;

138. This limitation is substantially that same as 1[c] and is obvious over the Field-Glick-Shukla combination for at least similar reasons.

7. 6[b.iv] a fourth instruction set configured to generate, in response to the monitoring, a record of events, the record of events including the pestware-related events;

139. This limitation is substantially that same as 1[d] and is obvious over the Field-Glick-Shukla combination for at least similar reasons.

8. 6[b.v] a fifth instruction set configured to analyze the record of events so as to identify the pestware-related events; and

140. This limitation is substantially that same as 1[e] and is obvious over the Field-Glick-Shukla combination for at least similar reasons.

9. 6[b.vi] a sixth instruction set configured to modify the set of behavior rules so as to prevent the pestware related events; and

141. This limitation is substantially that same as 1[f] and is obvious over the Field-Glick-Shukla combination for at least similar reasons.

10.6[b.vii] a seventh instruction set configured to scan data in a registry of the computer for pestware during the second period in the boot sequence.

142. This limitation is substantially that same as 1[g] and is obvious over the Field-Glick-Shukla combination for at least similar reasons.

G. Claim 7

143. Claim 7 of the '505 Patent recites:

> **7**. The system of claim **6**, wherein the memory further comprises an eighth instruction set configured to launch, after an operating system is initiated, a pestware management engine; and wherein the set of behavior rules includes behavior rules compiled by the pestware management engine.

144. This limitation is substantially that same as [2] and is obvious over the Field-Glick-Shukla combination for at least similar reasons.

H. Claim 8

145. Claim 8 of the '505 Patent recites:

> **8**. The system of claim **1**, wherein the record of events includes information selected from a group consisting of: process identification information, file identification information, and hook generation information.

146. This limitation is substantially that same as [3] and is obvious over the Field-Glick-Shukla combination for at least similar reasons.

I. Claim 9

147. Claim 9 of the '505 Patent recites:

> **9**. The system of claim **6**, wherein the memory further comprises:
>
> a ninth instruction set configured to manage pestware-related events after the computer becomes configured to run native applications.

148. This limitation is substantially that same as [4] and is obvious over the Field-Glick-Shukla combination for at least similar reasons.

J. Claim 10

149. Claim 10 of the '505 Patent recites:

> **10**. The system of claim **6**, wherein the first instruction set is further configured to monitor the boot sequence while boot drivers are initiated.

150. This limitation is substantially that same as [5] and is obvious over the Field-Glick-Shukla combination for at least similar reasons.

K. Claim 11

151. Claim 11 of the '505 Patent recites:

11. A non-transitory tangible computer-readable storage medium comprising program instructions for:
monitoring events during a boot sequence of a computer;
managing pestware-related events during a first period in a boot sequence of the computer, the first period in the boot sequence occurring before the computer becomes configured to run native applications, before a WIN32 subsystem is loaded, and after a kernel is loaded;
managing pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occurring when the computer is configured to run native applications;
generating, in response to the monitoring, a record of events, the record of events including the pestware-related events;
analyzing the record of events so as to identify the pestware-related events;
modifying the set of behavior rules so as to prevent the pestware related events; and
scanning data in a registry of the computer for pestware during the second period in the boot sequence.

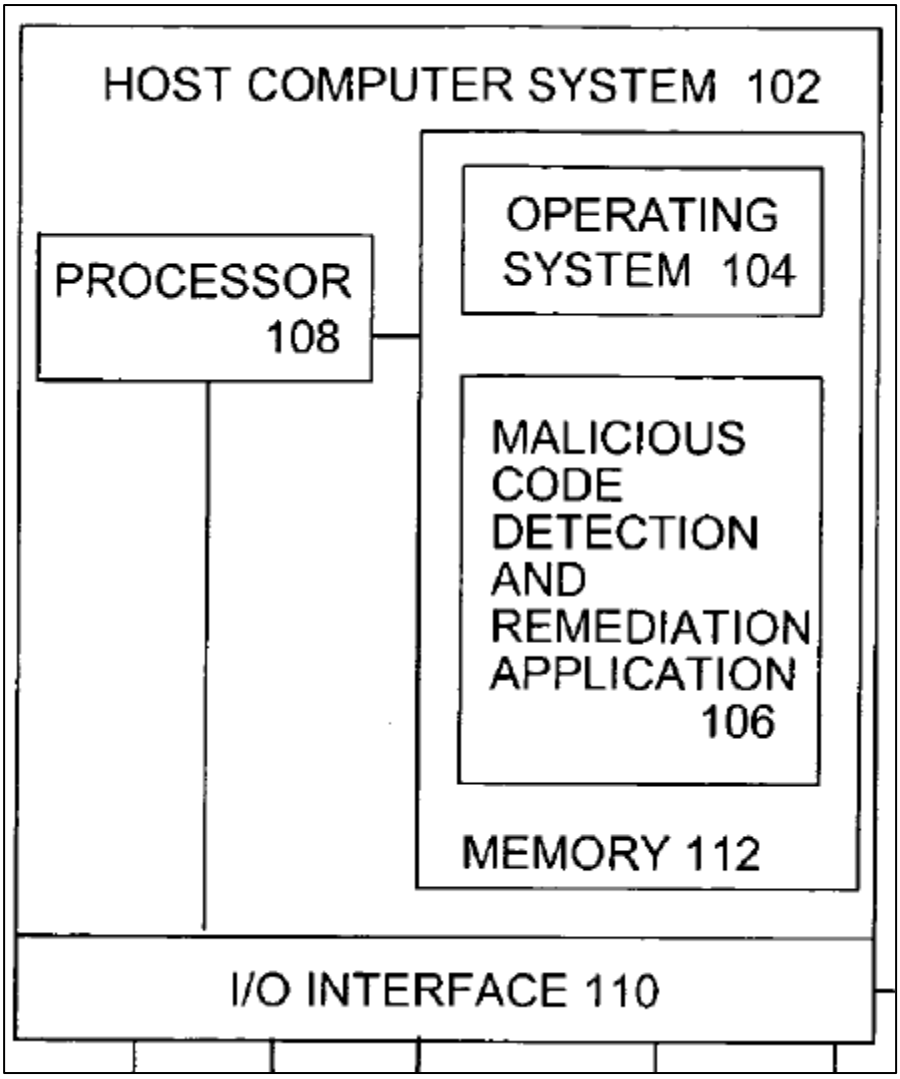
152. As I explain in more detail below through my feature-by-feature analysis, Field, Glick, and Shukla disclose or render obvious claim 11.

1. 11[pre] A non-transitory tangible computer-readable storage medium comprising program instructions for:

153. Field describes that its system operates on a typical computer 200 that "may be any one of a variety of devices including, but not limited to, personal computing devices, server-based computing devices, mini- and main frame computers, laptops, personal digital assistants ('PDAs'), set-top boxes, entertainment and gaming systems, or other electronic devices having some type of memory." EX1004, [0014]. Field describes that these types of computers typically

included "a keyboard, a mouse, a printer, a display, CPU, ***memory***, etc." *Id.* Indeed, Field teaches that "the boot detection module causes the scan engine 208 to be loaded into memory." EX1004, [0016]. A POSITA would have understood or at least found it obvious that Field's memory was a non-transitory tangible computer-readable storage medium comprising program instructions.

154. Glick's Figure 1 similarly shows a "malicious code detection and remediation application" on a host computer system with a memory 112, and that the "malicious code detection and remediation application 106 is typically stored in memory 112 of host computer system 102 and executed on host computer system 102." EX1007, FIG. 1, 2:3-13, 3:8-11. A POSITA would have understood or at least found it obvious that Glick's memory was a non-transitory tangible computer-readable storage medium comprising program instructions.



2. 11[a] monitoring events during a boot sequence of a computer;

155. This limitation is substantially that same as 1[a] and is obvious over the Field-Glick-Shukla combination for at least similar reasons.

3. 11[b] managing pestware-related events during a first period in a boot sequence of the computer, the first period in the boot sequence occurring before the computer becomes configured to run native applications, before a WIN32 subsystem is loaded, and after a kernel is loaded;

156. This limitation is substantially that same as 1[b] and 6[b.ii], and is obvious over the Field-Glick-Shukla combination for at least similar reasons.

4. 11[c] managing pestware-related events in accordance with a set of behavior rules during a second period in the boot sequence occurring when the computer is configured to run native applications;

157. This limitation is substantially that same as 1[c], and is obvious over the Field-Glick-Shukla combination for at least similar reasons.

5. 11[d] generating, in response to the monitoring, a record of events, the record of events including the pestware-related events;

158. This limitation is substantially that same as 1[d], and is obvious over the Field-Glick-Shukla combination for at least similar reasons.

6. 11[e] analyzing the record of events so as to identify the pestware-related events;

159. This limitation is substantially that same as 1[e], and is obvious over the Field-Glick-Shukla combination for at least similar reasons.

7. 11[f] modifying the set of behavior rules so as to prevent the pestware related events; and

160. This limitation is substantially that same as 1[f], and is obvious over the Field-Glick-Shukla combination for at least similar reasons.

8. 11[g] scanning data in a registry of the computer for pestware during the second period in the boot sequence.

161. This limitation is substantially that same as 1[g], and is obvious over the Field-Glick-Shukla combination for at least similar reasons.

L. Claim12

162. Claim 12 of the '505 Patent recites:

> **12**. The non-transitory tangible computer-readable storage medium of claim **11** further comprising program instructions for launching, after an operating system is initiated, a pest-ware management engine; and
> wherein the set of behavior rules includes behavior rules compiled by the pestware management engine.

163. This limitation is substantially that same as [2] and is obvious over the Field-Glick-Shukla combination for at least similar reasons.

M. Claim 13

164. Claim 13 of the '505 Patent recites:

> **13**. The non-transitory tangible computer-readable storage medium of claim **11**, wherein the record of events includes information selected from a group consisting of: process identification information, file identification information, and hook generation information.

165. This limitation is substantially that same as [3] and is obvious over the Field-Glick-Shukla combination for at least similar reasons.

N. Claim 14

166. Claim 14 of the '505 Patent recites:

> **14**. The non-transitory tangible computer-readable storage medium of claim **11** further comprising program instructions for:
>
> managing pestware-related events after the computer becomes configured to run native applications.

167. This limitation is substantially that same as [4] and is obvious over the Field-Glick-Shukla combination for at least similar reasons.

O. Claim 15

168. Claim 15 of the '505 Patent recites:

> **15**. The non-transitory tangible computer-readable storage medium of claim **11**, wherein the monitoring includes monitoring the boot sequence while boot drivers are initiated.

169. This limitation is substantially that same as [5] and is obvious over the Field-Glick-Shukla combination for at least similar reasons.

P. Claim 16

170. Claim 16 of the '505 Patent recites:

> **16**. The method of claim **2**, further comprising modifying, by the pestware management engine, the behavior rules utilized by a kernel-level monitor, wherein the kernel-level monitor manages pestware-related events in the first and second periods in the boot sequence.

171. As I described above with respect to [1f], a POSITA would have understood that iteratively observing new behavior and assessing it would have allowed the system of the Field-Glick-Shukla combination to better adapt to new

threats by allowing for the creation of new behavior rules and the modification of old behavior rules. *See* EX1005, [0078]; EX1006, 18. As Shukla recognized, using rigid signature-based detection techniques was "ineffective" (*see* EX1005, [0078]; EX1006, 18) and that using rigid behavior rules leads to a malware detection system that is either overly burdensome on the user or overly permissive to malware (*see* EX1005, [0119]-[0123]; EX1006, 25-26). Thus, Shukla teaches the benefits of flexibility in the creation, maintenance, and application of behavior rules in order to adapt to new malware threats. From these teachings, a POSITA would have found it obvious for the Field-Glick-Shukla combination to **modify[] the set of behavior rules so as to prevent the pestware related events**.

172. As described with respect to claim 2, given that the application-level components of these types of malware detection systems were known to have broader processing and remediation capabilities (*see, e.g.*, EX1004, [0034]), a POSITA would have understood or at least found it obvious that the user-mode/application-level components of the Field-Glick-Shukla combination would have been configured to compile the behavior rules. Indeed, this is consistent with Glick's general descriptions of its user mode application 106A performing the primary analysis and coordination functions, and calling upon the kernel mode driver 106B for specific functional assistance. *See, e.g.*, EX1007, 12:50-13:44, 18:6-33. Thus, in the Field-Glick-Shukla combination, a POSITA would have understood or

at least found it obvious that user-mode/application-level components (i.e., pestware management engine) would have managed the set of behavior rules, including the modification of those rules based on the mathematical/statistical risk analysis described by Shukla. EX1005, [0110]-[0118].

173. As I described above with respect to [1b] and [1c], a POSITA would have found it obvious that Field's scan engine 208 and a boot detection module 210—which together operate as a kernel-level monitor (*see* EX1004, [0005], [0017]; EX1007, 6:23-32)—would utilize the above-described behavior rules to **manage[] pestware-related events in the first and second periods in the boot sequence**. Field describes that "the scan engine may ***search for suspicious activities*** such as jump instructions in unexpected locations, hidden processes, references to memory addresses that are outside of a range allocated to the operating system, and the like" and that "[a]ntivirus software ***may identify activity*** that could be characteristic of malware." EX1004, [0017], [0027]. In one example, Field describes that "[r]emoving an entry from a process table or a similar data structure may be ***a strong heuristic indicator*** that a RootKit is present." EX1004, [0017]. While Field does not specifically describe this functionality as the use of behavior rules to managing pestware-related events, a POSITA would nonetheless have understood or at least found it obvious that Field's search for suspicious activities would have included the use of behavior rules based on the teachings of Shukla.

174. Shukla describes that its kernel-level component "hooks into kernel-mode API function calls or even unexported API calls and monitors various components of the kernel." EX1005, [0041]. Shukla teaches that the kernel-level component enforces a set of behavior rules "blocking, permitting, or monitoring the API function calls." EX1005, [0040], *see also id.*, [0043]-[0057]. Thus, in the Field-Glick-Shukla combination, a POSITA would have found it obvious that Field's scan engine 208 and a boot detection module 210, modified to utilize behavior rules based on the teachings of Shukla, were a kernel-level monitor that utilize behavior rules to manage pestware-related events in the first and second periods in the boot sequence.

Q. Claim 17

175. Claim 17 of the '505 Patent recites:

> **17**. The method of claim **1**, further comprising modifying the set of behavior rules based on the scanned registry so as to prevent the pestware-related events.

176. As I described with respect to [1g], the Field-Glick-Shukla combination scans data in a registry of the computer for pestware during the second period in the boot sequence. Shukla describes that, in "Learn mode," the analysis performed by the kernel-mode portion of the sandbox may be used to generate and test the behavior rules employed by Shukla's overall malware detection system. *See* EX1005, [0110]-[0118]; EX1006, 24-25. From these disclosure a POSITA would have understood or at least found it obvious that **modifying the set of behavior rules based on the**

scanned registry so as to prevent the pestware-related events, because Shukla teaches that registry scanning is one of the analyses performed by Shukla's kernel module, and a POSITA would have understood or at least found it obvious to use this analysis during "Learn mode" to develop the behavior rules. *Id.*

R. Claim 18

177. Claim 18 of the '505 Patent recites:

> **18**. The method of claim **1**, wherein the pestware-related invents include events associated with at least one of loading pestware, executing pestware, and one or more behaviors of pestware.

178. Field teaches monitoring **events associated with… loading pestware, executing pestware, and one or more behaviors of pestware.** For example, Field teaches that a scan engine starts executing early in the boot process so that "subsequently initialized boot drivers are scanned for malware as they are loaded into memory." EX1004, [0030]. A POSITA would have understood or at least found obvious from this teaching that pestware being loaded into memory is a recognized event. Field also teaches "one or more" other "behaviors of pestware," when it states that "some malware implement self-preservation techniques in which the malware's resources (e.g. files, processes, entries in configuration files, and the like) are monitored. When an attempt to remove the malware is identified, functionality

designed to preserve the malware's resources and maintain the infection are performed." EX1004, [0034].

179. Similarly, Glick teaches monitoring **pestware-related events associated with… loading pestware**. For example, Glick teaches that "In MALICIOUS CODE DETECTED check operation 610, user mode application 106A evaluates the list of running processes and/or dlls loaded into each running process 608 and determines whether malicious code is detected on host computer system 102. EX1007, 11:7-31.

S. Claim 19

180. Claim 19 of the '505 Patent recites:

> **19**. The system of claim **6**, wherein the pestware-related invents include events associated with at least one of loading pestware, executing pestware, and one or more behaviors of pestware.

181. This limitation is substantially that same as [18] and is obvious over the Field-Glick-Shukla combination for at least similar reasons.

T. Claim 20

182. Claim 20 of the '505 Patent recites:

> **20**. The non-transitory tangible computer-readable storage medium of claim **11**, wherein the pestware-related invents include events associated with at least one of loading pestware, executing pestware, and one or more behaviors of pestware.

183. This limitation is substantially that same as [18] and is obvious over the Field-Glick-Shukla combination for at least similar reasons.

X. ADDITIONAL REMARKS

I currently hold the opinions expressed in this declaration. But my analysis may continue, and I may acquire additional information and/or attain supplemental insights that may result in added observations.

I hereby declare that all statements made are of my own knowledge are true and that all statements made on information and belief are believed to be true. I further declare that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of the Title 18 of the United States Code and that such willful false statements may jeopardize the validity of this proceeding.

Dated: June 30, 2023 By: 

Dr. Markus Jakobsson

APPENDIX A

CV and Research Statement

Markus Jakobsson
www.linkedin.com/in/markusjakobsson
www.markus-jakobsson.com

1 At a Glance

- **Focus.** *Identification of security problems, trends and solution along four axes* – computational, structural, physical *and* social*; quantitative and qualitative fraud analysis; development of disruptive security technologies.*
- **Education.** *PhD* (Computer Science/Cryptography, University of California at San Diego, 1997); *MSc* (Computer Science, University of California at San Diego, 1994); *MSc* (Computer Engineering, Lund Institute of Technology, Sweden, 1993).
- **Large research labs.** *San Diego Supercomputer Center* (Researcher, 1996-1997); *Bell Labs* (Member of Technical Staff, 1997-2001); *RSA Labs* (Principal Research Scientist, 2001-2004); *Xerox PARC* (Principal Scientist, 2008-2010); PayPal (Principal Scientist of Consumer Security, Director, 2010-2013); Qualcomm (Senior Director, 2013-2015); Agari (Chief Scientist, 2016–2018); Amber Solutions Inc (Chief of Security and Data Analytics, 2018 – 2019); ByteDance (Principal Scientist, 2020-2021)
- **Academia.** *New York University* (Adjunct Associate Professor, 2002-2004); *Indiana University* (Associate Professor & Associate Director, 2004-2008; Adjunct Associate Professor, 2008-2016).
- **Entrepreneurial activity.** *ZapFraud* (Anti-scam technology; CTO and founder, 2012-current); *RavenWhite Security* (Authentication solutions; CTO and founder, 2005-); *RightQuestion* (Consulting; Founder, 2007-current); *FatSkunk* (Malware detection; CTO and founder, 2009-2013 – FatSkunk was acquired by Qualcomm); *LifeLock* (Id theft protection; Member of fraud advisory board, 2009-2013); *CellFony* (Mobile security; Member of technical advisory board, 2009-2013); *PopGiro* (User Reputation; Member of technical advisory board, 2012-2013); *MobiSocial* (Social networking, Member of technical advisory board, 2013); *Cequence Security* (Anti-fraud, Member of technical advisory board, 2013–current)
- **Anti-fraud consulting.** *KommuneData* [Danish govt. entity] (1996); *J.P. Morgan Chase* (2006-2007); *PayPal* (2007-2011); *Boku* (2009-2010); *Western Union* (2009-2010).

- **Intellectual Property, Testifying Expert Witness.** *Inventor of 100+ patents; expert witness in 25+ patent litigation cases* (McDermott, Will & Emery; Bereskin & Parr; WilmerHale; Hunton & Williams; Quinn Emanuel Urquhart & Sullivan; Freed & Weiss; Berry & Domer; Fish & Richardson; DLA Piper; Cipher Law Group; Keker & Van Nest). Details and references upon request.

- **Publications**. Books: *Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft* (Wiley, 2006); *Crimeware: Understanding New Attacks and Defenses* (Symantec Press, 2008); *Towards Trustworthy Elections: New Directions in Electronic Voting* (Springer Verlag, 2010); *Mobile Authentication: Problems and Solutions*)Springer Verlag, 2012); *The Death of the Internet* (Wiley, 2012); *Understanding Social Engineering* (Springer Verlag, 2016); *Security, Privacy and User Interaction* (Springer Verlag, 2020); *100+ peer-reviewed publications*

2 At a Glance

Ten years before Bitcoin was created, I formalized the notion of Proof of Work and described its use for mining of crypto payments. I later developed energy-efficient alternatives to this paradigm, and showed how to enable mining on mobile devices, which is not possible for Bitcoin. I am the founder of the academic discipline of phishing and have developed techniques to predict fraud trends years before they emerge, enabling countermeasures to be developed before they are needed. I developed the notion of implicit authentication, which is now ubiquitous; I also founded a company that developed the first retroactive virus detection technology, with guarantees of detection; the company was acquired by Qualcomm in 2013. I have worked as chief scientist and similar positions in startups as well as industry behemoths, such as PayPal. I have several hundred patents to my name and am a prominent security researcher with hundreds of peer reviewed publications and an array of textbooks. My 1997 PhD thesis, from University of California at San Diego, was on distributed electronic payment systems with revocable privacy.

3 Work History (Highlights)

1. **Member of Technical Staff, Bell Labs (1997-2001).** Markus was part of the security research group at Bell Labs. He formalized the notion of *proof of work*, later an integral part of BitCoin.

2. **Principal Scientist, RSA Labs (2001-2005).** Markus posited that phishing would become a mainstream problem, and developed ethical techniques for identifying likely trends based on human subject experiments.

3. **Associate Professor, Indiana University (2005-2008).** Markus was hired to lead the newly formed security group at Indiana University, and

created a research group comprising approximately 10 professors and 30 students, studying social engineering and fraud.

4. **Principal Scientist, Xerox PARC (2008-2010).** Markus was hired to lead the research efforts of the Xerox PARC security group, and developed the notion of *implicit authentication*, a technology that is now ubiquitous.

5. **Principal Scientist, PayPal (2010-2013).** Markus did research on security and user interfaces, and developed techniques to reduce the losses associated with *liar buyer fraud*.

6. **Senior Director, Qualcomm (2013-2016).** Qualcomm acquires FatSkunk, a company founded by Markus. At FatSkunk, Markus developed *retroactive* malware detection with provable security guarantees. A simplified version of this is now deployed with almost all Qualcomm chipsets.

7. **Chief Scientist, Agari (2016-2018).** Markus developed a technique to acquire fraudster intelligence by compromising scammer email accounts – *while staying within the law* – resulting in the extradition of several African scam lords to the U.S.

8. **Chief of Security and Data Analytics, Amber Solutions (2018-2020).** Markus developed usable configuration methods supporting improved security and privacy for IoT installations.

9. **Chief Scientist, ByteDance (2020-2021).** Markus oversaw the establishment of a research group and a research agenda at ByteDance, and contributed to their intellectual property and product security.

10. **Chief Scientist, Artema LABS (2021-).** Managing the research, product strategy, and patent strategy for Artema LABS, a Los Angeles based startup in the Crypto/NFT sector.

4 Publication List

Books (1-8); book chapters, journals, conference publications and other scientific publications (9-147), issued /published U.S. patents (148-234). For an updated list, and for international patents, please see `www.markus-jakobsson.com/publications` and appropriate patent search engines.

References

[1] M. Jakobsson, *Security, Privacy and User Interaction*, ISBN 978-3-030-43753-4, 110 pages, 2020.

[2] M. Jakobsson, *Understanding Social Engineering Based Scams*, ISBN 978-1-4939-6457-4, 130 pages, Springer, 2016.

[3] M. Jakobsson, *Mobile Authentication: Problems and Solutions*, ISBN 1461448778, 125 pages, Springer, 2013.

[4] M. Jakobsson, (editor) *The Death of the Internet*, ASIN B009CN2JVE, 359 pages, IEEE Computer Society Press, 2012.

[5] D. Chaum, M. Jakobsson, R. L. Rivest, P. Y. Ryan, J. Benaloh, and M. Kutylowski, (editors), *Towards Trustworthy Elections: New Directions in Electronic Voting*, 411 pages, (Vol. 6000), Springer, 2010.

[6] M. Jakobsson and Z. Ramzan (editors), *Crimeware: Trends in Attacks and Countermeasures*, ISBN 0321501950, Hardcover, 582 pages, Symantec Press / Addison Wesley, 2008.

[7] M. Jakobsson and S. A. Myers (editors), *Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft*, ISBN 0-471-78245-9, Hardcover, 739 pages, Wiley, 2006.

[8] M. Jakobsson, M. Yung, J. Zhou, *Applied Cryptography and Network Security: Second International Conference , Yellow Mountain, China, 2004*, 511 pages, Lecture Notes in Computer Science (Book 3089), 2004.

[9] M. Jakobsson, "Permissions and Privacy," in IEEE Security & Privacy, vol. 18, no. 2, pp. 46-55, March-April 2020

[10] M. Jakobsson, "The Rising Threat of Launchpad Attacks," in IEEE Security & Privacy, vol. 17, no. 5, pp. 68-72, Sept.-Oct. 2019

[11] J Koven, C Felix, H Siadati, M Jakobsson, E Bertini, "Lessons learned developing a visual analytics solution for investigative analysis of scamming activities," IEEE transactions on visualization and computer graphics 25 (1), 225-234

[12] M Jakobsson, "Two-factor inauthentication?the rise in SMS phishing attacks" Computer Fraud & Security 2018 (6), 6-8

[13] M. Dhiman, M. Jakobsson, T.-F. Yen, "Breaking and fixing content-based filtering," 2017 APWG Symposium on Electronic Crime Research (eCrime), 52-56

[14] M. Jakobsson, "Addressing sophisticated email attacks," 2017 International Conference on Financial Cryptography and Data Security, 310-317

[15] M. Jakobsson, "User trust assessment: a new approach to combat deception," Proceedings of the 6th Workshop on Socio-Technical Aspects in Security and Trust, 2016, pages 73-78

[16] H. Siadati, T. Nguyen, P. Gupta, M. Jakobsson, "Mind your SMSes: Mitigating social engineering in second factor authentication," Computers and Security, 2016

[17] M. Jakobsson, W. Leddy, "Could you fall for a scam? Spam filters are passe. What we need is software that unmasks fraudsters," IEEE Spectrum 53 (5), 2016, 40-55

[18] N. Sae-Bae, M. Jakobsson, *Hand Authentication on Multi-Touch Tablets*, HotMobile 2014

[19] Y. Park, J. Jones, D. McCoy, E. Shi, M. Jakobsson, *Scambaiter: Understanding Targeted Nigerian Scams on Craigslist*, NDSS 2014

[20] D. Balfanz, R. Chow, O. Eisen, M. Jakobsson, S. Kirsch, S. Matsumoto, J. Molina, and P. van Oorschot, "The future of authentication," Security & Privacy, IEEE, 10(1), 22-27, 2012.

[21] M. Jakobsson, and H. Siadati, *Improved Visual Preference Authentication: Socio-Technical Aspects in Security and Trust*, (STAST), 2012 Workshop on IEEE, 27–34, 2012.

[22] M. Jakobsson, R. I. Chow, and J. Molina, "Authentication-Are We Doing Well Enough?[Guest Editors' Introduction]" Security & Privacy, IEEE, 10(1), 19-21, 2012.

[23] E. Shi, Y. Niu, M. Jakobsson, and R. Chow, "Implicit authentication through learning user behavior," Information Security, 99-113, Springer Berlin Heidelberg, 2011.

[24] M. Jakobsson and K. Johansson, "Practical and Secure Software-Based Attestation," Lightweight Security & Privacy: Devices, Protocols and Applications (LightSec), 1–9, 2011.

[25] A. Juels, D. Catalano, and M.Jakobsson, *Coercion-resistant electronic elections: Towards Trustworthy Elections*, 37–63, Springer Berlin Heidelberg, 2010.

[26] M. Jakobsson and F. Menczer, "Web Forms and Untraceable DDoS Attacks," in Network Security, Huang, S., MacCallum, D., and Du, D. Z., Eds.,77–95, Springer, 2010.

[27] R. Chow, M. Jakobsson, R. Masuoka, J. Molina, Y. Niu, E. Shi, and Z. Song, "Authentication in the Clouds: A Framework and its Application to Mobile Users," 2010.

[28] X. Wang, P. Golle, M. Jakobsson, and A. Tsow, "Deterring voluntary trace disclosure in re-encryption mix-networks," ACM Trans. Inf. Syst. Secur., 13(2), 1-24, 2010.

[29] X. Wang, P. Golle, M. Jakobsson, A.Tsow, "Deterring voluntary trace disclosure in re-encryption mix-networks," ACM Trans. Inf. Syst. Secur. 13(2): (2010)

[30] M. Jakobsson, and C. Soghoian, "Social Engineering in Phishing," Information Assurance, Security and Privacy Services, 4, 2009.

[31] M. Jakobsson, C. Soghoian and S. Stamm, "Phishing," Handbook of Financial Cryptography (CRC press, 2008)

[32] M. Jakobsson and A. Tsow, "Identity Theft," In John R. Vacca, Editor, "Computer And Information Security Handbook" (Morgan Kaufmann, 2008)

[33] S. Srikwan and M. Jakobsson, "Using Cartoons to Teach Internet Security," Cryptologia, vol. 32, no. 2, 2008

[34] M. Jakobsson, N. Johnson and P. Finn, "Why and How to Perform Fraud Experiments," IEEE Security and Privacy, March/April 2008 (Vol. 6, No. 2) pp. 66-68

[35] M. Jakobsson and S. Myers, "Delayed Password Disclosure," International Journal of Applied Cryptography, 2008, pp. 47-59.

[36] M. Jakobsson and S. Stamm, "Web Camouflage: Protecting Your Clients from Browser Sniffing Attacks," IEEE Security & Privacy Magazine. November/December 2007

[37] P. Finn and M. Jakobsson, "Designing and Conducting Phishing Experiments," IEEE Technology and Society Magazine, Special Issue on Usability and Security

[38] T. Jagatic, N. Johnson, M. Jakobsson and F. Menczer. "Social Phishing," The Communications of the ACM, October 2007

[39] A. Tsow, M. Jakobsson, L. Yang and S. Wetzel, "Warkitting: the Drive-by Subversion of Wireless Home Routers," Anti-Phishing and Online Fraud, Part II Journal of Digital Forensic Practice, Volume 1, Special Issue 3, November 2006

[40] M. Gandhi, M. Jakobsson and J. Ratkiewicz, "Badvertisements: Stealthy Click-Fraud with Unwitting Accessories," Anti-Phishing and Online Fraud, Part I Journal of Digital Forensic Practice, Volume 1, Special Issue 2, November 2006

[41] N. Ben Salem, J.-P. Hubaux and M. Jakobsson. "Reputation-based Wi-Fi Deployment," Mobile Computing and Communications Review, Volume 9, Number 3 (Best papers of WMASH 2004)

[42] N. Ben Salem, J. P. Hubaux, and M. Jakobsson. "Node Cooperation in Hybrid Ad hoc Networks," IEEE Transactions on Mobile Computing, Vol. 5, No. 4, April 2006.

[43] P. MacKenzie, T. Shrimpton, and M. Jakobsson. "Threshold Password-Authenticated Key Exchange," Journal of Cryptology, 2005

[44] A. Juels, M. Jakobsson, E. Shriver, and B. Hillyer. "How To Turn Loaded Dice Into Fair Coins." IEEE Transactions on Information Theory, vol. 46(3). May 2000. pp. 911–921.

[45] M. Jakobsson, P. MacKenzie, and J.P. Stern. "Secure and Lightweight Advertising on the Web," Journal of Computer Networks, vol. 31, issue 11–16, Elsevier North-Holland, Inc., 1999. pp. 1101–1109.

[46] M. Jakobsson, "Cryptographic Protocols," Chapter from The Handbook of Information Security. Hossein Bidgoli, Editor-in-Chief. Copyright John Wiley & Sons, Inc., 2005, Hoboken, N.J.

[47] M. Jakobsson, "Cryptographic Privacy Protection Techniques," Chapter from The Handbook of Information Security. Hossein Bidgoli, Editor-in-Chief. Copyright John Wiley & Sons, Inc., 2005, Hoboken, N.J.

[48] M. Jakobsson, E. Shi, P. Golle, R. Chow, "Implicit authentication for mobile devices," 4th USENIX Workshop on Hot Topics in Security (HotSec '09); 2009 August 11; Montreal, Canada.

[49] R. Chow, P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka, J. Molina, "Controlling data in the cloud: outsourcing computation without outsourcing control," Proceedings of the 2009 ACM Workshop on Cloud Computing Security (CCSW 2009); 2009 November 13; Chicago, IL. NY: ACM; 2009; pp. 85–90.

[50] M. Jakobsson, H. Siadati, M. Dhiman, "Liar Buyer Fraud, and How to Curb It," NDSS, 2015

[51] M. Jakobsson, T.-F. Yen, "How Vulnerable Are We To Scams?," BlackHat, 2015

[52] M. Jakobsson, "How to Wear Your Password," BlackHat, 2014

[53] M. Jakobsson and G. Stewart, "Mobile Malware: Why the Traditional AV Paradigm is Doomed, and How to Use Physics to Detect Undesirable Routines," in BlackHat, 2013.

[54] M. Jakobsson, and H. Siadati,"SpoofKiller: You Can Teach People How to Pay, but Not How to Pay Attention" in Socio-Technical Aspects in Security and Trust (STAST), 2012 Workshop on, 3-10, 2012.

[55] M. Jakobsson, and M. Dhiman,"The benefits of understanding passwords," in Proceedings of the 7th USENIX conference on Hot Topics in Security, Berkeley, CA, USA, 2012.

[56] M. Jakobsson, and S. Taveau, "The Case for Replacing Passwords with Biometrics," Mobile Security Technologies, 2012.

[57] M. Jakobsson and D. Liu, "Bootstrapping mobile PINs using passwords," W2SP, 2011.

[58] M. Jakobsson and R. Akavipat, "Rethinking passwords to adapt to constrained keyboards," 2011.

[59] Y. Niu, E. Shi, R. Chow, P. Golle, and M. Jakobsson, "One Experience Collecting Sensitive Mobile Data," In USER Workshop of SOUPS, 2010.

[60] E. Shi, Y. Niu, M. Jakobsson, and R. Chow, "Implicit Authentication through Learning User Behavior," 2010.

[61] M. Jakobsson and K. Johansson, Assured Detection of Malware With Applications to Mobile Platforms, 2010.

[62] M. Jakobsson and K. Johansson, "Retroactive Detection of Malware With Applications to Mobile Platforms," in HotSec 2010, Washington, DC, 2010.

[63] M. Jakobsson, A Central Nervous System for Automatically Detecting Malware, 2009.

[64] R. Chow, P. Golle, M. Jakobsson, R. Masuoka, J. Molina, E. Shi, and J. Staddon, "Controlling data in the cloud: outsourcing computation without outsourcing control," ACM workshop on Cloud computing security (CCSW), 2009.

[65] M. Jakobsson and A. Juels, "Server-Side Detection of Malware Infection," in New Security Paradigms Workshop (NSPW), Oxford, UK, 2009.

[66] M. Jakobsson, "Captcha-free throttling," Proceedings of the 2nd ACM workshop on Security and artificial intelligence, 15–22, 2009.

[67] M. Jakobsson, E. Shi, P. Golle, and R. Chow, "Implicit authentication for mobile devices," Proceedings of the 4th USENIX conference on Hot topics in security, 9–9, 2009.

[68] C. Soghoian, O. Friedrichs and M. Jakobsson, "The Threat of Political Phishing," International Symposium on Human Aspects of Information Security & Assurance (HAISA 2008)

[69] R. Chow, P. Golle, M. Jakobsson, L. Wang and X. Wang, "Making CAPTCHAs Clickable," In proc. of HotMobile 2008.

[70] M. Jakobsson, A. Juels, and J. Ratkiewicz, "Privacy-Preserving History Mining for Web Browsers," Web 2.0 Security and Privacy, 2008.

[71] M. Jakobsson, E. Stolterman, S. Wetzel, L. Yang, "Love and Authentication," (Notes) ACM Computer/Human Interaction Conference (CHI), 2008. Also see `www.I-forgot-my-password.com`

[72] M. Jakobsson and S. Myers, "Delayed Password Disclosure," Proceedings of the 2007 ACM workshop on Digital Identity Management

[73] M. Jakobsson, S. Stamm, Z. Ramzan, "JavaScript Breaks Free," W2SP '07

[74] A. Juels, S. Stamm, M. Jakobsson, "Combatting Click Fraud via Premium Clicks," USENIX Security 2007

[75] R. Chow, P. Golle, M. Jakobsson, X. Wang, "Clickable CAPTCHAs," Ad-Fraud '07 Workshop; 2007 September 14; Stanford, CA, USA

[76] S. Stamm, Z. Ramzan, and M. Jakobsson, "Drive-by Pharming," In Proceedings of Information and Communications Security, 9th International Conference, ICICS 2007

[77] M. Jakobsson, A. Tsow, A. Shah, E. Blevis, Y.-K. Lim, "What Instills Trust? A Qualitative Study of Phishing," USEC '07.

[78] R. Akavipat, V. Anandpara, A. Dingman, C. Liu, D. Liu, K. Pongsanon, H. Roinestad and M. Jakobsson, "Phishing IQ Tests Measure Fear, not Ability," USEC '07.

[79] M. Jakobsson, "The Human Factor in Phishing," American Conference Institute's Forum on Privacy & Security of Consumer Information, 2007

[80] S. Srikwan, M. Jakobsson, A. Albrecht and M. Dalkilic, "Trust Establishment in Data Sharing: An Incentive Model for Biodiversity Information Systems," TrustCol 2006

[81] J.Y. Choi, P. Golle, M. Jakobsson, "Tamper-Evident Digital Signatures: Protecting Certification Authorities Against Malware," DACS '06

[82] L. Yang, M. Jakobsson, S. Wetzel, "Discount Anonymous On Demand Routing for Mobile Ad hoc Networks," SECURECOMM '06

[83] P. Golle, X. Wang, M. Jakobsson, A. Tsow, "Deterring Voluntary Trace Disclosure in Re-encryption Mix Networks." IEEE S&P '06

[84] M. Jakobsson, A. Juels, T. Jagatic, "Cache Cookies for Browser Authentication (Extended Abstract)," IEEE S&P '06

[85] M. Jakobsson and J. Ratkiewicz, "Designing Ethical Phishing Experiments: A study of (ROT13) rOnl auction query features.", WWW '06

[86] M. Jakobsson and S. Stamm. "Invasive Browser Sniffing and Countermeasures," WWW '06

[87] J.Y. Choi, P. Golle and M. Jakobsson. "Auditable Privacy: On Tamper-Evident Mix Networks," Financial Crypto '06

[88] A. Juels, D. Catalano and M. Jakobsson. "Coercion-Resistant Electronic Elections," WPES '05

[89] V. Griffith and M. Jakobsson. "Messin' with Texas, Deriving Mother's Maiden Names Using Public Records," ACNS '05, 2005.

[90] M. Jakobsson and L. Yang. "Quantifying Security in Hybrid Cellular Networks," ACNS '05, 2005

[91] Y.-C. Hu, M. Jakobsson, and A. Perrig. "Efficient Constructions for One-way Hash Chains," ACNS '05, 2005

[92] M. Jakobsson. "Modeling and Preventing Phishing Attacks," Phishing Panel in Financial Cryptography '05. 2005, abstract in proceedings.

[93] N. Ben Salem, J.-P. Hubaux, and M. Jakobsson. "Reputation-based Wi-Fi Deployment Protocols and Security Analysis," In WMASH '04. ACM Press, 2004. pp. 29–40.

[94] M. Jakobsson and S. Wetzel. "Efficient Attribute Authentication with Applications to Ad Hoc Networks," In VANET '04. ACM Press, 2004. pp. 38–46.

[95] M. Jakobsson, X. Wang, and S. Wetzel. "Stealth Attacks in Vehicular Technologies," Invited paper. In Proceedings of IEEE Vehicular Technology Conference 2004 Fall (VTC-Fall 2004). IEEE, 2004.

[96] A. Ambainis, H. Lipmaa, and M. Jakobsson. "Cryptographic Randomized Response Technique," In PKC '04. LNCS 2947. Springer-Verlag, 2004. pp. 425–438.

[97] P. Golle, M. Jakobsson, A. Juels, and P. Syverson. "Universal Re-encryption for Mixnets," In CT-RSA '04. LNCS 2964. Springer-Verlag, 2004. pp. 163–178.

[98] P. Golle and M. Jakobsson. "Reusable Anonymous Return Channels," In WPES '03. ACM Press, 2003. pp. 94–100.

[99] M. Jakobsson, S. Wetzel, B. Yener. "Stealth Attacks on Ad-Hoc Wireless Networks," In IEEE VTC '03, 2003.

[100] N. Ben Salem, L. Buttyan, J.-P. Hubaux, and M. Jakobsson. "A Charging and Rewarding Scheme for Packet Forwarding in Multi-hop Cellular Networks," In ACM MobiHoc '03. ACM Press, 2003. pp. 13–24.

[101] M. Jakobsson, J.-P.Hubaux and L. Buttyan. "A Micro-Payment Scheme Encouraging Collaboration in Multi-Hop Cellular Networks," In FC '03. LNCS 2742. Springer-Verlag, 2003. pp. 15–33.

[102] M. Jakobsson, T. Leighton, S. Micali and M. Szydlo. "Fractal Merkle Tree Representation and Traversal," In RSA-CT '03 2003.

[103] A. Boldyreva and M Jakobsson. "Theft protected proprietary certificates," In DRM '02. LNCS 2696, 2002. pp. 208–220.

[104] P. Golle, S. Zhong, M. Jakobsson, A. Juels, and D. Boneh. "Optimistic Mixing for Exit-Polls," In Asiacrypt '02. LNCS 2501. Springer-Verlag, 2002. pp. 451–465.

[105] P. MacKenzie, T. Shrimpton, and M. Jakobsson. "Threshold Password-Authenticated Key Exchange," In CRYPTO '02. LNCS 2442. Springer-Verlag, 2002. pp. 385–400.

[106] M. Jakobsson. "Fractal Hash Sequence Representation and Traversal," In Proceedings of the 2002 IEEE International Symposium on Information Theory (ISIT '02). 2002. pp. 437–444.

[107] M. Jakobsson, A. Juels, and R. Rivest. "Making Mix Nets Robust For Electronic Voting By Randomized Partial Checking," In Proceedings of the 11th USENIX Security Symposium. USENIX Association, 2002. pp. 339–353.

[108] D. Coppersmith and M. Jakobsson. "Almost Optimal Hash Sequence Traversal," In Financial Crypto '02. 2002.

[109] M. Jakobsson. "Financial Instruments in Recommendation Mechanisms," In Financial Crypto '02. 2002.

[110] J. Garay, and M. Jakobsson. "Timed Release of Standard Digital Signatures," In Financial Crypto '02. 2002.

[111] F. Menczer, N. Street, N. Vishwakarma, A. Monge, and M. Jakobsson. "Intellishopper: A Proactive, Personal, Private Shopping Assistant," In AAMAS '02. ACM Press, 2002. pp. 1001–1008.

[112] M. Jakobsson, A. Juels, and P. Nguyen. "Proprietary Certificates," In CT-RSA '02. LNCS 2271. Springer-Verlag, 2002. pp. 164–181.

[113] M. Jakobsson and A. Juels. "An Optimally Robust Hybrid Mix Network," In PODC '01. ACM Press. 2001. pp. 284–292.

[114] M. Jakobsson and M. Reiter. "Discouraging Software Piracy Using Software Aging," In DRM '01. LNCS 2320. Springer-Verlag, 2002. pp. 1–12.

[115] M. Jakobsson and S. Wetzel. "Security Weaknesses in Bluetooth," In CT–RSA '01. LNCS 2020. Springer-Verlag, 2001. pp. 176–191.

[116] M. Jakobsson and D. Pointcheval. "Mutual Authentication for Low-Power Mobile Devices," In Financial Crypto '01. LNCS 2339. Springer-Verlag, 2001. pp. 178–195.

[117] M. Jakobsson, D. Pointcheval, and A. Young. "Secure Mobile Gambling," In CT–RSA '01. LNCS 2020. Springer-Verlag, 2001. pp. 110–125.

[118] M. Jakobsson and S.Wetzel. "Secure Server-Aided Signature Generation," In PKC '01. LNCS 1992. Springer-Verlag, 2001. pp. 383–401.

[119] M. Jakobsson and A. Juels. "Addition of ElGamal Plaintexts," In T. Okamoto, ed., ASIACRYPT '00. LNCS 1976. Springer-Verlag, 2000. pp. 346–358.

[120] M. Jakobsson, and A. Juels. "Mix and Match: Secure Function Evaluation via Ciphertexts," In ASIACRYPT '00. LNCS 1976. Springer-Verlag, 2000. pp. 162–177.

[121] R. Arlein, B. Jai, M. Jakobsson, F. Monrose, and M. Reiter. "Privacy-Preserving Global Customization," In ACM E-Commerce '00. ACM Press, 2000. pp. 176–184.

[122] C.-P. Schnorr and M. Jakobsson. "Security of Signed ElGamal Encryption," In ASIACRYPT '00. LNCS 1976. Springer-Verlag, 2000. pp. 73–89.

[123] P. Bohannon, M. Jakobsson, and S. Srikwan. "Cryptographic Approaches to Privacy in Forensic DNA Databases," In Public Key Cryptography '00. LNCS 1751. Springer-Verlag, 2000, pp. 373–390.

[124] J. Garay, M. Jakobsson, and P. MacKenzie. "Abuse-free Optimistic Contract Signing," In CRYPTO '99. LNCS 1666. Springer-Verlag, 1999. pp. 449–466.

[125] M. Jakobsson. "Flash Mixing," In PODC '99. ACM Press, 1999. pp. 83–89.

[126] G. Di Crescenzo, N. Ferguson, R. Impagliazzo, and M. Jakobsson. "How To Forget a Secret," In STACS '99. LNCS 1563. Springer-Verlag, 1999. pp. 500–509.

[127] M. Jakobsson, D. M'Raihi, Y. Tsiounis, and M. Yung. "Electronic Payments: Where Do We Go from Here?," In CQRE (Secure) '99. LNCS 1740. Springer-Verlag, 1999. pp. 43–63.

[128] C.P. Schnorr and M. Jakobsson. "Security Of Discrete Log Cryptosystems in the Random Oracle + Generic Model," In Conference on The Mathematics of Public-Key Cryptography. 1999.

[129] M. Jakobsson and A. Juels "Proofs of Work and Breadpudding Protocols," In CMS '99. IFIP Conference Proceedings, Vol. 152. Kluwer, B.V., 1999. pp. 252 – 272.

[130] M. Jakobsson and C-P Schnorr. "Efficient Oblivious Proofs of Correct Exponentiation," In CMS '99. IFIP Conference Proceedings, Vol. 152. Kluwer, B.V., 1999. pp. 71–86.

[131] M. Jakobsson, P. MacKenzie, and J.P. Stern. "Secure and Lightweight Advertising on the Web," In World Wide Web '99

[132] M. Jakobsson, J.P. Stern, and M. Yung. "Scramble All, Encrypt Small," In Fast Software Encryption '99. LNCS 1636. Springer-Verlag, 1999. pp. 95–111.

[133] M. Jakobsson and J. Mueller. "Improved Magic Ink Signatures Using Hints," In Financial Cryptography '99. LNCS 1648. Springer-Verlag, 1999. pp. 253–268.

[134] M. Jakobsson. "Mini-Cash: A Minimalistic Approach to E-Commerce," In Public Key Cryptography '99. LNCS 1560. Springer-Verlag, 1999. pp. 122–135.

[135] M. Jakobsson. "On Quorum Controlled Asymmetric Proxy Re-encryption," In Public Key Cryptography '99. LNCS 1560. Springer-Verlag, 1999. pp. 112–121.

[136] M. Jakobsson and A. Juels. "X-Cash: Executable Digital Cash," In Financial Cryptography '98. LNCS 1465. Springer-Verlag, 1998. pp. 16–27.

[137] M. Jakobsson and D. M'Raihi. "Mix-based Electronic Payments," In Proceedings of the Selected Areas in Cryptography. LNCS 1556. Springer-Verlag, 1998. pp. 157173.

[138] M. Jakobsson, E. Shriver, B. Hillyer, and A. Juels. "A Practical Secure Physical Random Bit Generator," In CCS '98: Proceedings of the 5th ACM conference on Computer and communications security. ACM Press, 1998. pp. 103–111.

[139] M. Jakobsson. "A Practical Mix," In Advances in Cryptology – EuroCrypt '98. LNCS 1403. Springer-Verlag, 1998. pp. 448–461.

[140] M. Jakobsson and M. Yung. "On Assurance Structures for WWW Commerce," In Financial Cryptography '98. LNCS 1465. Springer-Verlag, 1998. pp. 141–157.

[141] E. Gabber, M. Jakobsson, Y. Matias, and A. Mayer. "Curbing Junk E-Mail via Secure Classification," In Financial Cryptography '98. LNCS 1465. Springer-Verlag, 1998. pp. 198–213.

[142] M. Jakobsson and M. Yung. "Distributed 'Magic Ink' Signatures," In Advances in Cryptology – EuroCrypt '97. LNCS 1233. Springer-Verlag, 1997. pp. 450–464.

[143] M. Jakobsson and M. Yung. "Applying Anti-Trust Policies to Increase Trust in a Versatile E-Money System," In Financial Cryptography '97. LNCS 1318. Springer-Verlag, 1997. pp. 217–238.

[144] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung. "Proactive public-key and signature schemes," In Proceedings of the 4th Annual Conference on Computer Communications Security. ACM Press, 1997. pp. 100–110.

[145] M. Bellare, M. Jakobsson, and M. Yung. "Round-Optimal Zero-Knowledge Arguments Based on any One-Way Function," In Advances in Cryptology – EuroCrypt '97. LNCS 1233. Springer-Verlag, 1997. pp. 280–305.

[146] M. Jakobsson and M. Yung. "Proving Without Knowing," In Crypto '96. LNCS 1109. Springer-Verlag, 1996. pp. 186–200.

[147] M. Jakobsson, K. Sako, and R. Impagliazzo. "Designated Verifier Proofs and Their Applications," In Advances in Cryptology – EuroCrypt '96. LNCS 1070. Springer-Verlag, 1996. pp. 143–154.

[148] M. Jakobsson and M. Yung. "Revokable and Versatile Electronic Money," In CCS '96: Proceedings of the 3rd ACM conference on Computer and communications security. ACM Press, 1996. pp. 76–87.

[149] M. Jakobsson. "Ripping Coins for a Fair Exchange," In Advances in Cryptology – EuroCrypt '95. LNCS 921. Springer-Verlag, 1995. pp. 220–230.

[150] M. Jakobsson. "Blackmailing using Undeniable Signatures," In Advances in Cryptology EuroCrypt '94. LNCS 950. Springer-Verlag, 1994. pp. 425–427.

[151] M. Jakobsson. "Reducing costs in identification protocols," Crypto '92, 1992.

[152] M. Jakobsson. "Machine-Generated Music with Themes," In International Conference on Artificial Neural Networks '92. Vol 2. Amsterdam: Elsevier, 1992. pp. 1645–1646

[153] M. Jakobsson, "Social Engineering 2.0: What's Next," McAfee Security Journal, Fall 2008

[154] M. Jakobsson and S. Myers, "Delayed Password Disclosure," ACM SIGACT News archive, Volume 38, Issue 3 (September 2007), pp. 56 - 75

[155] V. Griffith and M. Jakobsson. "Messin' with Texas, Deriving Mother's Maiden Names Using Public Records," CryptoBytes, 2007.

[156] M. Jakobsson, A. Juels, J. Ratkiewicz, "Remote-Harm Detection," Beta-version available at `rhd.ravenwhitedevelopment.com/`

[157] S. Stamm, M. Jakobsson, "Social Malware," Experimental results available at `www.indiana.edu/ phishing/verybigad/`

[158] M. Jakobsson. "Privacy vs. Authenticity," Ph.D. Thesis, University of California at San Diego, 1997

[159] Markus Jakobsson, Automatic PIN creation using passwords, US20130125214 A1, 2012.

[160] Markus Jakobsson, Systems and methods for creating a user credential and authentication using the created user credential, US 20130111571 A1, 2012.

[161] Markus Jakobsson, Password check by decomposing password, US 20120284783 A1, 2012.

[162] Markus Jakobsson, William Leddy, System and methods for protecting users from malicious content, US 20120192277 A1, 2011.

[163] Markus Jakobsson, Karl-Anders R. Johansson, Auditing a device, US8370935 B1, 2011.

[164] Markus Jakobsson, Methods and Apparatus for Efficient Computation of One-Way Chains in Cryptographic Applications, US20120303969 A1, 2011.

[165] Markus Jakobsson, Automatic PIN creation using password, US 20120110634 A1, 2011.

[166] Markus Jakobsson, Jim Roy Palmer, Gustavo Maldonado, Interactive CAPTCHA, US20130007875 A1, 2011.

[167] Markus Jakobsson, System access determination based on classification of stimuli, US 20110314559 A1, 2011.

[168] Markus Jakobsson, System access determination based on classification of stimuli, WO 2011159356 A1, 2011.

[169] Markus Jakobsson, Method, medium, and system for reducing fraud by increasing guilt during a purchase transaction, US8458041 B1, 2011.

[170] Markus Jakobsson, Visualization of Access Information, US 20120233314 A1, 2011.

[171] Markus Jakobsson, Richard Chow,Runting Shi, Implicit authentication, US20120137340 A1, 2010.

[172] Markus Jakobsson, Karl-Anders R. Johansson, Auditing a device, EP2467793 A1, 2010.

[173] Markus Jakobsson, Event log authentication using secure components, US 20110314297 A1, 2010.

[174] Markus Jakobsson, Philippe J.P. Golle, Risk-based alerts, US 20110314426 A1, 2010.

[175] Markus Jakobsson, Karl-Anders R. Johansson, Auditing a device, US 20110041178 A1, 2010.

[176] M. Jakobsson, A. Juels, J. Kaliski Jr, S. Burton and others, Identity authentication system and method, US Patent 7,502,933, 2009.

[177] Markus Jakobsson, Method and system for facilitating throttling of interpolation-based authentication, US8219810 B2, 2009.

[178] Markus Jakobsson, Karl-Anders R. Johansson, Auditing a device, US8375442 B2, 2009.

[179] Markus Jakobsson, Karl-Anders R. Johansson, Auditing a device, US 20110041180 A1, 2009.

[180] Markus Jakobsson, Pattern-based application classification, US 20110055925 A1, 2009.

[181] Markus Jakobsson, Method and apparatus for detecting cyber threats, US8286225 B2, 2009.

[182] Markus Jakobsson, Method and apparatus for detecting cyber threats, US20110035784 A1, 2009.

[183] Markus Jakobsson, CAPTCHA-free throttling, US8312073 B2, 2009.

[184] Markus Jakobsson, Captcha-free throttling, US 20110035505 A1, 2009.

[185] Markus Jakobsson, Christopher Soghoian, Method and apparatus for throttling access using small payments, US 20100153275 A1, 2008.

[186] Markus Jakobsson, Christopher Soghoian, Method and apparatus for mutual authentication using small payments, US 20100153274 A1, 2008.

[187] Philippe J.P. Golle, Markus Jakobsson,Richard Chow, Resetting a forgotten password using the password itself as authentication, US 20100125906 A1, 2008.

[188] Philippe J. P. Golle, Markus Jakobsson,Richard Chow, Authenticating users with memorable personal questions, US8161534 B2, 2008.

[189] Richard Chow, Philippe J.P. Golle, Markus Jakobsson, Jessica N. Staddon, Authentication based on user behavior, US20100122329 A1, 2008.

[190] Richard Chow, Philippe J.P. Golle, Markus Jakobsson, Jessica N. Staddon, Enterprise password reset, US 20100122340 A1, 2008.

[191] Markus Jakobsson, Methods and apparatus for efficient computation of one-way chains in cryptographic applications, US8086866 B2, 2008.

[192] Richard Chow, Philippe J. P. Golle, Markus Jakobsson, Selectable captchas, US8307407 B2, 2008.

[193] Markus Jakobsson, Ari Juels, Sidney Louis Stamm, Method and apparatus for combatting click fraud, US 20080162227 A1, 2007.

[194] Jakobsson, Method and apparatus for evaluating actions performed on a client device, US 20080037791 A1, 2007.

[195] Jakobsson, Ari Juels, Method and apparatus for storing information in a browser storage area of a client device, US 20070106748 A1, 2006.

[196] Markus Jakobsson, Steven Andrew Myers, Anti-phising logon authentication object oriented system and method, WO 2006062838 A1, 2005.

[197] Jakobsson, Jean-Pierre Hubaux,Levente Buttyan, Micro-payment scheme encouraging collaboration in multi-hop cellular networks, US 20050165696 A1, 2004.

[198] Andrew Nanopoulos, Karl Ackerman, Piers Bowness, William Duane, Markus Jakobsson, Burt Kaliski, Dmitri Pal, Shane D. Rice,Less , System and method providing disconnected authentication, WO2005029746 A3, 2004.

[199] Andrew Nanopoulos, Karl Ackerman, Piers Bowness, William Duane, Markus Jakobsson, Burt Kaliski, Dmitri Pal, Shane Rice, Ronald Rivest, Less , System and method providing disconnected authentication, US 20050166263 A1, 2004.

[200] Andrew Nanopoulos, Karl Ackerman, Piers Bowness, William Duane, Markus Jakobsson, Burt Kaliski, Dmitri Pal, Shane D. Rice,Less , Systeme et procede d'authentification deconnectee, WO 2005029746 A2, 2004.

[201] Markus Jakobsson, Ari Juels, Burton S. Kaliski Jr., Identity authentication system and method, WO2004051585 A3, 2003.

[202] Markus Jakobsson, Ari Juels, Burton S. Kaliski, Jr., Identity authentication system and method, US 7502933 B2, 2003.

[203] Markus Jakobsson, Ari Juels, Burton S. Kaliski Jr., Systeme et procede de validation d'identite, WO 2004051585 A2, 2003.

[204] Markus Jakobsson, Burton S. Kaliski, Jr., Method and apparatus for graph-based partition of cryptographic functionality, US7730518 B2, 2003.

[205] Markus Jakobsson, Phong Q. Nguyen, Methods and apparatus for private certificates in public key cryptography, US7404078 B2, 2002.

[206] Jakobsson, Philip MacKenzie, Thomas Shrimpton, Method and apparatus for performing multi-server threshold password-authenticated key exchange, US20030221102 A1, 2002.

[207] Markus Jakobsson, Philip D MacKenzie, Method and apparatus for distributing shares of a password for use in multi-server password authentication, US 7073068 B2, 2002.

[208] Markus Jakobsson, Methods and apparatus for efficient computation of one-way chains in cryptographic applications, EP 1389376 A1 (text from WO2002084944A1), 2002.

[209] Markus Jakobsson, Adam Lucas Young, Method and apparatus for identification tagging documents in a computer system, US 7356845 B2, 2002.

[210] Juan A. Garay, Markus Jakobsson, Methods and apparatus for computationally-efficient generation of secure digital signatures, US 7366911 B2, 2001.

[211] Markus Jakobsson, Methods and apparatus for efficient computation of one-way chains in cryptographic applications, US 7404080 B2, 2001.

[212] Markus Jakobsson, Susanne Gudrun Wetzel, Securing the identity of a bluetooth master device (bd addr) against eavesdropping by preventing the association of a detected channel access code (cac) with the identity of a particular bluetooth device, WO2002019641 A3, 2001.

[213] Markus Jakobsson, Susanne Gudrun Wetzel, Procede et appareil permettant d'assurer la securite d'utilisateurs de dispositifs a capacites bluetooth, WO 2002019641 A2, 2001.

[214] Robert M. Arlein, Ben Jai, Markus Jakobsson, Fabian Monrose, Michael Kendrick Reiter, Less , Methods and apparatus for providing privacy-preserving global customization, US 7107269 B2, 2001.

[215] Markus Jakobsson, Susanne Gudrun Wetzel, Secure distributed computation in cryptographic applications, US6950937 B2, 2001.

[216] Markus Jakobsson, Susanne Gudrun Wetzel, Method and apparatus for ensuring security of users of short range wireless enable devices, US6981157 B2, 2001.

[217] Markus Jakobsson, Susanne Gudrun Wetzel, Method and apparatus for ensuring security of users of bluetooth TM-enabled devices, US 6574455 B2, 2001.

[218] Garay; Juan A. (West New York, NJ), Jakobsson; M. (Hoboken, NJ), Kristol; David M. (Summit, NJ), Mizikovsky; Semyon B.(Morganville, NJ), Cryptographic key processing and storage , 7023998, 2001.

[219] Markus Jakobsson, Method, apparatus, and article of manufacture for generating secure recommendations from market-based financial instrument prices, US 6970839 B2, 2001.

[220] Markus Jakobsson, Encryption method and apparatus with escrow guarantees, US7035403 B2, 2001.

[221] Jakobsson, Call originator access control through user-specified pricing mechanism in a communication network, US20020099670 A1, 2001.

[222] Markus Jakobsson, Claus Peter Schnorr, Tagged private information retrieval, US7013295 B2, 2000.

[223] Markus Jakobsson, Secure enclosure for key exchange, US 7065655 B1, 2000.

[224] Markus Jakobsson, Michael Kendrick Reiter, Software aging method and apparatus for discouraging software piracy, US 7003110 B1, 2000.

[225] Markus Jakobsson, Probabilistic theft deterrence, US6501380 B1, 2000.

[226] Markus Jakobsson, Michael Kendrick Reiter, Abraham Silberschatz, Anonymous and secure electronic commerce, EP 1150227 A1, 2000.

[227] Markus Jakobsson, Ari Juels, Proofs of work and bread pudding protocols, US 7356696 B1, 2000.

[228] Markus Jakobsson, Joy Colette Mueller, Methods of protecting against spam electronic mail, US7644274 B1, 2000.

[229] Philip L. Bohannon, Markus Jakobsson,Fabian Monrose, Michael Kendrick Reiter,Susanne Gudrun Wetzel, Less , Generation of repeatable cryptographic key based on varying parameters, EP1043862 B1, 2000.

[230] Markus Jakobsson, Ari Juels, Mix and match: a new approach to secure multiparty computation, US6772339 B1, 2000.

[231] Markus Jakobsson, Ari Juels, Mixing in small batches, US6813354 B1, 2000.

[232] Philip L. Bohannon, Markus Jakobsson,Fabian Monrose, Michael Kendrick Reiter,Susanne Gudrun Wetzel, Less , Generation of repeatable cryptographic key based on varying parameters, US 6901145 B1, 36566

[233] Markus Jakobsson, Claus Peter Schnorr, Non malleable encryption method and apparatus using key-encryption keys and digital signature, US6931126 B1, 2000.

[234] Markus Jakobsson, Claus Peter Schnorr, Non malleable encryption method and apparatus using key-encryption keys and digital signature, US 6931126 B1, 2000.

[235] Markus Jakobsson, Flash mixing apparatus and method, US 6598163 B1, 1999.

[236] Markus Jakobsson, Minimalistic electronic commerce system, US 6529884 B1, 1999.

[237] Markus Jakobsson, Method and system for providing translation certificates, US 6687822 B1, 1999.

[238] Markus Jakobsson, Verification of correct exponentiation or other operations in cryptographic applications, US6978372 B1, 1999.

[239] Markus Jakobsson, Non malleable encryption apparatus and method, US 6507656 B1, 1999.

[240] Markus Jakobsson, Method and system for quorum controlled asymmetric proxy encryption, US 6587946 B1, 1998.

[241] Markus Jakobsson, Practical mix-based election scheme, US 6317833 B1, 1998.

[242] Markus Jakobsson, Ari Juels, Method and apparatus for extracting unbiased random bits from a potentially biased source of randomness, US 6393447 B1, 1998.

[243] Markus Jakobsson, Ari Juels, Executable digital cash for electronic commerce, US6157920 A, 1998.

[244] Bruce Kenneth Hillyer, Markus Jakobsson, Elizabeth Shriver, Storage device random bit generator, US 6317499 B1, 1998.

[245] Markus Jakobsson, Method and apparatus for encrypting, decrypting, and providing privacy for data values, US 6049613 A, 1998.