

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

PALO ALTO NETWORKS, INC.
Petitioner

v.

CROGA INNOVATIONS LTD.
Patent Owner

Case IPR2024-01421
U.S. Patent No. 11,223,601

DECLARATION OF SAMRAT BHATTACHARJEE, PH.D.

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

PAN EX1003
Palo Alto Networks v. Croga Innovations
IPR2024-01421

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	LIST OF DOCUMENTS CONSIDERED.....	1
III.	Qualifications.....	4
IV.	LEGAL PRINCIPLES	7
	A. Claim Construction	7
	B. Obviousness.....	8
	C. A Person of Ordinary Skill In The Art	10
	D. Level of Ordinary Skill in the Art	11
V.	U.S. PATENT. 11,223,601	11
	A. Overview of the '601 Patent	11
	B. Prosecution History.....	18
	C. Claims.....	22
VI.	TECHNOLOGY OVERVIEW	24
	A. Isolated Computing Environments	24
	B. Multi-User Interactive Software Applications.....	34
VII.	IDENTIFICATION OF THE CHALLENGE (37 C.F.R. § 42.104(B))	44
	A. Citation of Prior Art.....	44
VIII.	OVERVIEW OF THE PRIOR ART	47
	A. Walsh.....	47
	B. Innes	54
	C. Austin	58
	D. Bloch	63
	E. Walker	67
	F. Franco.....	70
IX.	GROUND OF UNPATENTABILITY.....	73
X.	GROUND 1: Claims 1-4, 7, 9-12, and 15 are obvious over Walsh, Applicant Admitted Prior Art (AAPA), and Innes.....	74

A.	A POSA would have been motivated to combine Walsh with Innes based on Walsh’s disclosure.	74
B.	A POSA would have been motivated to combine Walsh-Innes with AAPA.	82
C.	Independent claims 1 and 9.....	83
1.	[1-pre] “A system” and [9-pre] “A method”.....	83
2.	[1A] “a memory”	84
3.	[1B] “a processor configured to”.....	85
4.	[1C] and [9A] “[implement/implementing] a workspace that uses a host operating system, wherein the workspace enables operation of a first set of one or more applications or processes via a first memory space”.....	86
5.	[1D] and [9B] “[implement/implementing] an isolated computing environment that uses the host operating system”	88
6.	[1E] and [9C] “wherein the isolated computing environment enables operation of a second set of one or more applications or processes via a second memory space”	89
7.	[1F] and [9D] “wherein the second set of one or more applications or processes comprises a multi-user interactive software application”	91
8.	[1G] and [9E] “[isolate/isolating] the isolated computing environment from the workspace using an internal isolation firewall”.....	93
9.	[1H] and [9F] “[authenticate/authenticating] the isolated computing environment with an authentication device”	94
10.	[1I] and [9G] “[send/sending] data to an untrusted destination from the multi-user interactive software application via a proxy device when the isolated computing environment has been authenticated.”	96
D.	Claims 2 and 10	99
E.	Claims 3 and 11	100
F.	Claims 4 and 12	102
G.	Claims 7 and 15	103

XI.	GROUND 2: Claims 5, 6, 13, And 14 Are Obvious Over Walsh, AAPA, Innes, and Netfilter.	104
A.	A POSA would have been motivated to combine Walsh, AAPA, and Innes with Netfilter based on Walsh’s disclosure	104
B.	Claims 5 and 13	106
C.	Claims 6 and 14	106
XII.	GROUND 3: Claims 8 And 16 Are Obvious Over Walsh, AAPA, Innes, and Franco.....	107
A.	A POSA would have been motivated to combine Walsh, AAPA, and Innes with Franco based on Walsh’s disclosure.	107
B.	Claims 8 and 16	110
XIII.	GROUND 4: Claims 1-4, 7, 9-12, and 15 Are Obvious Over Austin, AAPA, And Bloch.....	112
A.	A POSA would have been motivated to combine Austin with Bloch based on Austin’s disclosure.	112
B.	A POSA would have been motivated to combine Austin-Bloch with AAPA.....	120
C.	Independent claims 1 and 9.....	121
1.	[1-pre] “A system” and [9-pre] “A method”.....	121
2.	[1A] “a memory”	122
3.	[1B] “a processor configured to”.....	123
4.	[1C] and [9A] “[implement/implementing] a workspace that uses a host operating system, wherein the workspace enables operation of a first set of one or more applications or processes via a first memory space”.....	124
5.	[1D] and [9B] “[implement/implementing] an isolated computing environment that uses the host operating system”.....	128
6.	[1E] and [9C] “wherein the isolated computing environment enables operation of a second set of one or more applications or processes via a second memory space”	130
7.	[1F] and [9D] “wherein the second set of one or more applications or processes comprises a multi-user interactive software application”	132

8.	[1G] and [9E] “[isolate/isolating] the isolated computing environment from the workspace using an internal isolation firewall”	133
9.	[1H] and [9F] “[authenticate/authenticating] the isolated computing environment with an authentication device”	135
10.	[1I] and [9G] “[send/sending] data to an untrusted destination from the multi-user interactive software application via a proxy device when the isolated computing environment has been authenticated.”	137
D.	Claims 2 and 10	139
E.	Claims 3 and 11	141
F.	Claims 4 and 12	143
G.	Claims 7 and 15	144
XIV.	GROUND 5: Claims 5, 6, 13, And 14 Are Obvious Over Austin, AAPA, Bloch, and Walker.	146
A.	A POSA would have been motivated to combine Austin, AAPA, and Bloch with Walker based on Austin’s disclosure.	146
B.	Claims 5 and 13	147
C.	Claims 6 and 14	148
XV.	GROUND 6: Claims 8 And 16 Are Obvious Over Austin, AAPA, Bloch, and Franco.....	150
A.	A POSA would have been motivated to combine Austin, AAPA, and Bloch with Franco based on Austin’s disclosure.	150
B.	Claims 8 and 16	152
XVI.	CONCLUSION	155

I, Samrat Bhattacharjee, hereby declare as follows.

I. INTRODUCTION

1. I have been retained as an expert witness on behalf of Petitioner Palo Alto Networks, Inc. for the above-captioned *inter partes* review (IPR). I am being compensated for my time in connection with this IPR at my standard consulting rate, which is \$700 per hour.

2. I understand that this Declaration accompanies a petition for IPR involving U.S. Patent No. 11,223,601 (“the ’601 patent”) (EX1001), which resulted from U.S. Patent Application No. 16/141,535 (“the ’535 application”), filed on September 25, 2018. I understand that the ’601 patent alleges a priority date of September 28, 2017. I refer to this date throughout this declaration.

3. In preparing this Declaration, I have reviewed the ’601 patent and each of the documents cited herein, in light of general knowledge in the art before September 28, 2017. In formulating my opinions, I have relied upon my experience, education, and knowledge in the relevant art. In formulating my opinions, I have also considered the viewpoint of a person of ordinary skill in the art (“POSA”) (as defined further below in Section IV.D) prior to September 28, 2017.

II. LIST OF DOCUMENTS CONSIDERED

4. A list of materials I have considered is below.

Exhibit No.	Description
1001	U.S. Patent No. 11,223,601 to Coleman et al.
1002	U.S. Patent Prosecution History of U.S. Patent No. 11,223,601
1003	Declaration of Dr. Samrat Bhattacharjee
1005	U.S. Publication No. 2016/0306964 to Austin et al.
1006	U.S. Patent No. 8,640,187 to Walsh
1007	U.S. Patent No. 7,849,502 to Bloch et al.
1008	U.S. Publication No. 2016/0149863 to Walker et al.
1009	U.S. Patent No. 8,161,563 to Franco et al. (“Franco”)
1010	“Productivity Tools,” Cisco WebEx (2016)
1011	“Sharing and Presenting Tools,” Cisco WebEx (2016)
1012	U.S. Patent No. 9,154,488 to Innes et al.
1013	“What is netfilter.org?,” Netfilter (2016)
1014	Declaration of Nathaniel E. Frank-White regarding the authenticity and public accessibility of WebEx, Netfilter, and X2Go
1015	Kamp, P. & Watson, R., “Jails: Confining the omnipotent root.,” The FreeBSD Project (2000)
1016	Bui, T., “Analysis of Docker Security,” Aalto University School of Science (Jan. 13, 2015)
1017	Meggers, J., “Ringin in 2015, the Webex Way,” Cisco (Jan. 14, 2015), accessed at https://blogs.cisco.com/collaboration/ringing-in-2015-the-webex-way#:~:text=3%20min%20read,Collaboration%20Applications%20team%20at%20Cisco_

Exhibit No.	Description
1018	“Getting Started Guide: Cisco WebEx Meetings,” Cisco WebEx, (2014)
1019	“Sharing and Presenting Tools Event Center,” Cisco WebEx University (2014)
1020	Forouzan, B.A., “Data Communications and Networking” 5th ed. (2013)
1021	Love, R., “Linux Kernel Development,” 3d ed. (2010)
1022	John L. Hennessy & David A. Patterson, Computer Architecture (5th ed. 2012)
1023	The Associated Press, “Cisco to Buy an Online Conferencing Company,” N.Y. Times (Mar. 16, 2007), accessed at https://www.nytimes.com/2007/03/16/technology/16cisco.html
1024	“GoToMeeting User Guide,” Cisco (2013)
1025	Vomel, S. & Freiling, F.C., “A Survey of Main Memory Acquisition and Analysis Techniques for the Windows Operating System,” Digital Investigation, 1(8):3-22 (2011)
1026	McKeen, F., et al., “Innovative Instructions and Software Model for Isolated Execution,” Intel Corporation (2013)
1027	Wahbe, R., et al., “Efficient Software-Based Fault Isolation,” SOSP '93: Proceedings of the fourteenth ACM Symposium on Operating Systems Principles, pp. 203-216 (1993)
1028	Felter, W., et al., “An Updated Performance Comparison of Virtual Machines and Linux Containers,” IBM (Jul. 21, 2014)
1029	Price, D. & Tucker, A., “Solaris Zones: Operating System Support for Consolidating Commercial Workloads,” Sun Microsystems, Inc. (2004)
1030	Russinovich, M.E. & Solomon, D.E., “Microsoft Windows Internals,” 4th ed. (2005)

Exhibit No.	Description
1031	Smalley, S., “Configuring the SELinux Policy,” National Security Agency (Feb. 2005)
1032	U.S. Patent No. 8,566,398 to Barnier et al. (“Barnier”)
1033	“Cisco WebEx Meetings Product Overview,” Cisco (Apr. 2015)
1034	Azab, A.M., et al., “SICE: A Hardware-Level Strongly Isolated Computing Environment for x86 Multi-Core Platforms,” Association for Computing Machinery (2011)

III. Qualifications

5. My qualifications for forming the opinions set forth in this declaration are summarized below and explained in more detail in my current curriculum vitae, provided as EX1004.

6. I am a tenured professor in the Computer Science Department at the University of Maryland. I am also an Affiliate Professor with the Department of Electrical and Computer Engineering at the University of Maryland. I have been an appointed faculty member at the University of Maryland since 1999.

7. I received a Ph.D. in Computer Science from the Georgia Institute of Technology in 1999. Before that, I received a B.S. degree in Mathematics and Computer Science with Highest Distinction (Summa Cum Laude) and Outstanding Department Major from Georgia College and State University, Milledgeville, Georgia in 1994.

8. I have been involved in the research and development of computing systems for over two decades. My research and teaching areas include computer security, operating systems, computer networks, distributed systems, and cryptography, among others. I have developed research software to isolate processes with potentially malicious software from the underlying host machine. I have also published papers on this topic.

9. The primary focus of my research is improving security and performance for computing systems as well as network communications between those systems. My research has examined problems ranging from network and operating system support for secure servers, power efficient operating systems, audio and video streaming, large-scale Internet video distribution, novel network architectures for secure content distribution and storage, efficient content distribution over wireless networks, and so on.

10. For example, starting in the late 1990s, the focus of my research was the development of network and operating system technology to design a new network architecture for efficient data transfer. Part of my Ph.D. research was to develop new and more efficient internet video delivery architectures. During my graduate studies, I published various papers on this architecture. I continued to work on digital content security, delivery, and storage as a faculty member, and have also published various papers in these areas.

11. In 2007, I was a visiting researcher at AT&T Labs, and one of the projects I focused on was a video content delivery platform. This work resulted in various publications and a U.S. patent (U.S. Pat. No. 8,752,100).

12. In addition to my research, I also teach graduate and undergraduate courses on computer security, computer networking, cryptography, and video distribution. I have collaborated with many of my University of Maryland students to publish papers in these and other topic areas.

13. Over the past 25 years, I have authored or co-authored over 100 articles in peer-reviewed journals, conference proceedings, and workshops relating to operating systems, computer networking, computer security, and data security. Over the years, I have received several fellowships, prizes, and awards for teaching excellence and technical papers, including the Alfred P. Sloan Jr. Fellowship in 2004, and the National Science Foundation CAREER award in 2001. I have also received the “Test of Time” paper award from the SIGCOMM foundation.

14. My research work has been supported by multiple grants from the US National Science Foundation and the Department of Defense. I have also started a Joint Ph.D. program between the University of Maryland and the Max Planck Society in Germany. Additionally, I co-founded the annual Cornell, Maryland, and Max Planck Research School providing 80 students research exposure during the weeklong experience.

15. I have served on numerous paper and proposal review panels, and participated on program committees for ACM/IEEE groups, technical journals, and conferences for computer system and network security. I have created research software for all of my projects, much of which is available online for others to build on.

16. I am a named inventor on four U.S. Patents. These patents are related to computer security, computer networking, network service delivery, and video delivery systems.

17. I have served as an expert witness and technical consultant in litigation and *Inter Partes* Review matters involving operating systems, computer networks, videoconferencing, data conferencing and screen sharing, voice over IP (VoIP) telephony, multimedia networking, distributed systems, and datacenter networking, among others. I have testified in several trials and depositions as an expert witness.

IV. LEGAL PRINCIPLES

18. I am not a lawyer. However, I have been advised that technical experts should apply certain legal standards in forming their opinions.

A. Claim Construction

19. I understand that claim terms should be interpreted under a claim construction standard called Phillips. I understand that, according to the Phillips

standard, claim terms should be construed according to the terms' meaning as understood by a person having ordinary skill in the art (POSA) at the time of the invention. I understand that the claim terms should be construed in light of the specification. I also understand that the prosecution history of the patent should be consulted in determining claim meaning. In this declaration, I have construed the terms according to their ordinary and customary meanings as understood by a POSA in view of the specification and prosecution history.

B. Obviousness

20. I understand a patent claim is unpatentable if the claimed invention would have been obvious to a person of ordinary skill in the art (POSA) at the time of the alleged invention. This means that a patent claim can be unpatentable even if all of the claim elements cannot be found in a single prior-art reference.

21. I understand an invention is obvious when the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious to a POSA at the time of the invention.

22. I understand that to prove that prior art renders a patent claim obvious, it is necessary to: (1) identify the particular references that, singly or in combination, teach or suggest every element of the claim; (2) identify which claim elements appear in which reference; and (3) explain why a POSA would have sought to combine the prior-art references.

23. I understand that when considering whether a patent claim is obvious, one should consider whether at the time of the alleged invention a teaching, suggestion, or motivation to combine the references existed so as to avoid impermissibly applying hindsight. I understand this test should not be rigidly applied, but that the test can be important to avoid such hindsight.

24. I understand the following rationales can support a motivation to combine prior-art references:

- (A) Combining prior art elements according to known methods to yield predictable results;
- (B) Simple substitution of one known element for another to obtain predictable results;
- (C) Use of known technique to improve similar devices (methods, or products) in the same way;
- (D) Applying a known technique to a known device (method, or product) ready for improvement to yield predictable results;
- (E) Choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success; and
- (F) 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.

25. I understand objective indicia (also called secondary considerations) can be important evidence as to whether a patent claim is obvious. Such indicia include: (1) commercial success of products covered by the patent claim; (2) a long-felt need for the invention; (3) failed attempts by others to make the invention; (4) copying of the invention by others; (5) unexpected results achieved by the invention; (6) praise of the invention by the infringer or others in the field; (7) taking of licenses under the patent by others; (8) expressions of surprise by experts and those skilled in the art at the making of the invention; and (9) the patentee proceeded contrary to the accepted wisdom of the prior art.

C. A Person of Ordinary Skill In The Art

26. I understand that a POSA is presumed to be aware of all pertinent art, thinks along conventional wisdom in the art, and is a person of ordinary creativity—not an automaton.

27. I have been asked to consider the level of ordinary skill in the field that someone would have had at the time the claimed invention was made. In deciding the level of ordinary skill, I considered the following:

- the levels of education and experience of persons working in the field;
- the types of problems encountered in the field; and
- the sophistication of the technology.

28. My opinion below explains how a POSA would have understood the

technology described in the references I have identified herein around the September 2016 timeframe. However, my opinions herein would not be affected if the '601 patent is found to be entitled to an earlier priority date.

29. Regardless of whether I use “I” or a “POSA” during my technical analysis below, all of my statements and opinions are always to be understood to be based on how a POSA would have understood or read a document at the time of the alleged invention.

D. Level of Ordinary Skill in the Art

30. A POSA in the technical field of the '601 patent at the relevant time would have had a B.S. degree in Computer Science, Computer Engineering, or an equivalent field, as well as at least 2-3 years of academic or industry experience in the design, analysis, and implementation of operating systems, as well as the design, analysis, and monitoring of computer networks, including issues of network security and network administration, or comparable industry experience.

V. U.S. PATENT. 11,223,601

A. Overview of the '601 Patent

31. The '601 patent describes a system for “isolation of collaboration software on a host computer system.” EX1001, Abstract. This system uses “a workspace” and an “isolated computing environment.” EX1001, 1:62-65.

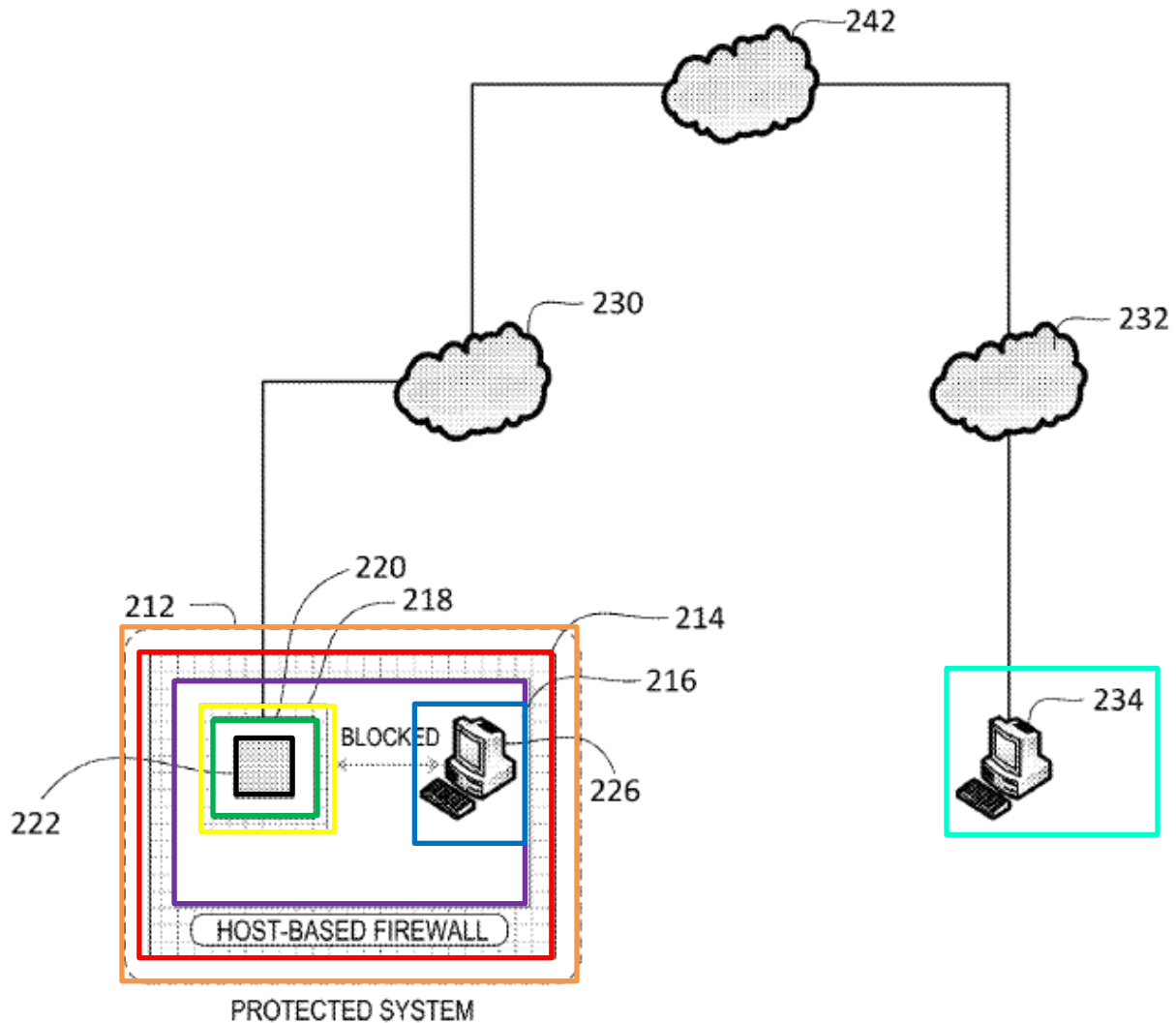
32. Trusted applications run in “the workspace via a first memory space (e.g., a trusted memory space).” EX1001, 1:62-64. Untrusted applications, or

applications that are to be “isolated,” run in “the isolated computing environment via a second memory space.” EX1001, 2:8-9. The isolated computing environment isolates the workspace by “an internal isolation firewall,” which prevents “communication between the isolated computing environment and the workspace.” EX1001, 1:65-2:2. The host computer system is “configured to run a collaboration software that enables interaction with one or more other host computer systems.” EX1001, 3:4-6. The collaboration software runs “within the untrusted memory space.” EX1001, 3:7-8. Example collaboration software include WebEx and GoToMeeting. EX1001, 4:45-47.

33. The workspace includes “one or more processes operating within the operating system that are not restricted by the sandbox container process.” EX1001, 5:44-46. The operating system includes resources enabling operation of both the workspace and the sandbox container. EX1001, 4:46-54. The applications within the sandboxed computing environment are permitted “to access untrusted network destinations via an intermediate connection device [e.g., a web proxy and/or an authentication device].” EX1001, 6:9-14.

34. Figure 2 (annotated below) presents an embodiment of the alleged invention. EX1001, 5. “The host computer system 212 [orange below] may include a host-based firewall 214 [red below], an operating system 226 [blue below], a sandboxed computing environment (e.g., an isolated computing environment) 220

[green below], an internal isolation firewall 218 [yellow below], and a workspace 216 [purple below].” EX1001, 9:63-10:3.



EX1001, FIG. 2 (annotated).

35. Collaboration software (black above) enables the host computer to participate in “software and/or web-based meeting[s].” EX1001, 11:43-45. The collaboration software allows another device to “access, view, interact with, download from, upload to, and/or modify” an application on the host. EX1001,

2:19-23.

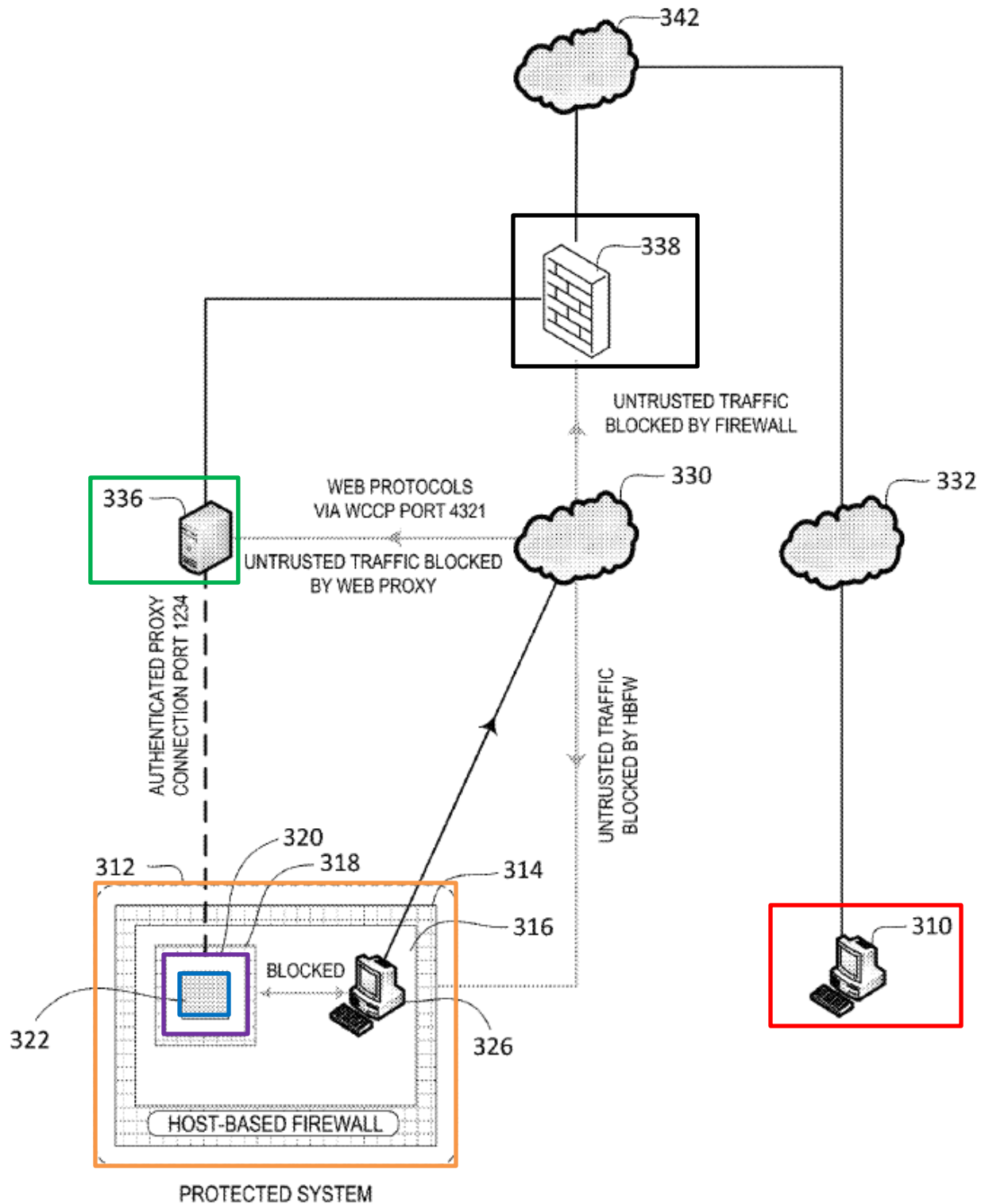
36. The collaboration software operates within “sandboxed computing environment 220,” restricting the other devices’ ability to access the trusted memory (e.g., the workspace) of the host computer. EX1001, 11-12:65-3. For example, “[i]f malware is transferred to the host computer system 212 due to interactions with the host computer system 234 [teal above], the malware may be isolated to the sandboxed computing environment 220” EX1001, 12:25-28.

37. Figure 3 (annotated below) “depicts an example host computer system 312 [orange below] (e.g., such as the host computer system 212 shown in FIG. 2) configured to participate in a software based meeting with one or more other host computer systems, the example host computer system 312.” EX1001, 14:30-34. “The host computer system 312 may be configured to connect to a network 330.” EX1001, 14:35-36. “[N]etwork 330 may send web traffic and/or other protocols (e.g., to support remote meeting software applications) to one or more untrusted network destinations using one or more of a border firewall 338 [black below] or a web proxy 336 [green below].” EX1001, 14:46-50.

38. “The border firewall 338 may enable isolation of the host computer system 312 (e.g., the workspace 316) from one or more untrusted network destinations.” EX1001, 17:34-36. “For example, the border firewall 338 may be configured to block untrusted traffic that is not routed through the web proxy 336. .

. [,] untrusted traffic that is not received from one or more predetermined devices. .
. [, and] untrusted traffic that is not received via one or more predetermined
network ports. EX1001, 17:41-52.

39. “[H]ost computer system 310 [red below] may be configured to
interact with the host computer system 312 via the collaboration software
application 322 [blue below].” EX1001, 16:51-53. “The host computer system 312
and the host computer system 310 may communicate via the respective
collaboration software applications running thereon.” EX1001, 16:60-62. “The
host computer system 312 may allow the host computer system 310 to access the
sandboxed computing environment 320 [purple below].” EX1001, 16:62-64.

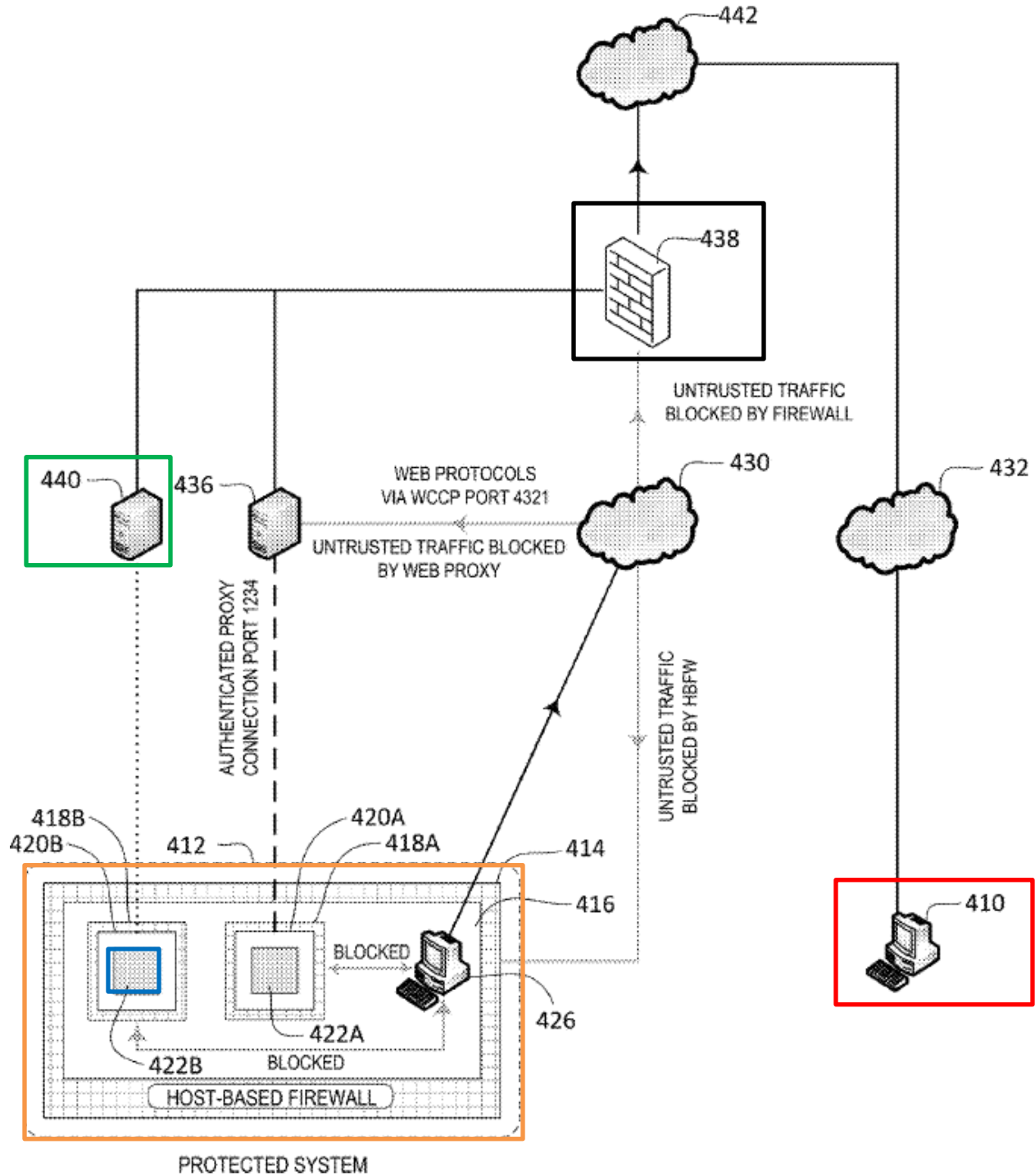


EX1001, FIG. 3 (annotated).

40. Figure 4 (annotated below) “depicts an example host computer system

412 [orange below] (e.g., such as the host computer system 212 shown in FIG. 2) configured to enable separate sandboxed computing environments 420A, 420B” EX1001, 17:53-57. “[H]ost computer system 410 [red below] may be configured to interact with the host computer system 412 via the collaboration software application 422B [blue below].” EX1001, 20:59-61. “The host computer system 412 may be configured to isolate communication with untrusted network destinations . . . using one or more of the host-based firewall 414, the border firewall 438 [black below], the web proxy 436, the termination device 440 [green below], the first internal isolation firewall 418A, the second internal isolation firewall 418B, or a segregation of untrusted memory space.” EX1001, 18:37-44.

41. “The termination device 440 may be associated with the collaboration software application 422B . . . [,] located at the edge of the public or private network and may enable remote access to the public or private network.” EX1001, 21:23-27. “[T]he second sandboxed computing environment 420B may authenticate with the termination device.” EX1001, 21:31-33. “The host computer system 412 may communicate with the host computer system 410 via the termination device 440, the border firewall 438, the Internet 442, and/or the disparate network 432.” EX1001, 21:33-36.



EX1001, FIG. 4 (annotated).

B. Prosecution History

42. The '601 patent was rejected three times during prosecution.

43. First Office Action. The first Office Action included several rejections, including, as relevant here, a rejection of application claims 1-16 under 35 U.S.C. § 103 over U.S. Pat. No. 8,966,464 (“Christopher”) in view of U.S. Pat. Pub. No. 2014/0040979 (“Barton”). EX1002, 103-107. The Examiner cited Christopher as disclosing: (1) a workspace; (2) an isolated computing environment; and (3) an internal isolation firewall. EX1002, 103-104. The Examiner cited Barton as disclosing multi-user interactive software communicating with an untrusted destination via a proxy device. EX1002, 104.

44. In response to the obviousness rejection, the Applicant filed an amendment and remarks. Specifically, Applicant amended independent claims 1 and 9 to recite the system sending data “to an untrusted destination from the multi-user interactive software application associated with the isolated computing environment via a proxy device[.]” EX1002, 132. And the Applicant argued the “managed applications” in Barton cannot send data to untrusted destinations because the managed applications are prohibited from communicating with “unmanaged applications.” EX1002, 139.

45. Second Office Action. The Examiner issued a Final Rejection with the same §103 rejection as in the First Office Action, concluding that Barton discloses managed apps may communicate with unmanaged apps (i.e. untrusted destinations). EX1002, 146-147.

46. The Applicant filed an amendment and a Request for Continued Examination. The Applicant narrowed independent claims 1 and 9 by adding the following element: “authenticat[ing] the isolated computing environment with an authentication device.” The Applicant further amended the sending step to “send data to an untrusted destination from the mulit-user interactive software application via a proxy device when, the isolated computing environment has been authenticated.” EX1002, 166, 167-168. The Applicant again argued that: (1) Christopher fails to send data to an untrusted application; and (2) in Barton, managed apps (e.g., trusted apps) cannot send data to unmanaged apps (e.g., untrusted apps). EX1002, 171.

47. Third Office Action. The Examiner issued a second non-final office action, which, as relevant here, rejected claims 1-16 under § 103 over “SICE: A Hardware-Level Strongly Isolated Computing Environment for x86 Multi-core Platforms” (“Azab”) in view of U.S. Pat. No. 8,566,398 (“Barnier”). EX1002, 183. Azab discloses “a novel framework to provide hardware-level isolation . . . [that] does not rely on any software component in the host environment (i.e., an OS or a hypervisor). EX1034, Abstract. The isolation is implemented via, “hardware, the BIOS, and the System Management Mode (SMM).” EX1034, Abstract. Barnier discloses an extranet including “non-related entities” that may share applications and data through the extranet connection. EX1032, 3:40-45.

48. In response, the Applicant narrowed both the claimed “workspace” and the “isolated computing environment” to use a “host operating system.” EX1002, 199, 201.

implement a workspace that uses a host operating system, wherein the workspace enables ~~is configured to enable~~ operation of a first set of one or more applications or processes via a first memory space;

implement an isolated computing environment that uses the host operating system, wherein the isolated computing environment enables ~~being configured to enable~~ operation of a second set of one or more applications or processes via a second memory space, and wherein the second set of one or more applications or processes comprises a multi-user interactive software application;

EX1002, 199.

49. The Applicant argued that Azab’s isolated environment doesn’t use the host operating system because Azab states that it doesn’t rely on software in the host environment. EX1002, 205. The applicant further argued that Barnier includes no discussion of operating systems. EX1002, 205.

50. Claims 1-16 were allowed and claim 17 was canceled in an Ex Parte Quayle action. The notice stated that the prior art supposedly did not teach the claim limitation that “isolated computing environment that uses the host operating system.” EX1002, 240. But as explained below, this limitation is taught by both Walsh and Austin.

C. Claims

51. The '601 patent issued with 16 claims, two of which are independent.

Independent claims 1 and 9 are shown below. As I show below, the claims recite

Independent Claim 1

A system comprising:

a memory; and

a processor configured to:

implement a workspace that uses a host operating system, wherein the workspace enables operation of a first set of one or more applications or processes via a first memory space;

implement an isolated computing environment that uses the host operating system, wherein the isolated computing environment enables operation of a second set of one or more applications or processes via a second memory space, and wherein the second set of one or more applications or processes comprises a multi-user interactive software application;

isolate the isolated computing environment from the workspace using an internal isolation firewall;

authenticate the isolated computing environment with an authentication device; and

send data to an untrusted destination from the multi-user interactive software application via a proxy device when the isolated computing environment has been authenticated.

EX1001, 22:42-63.

Independent Claim 9

A method comprising:

implementing a workspace that uses a host operating system, wherein the workspace enables operation of a first set of one or more applications or processes via a first memory space;

implementing an isolated computing environment that uses the host operating system, wherein the isolated computing environment enables operation of a second set of one or more applications or processes via a second memory space, wherein the second set of one or more applications or processes comprises a multi-user interactive software application;

isolating the isolated computing environment from the workspace using an internal isolation firewall;

authenticating the isolated computing environment with an authentication device; and

sending data to an untrusted destination from the multi-user interactive software application via a proxy device when the isolated

computing environment has been authenticated.

EX1001, 23:33-24:8.

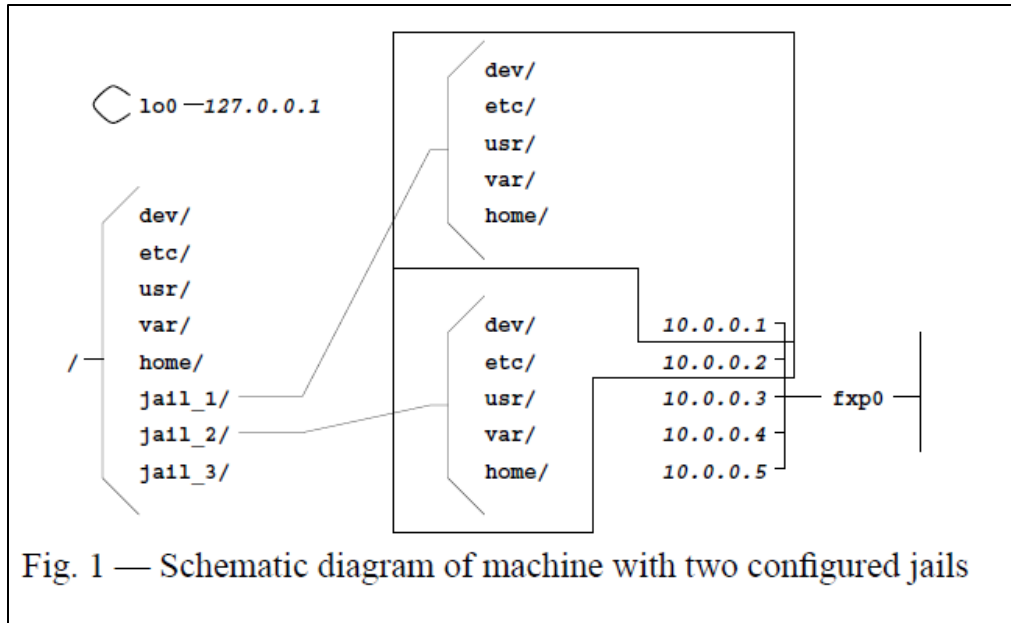
VI. TECHNOLOGY OVERVIEW

52. As I explain in the following sections, process isolation, interactive software applications, device authentication, and proxies existed long before the '601 patent.

A. Isolated Computing Environments

53. Process isolation existed and was ubiquitous before the '601 patent.

54. In 2000, FreeBSD introduced Jails, which are partitioned environments including processes, files systems, and network resources, supporting multiple users while limiting the scope of a jailed process's activities to the jail. EX1015, 4. Once a process is placed in a jail, it cannot leave the jail, and any processes it creates are also placed within the jail. EX1015, 5. A diagram showing two jails is depicted below.

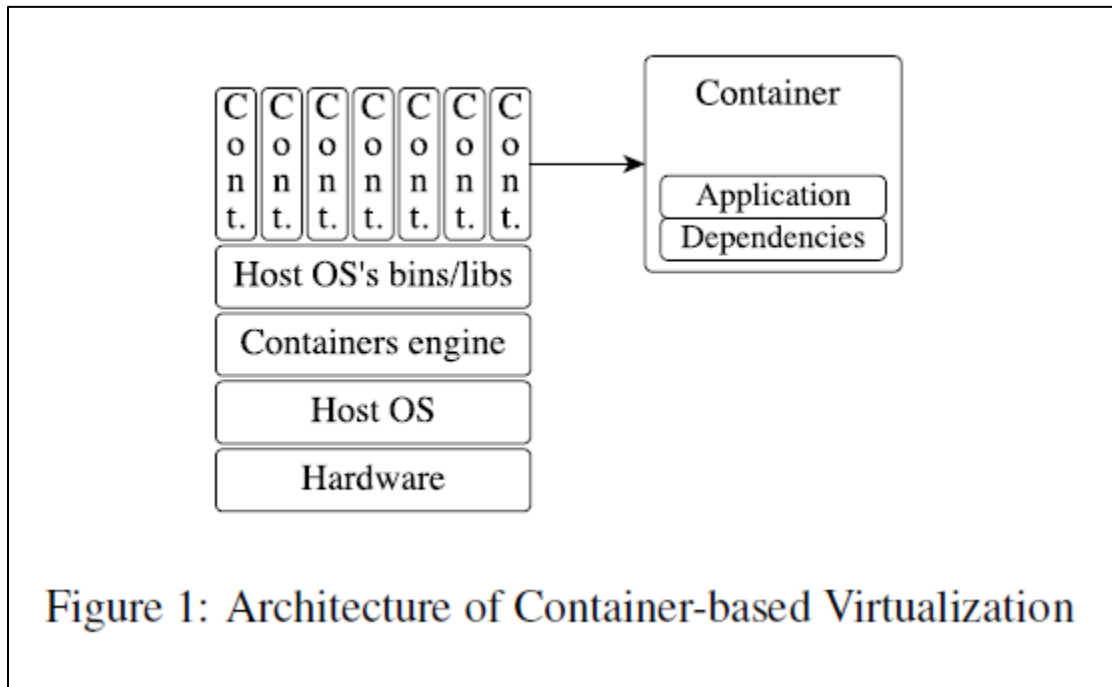


EX1015, 5.

55. Once placed in jail, the process is limited to files of the jail file system root. EX1015, 5. As a result, processes cannot address or manipulate files outside of the jail. EX1015, 5. Additionally, jailed processes are unable to interact with, or even detect processes outside of the jail. EX1015, 6. Additionally, each jail has a single IP address and cannot bind to any other address. EX1015, 5. Unlike virtual machines or other container implementations, FreeBSD jails use the host operating system because they are created and configured using the host operating system. EX1015, 9.

56. Another well-known process isolation method is the use of containers, for example, the use of Docker. Docker implements container-based virtualization where the host operating system runs one or more virtual environments. EX1016,

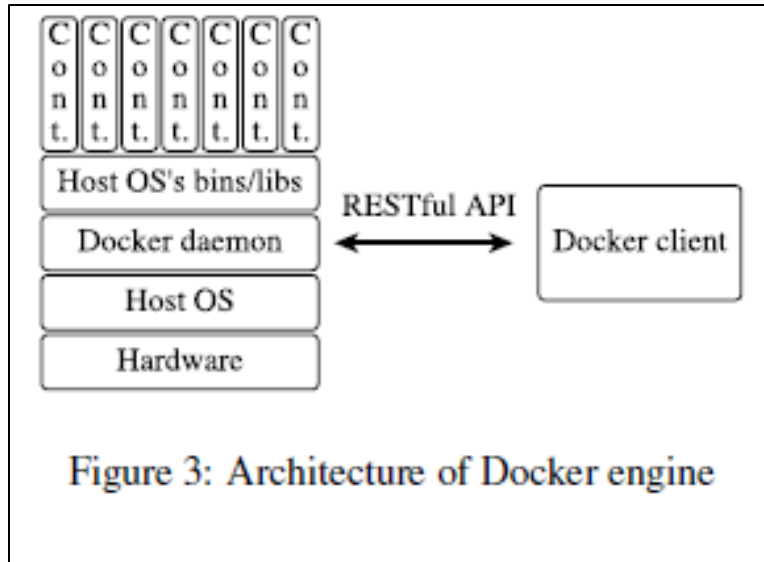
1. A diagram of container-based virtualization is depicted below.



EX1016, Figure 1.

57. As shown, multiple containers are executed without redundantly executing multiple operating systems on the host machine. EX1016, 1. Each container includes the resources required to run the containerized applications. EX1016, 1. Furthermore, containerized applications can share resources with the host operating system. EX1016, 1. For example, containers appear as normal processes running on top of the host operating system. EX1016, 1. However, the containers and the applications executed within them are isolated. EX1016, 1.

58. Docker is a type of container-based virtualization with certain modifications.



EX1016, Figure 3.

59. Docker uses two Linux features for process isolation: namespaces and cgroups. EX1016, 3. Cgroups (control groups) limit resources that a container can access. For example, cgroups can be used to limit the amount of memory and CPU resources a container can consume. EX1028, 4. Similarly, cgroups can limit the amount of disk I/O a process can use. EX1016, 5. Namespaces bundle OS resources so that a process can only access resources allocated to its assigned namespace. EX1016, 3. For example, once a process is assigned to a namespace, it cannot view or access objects outside its assigned namespace. EX1028, 4. However, processes isolated across multiple container namespaces still share the underlying operating system. EX1028, 3-4.

60. Docker implements five types of namespace isolation: (1) process isolation; (2) file system isolation; (3) device isolation; (4) IPC isolation; and (5)

network isolation. EX1016, 3. To achieve process isolation, Docker isolates the process identifier (PID) space of a container from that of the host. EX1016, 4. Additionally, since PID namespaces are hierarchical, a process can only see processes in its own namespace and any child namespaces. EX1016, 4. Since the container PID namespace is isolated from the host PID namespaces, processes in the container cannot view or interfere with host processes (e.g., processes outside the container). EX1016, 4.

61. Docker also isolates the container file system from the host file system. EX1016, 4. Docker achieves this by mounting a different file namespace for each container. EX1016, 4. As a result, processes in a container can only view and interact with file systems mounted to the container. EX1016, 4. For example, if multiple containers are executing on a single host machine, changes made by a process in one container are undetectable by processes in other containers. EX1016, 4. Since Docker still uses the host operating system, containers are provided read-only access to the operating system files. EX1016, 4. Furthermore, Docker containers are prevented from remounting to other file systems. EX1016, 4.

62. Docker further isolates the hardware devices a container can access. EX1016, 4. For example, Docker containers have limited access to the host's physical memory space, storage, and terminal. EX1016, 4. Docker uses a Device

Whitelist Controller to determine what devices a container can access. EX1016, 4.

Isolated processes within containers are also prevented from creating new device nodes to access additional hardware devices. EX1016, 4.

63. Docker also limits inter-process communications (IPC) of containers. EX1016, 4. Processes communicate using objects such as semaphores, message queues, and shared memory. EX1016, 4. To isolate container processes and protect the host, Docker creates an IPC namespace for each container. EX1016, 4. Similar to the file system namespace I discussed above, processes in an IPC namespace cannot access IPC resources outside their assigned namespace. EX1016, 4. As a result, processes in a container cannot interfere with processes executing in other containers through IPC. EX1016, 4.

64. Furthermore, Docker implements network isolation to prevent processes within containers from manipulating network traffic of the host or other containers. EX1016, 4. Each container is assigned its own network stack. EX1016, 4. As a result, a container has its own set of IP addresses, network devices, and IP routing tables. EX1016, 4. As a result, containers can interact with each other as if they were interacting with external network devices. EX1016, 4.

65. Docker is also compatible with various versions of Linux, such as SELinux. EX1016, 1. In SELinux, all files, directories, processes, and system objects have a label defining access control. EX1016, 5. In SELinux, the kernel, as

opposed to the object owner, mediates access control. EX1016, 5-6. As a result, even if an attacker compromises the object owner, the attacker would not have access to the owner's objects. EX1016, 5-6. When Docker is used with SELinux, all content within a container is given a predefined label. EX1016, 6. Additionally, processes within the container are configured to only access content having the predefined label. EX1016, 6. As a result, containerized processes cannot access content or processes in other containers. EX1016, 6.

66. As early as 1993, the need for process isolation was recognized because of the ability of untrusted processes to write and/or read data outside assigned address spaces. EX1027, 1-2. For example, an untrusted process could make a system call to close or delete a file, causing other applications fail. EX1027, 5. To address this challenge, IBM introduced a software encapsulation technique called sandboxing. EX1027, 3. To sandbox an application, the application's virtual address space is divided into segments that each include an identifier. EX1027, 3. The application's object code is modified so that it can only jump to addresses within a code segment of the virtual address space. EX1027, 3. The object code is further modified so that it can only write to addresses within a data segment. EX1027, 3. By separating code and data segments, the application cannot modify its own code. EX1027, 3. By using the segment identifier, attempts by the application to access addresses outside its assigned space are detected and

blocked. EX1027, 4.

67. Process isolation has been implemented at the hardware level. Intel's Software Guard Extensions (SGX) "enables applications to execute with confidentiality and integrity in the native OS environment." EX1026, 1. SGX is implemented as an instruction set architecture extension for "generating hardware enforceable containers" managed by the OS. EX1026, 1. A container, referred to as an "enclave," is a protected area of an application's address space. EX1026, 1. In SGX, each enclave is isolated. EX1026, 2. As a result, attempts to access code or data within an enclave are blocked. EX1026, 1. Specifically, an attempt by a process to access the memory of an enclave is treated as if the memory address does not exist. EX1026, 5.

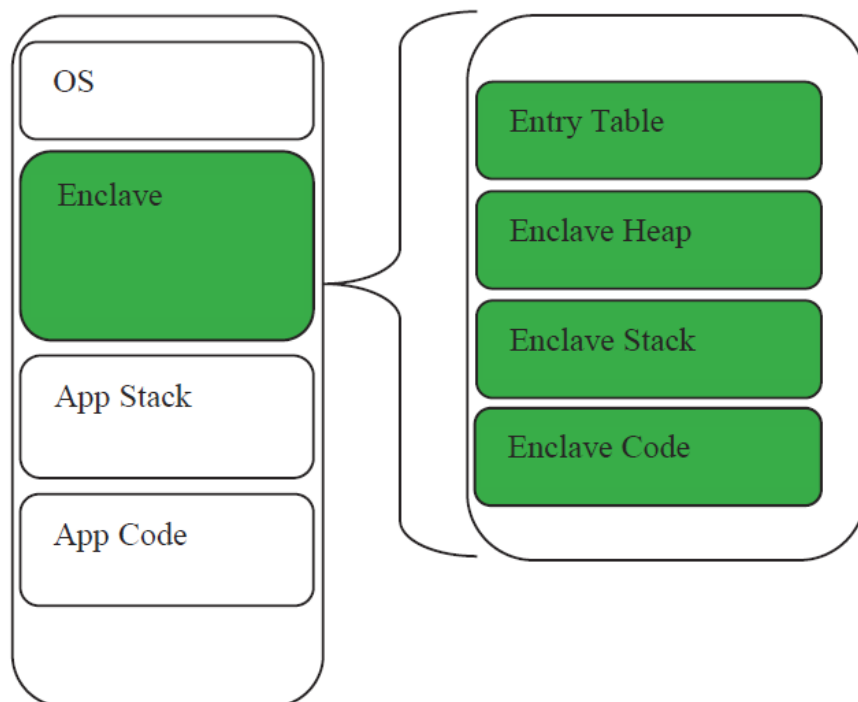


Figure 1: Enclave within Application's Virtual Address Space

EX1026, Figure 1.

68. Solaris Zones, introduced by Sun Microsystems in 2004, is another process isolation solution. Solaris zones are “lightweight ‘sandboxes’ within an operating system instance, in which one or more applications may be installed and run without affecting or interacting with the rest of the system.” EX1029, 2. Applications within a zone cannot “locate, examine, or signal a process in another zone.” EX1029, 3.

69. The operating system kernel assigns each zone an identifier. EX1029, 6. The identifier is also associated with each process executing within the zone. EX1029, 5. The operating system uses the identifier to enforce isolation between zones by inspecting process identifiers. For example, if a process invokes the process identifier of another zone, the operating system blocks the call and returns an error indicating that the process does not exist. EX1029, 7. Each zone is also assigned its own set of IP addresses. EX1029, 8. As a result, processes within the zone can send and receive network traffic from only the assigned IP addresses. EX1029, 8. From an external perspective, the zones appear like a set of distinct servers. EX1029, 8.

70. Zones also support resource pools where an administrator assigns resources exclusively to the processes of a zone. EX1029, 10. For example, a portion of CPU can be bound to a zone. EX1029, 10. As a result, processes within

the zone can only use the bound resource, and are “completely isolated in terms of resource usage.” EX1029, 10.

71. Most, if not all of the isolation techniques allegedly invented by the '601 patent are described above. For example, the '601 patent claims “an isolated computing environment” and states that the isolated computing environment “may refer to a memory space that is isolated . . . from another memory space.” EX1001, 5:33-37, 22:49. As I discussed above, the sandboxing technique disclosed by IBM in 1993, isolates untrusted software by restricting the memory the isolated software can access. EX1027, 3-4. Intel’s SGX isolates processes within enclaves, and attempts by processes to access memory inside the enclave are blocked. EX1026, 5. Solaris assigns resources pools to processes isolated within a zone, preventing them from accessing any resources outside the pool. EX1029, 10.

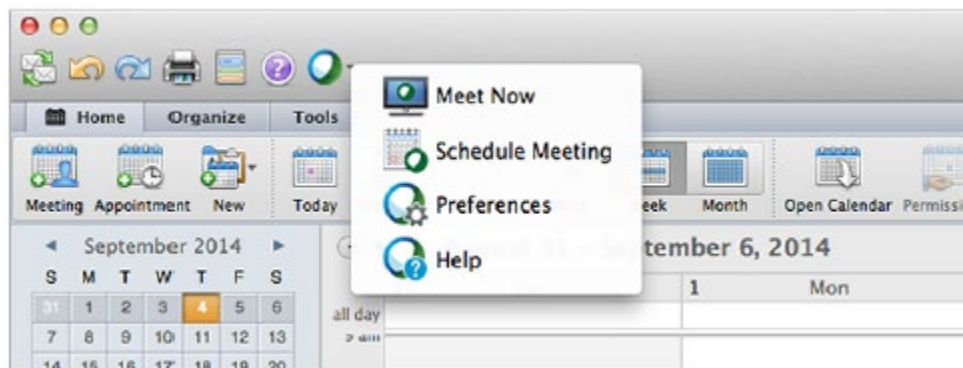
72. The '601 patent states that the “isolated computing environment . . . uses the host operating system” EX1001, 22:49-51. Again, IBM’s sandboxed processes use the host because the operating system prevents sandboxed software from accessing memory outside the sandbox. EX1027, 3-4. FreeBSD Jails isolate processes within jails, thereby preventing them from accessing files or processes outside of the jail. EX1015, 5-6. FreeBSD jails use the host operating system. EX1015, 9. Docker executes processes within isolated containers that share the underlying host operating system. EX1016, 1. Solaris isolates processes within

zones, and then uses the operating system to block attempts by a process to invoke the identifier of a process outside the zone. EX1029, 7.

B. Multi-User Interactive Software Applications

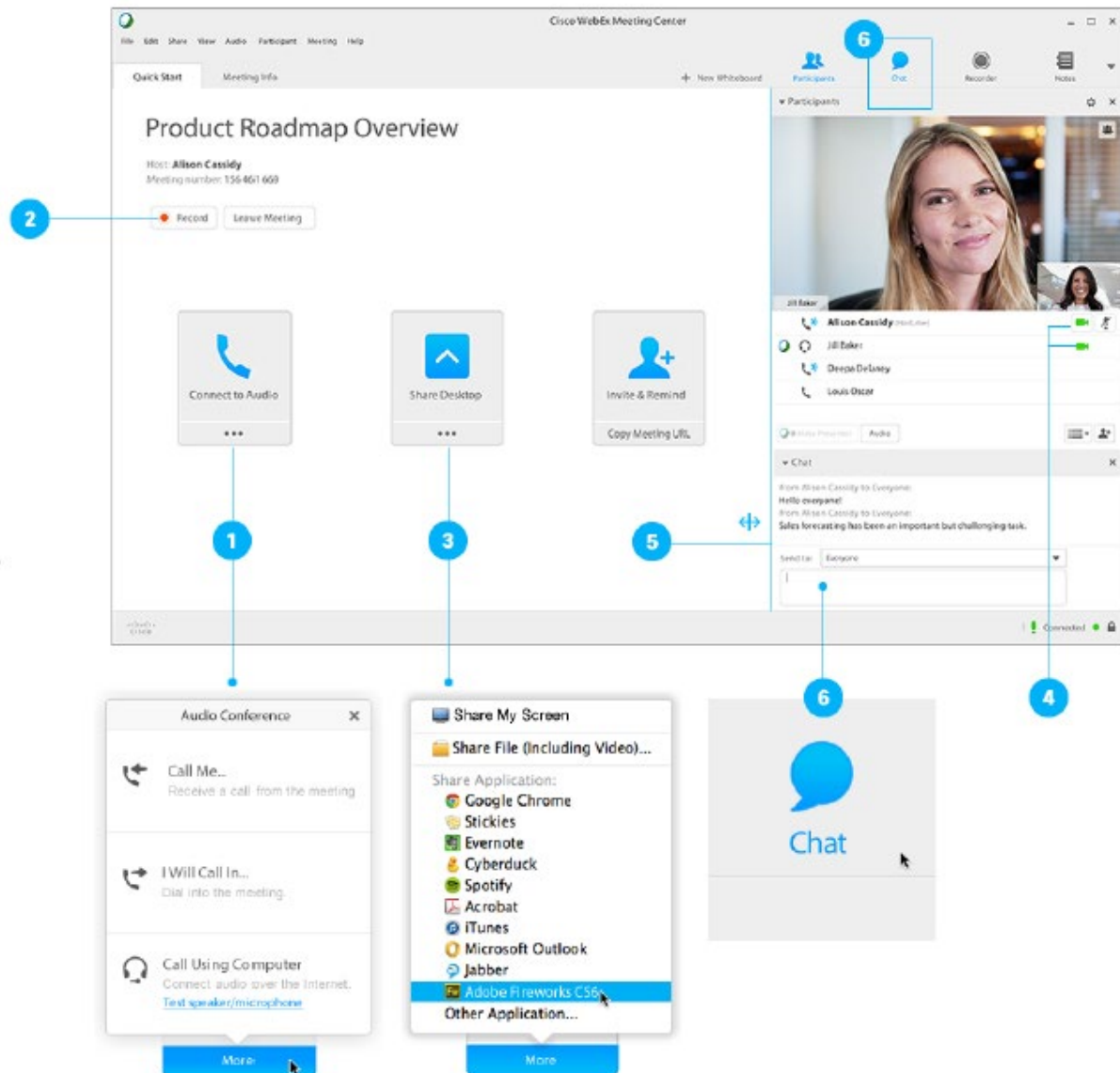
73. The '601 patent itself recognizes that multi-user interactive software such as WebEx and GoToMeeting existed before the alleged invention. EX1001, 4:45-47.

74. WebEx is a video conferencing tool allowing individuals to “[d]eliver presentations, discuss ideas, and brainstorm” using integrated audio and video. EX1033, 1. In 2007, Cisco purchased WebEx, which at the time had 64% of the online meeting market and more than 3.5 million monthly WebEx users. EX1023, 2. By 2014, WebEx had over 60 million users, handling over 1 million meetings per day. EX1017, 1. With WebEx, users could start and join meetings by using a web browser or by using plug-ins, such as the one integrated with Microsoft Outlook below.



EX1018, 2.

75. Once the meeting is started, WebEx supported various multi-user interactive features. For example, users could record audio, video, and materials presented during the meeting. EX1018, 1. The user can also choose to share their desktop, files, applications, and videos. EX1018, 1.

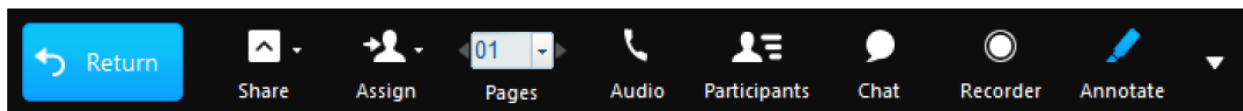


EX1018, 1

76. When sharing a desktop, the presenter can select which specific

screen to share with other users. EX1011, 2; EX1019, 1. Presenters can also share web browsers. For example, when the presenter navigates to a URL, participants see the same website displayed in the presenter's browser. EX1011, 2; EX1019, 1. Since the presenter controls what is displayed, when the presenter navigates the webpage, the participant's view is updated. WebEx users can also connect to and remotely control computers via the Access Anywhere agent. EX1011, 2; EX1019, 1

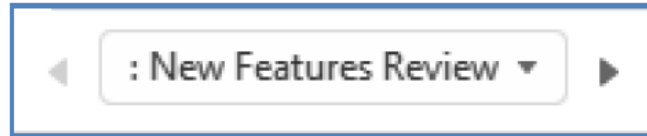
77. When presenters are sharing content, they can annotate the content and, if desired, save it for redistribution. EX1011, 1; EX1019, 1. For example, when the presenter shares content in a full screen mode, a control pane is displayed at the top of their screen including an "Annotate" option. EX1011, 3; EX1019, 1.



EX1011, 3.

78. With annotation, the presenter can use a pointer to draw attention to parts of the shared screen. EX1011, 1; EX1019, 2. The presenter can also use the text tool to type and display text that is shown to participants. EX1011, 1; EX1019, 2. The presenter can also use a line tool to create lines, a shape tool to create various shapes, and a pen tool to draw freehand. EX1011, 1; EX1019, 2-3. Finally, the presenter can delete annotations using an eraser tool. EX1011, 1; EX1019, 3.

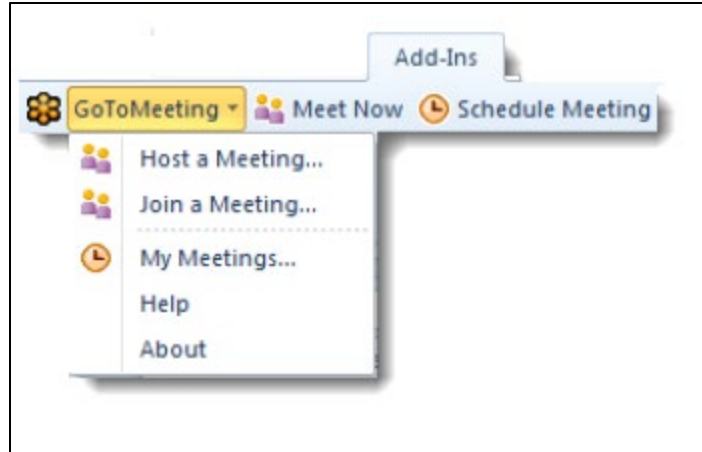
When presenting, the presenter can use page controls to change what participants see. EX1011, 1; EX1019, 2. For example, the presenter can use the page controls shown below to move through slides.



EX1011, 1.

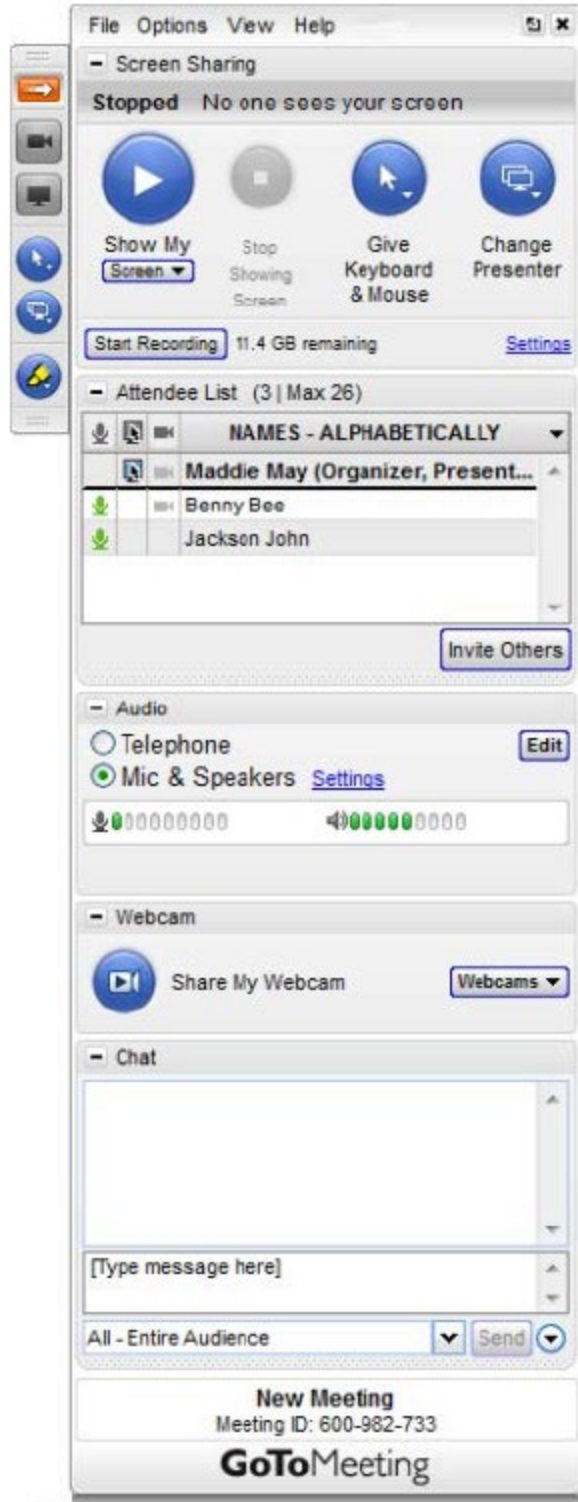
79. Similarly, GoToMeeting was another video conferencing software available before the '601 patent. With GoToMeeting, users “can collaborate on documents, deliver presentations, perform product demonstrations and securely share confidential information from anywhere, at any time.” EX1024, 10. GoToMeeting was configured to run on devices executing Windows operating system, Mac OS, iOS, and Android. EX1024, 12-13. GoToMeeting was configured to run from a browser or as a downloaded application. EX1024, 12-13.

80. Once installed, GoToMeeting integrated with other applications. For example, as shown below GoToMeeting integrated with Outlook, allowing users to host and join meetings via the Outlook toolbar. EX1024, 31. GoToMeeting could also be integrated with Microsoft Word, Excel, and PowerPoint. EX1024, 31.



EX1024, 31.

81. Once a GoToMeeting was started, the organizer was presented with the Presenter Control Panel shown below. EX1024, 92. The Presenter Control Panel includes panes such as “Screen Sharing, Attendee List, Audio, Chat and the Grab Tab.” EX1024, 92.



EX1024, 92.

82. Using the Screen Sharing pane shown below, the organizer can share

their screen, pause screen sharing, pass keyboard control, and change the presenter.

EX1024, 92. “The Show My Screen button starts or pauses a presenter's ability to

show the entire desktop to attendees.” EX1024, 93. The “Screen drop-down

button” allows the presenter to select specific application windows to share.

EX1024, 93. The “Give Keyboard & Mouse button” allows another attendee to

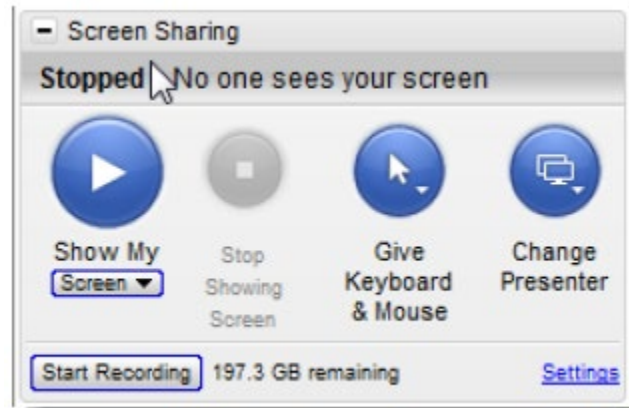
control the mouse and keyboard of the presenter. EX1024, 93. Once the presenter

gives control of the keyboard and mouse to another attendee, they can regain

control by pressing a key or moving the mouse. EX1024, 95. Attendees can also

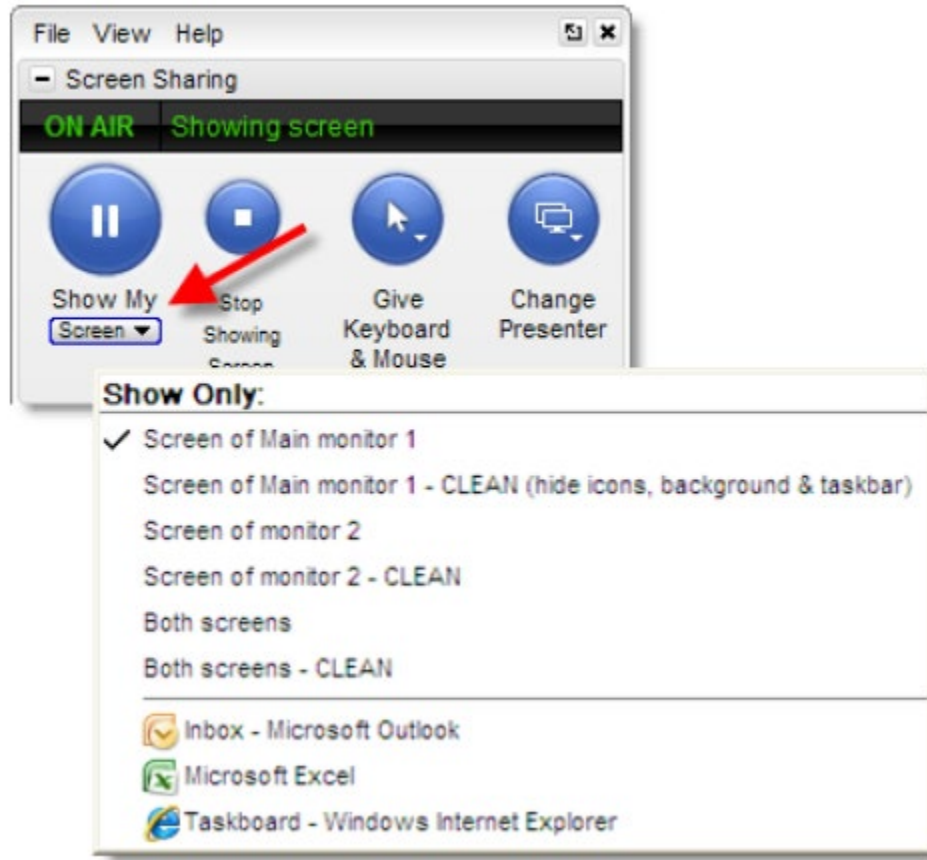
request keyboard and mouse control from the presenter. EX1024, 95. The “Change

Presenter button” gives the presenter role to another attendee. EX1024, 93.



EX1024, 93.

83. For example, shown below the presenter can indicate which monitor or application to share with meeting attendees. EX1024, 94.



EX1024, 94.

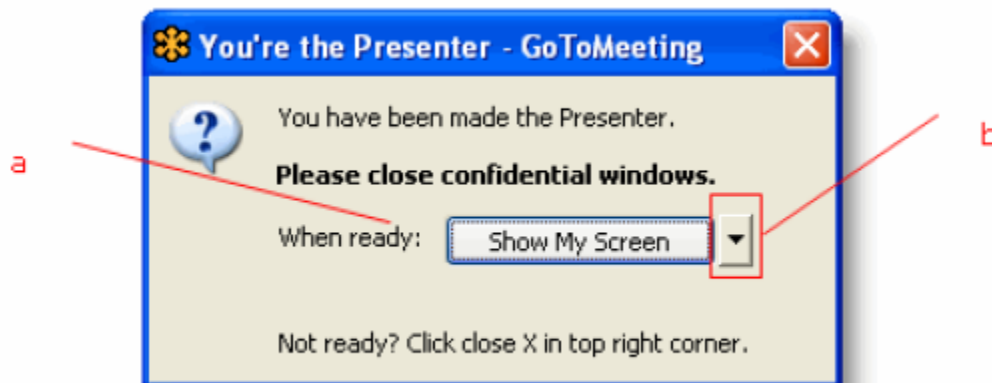
84. The presenter can also use drawing tools to draw, highlight, and point to items on the screen. EX1024, 16. For example, the presenter can use the drawing tools to “annotate directly on any screen being shared.” EX1024, 59.

85. In GoToMeeting, the attendee interacts with the presenter’s shared screen via a Viewer window depicted below. EX1024, 143. The Viewer window can be displayed in a window or full-screen mode. EX1024, 143.



EX1024, 143.

86. When an attendee is made presenter, a dialog box is displayed allowing them to their share their screen with the other attendees. EX1024 146. As I discussed above, the attendee can “present his or her entire desktop (a) or click the down arrow to select a specific application window (b).” EX1024 146.



EX1024, 146.

87. Indeed, both WebEx and GoToMeeting disclose features of the multi-user interactive software application as recited in the '601 patent. Both WebEx and GoToMeeting allow a remote machine to view the host application. *See* EX1011, 2

(discloses sharing a desktop, a running application, and a web browser with meeting participants); EX1019, 1 (discloses sharing a desktop, a running application, and a web browser with meeting participants); EX1024, 93 (discloses sharing the presenter's entire desktop or specific applications with meeting attendees). Both WebEx and GoToMeeting allow a remote machine to interact with the host application. *See* EX1011, 3 (discloses assigning another participant the meeting presenter); EX1019, 1 (discloses making another participant the presenter); EX1024, 96 (discloses assigning another attendee the role of presenter). Both WebEx and GoToMeeting allow a remote machine to upload to the host machine and download data from the host machine. *See* EX1011, 2 (discloses sharing files with meeting participants); EX1019, 1 (discloses sharing files during a meeting); EX1024, 112 (discloses sharing files with participants); EX1024, 182 (discloses uploading files during a meeting). Both WebEx and GoToMeeting allow a remote device to control a host's display. *See* EX1011, 2-3 (discloses synchronizing participant's view with presenter's view); EX1011, 1 (discloses sharing content in full-screen); EX1019, 2 (discloses displaying shared content in a full-screen view); EX1019, 2 (discloses synchronizing participant's and presenters views so that both are seeing the same content); EX1024, 93 (discloses sharing the presenter's entire desktop or specific applications with meeting attendees).

VII. IDENTIFICATION OF THE CHALLENGE (37 C.F.R. § 42.104(B))

A. Citation of Prior Art

88. The '601 patent was filed September 25, 2018. EX1001, Cover (22). It claims priority to a provisional application filed September 28, 2017. EX1001, Cover (60). I have been informed that Petitioner does not concede that the challenged claims of the '601 patent are entitled to the provisional application filing date. Nonetheless, I have been informed that all references relied on here qualify as prior art even if it is.

89. **Austin:** U.S. Patent Application Publication No. 2016/0306964 to Austin et al. ("Austin," EX1005) describes methods for isolating untrusted content in a secondary user account that does not have access to a primary user account. EX1005, Abstract. Austin was published on October 20, 2016, and I have been informed that it qualifies as prior art under §102(a)(1). EX1005, cover (43).

90. **Walsh:** U.S. Patent No. 8,640,187 to Walsh et al. ("Walsh," EX1006) describes methods for creating an isolated execution environment to access untrusted content "without exposing other processes and resources of the computing system to the untrusted content." EX1006, Abstract. Walsh issued on January 28, 2014, and I have been informed that it qualifies as prior art under §102(a)(1). EX1006, cover (45).

91. **Bloch:** U.S. Patent No. 7,849,502 to Bloch et al. ("Bloch," EX1007)

describes methods for using a proxy device and authentication server to provide integrated authentication services. EX1007, 8:20-24, FIG. 6B. Bloch issued on December 7, 2010, and I have been informed that it qualifies as prior art under §102(a)(1). EX1007, cover (45).

92. **Innes:** U.S. Patent No. 9,154,488 to Innes et al. (“Innes,” EX1012) describes methods for authenticating and accessing remote resources using a proxy device. EX1012, Abstract. Innes issued on October 6, 2015, and I have been informed that it qualifies as prior art under §102(a)(1). EX1012, cover (45).

93. **Walker:** U.S. Patent Application Publication No. 2016/0149863 to Walker et al. (“Walker,” EX1008) describes methods for managing a firewall on a host computing device. EX1008, Abstract. Walker published on May 26, 2016, and I have been informed that it qualifies as prior art under §102(a)(1). EX1008, cover (43).

94. **Franco:** U.S. Patent No. 8,161,563 to Franco et al. (“Franco,” EX1009) describes methods for executing an Internet-related application “with a reduced privilege level” to “prohibit the application from accessing portions of an associated computing device.” EX1009, Abstract. Franco issued on April 17, 2012, and I have been informed that it qualifies as prior art under §102(a)(1). EX1009, cover (45).

95. **WebEx:** This petition relies on two papers describing functionality of

Cisco's WebEx product. These papers are collectively referred to as "WebEx":

- a. Cisco WebEx, Productivity Tools (2016) (EX1010) was published on the Internet by Cisco. Its public availability is confirmed by an Internet Archive WayBack Machine capture of the publication in its entirety on the Cisco website at least as early as April 23, 2016. EX1014, Exhibit A at 10-12. I have been informed that it that EX1010 qualifies as prior art under §102(a)(1).
- b. Cisco WebEx, Sharing and Presentation Tools (2016) (EX1011) was published on the Internet by Cisco. Its public availability is confirmed by an Internet Archive WayBack Machine capture of the publication in its entirety on the Cisco website at least as early as April 23, 2016. EX1014, Exhibit B at 1-4. I have been informed that it that EX1011 qualifies as prior art under §102(a)(1).

96. **Netfilter:** Netfilter, What is netfilter.org? (2016) (EX1013) was published on the Internet in 2016 by the netfilter.org project. Its public availability is confirmed by an Internet Archive WayBack Machine capture of the publication in its entirety on the netfilter.org website at least as early as October 11, 2016. EX1014, Exhibit A at 6-8. I have been informed that it that EX1013 qualifies as

prior art under §102(a)(1).

VIII. OVERVIEW OF THE PRIOR ART

97. This section provides an overview of the prior art and demonstrates that process isolation and proxy network communications were well-known in the industry well-before the priority date of the '601 patent.

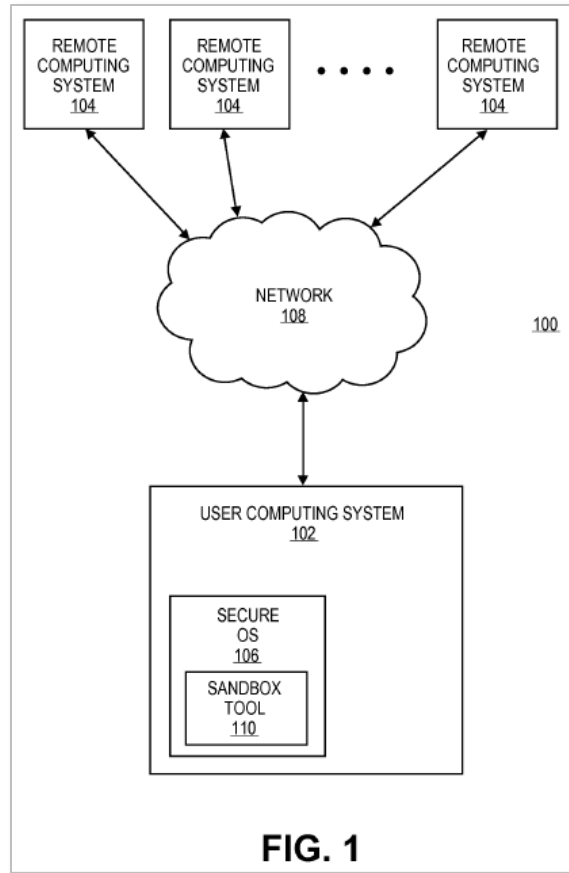
A. Walsh

98. Walsh is directed to “computer security” in “distributed computing environments.” EX1006, 1:8-13. In these environments, Walsh acknowledges that “malicious entities can plant viruses, Trojans, or other malicious agents in publicly available content in order to attack the users’ computing systems and steal sensitive information from the users.” EX1006, 1:19-23. Although anti-virus programs may attempt to identify and block dangerous content, these programs need to know the identity of the dangerous content before they are able to block them. EX1006, 1:25-34. Furthermore, while modern operating systems block programs that attempt to go beyond their levels of access, it hinders the user’s ability to use the program. EX1006, 1:42-49. Thus, Walsh recognizes the need to run untrusted programs while also protecting the user’s computer. EX1006, 1:51-53.

99. To address these issues, Walsh discloses a sandbox tool that works with a secure operating system to create isolated execution environments. EX1006,

Abstract. As a result, untrusted content (such as, “programs” and “applications”) is accessed “without exposing a computing system to [the] untrusted content and possibly malicious content.” EX1006, 2:32-36.

100. Walsh’s Figure 1 (annotated below) depicts an exemplary network including “user computing system 102 and remote computing systems 104.” EX1006, 3:24-38. The secure OS on the user computing device is used to access data and run programs such as word processors, web browsers, file sharing apps, email clients, multimedia apps, and chat applications. EX1006, 3:64-4:4. The secure OS is also used to access files and data, for running programs or by programs. EX1006, 4:4-6. The content on the user computing system is acquired from physical media such as CDs/DVDs or downloaded from other devices such as a remote computing system. EX1006, 4:8-13. Since the content might be unsecure, “sandbox tool” is configured to cooperate with components of the secure OS 106 to create an isolated execution environment for accessing content (trusted or untrusted) without exposing other processes and resources of the user computing system 102 to the content. EX1006, 4:23-35.



EX1006, FIG. 1.

101. Exemplary software for providing such content isolation is illustrated in Walsh’s Figure 2 (annotated below). “User execution environment 220 allows users to interact with the user computing system 102 to access content such as run application[s] and programs, view files, etc.” EX1006, 5:30-62. The sandbox tool allocates resources such as storage space and memory to create the isolated execution environment for accessing content. EX1006, 5:57-61. As a result, content in the isolated execution environment only has access to the resources allocated to the isolated execution environment. EX1006, 5:61-65. Accordingly,

untrusted content executed in the “isolated execution environment 230” (in red) does not expose “other processes such as the user execution environment 220” to the untrusted content. EX1006, 5:50-57.

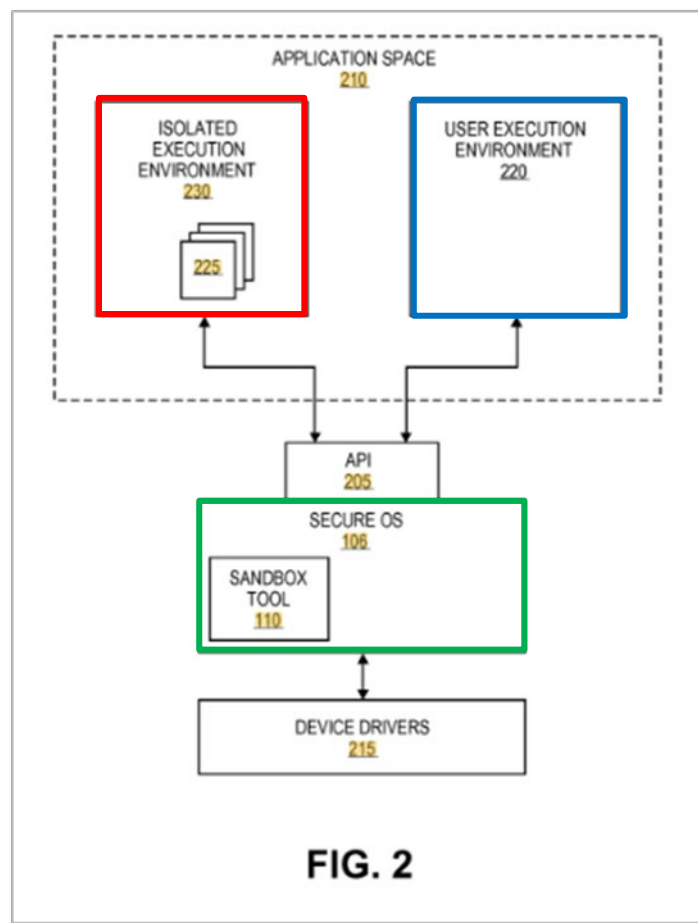
102. The “user execution environment 220” and the “isolated execution environment 230” both use a “secure OS” (in green), but each are provided with their own memory “namespace” so that “the processes of isolated execution environment 230 cannot access files and processes of any other namespace” EX1006, 6:23-33.

103. In an embodiment, the sandbox tool creates the isolated execution environment by using the secure OS to create a namespace for the isolated environment that is separate from the user execution environment. EX1006, 6:4-9. For example, the sandbox tools copies the necessary content (e.g., files, programs) into the separate namespace allowing the isolated execution environment to function. EX1006, 6:9-14. The sandbox tools also copies the untrusted content into the separate namespace. EX1006, 6:17-22.

104. To further protect the computer and user environment, the sandbox tools and secure OS unshare any other namespaces with the separate namespace of the isolated execution environment. EX1006, 6:26-29, such that, processes in the isolated execution environment cannot access files or programs of other namespaces. EX1006, 6:31-34. Furthermore, the sandbox tool in cooperation with

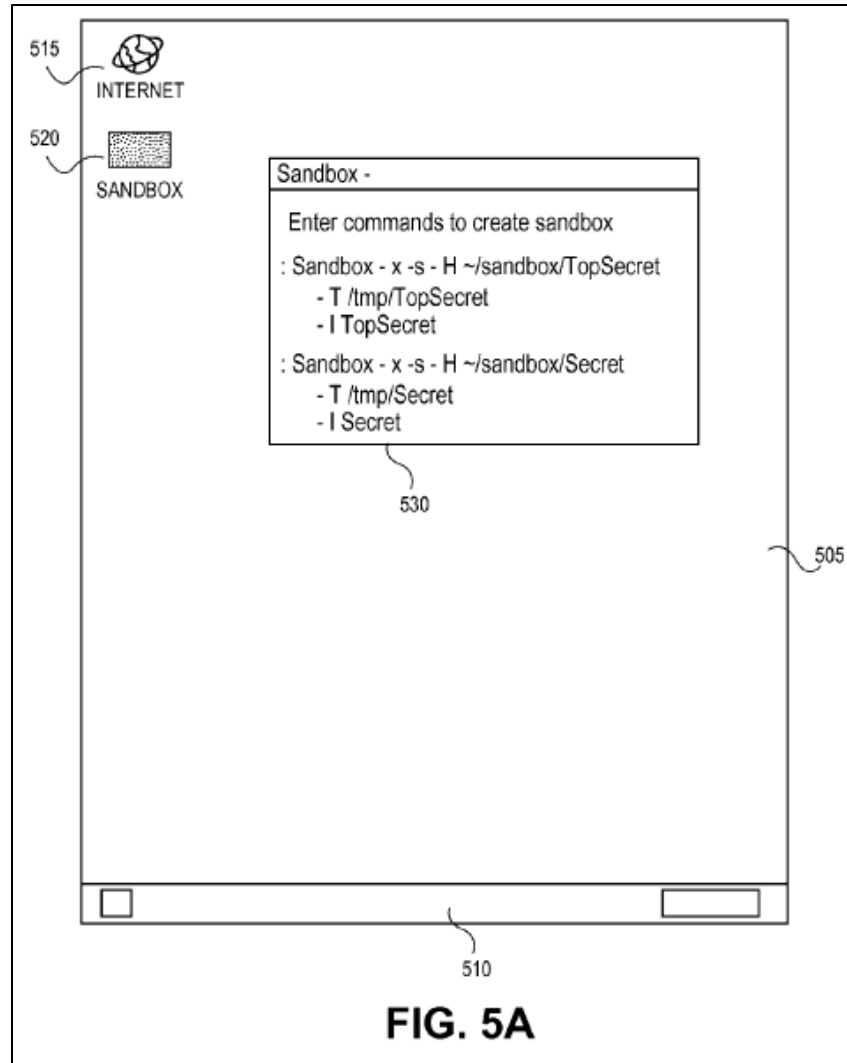
the secure OS creates a separate user interface allowing the user to interact the isolated computing environment. EX1006, 6:37-40. Additionally, the sandbox tool and secure OS assign security policies to the isolated execution environment that define the level of access processes within the isolated environment have to the computer's resources via the device drivers. EX1006, 6:49-59.

105. “The content,” Walsh explains, can be accessed over the Internet via a “web browser” application. EX1006, 3:67-4:4. Trusted content is executed in a “user execution environment 220” (in blue). EX1006, 5:30-32.



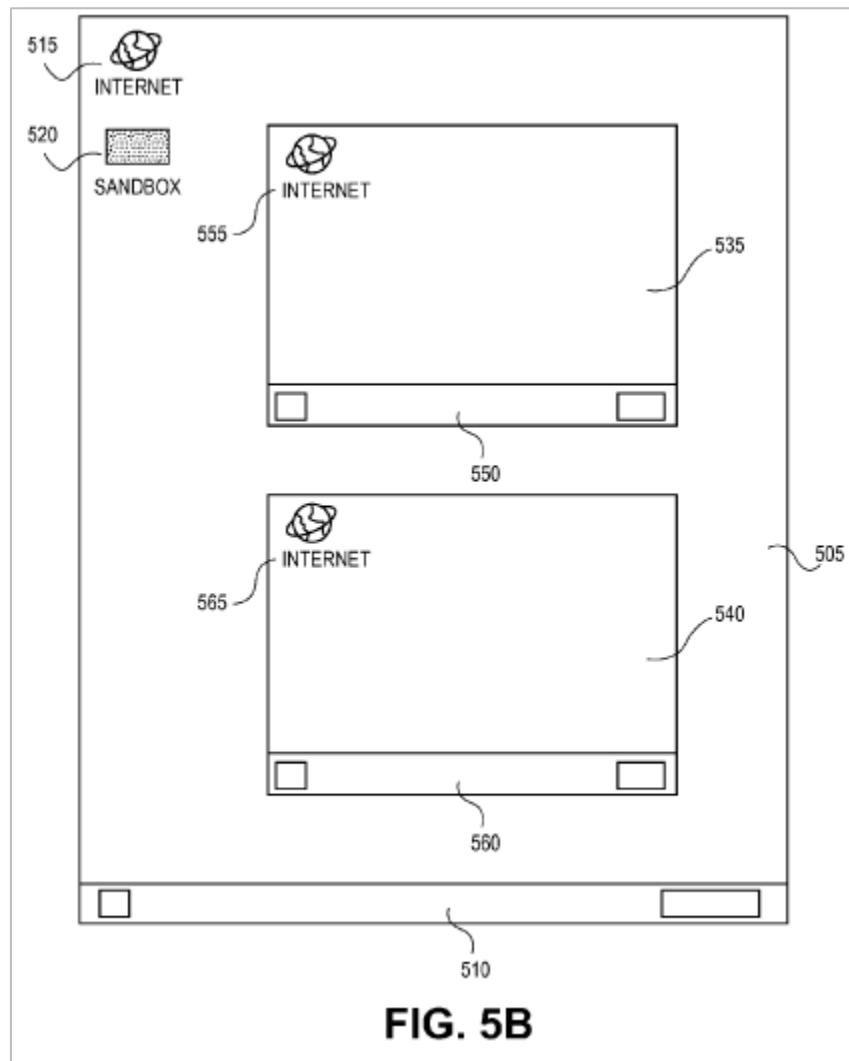
EX1006, FIG. 2 (Annotated).

106. Figure 5A below shows the desktop of the user execution environment where the user can interact with “sandbox icon 520” to initiate the sandbox tool. EX1006, 11:58-12:2. The sandbox tool launches “sandbox interface 530” where the user defines specifications of the isolated execution environment. EX1006, 12:2-7. For example, the user can create two isolated execution environments, assigning each to a namespace and providing the security context. EX1006, 12:9-12.



EX1006, FIG. 5A.

107. Figure 5B below depicts a user interface including both isolated execution environments. EX1006, 12:22-26. As depicted, the user interfaces of the isolated environments include menus and widgets such as “tool bar 550 and application icon 555” to, for example, access a web browser. EX1006, 12:26-29.

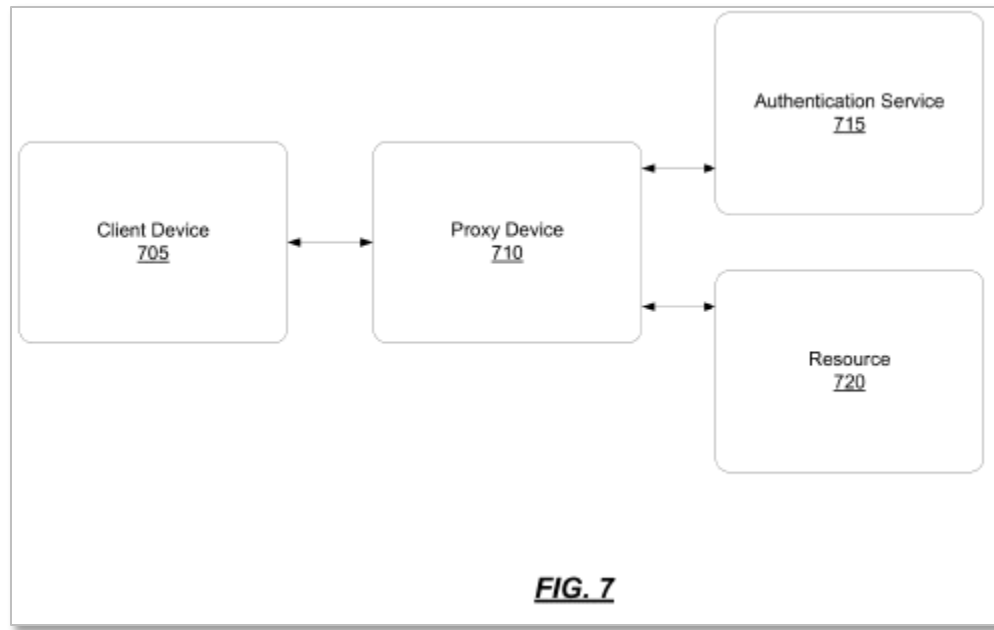


EX1006, FIG. 5B.

B. Innes

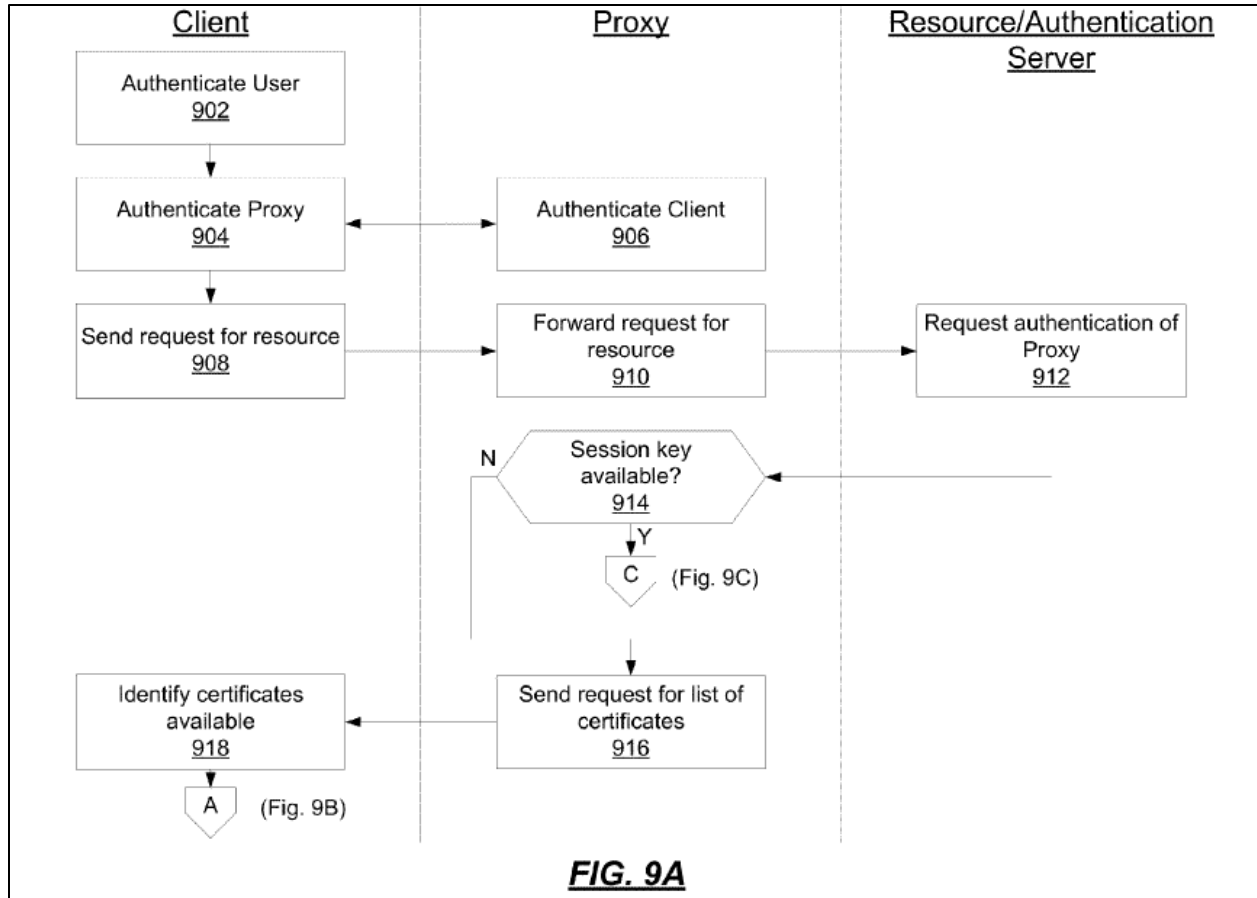
108. Innes recognizes that in traditional private networks, proxy devices act as transparent pipes allowing client devices to directly access private network resources. EX1012, 1:12-15. As a result, the proxy device is unable to “control or monitor the activities of the client device.” EX1012, 1:23-24. Innes further recognizes that the client device needs to be configured to communicate on the private network, and access the resource. EX1012, 1:20-22.

109. To overcome these problems, Innes describes systems and methods for accessing secured resources using a proxy. EX1012, 21:62. As depicted below in Figure 7, a client device communicates with “resources 720” and “authentication services 715 using a proxy device 710.” EX1012, 22:2-4. For example, the client device may send a request for resources to the proxy device. EX1012, 22:13-15. Resources include documents, emails, services, or files. EX1012, 22:14. The resource may authenticate the client devices’ request by, for example, requesting a signature from a client certificate. EX1012, 22:18-20.



EX1012, FIG. 7.

110. Figures 9A-9C below depict steps for authentication and secure resource access via a proxy device. EX1012, 27:43-45. At 902, the user is authenticated by providing credentials to the client device. EX1012, 27:62-65. Next the client device authenticates with the proxy using SSL server authentication. EX1012, 27:66-28:5. Subsequently, the client device requests access to a resource (e.g., web resources, enterprise resources). EX1012, 28:35-39. The request is sent to an authentication service. EX1012, 28:45-48. The proxy can authenticate the request using the client's certificate provided during a prior authentication session. EX1012, 28:62-29:14.



EX1012, FIG. 9A (Annotated).

111. If the proxy device does not have access to a client device certificate, at 928 the proxy device generates a request for a signature from the client device. EX1012, 31:33-35. The proxy device includes context information with the request. EX1012, 31:53-55. The context information is used by the client device to verify the authentication session between the proxy device and authentication server. EX1012, 31:62-65.

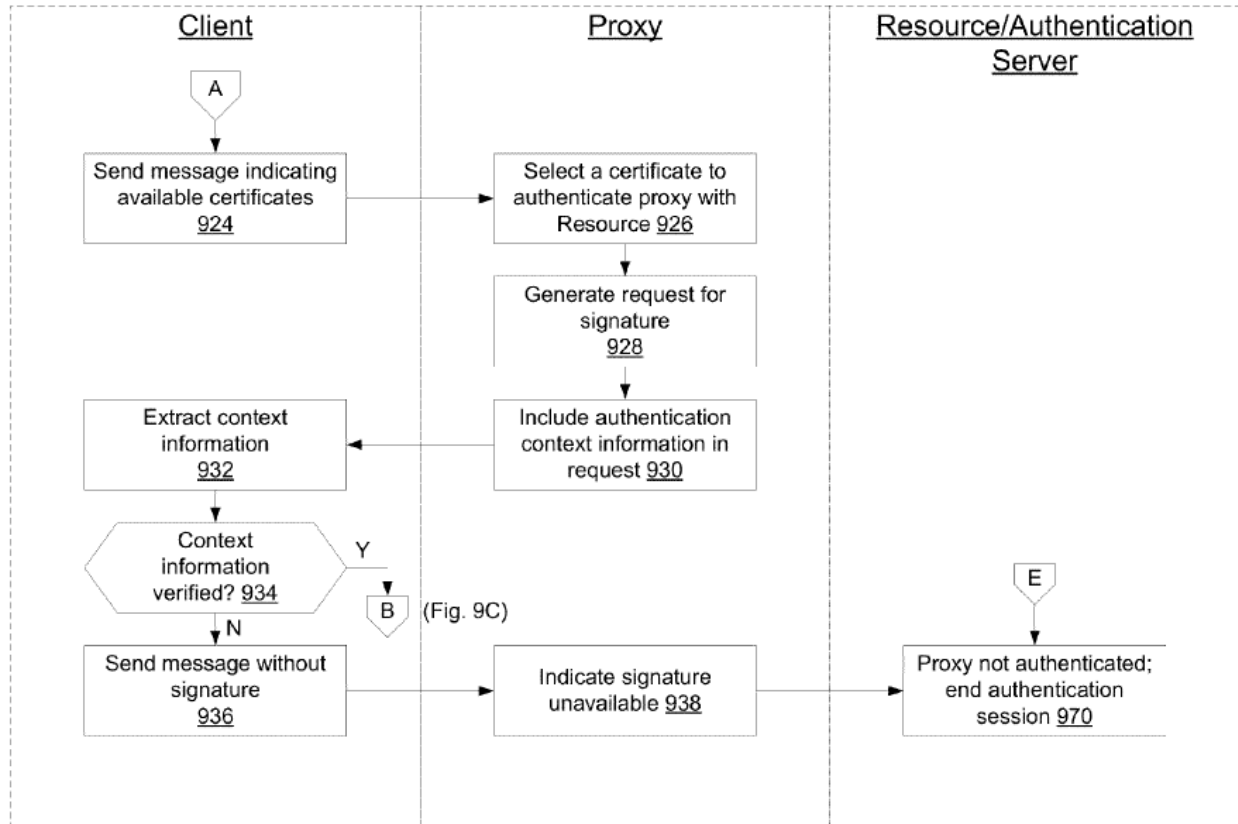


FIG. 9B

EX1012, FIG. 9B (Annotated).

112. For example, as shown below at 942, the client device signs a message and sends it to the proxy device. EX1012, 34:64-66. The proxy device then extracts the signature and forwards it to the authentication server. EX1012, 35:11-23. The authentication server validates the signature and sends the proxy device a session key to obtain the resource. EX1012, 42-62. The proxy device obtains the resource using the session key and sends it to the client device. EX1012, 37:19-46.

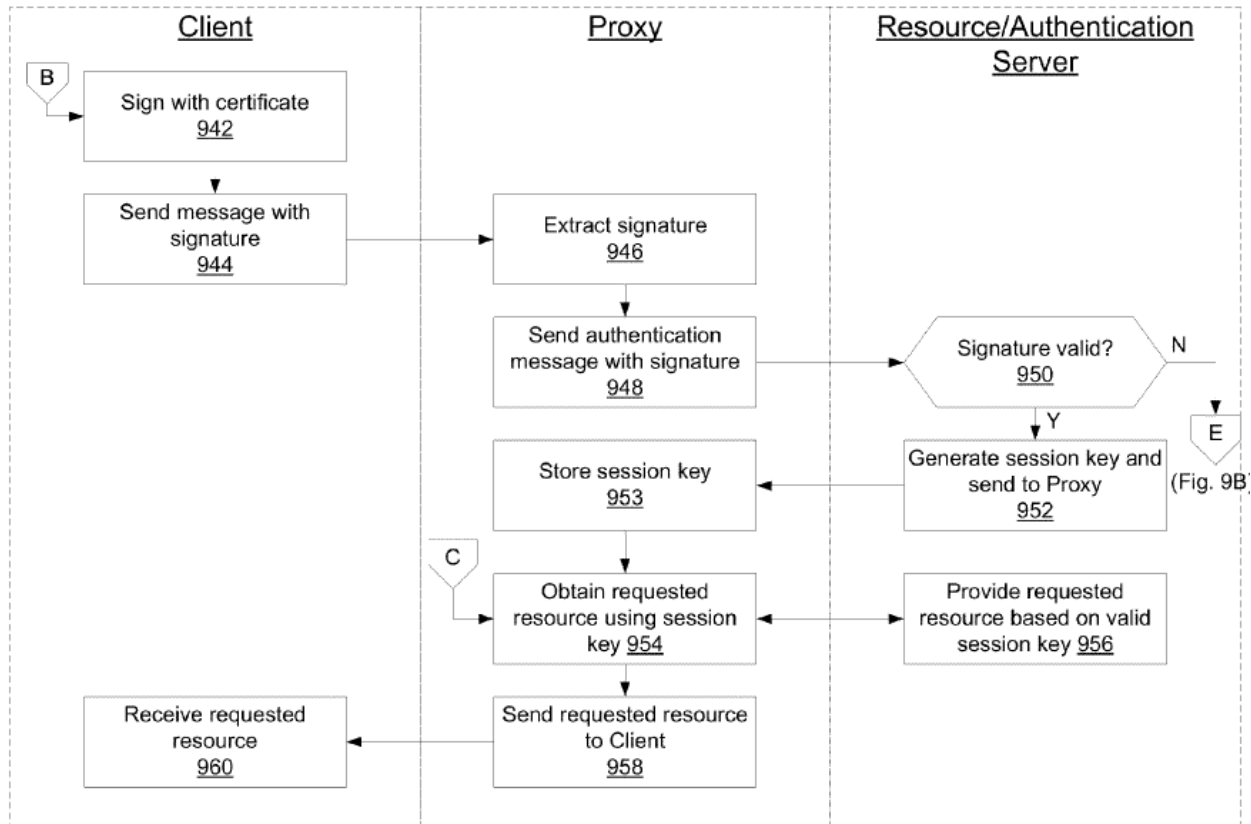


FIG. 9C

EX1012, FIG. 9C (Annotated).

C. Austin

113. Austin recognizes the “on-going need to protect computer devices from malicious content.” EX1005, ¶5. For example, Austin recognizes “untrusted content may be copied by the user from an untrusted source (e.g. untrusted content, untrusted application) . . .” EX1005, ¶13. Austin further recognizes risks posed by “untrusted tasks” such as “certain forms of Web browsing, viewing email attachments, starting an untrusted application program, or accessing a particular file” EX1005, ¶72.

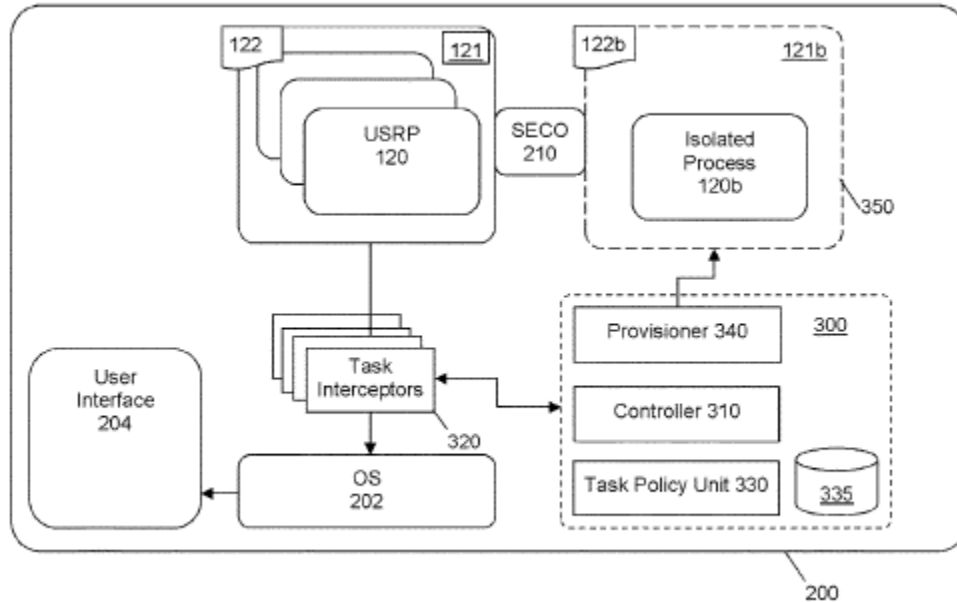
114. To address these challenges, Austin leverages isolation because it inhibits untrusted tasks “from interfering with other tasks or components of the computer device.” EX1005, ¶72. Isolation may involve “limiting the resources of the computer device which are accessible by the untrusted content.” EX1005, ¶15. As a result, the untrusted task and content may be manipulated. EX1005, ¶73.

115. Austin discloses a computer device with an operating system where “each process 120 that is initiated for a particular user will be run in a security context 121 that derives access rights and permissions from the user’s security account.” EX1005, ¶66. “For example, a particular process 120 is able to read from, but not write to, a file of the first resource 115.” EX1005, ¶68. A “security module (SECO) 210 in the operating system” is leveraged to control process’s access to resources. EX1005, ¶68.

116. Austin handles untrusted tasks by opening them in a “task isolation environment or sandbox 350.” EX1005, ¶86. “Where the controller 310 determines that the task in question relates to untrusted content, i.e. is determined to be an untrusted task, then the controller 310 causes the untrusted task to be opened in a task isolation environment or sandbox 350.” EX1005, ¶86. Austin discloses “task interceptor units 320” that intercept, and pass task data to controller 310. EX1005, ¶75. The controller then blocks the task, allows the task to continue executing, or isolates the task within an isolation environment. EX1005, ¶77.

117. For example, a browser task interceptor unit registers a plugin within a web browser application, collects each web page requested by the web browser, and forwards it to the controller. EX1005, ¶80. Similarly, an email task interceptor operates as a plugin to an email client (e.g., Outlook) and is notified when the email client attempts to access a file on the computer storage. EX1005, ¶81. As a result, the email task interceptor sends information including the file and the email requesting access to the controller. EX1005, ¶81. Austin also discloses an application interceptor that notifies the controller each time a new application is started. EX1005, ¶82. Austin further discloses a file unit interceptor that notifies the controller when a file is opened. EX1005, ¶83.

118. When the controller isolates a task within a sandbox, a temporary user account is created for the sandbox, such that the isolated task uses the temporary user account for data and registry access. EX1005, ¶91. As a result, “[a]ny malicious activity only affects the temporary user account 121b and does not affect the primary user account 121.” EX1005, ¶91. For example, the temporary user account is unable to access data of the primary user account, denied administrative access, and isolated from certain OS resources. EX1005, ¶91.

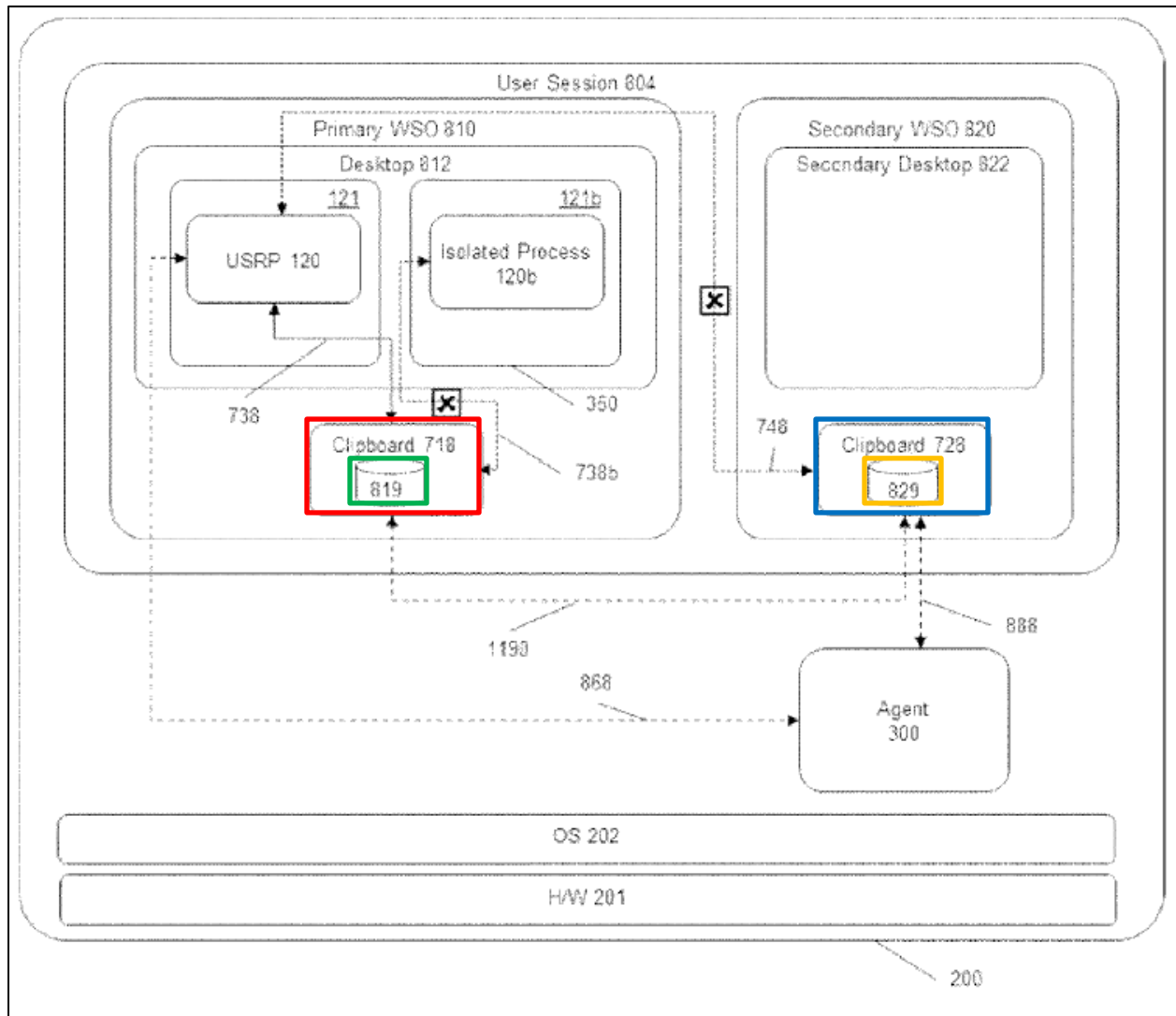


EX1005, FIG. 4 (Annotated)

119. In addition to isolating untrusted processes, Austin discloses another mechanism to protect the host from malware that may attempt to access the host system’s clipboard. EX1005, ¶13. Austin recognizes that “most conventional operating systems” would allow the isolated untrusted process “to access the primary clipboard,” introducing an additional vulnerability. EX1005, ¶102.

120. Austin addresses this issue via “a primary clipboard [red below] accessible from a primary user account” and a “secondary clipboard [blue below] . . . provisioned and associated with the secondary user account” within the isolated sandbox. EX1005, ¶20. Each clipboard is supported by its own respective memory source. The primary clipboard 718 is supported by a primary clipboard memory 819 (green below) (i.e., “physical memory within the hardware 201”). EX1005, ¶119. “[T]he secondary clipboard 728 is supported by appropriate physical

memory 829 [orange below].” EX1005, ¶119.



EX1005, FIG. 8 (Annotated).

121. Austin further discloses the ability to selectively transfer data between isolated and non-isolated applications via the clipboard mechanism. EX1005, ¶134. For example, data may be transferred from non-isolated applications to isolated applications, but not from isolated applications to non-isolated applications. EX1005, ¶134. To provide additional protection, content may be modified before

being shared between isolated and non-isolated applications via the clipboard mechanisms. For example, executable content and attachments may be removed, and parts of content (e.g., hyperlinks) may be replaced. EX1005, ¶129.

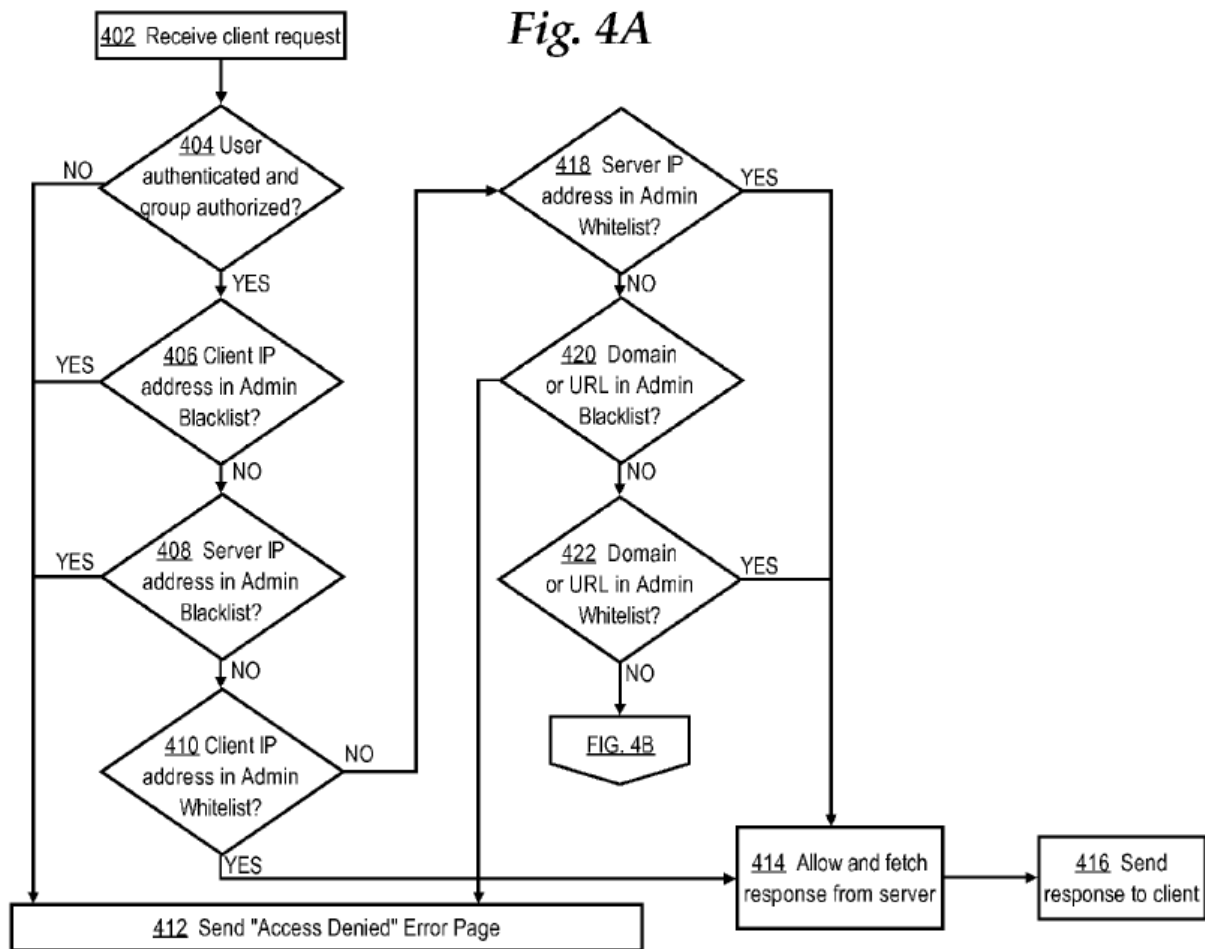
D. Bloch

122. Bloch is directed towards “preventing spyware and other threats from harming computer networks.” EX1007, 1:15-17. Bloch recognizes that “the speed, variety, and maliciousness of spyware and other web-based malware attacks highlight the importance of protecting enterprise networks at the perimeter from such threats.” EX1007, 1:31-34. To mitigate these threats, Bloch discloses a proxy appliance that manages network traffic by blocking and redirecting traffic. EX1007, 3:35-40.

123. Specifically, Bloch’s proxy appliance uses a combination of web reputation filters, malware signatures, and port scanning to protect networks and computers therein. EX1007, 3:39-58. For example, the proxy appliance scans for and blocks traffic attempting to bypass port 80. EX1007, 3:49-52. Similarly, traffic and network parameters are used to evaluate the trustworthiness of a URL. Based on the parameters, a reputation score is generated for the traffic, and a policy (e.g., allow, block) is applied based on the score. EX1007, 3:55-62.

124. Figures 4A and 4B below depict detection techniques of Bloch’s proxy appliance. EX1007, 5:20-21. As depicted in Figure 4A, a client request for

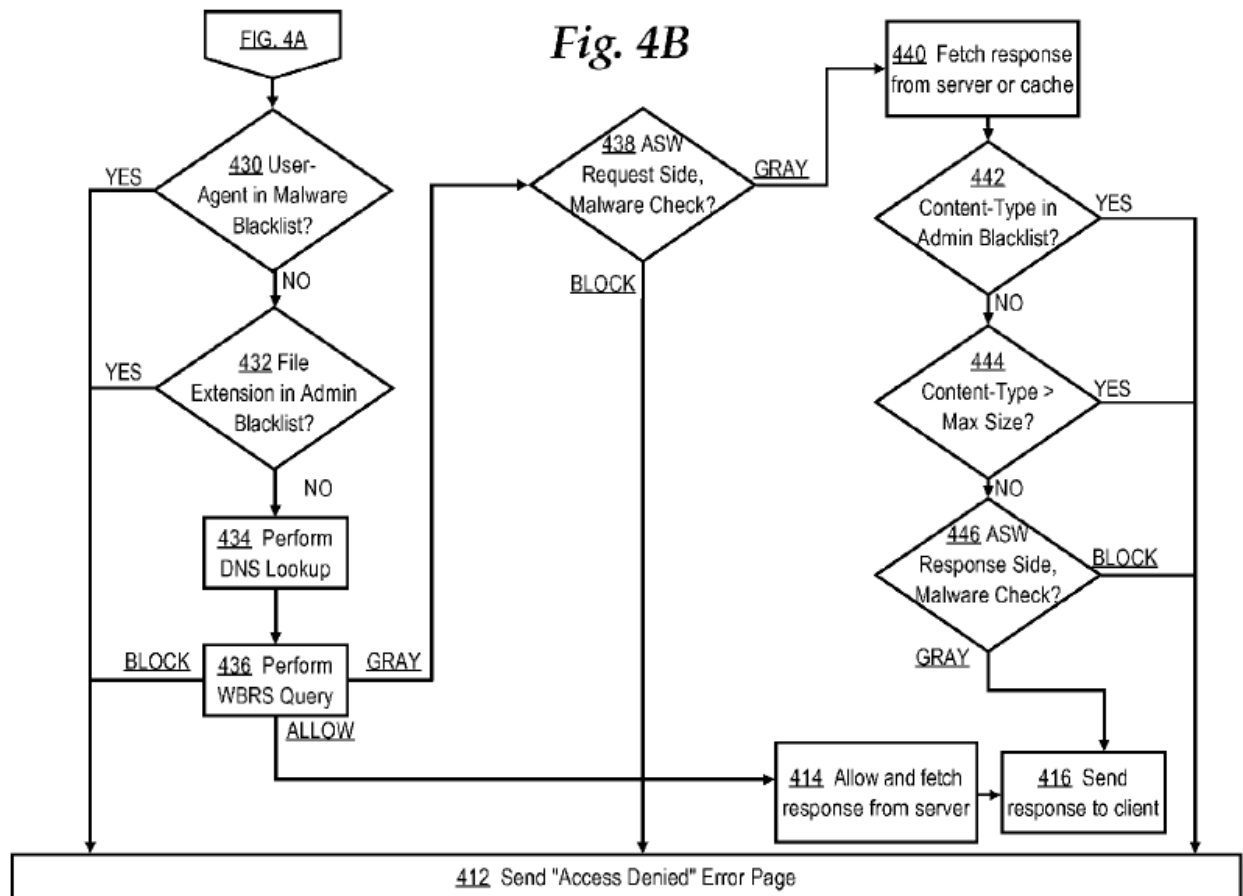
information is received, such as from an HTTP server, FTP server, or other network resource. EX1007, 21-25. Subsequently, the proxy appliance performs various checks. For example, the proxy appliance determines whether the user associated with the requesting client device is authenticated. EX1007, 5:25-2. The proxy appliance also compares the client IP address, server IP address, and requested URL to both whitelists and blacklists. EX1007, 5:35-54. Based on the outcomes of the comparisons, the content is retrieved and sent to the client device. EX1007, 5:54-59.



EX1007, FIG. 4A.

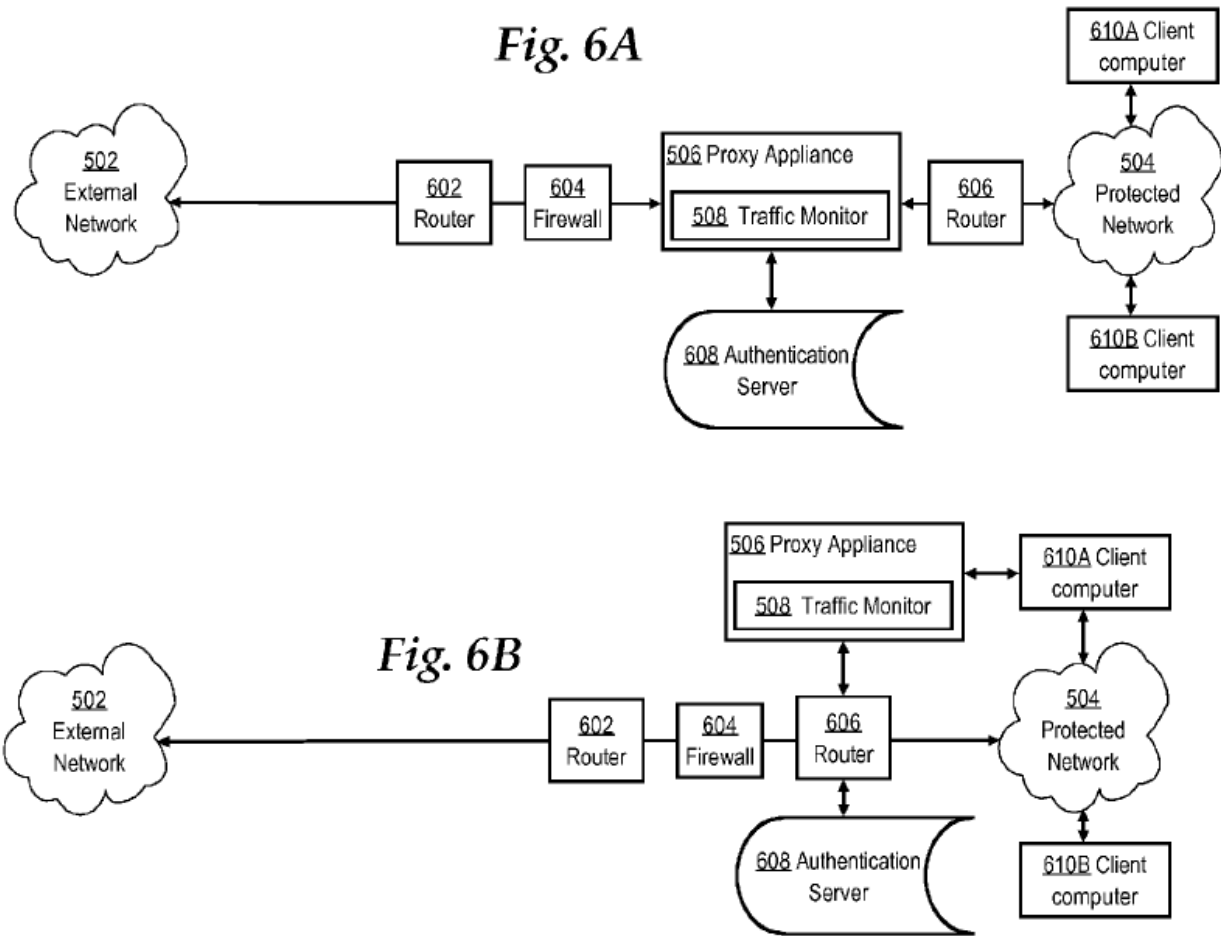
125. As shown in Figure 4B below, various checks are performed on the content requested by the client. For example, the user-agent and file extension of the content are checked against malware blacklists. EX1007, 5:60-64.

Additionally, a web reputation score is generated for the domain or URL of the request. EX1007, 6:8-16. Furthermore, the content type is compared to a blacklist and the content is scanned for spyware. EX1007, 6:38-45. Based on the results, the content is sent to the client. EX1007, 6:48-50.



EX1007, FIG. 4B.

126. Figures 6A and 6B below depict example layouts of Bloch's proxy appliance. EX1007, 8:16-17. As shown in Figure 6A, "proxy appliance 506 is coupled to protected network 504 through router 606." EX1007, 8:24-26. Whereas in Figure 6B, the client computers are connected directly to the proxy appliance. EX1007, 8:35-37. In both embodiments, "authentication server 608" provides "integrated user authentication service[s]" such as lightweight directory access protocol (LDAP) and Microsoft Active Directory. EX1007, 8:21-24. The "traffic monitor 508" inspects all traffic between the internet and the protected network. EX1007, 8:40-42. Based on the IP address and associated policies, the traffic monitor or "firewall 604" log or block the traffic. EX1007, 9:18-27.



EX1007, FIG. 6A, 6B.

E. Walker

127. Walker discloses systems and methods for managing host-based firewalls. EX1008, Abstract. As noted in Walker, firewalls regulate incoming and outgoing network traffic based on rules. EX1008, ¶2. The rules determine whether to allow a packet to pass through the firewall or not. EX1008, ¶3. Thus, the firewall and the associated rules protect computers and computer networks “from malicious attacks originating from a communication network.” EX1008, ¶3.

128. Depicted in Figure 1B below is an example computing device running the Windows operating system. EX1008, ¶70. In Windows, the firewall is implemented using Windows Filtering Platform (WFP). EX1008, ¶70. WFP is a kernel application programming interface (API) for controlling firewall parameters. EX1008, ¶70. Figure 1C depicts a Linux computing device where the firewall is implemented using Netfilter. EX1008, ¶70. Similar to WFP, Netfilter is a kernel API to control firewall parameters. EX1008, ¶70.

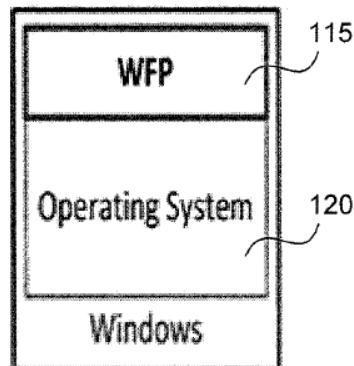


Fig. 1b

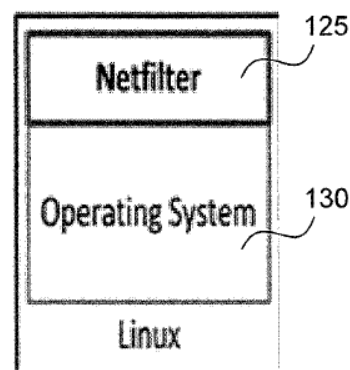


Fig. 1c

EX1008, FIGs. 1B, 1C.

129. Figure 2A below depicts a host computing device and “central management suite 260.” EX1008, ¶71. As I discussed above, the host computing device includes “operating system 210” and “firewall 205” that may be Windows and WFP, or Linux and Netfilter, respectively. EX1008, ¶71. “Driver 215” exists within the kernel space of the operating system and communicates with the

firewall. EX1008, ¶71. “Host policy module 220” and “host logging module 225” exist in user space and communicate with “driver 215.” EX1008, ¶72. The policy module downloads firewall policies from the central management suite and forwards them to the driver to configure the firewall. EX1008, ¶72. Using the policies, the firewall either allows or denies a network packet. EX1008, ¶79. The firewall also uses the policies to determine whether to log the network packet. EX1008, ¶79.

130. Firewall rules are based on information extracted from a packet’s “[n]etwork layer (layer 3) header, Transport layer (layer 4) header, Session layer (layer 5) header, Presentation layer (layer 6) header and/or Application layer (layer 7) header.” EX1008, ¶142. For example, rules can be based on source IP address, destination IP address, and the port used (e.g., port 80). EX1008, ¶143. Rules can also be based on application layer information, such as the URL included within a packet. EX1008, ¶144. This provides the firewall with greater scope since the IP address of a URL may be updated. EX1008, ¶144. Furthermore, rules can be based on transportation layer information. EX1008, ¶145. For example, since HTTP traffic is typically sent as TCP via port 80, traffic that deviates from this criteria can be blocked. EX1008, ¶146. Similarly, the firewall can allow or block FTP traffic or SMTP traffic. EX1008, ¶147.

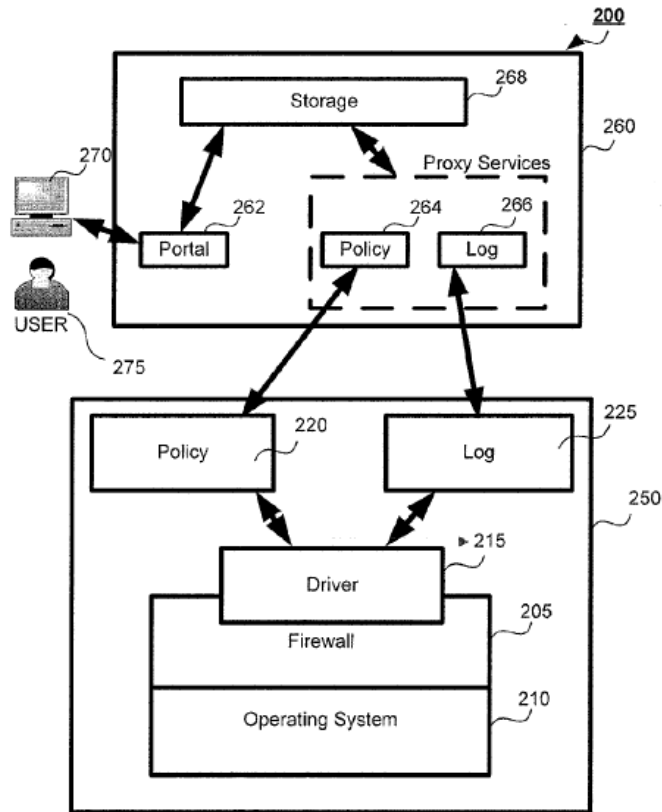


Fig. 2a

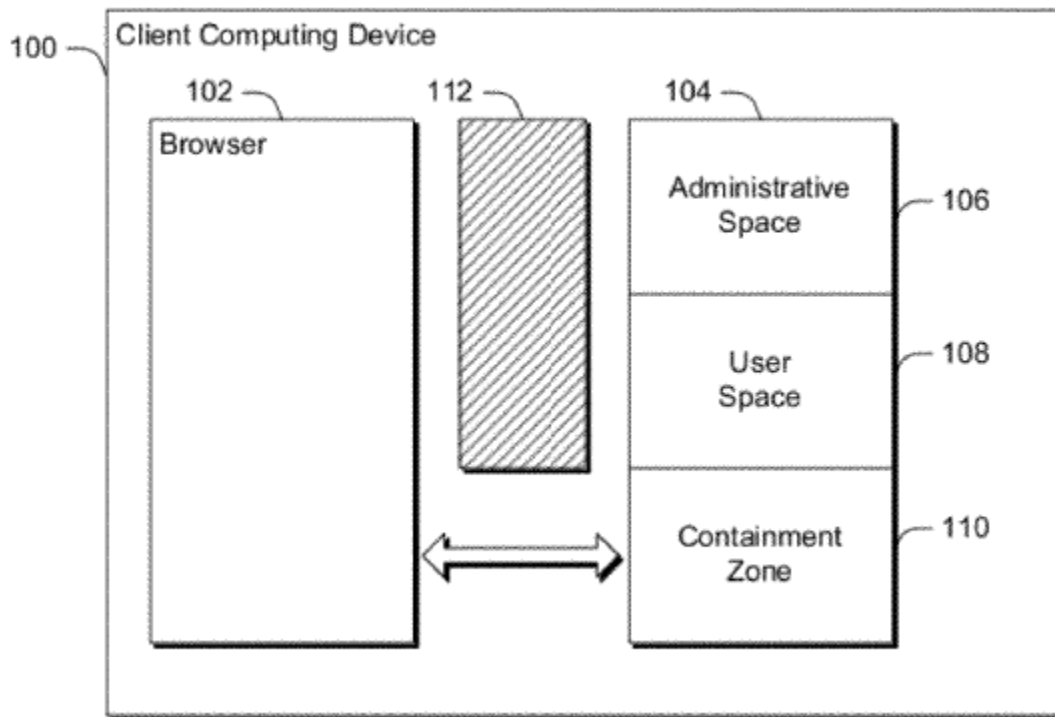
EX1008, FIG. 2A.

F. Franco

131. Franco recognizes that applications that interact with, and download content from the internet can inadvertently install malware or spyware onto the host computer. EX1009, 1:22-27. Not only do these programs degrade the host system's performance, they can also tamper with the host by changing application or system security settings. EX1009, 1:30-34. As a result, Franco discloses restricting applications to prohibit them from accessing portions of the computing device. EX1009, 1:42-46.

132. As depicted in Figure 1 below, the client computing device includes “administrative space 106” including “operating system folders, computer system folders, permanent file folders and the like.” EX1009, 3:27-32. The computing device also includes “user space 108” for user data, such as “My Documents, My Music, Desktop and the like.” EX1009, 3:36-40. To protect the administrative and user spaces, “blocking mechanism 112 is logically interposed between browser 102 and certain predefined spaces, such as the administrative 106 and user spaces 108, to prevent the browser from directly accessing such spaces.” EX1009, 4:7-11.

133. Instead of writing to the administrative or user spaces, the blocking mechanism allows the browser to write to “containment zone 110” that is considered untrusted. EX1009, 3:42-48, 2:43-45. For example, application settings and data (e.g., cache data) are written to the containment zone. EX1009, 3:49-56. As a result, settings and data for the restricted application can be saved without interfering with other applications on the device. EX1009, 3:49-52.

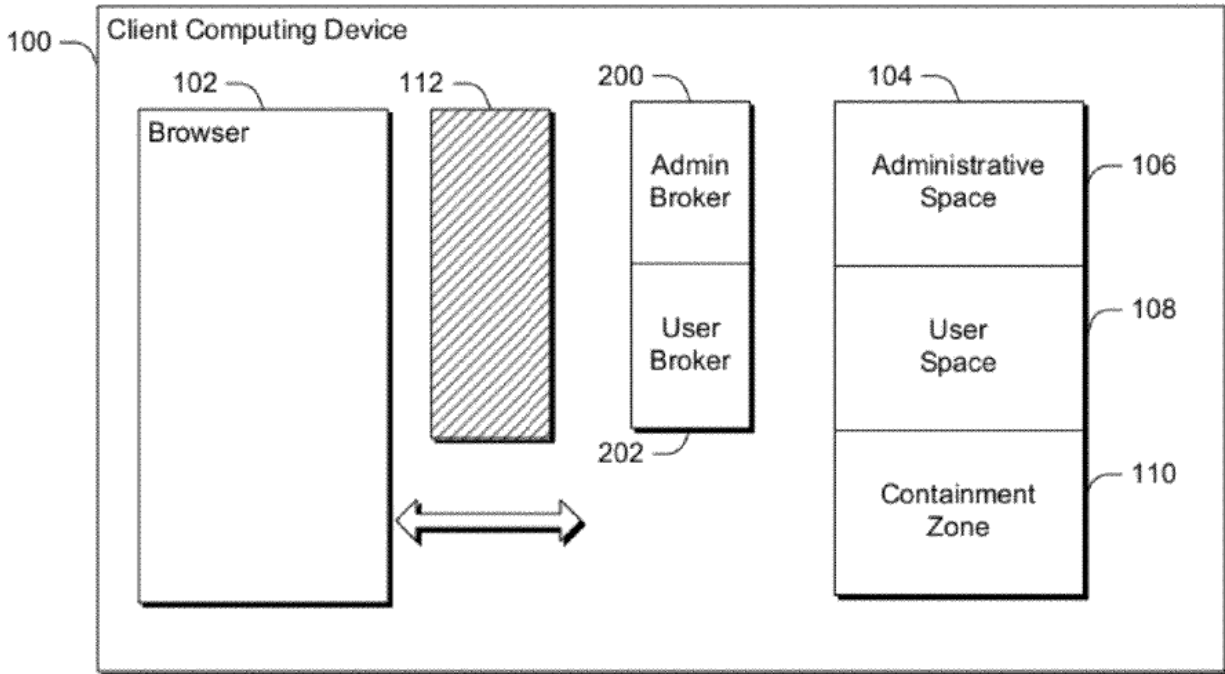


EX1009, FIG. 1 (Excerpted).

134. As depicted in Figure 2 below, the broker mechanism is configured as two broker objects, one for the administrative space and one for the user space.

EX1009, 4:26-28. When the browser attempts to access a restricted area, such as the administrative space, the administrative broker object inspects the request to allow or deny the request. EX1009, 4:41-45. For example, if the request includes downloading files, the broker object inspects electronic signatures of the files.

EX1009, 4:45-48.



EX1009, FIG. 2 (Excerpted).

135. The broker objects also brokers access to the restricted spaces. For example, a process such as the browser may attempt to save a file in the user space of the computing device. EX1009, 5:4-6. To facilitate the request, the broker object displays a dialog box for the user to indicate whether to allow the request. EX1009, 5:4-8. Similarly, if the process attempts to access the administrative space, the broker object prompts the user for administrator credentials to facilitate the request. EX1009, 5:8-11.

IX. GROUNDS OF UNPATENTABILITY

Ground	References	Claims
1	Walsh and Innes	1-4, 7, 9-12, 15

2	Walsh, Innes, and Netfilter	5, 6, 13, 14
3	Walsh, Innes, and Franco	8, 16
4	Austin and Bloch	1-4, 7, 9-12, 15
5	Austin, Bloch, and Walker	5, 6, 13, 14
6	Austin, Bloch, and Franco	8, 16

X. GROUND 1: Claims 1-4, 7, 9-12, and 15 are obvious over Walsh, Applicant Admitted Prior Art (AAPA), and Innes.

A. A POSA would have been motivated to combine Walsh with Innes based on Walsh’s disclosure.

136. Walsh is directed to “computer security” in “distributed computing environments.” EX1006, 1:8-13. In these environments, Walsh acknowledges that “malicious entities can plant viruses, Trojans, or other malicious agents in publicly available content in order to attack the users’ computing systems and steal sensitive information from the users.” EX1006, 1:19-23. Walsh recognizes that the untrusted content (e.g., programs, files, and data) can be copied onto the user computing system through physical media and/or downloaded from a remote computing system. EX1006, 3:67-4:12.

137. Although antivirus software prevents attacks by dangerous content, the antivirus software needs to know the identity of the dangerous content before it

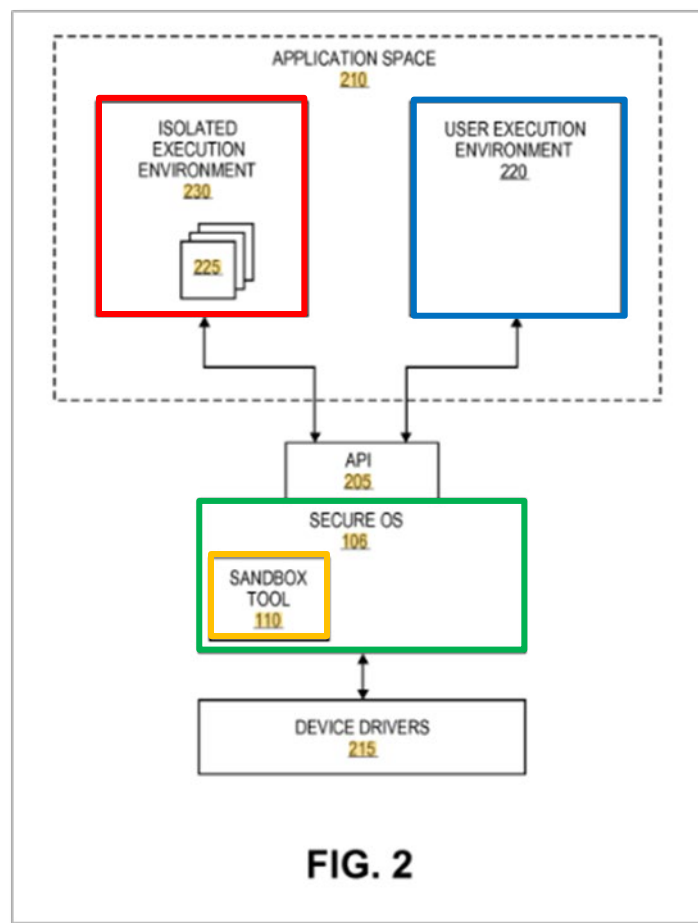
can be blocked. EX1006, 1:25-34. Furthermore, modern operating systems typically employ access controls that shut down programs exceeding predefined access levels. EX1006, 1:42-47. However, this also hinders the user's ability to use the program. EX1006, 1:47-49. Thus, Walsh recognizes the need to run untrusted programs while also protecting the user's computer. EX1006, 1:51-53.

138. To address these issues, Walsh discloses techniques for accessing untrusted content (such as, “programs” and “applications”) “without exposing a computing system to [the] untrusted content and possibly malicious content.” EX1006, 2:32-36. Specifically, Walsh provides “an isolated execution environment,” which uses “a secure operating system (OS), such as security enhanced LINUX (SELinux),” to access the untrusted content. EX1006, 2:36-44.

139. Exemplary software for providing such content isolation is illustrated in Walsh's Figure 2 (annotated below). “The content,” Walsh explains, can be accessed over the Internet via a “web browser” application. EX1006, 3:67-4:4. Trusted content is executed in a “user execution environment 220” (in blue). EX1006, 5:30-32. Untrusted content, in contrast, is executed in an “isolated execution environment 230” (in red) “without exposing other processes such as the user execution environment 220” to the untrusted content. EX1006, 5:50-57. The “user execution environment 220” and the “isolated execution environment 230” both use a “secure OS” (in green), but each are provided with their own memory

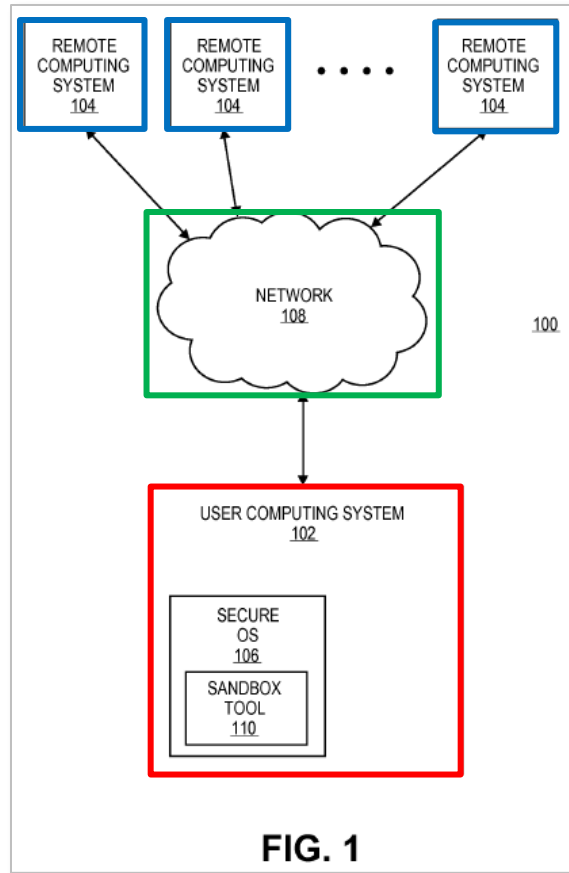
“namespace” so that “the processes of isolated execution environment 230 cannot access files and processes of any other namespace” EX1006, 6:23-33.

140. For example, the secure OS cooperates with a “sandbox tool” (in orange) to copy programs, files, and data necessary to create the isolated execution environment, and have the same functionality as the user execution environment. EX1006, 6:4-17. The secure OS and sandbox tool also copy content (e.g., an untrusted program, untrusted data) to use within the isolated execution environment. EX1006, 10:11-23.



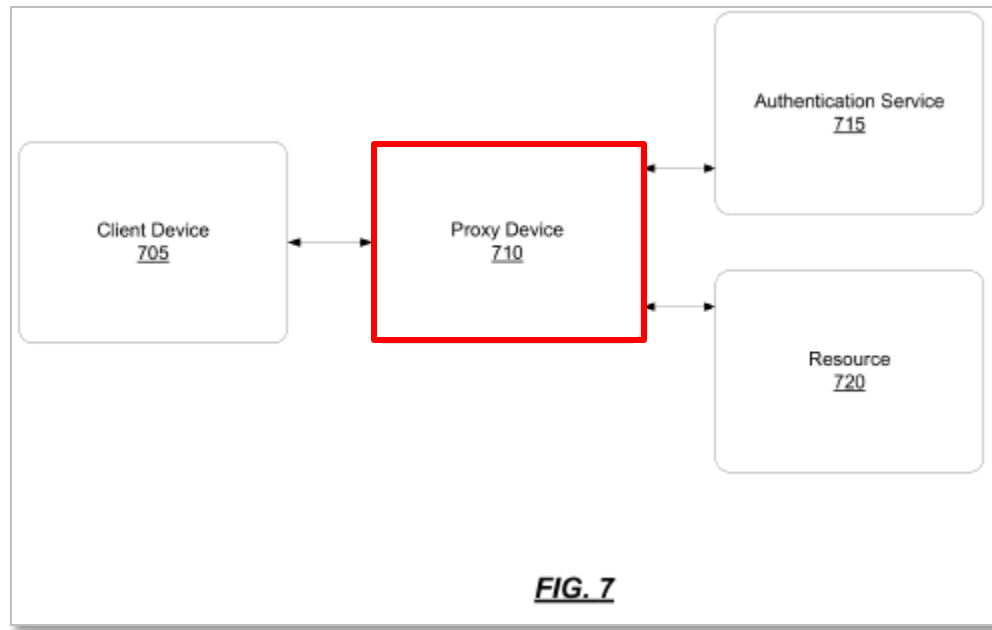
EX1006, FIG. 2 (Annotated).

141. A POSA would have been motivated to combine Walsh with Innes based on the teachings in Walsh itself. Referring to annotated Figure 1 (below), Walsh discloses that a “user computing system 102” (in red) is connected to other “remote computing systems 104” (in blue) via “network 108” (in green). EX1006, 3:24-63. Walsh says that the “user computing system 102” can access content via “network 108” using “a client-server model.” EX1006, 4:66-5:3. For example, Walsh discloses that “user computing system 102” can download content from “one or more of the remote computing systems 104 via the network 108.” EX1006, 4:8-13. Noted above, the content may be insecure. EX1006, 4:17-19. To the extent that Walsh does not provide details about the client-server model, Innes does.



EX1006, FIG. 1 (Annotated).

142. Like Walsh, Innes discloses that remote computers can access content via a network “using a web browser.” EX1012, 4:27-47, FIG. 1. The remote computers “may wish to access enterprise resources that require authentication” EX1012, 23:43-45. To provide such authentication, Inness discloses a “proxy device 710” (in red below) that can provide “authentication service 715” to a “client device 705”. EX1012, 22:2-4.



EX1012, FIG. 7 (Annotated).

143. And Innes discloses “example method steps for authentication” and for “providing secured access to resources using a proxy,” such as “proxy device 710.” EX1012, 27:43-44, FIGs. 9A-C. For example, when the client device requests access to a resource, the request is sent to an authentication server to determine whether to grant the request. EX1012, 28:35-48. Specifically, the client device signs a message and provides it to the proxy device. EX1012, 34:63-35:12. The proxy device sends the signed message to the authentication server that determines whether the client device provided a valid signature. EX1012, 35:21-44. When the authentication server validates the client signature, the resource is sent to the client via the proxy device. EX1012, 37:44-46. A POSA would have understood that in the Walsh-Innes combination, the authentication would be

performed independently by the user execution environment and the isolated execution environment because Walsh discloses using role-based access control (RBAC) to derive permissions for each environment. EX1006, 3:49-53.

144. In Walsh, the user execution environment and isolated execution environment are each assigned a security context. EX1006, 8:14-16. The security context includes a role. EX1006, 6:52-54. A POSA would have understood that the user execution environment and isolated execution environments would have different roles, and that the roles would be accessed through account credentials. *See* EX1031, 19 (assigning roles to each user account). In the Walsh-Innes combination, the credentials would be used to authenticate with the authentication server.

145. A POSA would have been further motivated to implement Walsh's user computing system on a network including Innes's proxy device and authentication services to improve computer and network security.

146. Walsh not only recognizes that content may be insecure because it could include a virus; Walsh also recognizes that the source of the content could be untrustworthy. For example, Walsh states that computing devices "may not know the source of the content, and cannot guarantee the content is secure." EX1006, 17-19. To provide protection, Walsh's computing devices implement content isolation by using a "user execution environment 220" (for trusted content) and an "isolated

execution environment 230” (for untrusted content). EX1006, FIG. 2.

147. To provide additional protection for Walsh’s computing device, a POSA would be motivated to place the computing device on a network with Innes’s proxy device. Noted above, Walsh’s computing devices access the trusted/untrusted content over a network using a web browser. EX1006, 3:64-4:4. The content, Innes teaches, can be “securely” accessed “using a proxy device.” EX1012, 1:6-8. For example, Innes’s proxy device is configured to establish an authenticated session with the requested resource, and provide session information to the computing device. EX1012, 22:15-17. The computing device uses the session information to verify the authentication session between the proxy device and resource. EX1012, 22:37-40. Combining Innes’s authentication techniques with Walsh’s computing device would have provided an additional layer of security for accessing the trusted/untrusted content over the network.

148. A POSA also would have had a reasonable expectation of success combining Walsh with Innes because the combination would amount to nothing more than placing Walsh’s computing device on a network with Innes’s proxy device. Walsh’s computing device is a standard computing system running, for example, SELinux. EX1006, 3:44-46. Walsh’s computing device communicates and shares data via a network that can be wired or wireless, to access both the internet and local area networks. EX1006, 3:57-63. Innes’s proxy device

communicates with a client device (e.g., Walsh’s computing device) over a network to provide resource access. EX1012, 22:13-46.

B. A POSA would have been motivated to combine Walsh-Innes with AAPA.

149. The ’601 patent admits that the claimed “multi-user interactive software application” can be implemented via commercially-available products, such as WebEx. EX1001, 4:45-49.

150. A POSA would have been motivated, with a reasonable expectation of success, to use Innes’s proxy appliance with Walsh’s client computer in order to “securely” access WebEx traffic via Walsh’s computer system. EX1012, 1:6-8. Walsh explains that a user computer system may access content over the Internet via a “web browser” application. EX1006, 3:67-4:4. A POSA would have understood that WebEx and GoToMeeting were commercially-available applications that would be (i) installed as standalone programs on Walsh’s computer system or (ii) used within a web browser or email client on Walsh’s computer system. EX1018, 1-4; EX1024, 43-46, 131, 204-05; EX1033, 2. The combined Walsh-AAPA system would then be used on a network including Innes’s proxy device to provide Walsh’s client device with secure access to network resources (e.g., an HTTP server). EX1006, 28:34-48. A POSA would not have experienced any unreasonable technical hurdles or conflicts using Walsh’s computing device to access WebEx or GoToMeeting on a network including

Innes's proxy device.

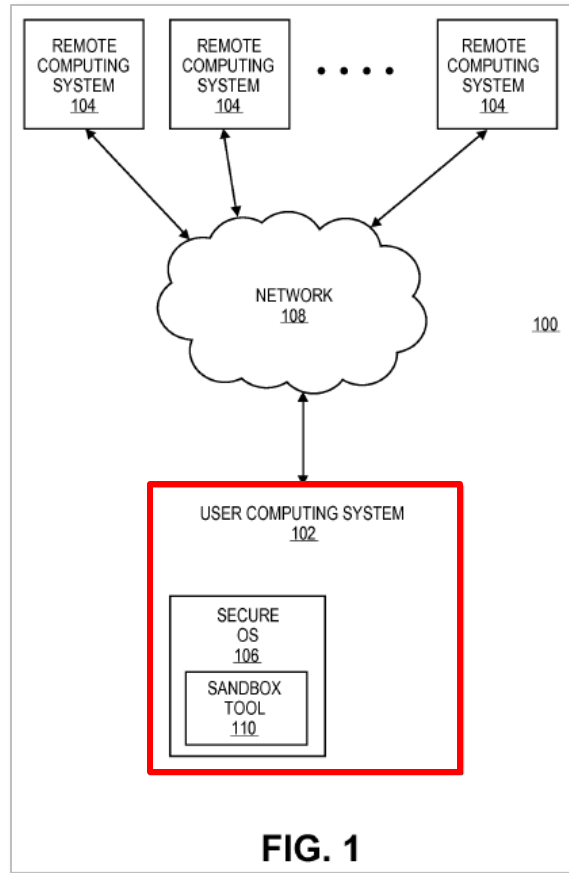
C. Independent claims 1 and 9

1. [1-pre] “A system” and [9-pre] “A method”

151. To the extent the preambles are limiting, Walsh discloses them.

Referring to Figure 1 (annotated below), Walsh discloses “a user computing system 102” (a “*system*”) included within a network 100. EX1006, 3:36-37.

Walsh's “user computing system 102” implements a “*method*” because it creates an isolated execution environment and accesses “content (trusted or untrusted) without exposing other processes and resources of the user computing system 102 to the content.” EX1006, 4:30-35.

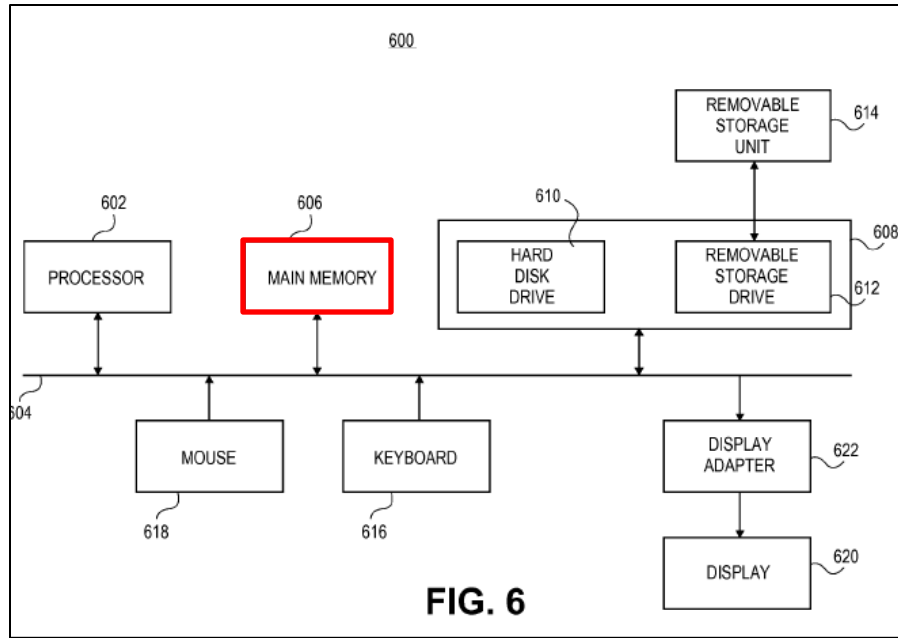


EX1006, FIG. 1 (Annotated).

2. [1A] “a memory”

152. Walsh discloses this element.

153. Walsh’s “user computing system 102” includes “hardware resources,” such as “processors” and “memory.” EX1006, 3:38-40. For example, “computing system 600” shown below can implement “user computing device 102” and includes “main memory 606.” EX1006, 12:44-56.

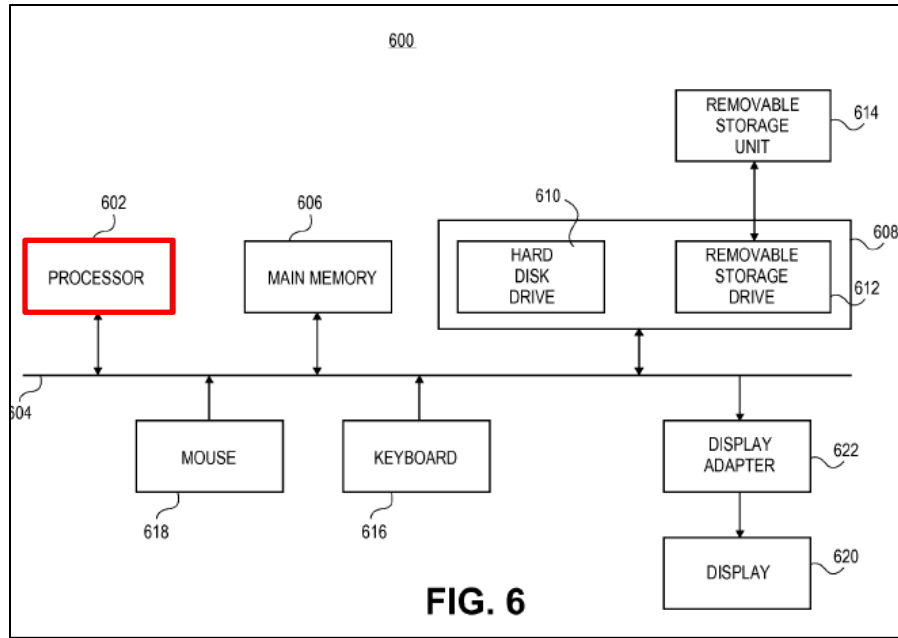


EX1006, FIG. 6 (Annotated).

3. [1B] “a processor configured to”

154. Walsh discloses this element.

155. Walsh’s “user computing system 102” includes “hardware resources,” such as “processors” and “memory.” EX1006, 3:38-40. For example, “computing system 600” shown below can implement “user computing device 102” and includes “processor 602.” EX1006, 12:44-51.



EX1006, FIG. 6 (Annotated).

4. [1C] and [9A] “[implement/implementing] a workspace that uses a host operating system, wherein the workspace enables operation of a first set of one or more applications or processes via a first memory space”

156. Walsh discloses this element.

157. Walsh’s “user execution environment 220” (“*a workspace*”) is created by a “secure OS 106” (“*a host operating system*”) of the “user computing system 102”: “secure OS 106 can be configured to create a user execution environment 220 in the application space 210.” EX1006, 5:28-29; see also FIG. 1. “The user execution environment 220” (“*workspace*”) “enables operation of a first set of one or more applications or processes” because Walsh discloses that the “user execution environment 220 allows users to interact with the user computing system 102 to access content such as run application and programs, view files, etc.”

EX1006, 5:30-32. For example, “user computing device 102” executes word processing applications, web browser applications, file sharing applications, electronic mail (e-mail) applications, multimedia applications, chat applications, etc.” EX1006, 3:64-4:4. The “user execution environment 220” (“*workspace*”) is supported by its own memory and storage, such as “user namespace such as home directory and temporary directory” (“*a first memory space*”). EX1006, 5:32:40. The “sandbox tool 110” is configured to “allocate resources (storage space, memory, etc.) of the user computing system 102, which are necessary to create the isolated execution environment.” *Id.*, 4:35-38.

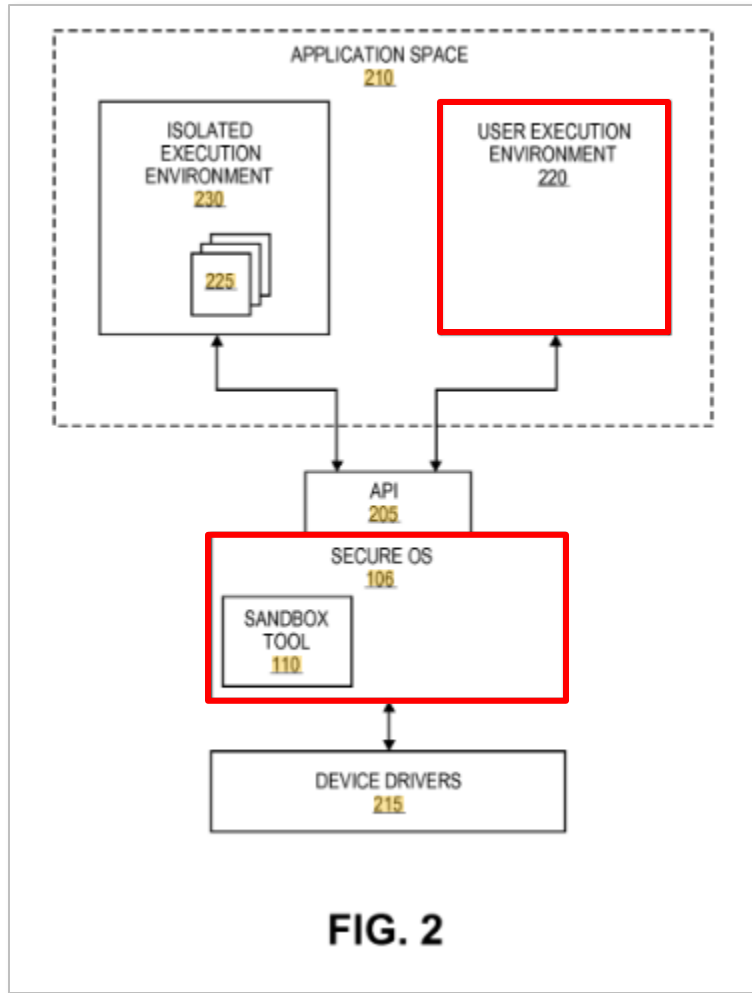


FIG. 2

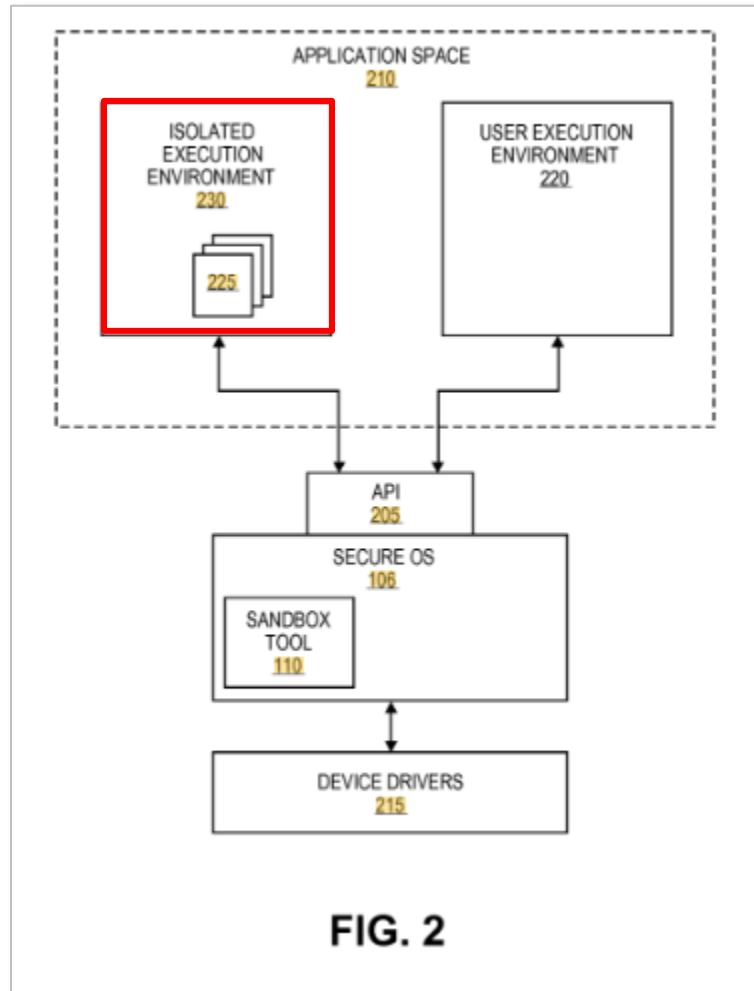
EX1006, FIG. 2 (Annotated).

5. [1D] and [9B] “[implement/implementing] an isolated computing environment that uses the host operating system”

158. Walsh discloses this element.

159. Walsh discloses an “isolated execution environment 230” (“*isolated computing environment*”) that is created by the “secure OS 106” (“*host operating system*”): “sandbox tool 110 can be configured to cooperate with components of a secure OS 106, to create an isolated execution environment 230” EX1006,

5:53-55. For example, the isolated execution environment uses the secure OS to gain access to hardware resources “via the device drivers 215.” EX1006, 6:54-59.



EX1006, FIG. 2 (Annotated).

6. [1E] and [9C] “wherein the isolated computing environment enables operation of a second set of one or more applications or processes via a second memory space”

160. Walsh discloses this element.

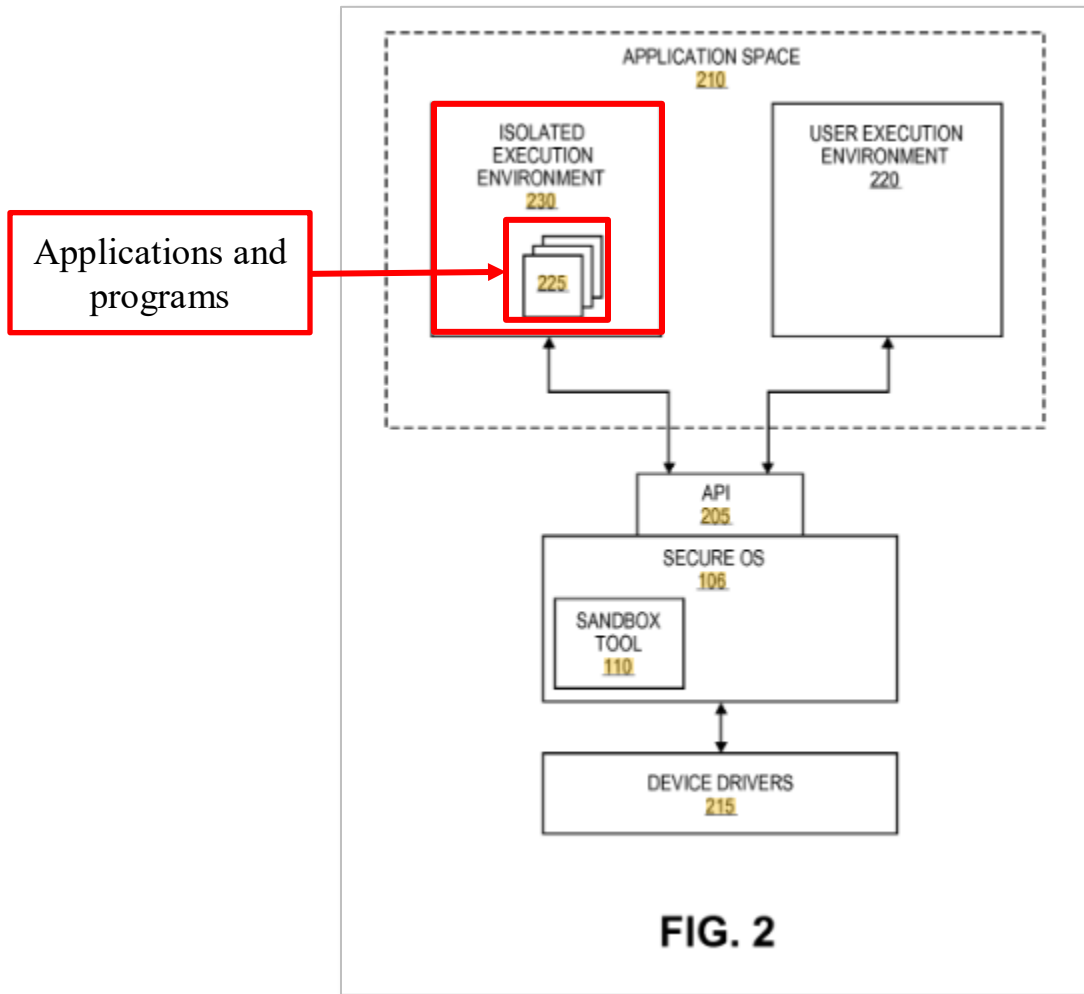
161. Walsh’s “isolated execution environment 230” (“*isolated computing environment*”) executes applications and programs (“*a second set of one or more*

applications or processes”) that require isolation. EX1006, 4:47-50; 5:50-55. For example, the sandbox tool creates a namespace for the isolated execution environment and copies content into the namespace. EX1006, 10:1-12. The content can include programs, applications, files and/or data. EX1006, 10:29-33.

162. Walsh discloses “storage space” and “memory” (“*a second memory space*”) that is allocated to the “isolated execution environment 230” by a “sandbox tool 110.” EX1006, 4:35-42; 5:57-61. For example, Walsh states that “sandbox tool 110 can be configured to allocate resources (storage space, memory, etc) of the user computing system 102 . . . necessary to create the isolated execution environment” EX1006, 4:35-38. For example, the “sandbox tool 110” creates “a namespace for the isolated execution environment 230 (e.g. home directory and tmp directory) which is separate from the namespace of the user execution environment 220.” EX1006, 6:5-9. The “isolated execution environment 230 can only access the resources” (e.g., the “memory”) allocated to it. EX1006, 5:61-65; *See also* EX1006, 4:40-42. As a result, the isolated content can be accessed without exposing the host to malicious or dangerous content. EX1006, 4:43-45.

163. Thus, the “*second memory space*” for the isolated execution environment is different from the “*first memory space*” for the user execution environment discussed above. As a result, the isolated execution environment is

prevented from accessing the user execution environment. EX1006, 10:42-47.



EX1006, FIG. 2 (Annotated).

7. [1F] and [9D] “wherein the second set of one or more applications or processes comprises a multi-user interactive software application”

164. Walsh alone or in combination with AAPA discloses this element.

165. Walsh discloses that the content executing in the isolated execution environment 230 (“*the second set of one or more applications or processes*”) may include “any number and type of data, applications programs such as word

processing applications, web browser applications, file sharing applications, electronic mail (e-mail) applications, multimedia applications, chat applications, etc.” EX1006, 3:67-4:4. The chat application disclosed by Walsh is a “*multi-user interactive software application*” because it allows multiple users to communicate (e.g., interact).

166. The ’601 patent admits that the claimed “*multi-user interactive software application*” existed in commercially available products, such as WebEx and GoToMeeting. EX1001, 4:45-49. And a POSA would have understood that both WebEx and GoToMeeting could be used (i) within a web browser, (ii) as a standalone application, or (iii) as an email client plugin. EX1018, 1-4; EX1024, 43-46, 131, 204-05; EX1033, 2.

167. As explained above, a POSA would have been motivated, with a reasonable expectation of success, to execute WebEx (“*multi-user interactive software application*”) as one of Walsh’s untrusted applications in the isolated execution environment 230 so that WebEx could not “access files and processes of any other namespace.” EX1006, 6:23-33. This combination, moreover, would have amounted to nothing more than combining familiar elements (e.g., Walsh’s content isolation and the commercially-available WebEx application) according to their known methods of use.

168. In the Walsh-AAPA combined system, Walsh’s untrusted application

(“the second set of one or more applications or processes”) would be WebEx (“a multi-user interactive software application”).

8. [1G] and [9E] “[isolate/isolating] the isolated computing environment from the workspace using an internal isolation firewall”

169. Walsh discloses this element.

170. Walsh discloses a “sandbox tool 110” implementing “security policies” (“internal isolation firewall”) to isolate the “isolated execution environment 230” (“isolated computing environment”) from the “user execution environment 220” (“workspace”):

[S]andbox tool 110 can be configured to cooperate with components of a secure OS 106, to create an isolated execution environment 230 for accessing content 225 (trusted or untrusted) **without exposing other processes such as the user execution environment 220 and resources of the user computing system 102 to the content 225.**”

EX1006, 5:52-57 (emphasis added).

171. In particular, the sandbox tool 110 is “configured to apply security policies of the secure OS 106 to the isolated execution environment 230 such that **the content 225 running in the isolated execution environment 230 can only access the resources allocated to the isolated execution environment 230.**”

EX1006, 5:61-65 (emphasis added).

9. [1H] and [9F] “[authenticate/authenticating] the isolated computing environment with an authentication device”

172. The Walsh-AAPA-Innes combination discloses this element.

173. A POSA would have been motivated to combine Walsh-AAPA with Innes in order to securely access content on the network disclosed by Walsh. Specifically, Walsh notes that, in distributed computing environments, local computing systems may be connected to remote computing systems via a network. EX1006, 3:24-54. Walsh says that the local computing systems can access content via the network using “a client-server model.” EX1006, 4:66-5:3. To the extent that Walsh does not provide implementation details about the client-server model, Innes does.

174. Like Walsh, Innes discloses that remote computers can access content via a network “using a web browser.” EX1012, 4:27-47, FIG. 1. The remote computers “may wish to access enterprise resources that require authentication” EX1012, 23:43-45. To provide such authentication, Innes discloses a “proxy device 710.”

175. And Innes discloses “example method steps for authentication” and for “providing secured access to resources using a proxy,” such as “proxy device 710.” EX1012, 27:43-44, FIGs. 9A-C. For example, a client device may request access to a network resource. EX1012, 28:35-40. The resource may require that the client device be authenticated. EX1012, 28:36-40. As a result, the client device

may sign an authentication message using a certificate, and send the signed message to the authentication server via the proxy device. EX1012, 34:64-35:23.

176. A POSA would have been motivated, with a reasonable expectation of success, to implement the Walsh-AAPA user computing system on a network including Innes's proxy device and authentication services in order to securely access WebEx content on Walsh's network. Walsh discloses that computing devices may implement content isolation by using a "user execution environment 220" (for trusted content) and an "isolated execution environment 230" (for untrusted content). EX1006, FIG. 2. The computing devices access the trusted/untrusted content over a network using a web browser. EX1006, 3:64-4:4. WebEx could be used within a web browser. EX1018, 1; EX1033, 2. GoToMeeting could be used within a browser. EX10124, 131, 204-05. The content, Innes teaches, can be "securely" accessed "using a proxy device." EX1012, 1:6-8. And Innes's "proxy device" allows computing devices to request certain authentication services, which "may be provided by an authentication server." EX1012, 27:43-61. Moreover, this combination would have been obvious because it would amount to nothing more than combining familiar elements according to their known methods of use.

177. The Walsh-Innes combination would also have been obvious to try because Walsh describes a client-server model. EX1006, 4:66-5:3. In the client-

server context, machines can access the Internet using known paradigms, for example, using a network-address-translation device or using a proxy device. As a result, it would have been obvious to try and connect Walsh's computing system to Innes's proxy device to access the internet.

178. In the Walsh-AAPA-Innes system, Walsh's "isolated execution environment 230" (*"isolated computing environment"*) would be authenticated via Innes's authentication server (*"authentication device"*). EX1006, 5:53-54.

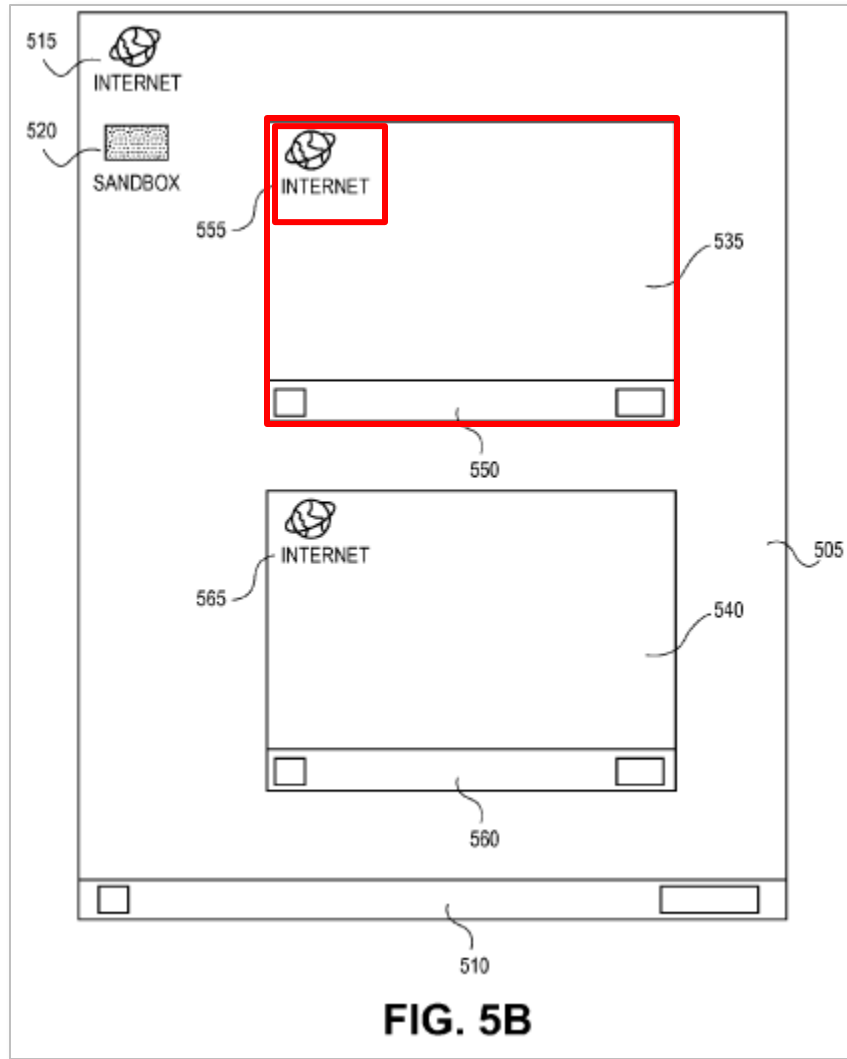
10. [1I] and [9G] "[send/sending] data to an untrusted destination from the multi-user interactive software application via a proxy device when the isolated computing environment has been authenticated."

179. The Walsh-AAPA-Innes combination discloses this element.

180. Walsh recognizes that untrusted tasks may be accessed over the Internet using a web browser. EX1006, 1:12-25. To address this issue, Walsh discloses that a web browser may be isolated when instructed to view (*"send data to"*) an untrusted website (*"an untrusted destination"*):

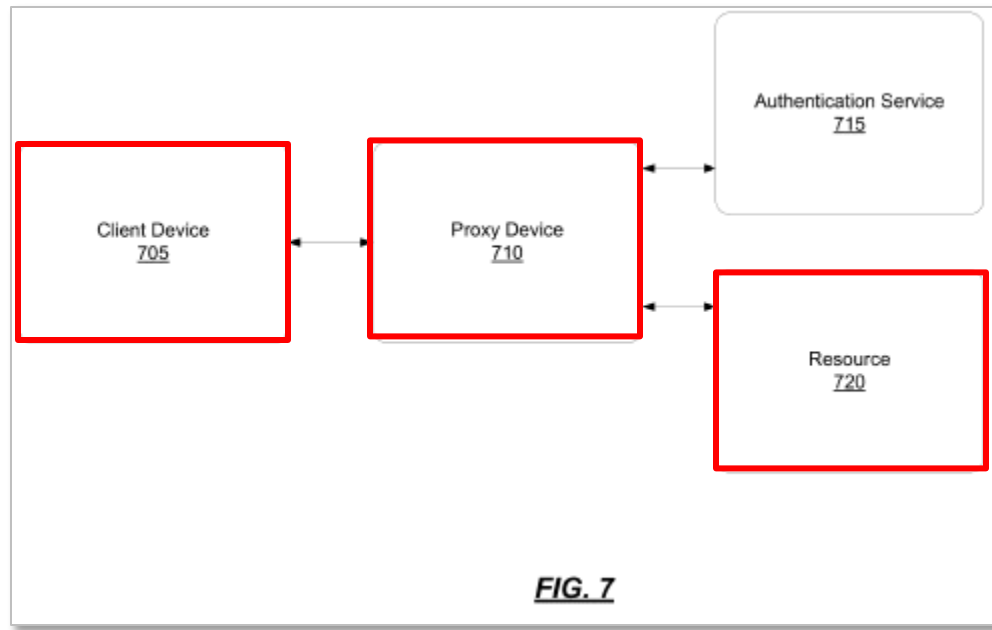
As illustrated in FIG. 5B [annotated below], the sandbox tool 110 can display a user **interface 535 for the first isolated execution environment . . .** in the desktop GUI 505. The user interface 535 can include various menus and widgets for accessing application programs, such as a tool bar 550 and application icon 555 for accessing **a web browser application program.**

EX1006, 12:22-29.



EX1006, FIG. 5B (Annotated).

181. As shown in Figure 7 below, Innes discloses “a proxy device 710” (“*proxy device*”) that provides a “client device 705” access to “resources 720, such as documents, emails, services, files, and the like.” EX1012, 22:13-15.



EX1012. FIG. 7 (Annotated).

182. As I explained above, a POSA would have been motivated, with a reasonable expectation of success, to implement the Walsh-AAPA user computing system on a network including Innes's proxy device and authentication services in order to securely access WebEx or GoToMeeting content on Walsh's network. Walsh discloses that computing devices may implement content isolation by using a "user execution environment 220" (for trusted content) and an "isolated execution environment 230" (for untrusted content). EX1006, FIG. 2. The computing devices access the trusted/untrusted content over a network using a web browser. EX1006, 3:64-4:4. Both WebEx and GoToMeeting could be used within web browsers. EX1018, 1; EX1024, 12, 131, 204; EX1033, 2. The content, Innes teaches, can be "securely" accessed "using a proxy device." EX1012, 1:6-8. And

Innes's "proxy device" allows computing devices to request certain authentication services, which "may be provided by an authentication server." EX1012, 27:43-61. Moreover, this combination would have been obvious because it would amount to nothing more than combining familiar elements according to their known methods of use.

183. In the Walsh-AAPA-Innes system, Walsh's web browser would access WebEx ("*multi-user interactive software*") to communicate ("*send data to*") with an untrusted website (e.g., a WebEx server) ("*an untrusted destination*") when Walsh's "isolated execution environment 230" ("*isolated computing environment*") has been "authenticated" by Innes's authentication server ("*authentication device*"). EX1006, FIG. 6.

D. Claims 2 and 10

184. Claims 2 and 10 depend from claims 1 and 9, respectively, and recite "*wherein the multi-user interactive software application allows a remote machine to control an application or a process of the second set of one or more applications or processes running in the isolated computing environment, wherein controlling the application or process comprises at least one of viewing, interacting with, modifying, uploading data to, or downloading data from the application or the process.*" The Walsh-AAPA-Innes combination discloses this limitation.

185. These claims recite functionality that was available through WebEx—

an application that was commercially available before the earliest priority date of the '601 patent. As I discussed for Element 1G, Walsh's "web browser application" would access WebEx ("*a multi-user interactive software application*"). And a POSA would have understood that Walsh's "web browser application" accessing WebEx enables "*viewing, interacting with, modifying, uploading,*" and "*downloading data.*" EX1006, 11:66-67. During a WebEx meeting, for example, a presenter can share their desktop, an application, or a document. EX1011, 0001-02.

E. Claims 3 and 11

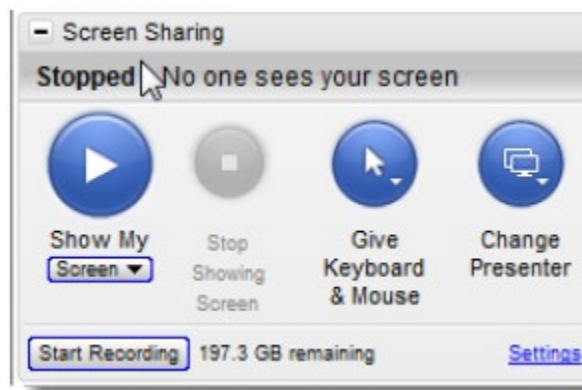
186. Claims 3 and 11 depend from independent claims 1 and 9, respectively, and recite, "*wherein the multi-user interactive software application allows a remote device to control one or more of a pointer device, a keyboard input, or a display of the [system/isolated computing environment].*" The Walsh-AAPA-Innes combination teaches this limitation.

187. As noted above, the '601 patent admits that WebEx and GoToMeeting disclose the "*multi-user interactive software application*" element. EX1001, 4:45-49. WebEx and GoToMeeting would be used on Walsh's computer system, either within a browser or as a standalone application. EX1018, 1-2; EX1024, 12-13, 204-05; EX1033, 2.

188. A POSA would understand that tools such as WebEx allow "*a remote*

device to control . . . a display of the system.” During a WebEx meeting, a presenter (“*a remote device*”) can share content with participants. For example, the presenter can share their desktop, an application, or a document. EX1011, 0001-02.

189. GoToMeeting also allows “*a remote device to control one or more of a pointer device, a keyboard input, or a display of the system.*” As I explained above, the Screen Sharing Pane in GoToMeeting is used to: (1) start or pause screen sharing; (2) select which application window or screen to share; (3) give keyboard and mouse control to another meeting participant; and (4) make another meeting participant the presenter. EX1024, 93.



EX1024, 93.

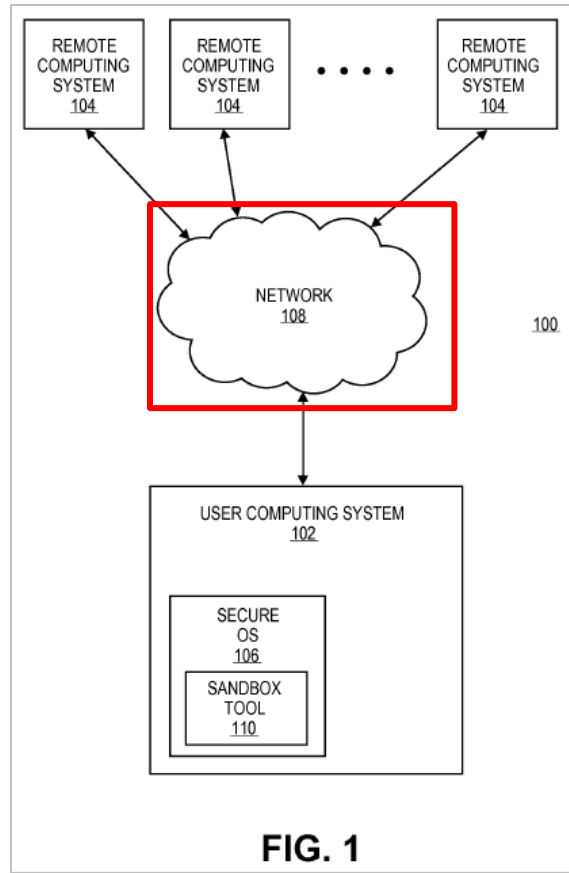
190. While sharing, the presenter can control what the other participants see. For example, the presenter can synchronize views of a participant (“*control . . . a display of the system*”), so the participant’s view is matched to the presenter’s. EX1011, 0001-02. A POSA would have understood that synchronizing the display of a participant controls the display of a system.

191. In the combined Walsh-AAPA-Innes system, Walsh’s web browser that runs a WebEx meeting (“*multi-user interactive software application*”) would enable a presenter (“*a remote device*”) to share and control content displayed on the client device (“*control . . . a display of the [system/isolated computing environment]*”).

F. Claims 4 and 12

192. Claims 4 and 12 depend from claims 1 and 9, respectively, and recite “[*wherein the system is connected/connecting to a network, the network corresponding to one or more of a local area network (LAN), a wide area network (WAN), a cellular network, a Wi-Fi network, a digital subscribers line (DSL) network, a cloud computing network, a data center, or a satellite network.*” Walsh discloses this claim.

193. Referring to Figure 1 (annotated below), Walsh discloses that “user computing system 102” is connected to “networks 108,” which can be “any type of communications networks, whether wired or wireless, to allow the computing system to communicate, such as *wide-area networks* (e.g. Internet) or *local-area networks*.” EX1006, 3:58-63.



EX1006, FIG. 1 (Annotated).

G. Claims 7 and 15

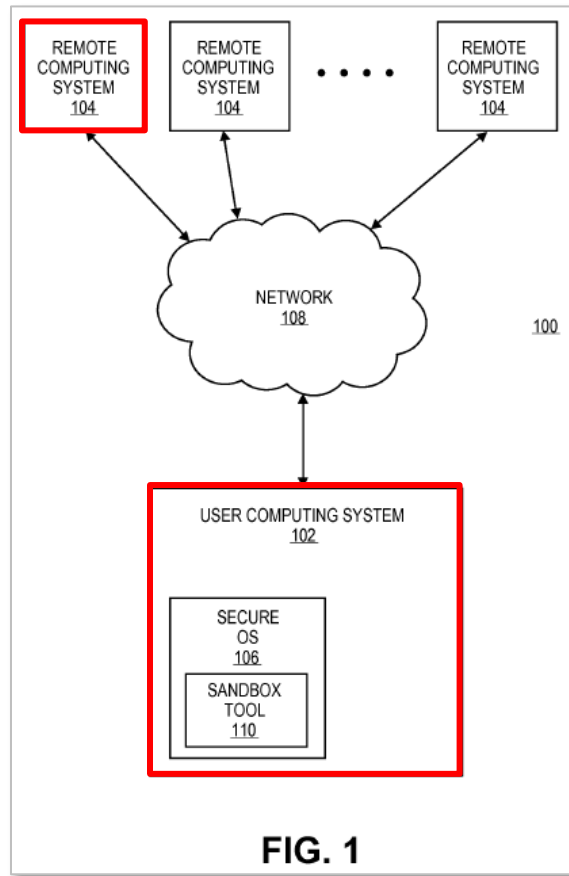
194. Claims 7 and 15 depend from claims 4 and 12, respectively, and recite “[wherein the processor is further configured to: run/running] one or more trusted applications in the first memory space, and wherein the one or more trusted applications communicate with one or more trusted machines via the network.”

Walsh discloses this limitation.

195. As described for Element 1C, Walsh discloses that a “user computing system 102” can execute “trusted applications” in “user execution environment

220,” which is supported by its own “user namespace” (“*first memory space*”).

EX1006, 5:32-40. The user execution environment 220 “allows a user to run applications programs” and “communicate with the remote computing systems 104” (“*trusted machines*”) via a network 108. EX1006, 11:58-64.



EX1006, FIG. 1 (Annotated).

XI. GROUND 2: Claims 5, 6, 13, And 14 Are Obvious Over Walsh, AAPA, Innes, and Netfilter.

A. A POSA would have been motivated to combine Walsh, AAPA, and Innes with Netfilter based on Walsh’s disclosure

196. As I described for Element 1I and claim 7, the Walsh-AAPA-Innes

system allows “user computer system 102” to access information from “remote computers 104” over a “network 108.” EX1006, 3:57-60. And Walsh discloses that the “user computer system 102” may use “security enhanced Linux (‘SELinux’).” EX1006, 3:44-45. To the extent that neither Walsh nor Innes provide implementation details about firewalls for Linux computers connected to the Internet, Netfilter does. Thus, a POSA would have been motivated to combine Walsh-AAPA-Innes with Netfilter to provide added security to the content-isolation disclosed by Walsh.

197. Netfilter discloses a “packet filtering framework” for use with the Linux operating system. EX1013, 1. In particular, Netfilter allows users to “build internet firewalls based on stateless and stateful packet filtering.” EX1013, 1.

198. A POSA would have been motivated, with a reasonable expectation of success, to use Netfilter’s firewall within Walsh’s computer system because this combination would amount to nothing more than combining familiar elements according to their known methods of use. Again, Walsh discusses protecting a computer system using the Linux operating system. And Netfilter discloses a firewall built into the Linux operating system. Thus, a POSA would have reasonably expected that Netfilter’s firewall would work in Walsh’s computer system to further protect it.

B. Claims 5 and 13

199. Claims 5 and 13 depend from claims 4 and 12, respectively, and recite “[*wherein the processor is further configured to: implement/implementing*] a *host-based firewall, wherein the host-based firewall isolates the [system/first memory space and the second memory space] from communicating with one or more other devices on the network.*” The Walsh-AAPA-Innes-Netfilter combination discloses this element.

200. Again, Walsh discloses computers that implement the Linux operating system. EX1006, 3:44-45. And Netfilter discloses a Linux-based software framework for building “internet firewalls” (“*a host-based firewall*”). EX1013, 1. Netfilter’s firewall regulates incoming and outgoing traffic by “stateless and stateful packet filtering” (“*isolates the system from communicating with one or more other devices on the network*”). EX1013, 2.

201. In the Walsh-AAPA-Innes-Netfilter combined system, Walsh’s “processor” (“*the processor*”) would implement Netfilter’s “internet firewall” (“*host-based firewall*”) that regulates incoming and outgoing traffic (“*isolates the [system/first memory space and the second memory space] from communicating with one or more other devices on the network*”).

C. Claims 6 and 14

202. Claims 6 and 14 depend from claims 5 and 13, respectively, and recite

“wherein the host-based firewall blocks incoming communications from the network that are sent to the workspace.” The Walsh-AAPA-Innes-Netfilter combination discloses this limitation.

203. Again, Walsh’s user execution environment (*“the workspace”*) allows a user to execute applications and programs. EX1006, 5:30-32. And Netfilter’s firewall (*“the host-based firewall”*) regulates incoming traffic (*“block[ing] incoming communications from the network”*) based on a set of packet-filtering rules. EX1013, 2.

204. In the Walsh-AAPA-Innes-Netfilter combined system, Walsh’s computing system would implement Netfilter’s firewall (*“host-based firewall”*) that regulates incoming and outgoing traffic (*“block[]incoming communications from the network”*) based on an application executing in the workspace (*“sent to the workspace”*).

XII. GROUND 3: Claims 8 And 16 Are Obvious Over Walsh, AAPA, Innes, and Franco.

A. A POSA would have been motivated to combine Walsh, AAPA, and Innes with Franco based on Walsh’s disclosure.

205. Again, Walsh discloses a “sandbox tool 110” (*“internal isolation firewall”*) that isolates the “isolated execution environment 230” (*“isolated computing environment”*) from the “user execution environment 220” (*“workspace”*). EX1006, 5:52-57. Notably, Walsh’s “sandbox tool 110” allocates

resources to “isolated execution environment 230[,]” and applies security policies of the secure operating system to the “isolated execution environment 230.”

EX1006, 4:35-42. As a result, “content 225” running in the isolated execution environment can only access resources within the isolated execution environment.

EX1006, 5:57-65.

206. In some instances, Walsh discloses that content from the user execution environment 220 can be transferred to the namespace for the isolated execution environment 230:

[S]andbox tool 110 can copy necessary content and content 225 to the namespace for the isolated execution environment 230. For example, the sandbox tool 110 can copy or can instruct the secure OS 106 to copy the necessary content and the content 225 to the home directory and tmp directory in the same location it would be found in the home directory and tmp directory of the user execution environment 220. The necessary content can be any content (files, applications, programs) which is required by the isolated execution environment 230 to function similarly as the user execution environment 220. The content 225 can be application programs, files, data, etc. that the user specified in the request.

EX1006, 10:11-23.

207. Walsh also discloses user interaction. For example, Walsh discloses that the security context for the isolated execution environment 230 can be

“provided by the user of the computing system 102.” EX1006, 10:58-60.

Additionally, Walsh discloses that “sandbox tool 110” can store and notify the user of malicious activity occurring within the isolated execution environment.

EX1006, 11:33-38. To the extent that Walsh does not provide details about such user interactions, Franco does. A POSA wishing to implement Walsh’s system thus would have been motivated to combine it with Franco.

208. Like Walsh, Franco is directed to preventing malware or other malicious software from compromising a host computer. EX1009, 1:22-29. To overcome this issue, Franco executes applications with “reduced privilege” so that they are restricted from accessing portions of the host. EX1009, 2:31-36. For example, the restricted processes are given access to a “containment zone” which is treated as untrusted by the system. EX1009, 2:41-46. When attempts to access a restricted area of the host are made, Franco captures explicit user input to authorize the access. EX1009, 2:36-56. Franco implements this functionality by placing a broker mechanism between the restricted process and the remaining area of the system. EX1009, 2:48-51. The broker mechanism inspects requests for resources outside the “containment zone.” EX1009, 4:41-46.

209. A POSA would have had a reasonable expectation of success using Franco’s user-prompt mechanism within Walsh’s user computing system because this combination would amount to nothing more than combining familiar elements

according to their known methods of use. Both Walsh and Franco disclose isolating processes within a computing device. Thus, a POSA would have understood that Walsh's security contexts could be modified with an additional rule requiring Franco's user input when data is transferred between isolated and non-isolated processes and/or environments.

B. Claims 8 and 16

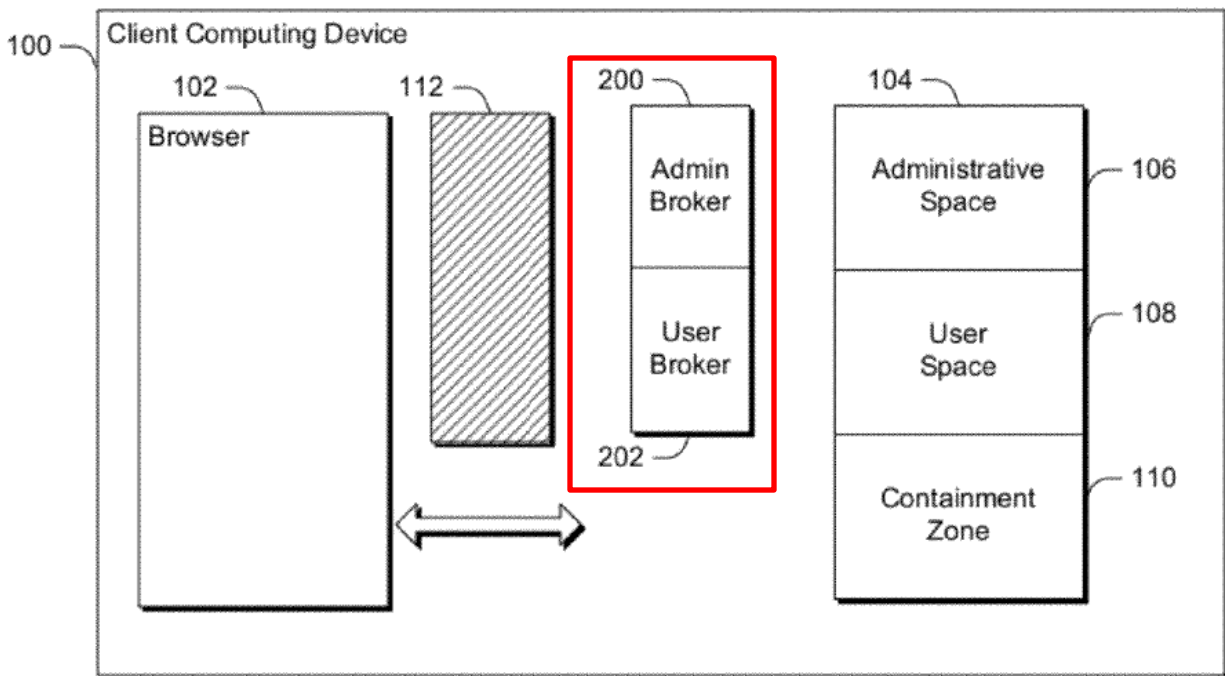
210. Claims 8 and 16 depend from claims 1 and 9, respectively, and recite *“wherein the internal isolation firewall prompts a user of the [system/workspace] to allow communication between the first memory space and the second memory space.”* The Walsh-AAPA-Innes-Franco combination discloses this limitation.

211. Again, Walsh discloses a “sandbox tool 110” (*“internal isolation firewall”*) that isolates the “isolated execution environment 230” (*“isolated computing environment”*) from the “user execution environment 220” (*“workspace”*). EX1006, 5:52-57. And Walsh discloses that the security context for the isolated execution environment 230 can be “provided by the user of the computing system 102.” EX1006, 10:58-60.

212. Franco prevents applications from accessing restricted areas of a computer and requires explicit user input when access attempts are made. EX1009, 2:36-56. When the restricted area is accessed, a broker object prompts the user with a dialog box, requiring the user to input credentials (*“prompts a user of the*

[system/workspace]”) to allow the access attempt (“to allow communication”).

EX1009, 5:4-11. For example, if a user is attempting to save a file within the My Documents folder, the broker object displays a dialog box prompting the user for input. EX1009, 5:4-7. Similarly, if a process attempts to access the administrative space of the host, the broker object prompts the user for administrator credentials to facilitate the request. EX1009, 5:8-11.



EX1009, FIG. 2 (annotated).

213. In the Walsh-AAPA-Innes-Franco combined system, Walsh’s “sandbox tool 110” (“internal isolation firewall”) would display Franco’s dialog box (“prompt[] a user of the [system/workspace]”) when content is transferred between the isolated and non-isolated applications (“to allow communication between the first memory space and the second memory space”).

XIII. GROUND 4: Claims 1-4, 7, 9-12, and 15 Are Obvious Over Austin, AAPA, And Bloch.

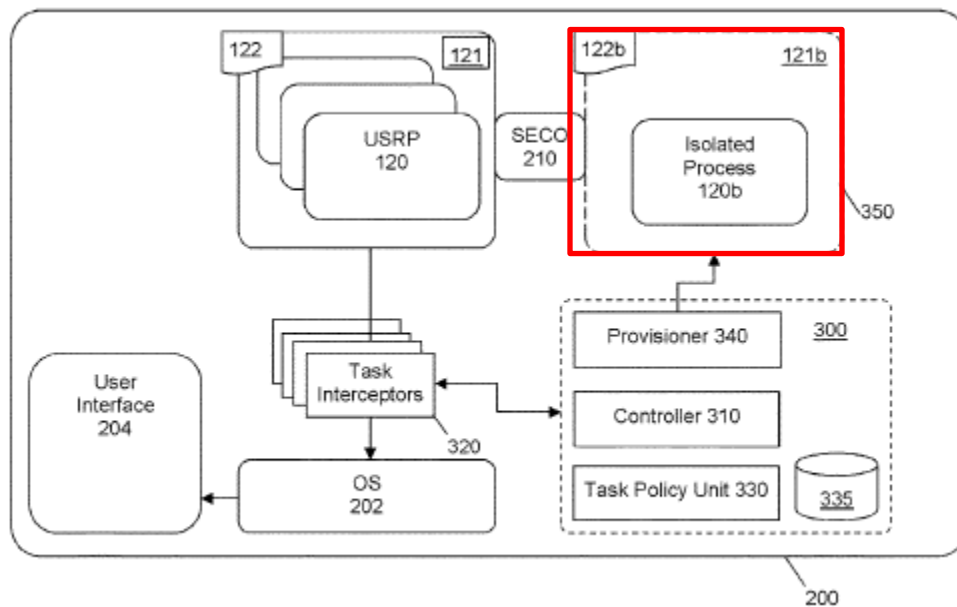
A. A POSA would have been motivated to combine Austin with Bloch based on Austin’s disclosure.

214. Austin recognizes the “on-going need to protect computer devices from malicious content.” EX1005, ¶5. For example, Austin recognizes “untrusted content may be copied by the user from an untrusted source (e.g. untrusted content, untrusted application) . . .” EX1005, ¶13. Austin further recognizes risks posed by “untrusted tasks” such as “certain forms of Web browsing, viewing email attachments, starting an untrusted application program, or accessing a particular file” EX1005, ¶72.

215. To address these challenges, Austin leverages isolation because it inhibits untrusted tasks “from interfering with other tasks or components of the computer device.” EX1005, ¶72. Isolation may involve “limiting the resources of the computer device which are accessible by the untrusted content.” EX1005, ¶15. As a result, the untrusted task and content may be manipulated. EX1005, ¶73.

216. Austin discloses a computer device with an operating system where “each process 120 that is initiated for a particular user will be run in a security context 121 that derives access rights and permissions from the user’s security account.” EX1005, ¶66. “For example, a particular process 120 is able to read from, but not write to, a file of the first resource 115.” EX1005, ¶68.

217. Austin handles untrusted tasks by opening them in a “task isolation environment or sandbox 350.” EX1005, ¶86. “Where the controller 310 determines that the task in question relates to untrusted content, i.e. is determined to be an untrusted task, then the controller 310 causes the untrusted task to be opened in a task isolation environment or sandbox 350.” EX1005, ¶86. A temporary user account may be created for the sandbox, such that the isolated task uses the temporary user account for data and registry access. EX1005, ¶91. As a result, “[a]ny malicious activity only affects the temporary user account 121b and does not affect the primary user account 121.” EX1005, ¶91.

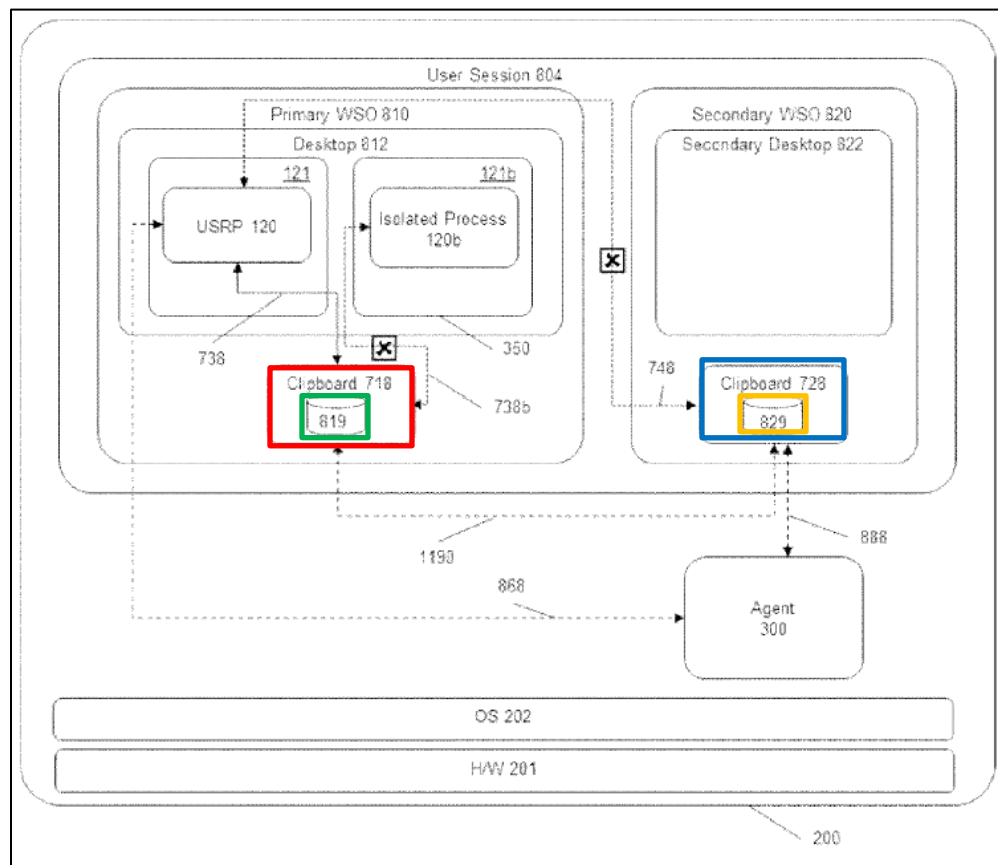


EX1005, FIG. 4 (Annotated).

218. In addition to isolating untrusted processes, Austin discloses another mechanism to protect the host from malware that may attempt to access the host system’s clipboard. EX1005, ¶13. Austin recognizes that “most conventional

operating systems” would allow the isolated untrusted process “to access the primary clipboard,” introducing an additional vulnerability. EX1005, ¶102.

219. Austin addresses this issue via “a primary clipboard accessible from a primary user account” and a “secondary clipboard . . . provisioned and associated with the secondary user account” within the isolated sandbox. EX1005, ¶20. Each clipboard is supported by its own respective memory source. The primary clipboard 718 (red below) is supported by a primary clipboard memory 819 (i.e., “physical memory within the hardware 201”) (green below). EX1005, ¶119. “[T]he secondary clipboard 728 [blue below] is supported by appropriate physical memory 829 [orange below].” EX1005, ¶119.



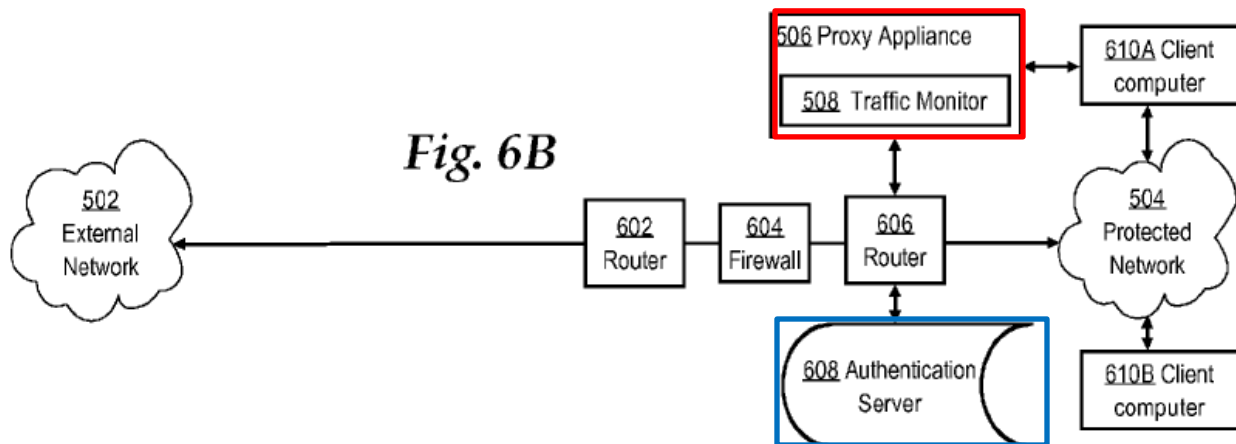
EX1005, FIG. 8 (annotated).

220. A POSA implementing Austin would not only have appreciated and understood Austin's process isolation, but also would have appreciated and understood Austin's disclosure of additional protection mechanisms, such as using two clipboards to prevent malware from compromising a single clipboard. These mechanisms "protect the computer device 200 from malicious attacks." EX1005, ¶74. Because Austin discloses networked based threats in the form of untrusted webpages and untrusted email attachments, a POSA reading Austin would have sought additional references such as Bloch to implement additional protection mechanisms directed at network-based threats. EX1005, ¶72.

221. Bloch addresses a problem associated with network-based threats, in that administrators may wish to manage (i.e., allow, block) network traffic. EX1007, 6:51-53. Bloch addresses this problem by using a proxy device to inspect and filter out suspicious content. EX1007, 3:35-40. The proxy device employs firewall rules to filter traffic, uses signature-based filters to avoid malware and spyware, and blocks certain downloads. EX1007, 3:30-48.

222. Bloch discloses an "apparatus and computer program product for managing and monitoring network traffic and filtering responses." EX1007, 1:62-64. Specifically, a proxy appliance manages "network traffic into and out of an internal network to block or redirect the traffic to avoid malware." EX1007, 3:36-

38. “The proxy appliance 506 is coupled to an authentication server 608 so that the proxy appliance can provide an integrated user authentication service, for example, using lightweight directory access protocol (LDAP), Microsoft Active Directory, etc.” EX1007, 8:20-24. The proxy appliance can also include “application proxies for hyper text transfer protocol (HTTP), hyper text transfer protocol secure (HTTPS), and file transfer protocol (FTP).” EX1007, 3:41-44. The proxy appliance uses “web reputations filters that analyze different web traffic and network-related parameters to evaluate the trustworthiness of a given URL.” EX1007, 3:55-57.



EX1007, FIG. 6B (Annotated).

223. As an example operation, the proxy appliance may receive a “client request” for “information from an HTTP server, FTP server, or other network resource that is located outside the protected network.” EX1007, 5:21-25. The system checks whether a user associated with the computer device has been

authenticated. EX1007, 5:25-28. If the user is authenticated, “then several checks are made for parameters associated with the client request.” EX1007, 5:35-37. For example, the client and server IP addresses, as well as the requested URL, are checked against both blacklists and whitelists. EX1007, 5:43-54. Additional checks, such as determining whether a user-agent and requested file extension are within a blacklist, are performed. EX1007, 5:60-66. Based on the checks, the content may be fetched from the server and sent to the client device. EX1007, 5:45-50.

224. A POSA would have been motivated to combine Austin with Bloch to “*authenticate the isolated computing environment*” and “*send data to an untrusted destination . . . via a proxy device*” because the combination would further protect Austin’s computing device. As stated above, Austin recognizes the threat posed by untrusted tasks such as accessing untrusted websites. EX1005, ¶73. To protect the computing device, Austin isolates untrusted tasks to prevent interference with other tasks and computing device components. EX1005, ¶72. As a further layer of protection, a POSA would use Bloch’s proxy appliance and authentication server to access an untrusted website (“*an untrusted destination*”).

225. In order to utilize Bloch’s proxy appliance, a POSA would first authenticate Austin’s isolated environment with Bloch’s proxy appliance to establish a layer of trust. As discussed, Austin is concerned with isolating untrusted

tasks, whether that involves accessing untrusted content or communicating with untrusted network entities (e.g., an untrusted website). EX1005, ¶¶72-73. To confirm that Bloch's proxy device can be trusted, a POSA would be motivated to perform authentication between the Austin's isolation environment and Bloch's proxy device via Bloch's authentication server. Once authenticated, Bloch's proxy device would be used to further protect Austin's computing device.

226. Bloch's proxy appliance can be used to block or redirect network traffic based on domains and/or IP addresses included within the traffic. EX1007, 6:4-22; EX1007, 10:18-29. For example, Bloch's proxy appliance can block traffic from a domain that is known to distribute spyware or other malware. EX1007, 7:52-61. When combined with Austin's isolation environment, Bloch's proxy appliance would be used to block traffic from sources confirmed to distribute malware or spyware, further protecting Austin's computing device.

227. In the Austin-Bloch combination, Austin's device would be able to access network content via through Bloch's proxy appliance and authentication server. This combination amounts to nothing more than using a known technique according to known methods of use. Connecting Bloch's proxy device and authentication server with Austin's device was a known technique to improve similar devices, methods, or products in the same way (e.g., to use a proxy device to protect computer functionality that is connected to a network). For example,

Austin discloses accessing content from websites. EX1005, ¶¶72-73. Bloch's proxy appliance is used to provide network resources (e.g., HTTP resources) to a computing device. EX1007, 4:21-50. Thus, Austin's computing device would access network content via Bloch's proxy appliance.

228. A POSA would also have success authenticating Austin's isolated environment with Bloch's authentication server. Austin discloses authenticating a user attempting to log into the computer by validating a set of user credentials (e.g., username and password). EX1005, ¶62. Austin states that the credentials are validated locally or remotely, such as by an active directory domain controller. EX1005, ¶62. Thus, Austin already contemplates performing authentication with a remote entity and could therefore authenticate with Bloch's authentication server.

229. A POSA would further have a reasonable expectation of success authenticating Austin's isolated environment with Bloch's authentication server because the isolated environment is provided its own set of credentials. When "controller 310" creates a new "task isolation environment 350," it also creates a new, temporary user account. EX1005, ¶87. The new user account is assigned credentials used to log into the new account. EX1005, ¶87. Therefore, a user could provide the credentials of the new user account associated with the isolated environment to Bloch's authentication server. Once authenticated, the user would have access to the new user account and the associated task isolation environment.

For example, once logged into the new user account, the isolated tasks are displayed within the desktop of the computing device. EX1005, ¶88.

B. A POSA would have been motivated to combine Austin-Bloch with AAPA

230. The '601 patent admits that the claimed “multi-user interactive software application” could be implemented via commercial products, such as WebEx. EX1001, 4:45-46. WebEx was a well-known product and was used as a standalone application, an email client plugin, or within a web browser. For example, by 2015 WebEx had over 60 million users. EX1017, 1.

231. A POSA would have been motivated, with a reasonable expectation of success, to use Bloch’s proxy appliance with Austin’s client computer in order to securely access WebEx traffic via Austin’s client computer. Austin explains that a client computer may access content over the Internet via a web browser application. That content, a POSA would have understood, would include WebEx or GoToMeeting traffic, and a POSA would have understood that WebEx or GoToMeeting would be (i) installed as a standalone programs on Austin’s client computer or (ii) used within a web browser or email client on Austin’s client computer. EX1018, 1-4; EX1024, 43-46, 131, 204-05; EX1033, 2. The combined Austin-AAPA system would then be used on a network including Bloch’s proxy appliance to provide Austin’s client computer with secure access to network resources (e.g., an HTTP server). Thus, it would have been obvious to make the

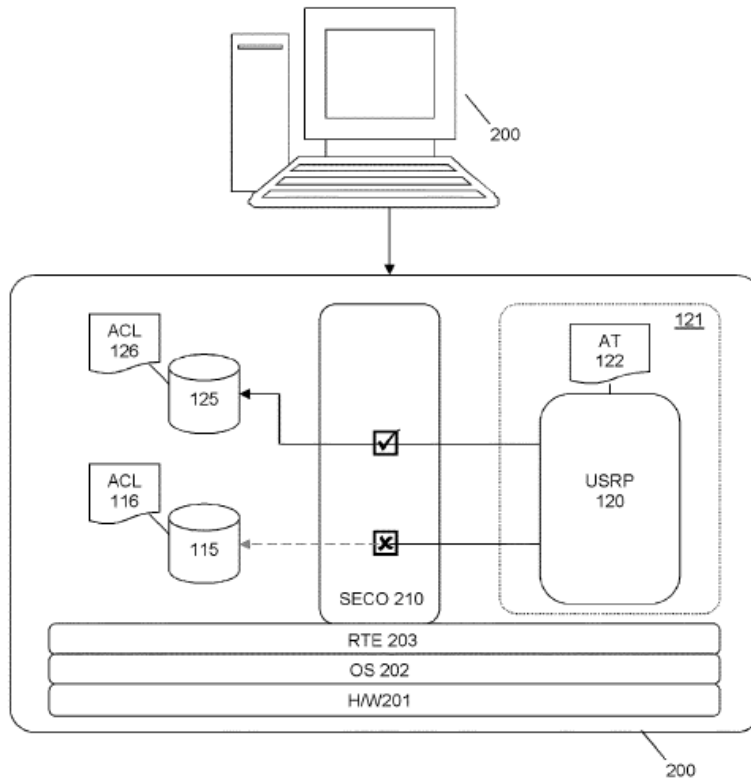
Austin-AAPA-Bloch combination because it would have amounted to nothing more than combining familiar elements according to their known methods of use.

C. Independent claims 1 and 9

1. [1-pre] “A system” and [9-pre] “A method”

232. To the extent the preambles are limiting, Austin discloses them.

233. Austin discloses a computer device, method, and system. EX1005, Abstract. Referring to Austin’s Figure 1 (reproduced below), “computer device 200 includes physical hardware (HW) 201 such as memory, processors (CPUs), I/O interfaces, backbone, power supply and so on.” EX1005, ¶63. “[H]ardware 201 supports an operating system (OS) 202 that provides a multitude of components, modules and units that together coordinate to provide a runtime environment (RTE) 203.” EX1005, ¶63. The runtime environment may execute “a plurality of processes (USRP) 120.” EX1005, ¶63. The computer device further includes “resources 115, 125,” such as “installed software, system services, drivers, files and/or registry settings.” EX1005, ¶63. Austin’s “computer device” implements a “*method*” because it creates a “task isolation environment” to run untrusted content in order to limit “the resources of the computer device which are accessible by the untrusted content.” EX1005, ¶15.

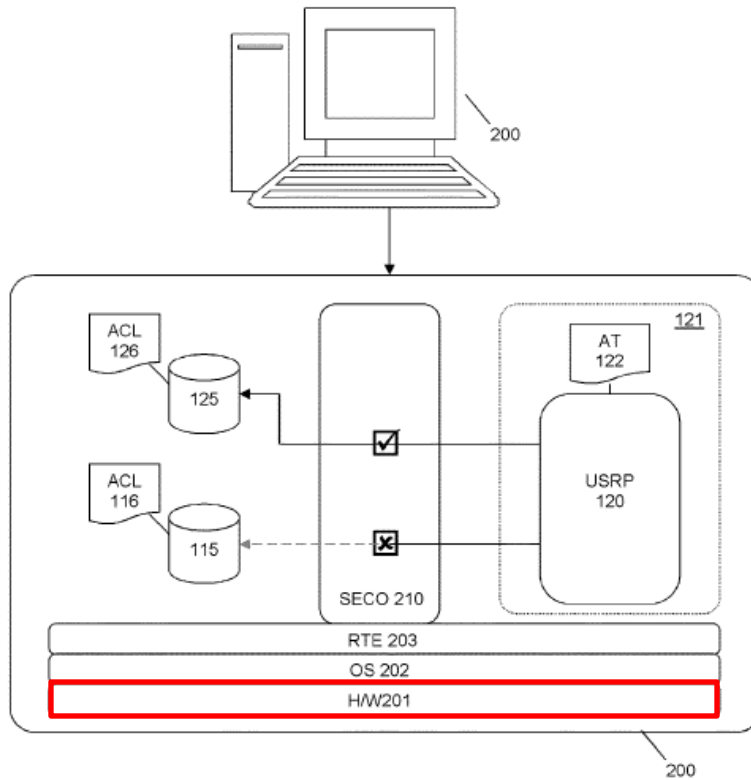


EX1005, FIG. 1.

2. [1A] “a memory”

234. Austin discloses “a memory.”

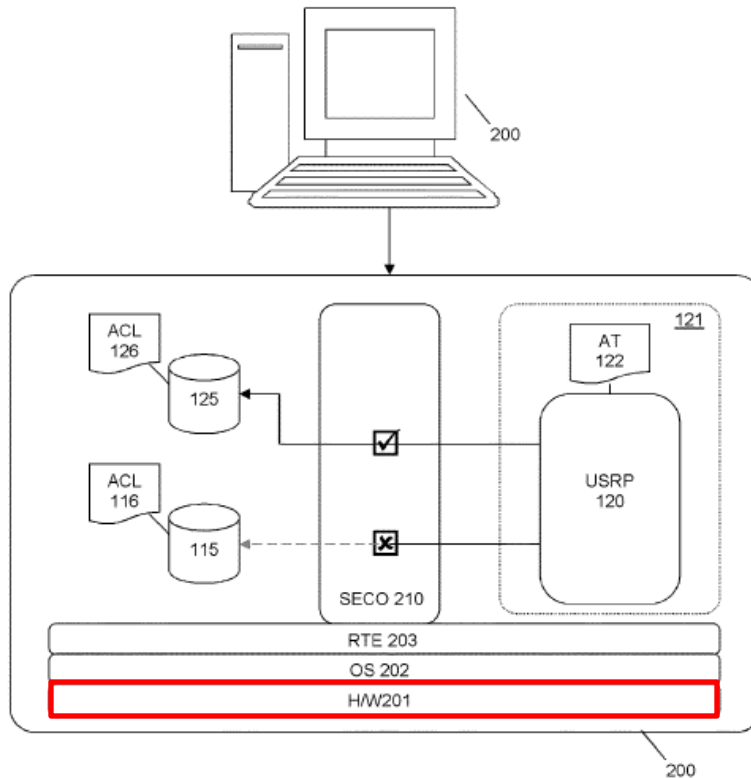
235. Austin’s Figure 1, shown below, depicts “a schematic overview of the computer device 200.” EX1005, ¶62. “[C]omputer device 200 includes physical hardware (HW) 201 such as *memory*. . . .” EX1005, ¶63 (emphasis added).



EX1005, FIG. 1 (Annotated).

3. [1B] “a processor configured to”

236. Referring again to Figure 1 (shown below), Austin’s “computer device 200 includes physical hardware (HW) 201 such as memory, *processors (CPUs)*.” EX1005, ¶63 (emphasis added).



EX1005, FIG. 1 (annotated).

4. [1C] and [9A] “[implement/implementing] a workspace that uses a host operating system, wherein the workspace enables operation of a first set of one or more applications or processes via a first memory space”

237. Austin discloses this element.

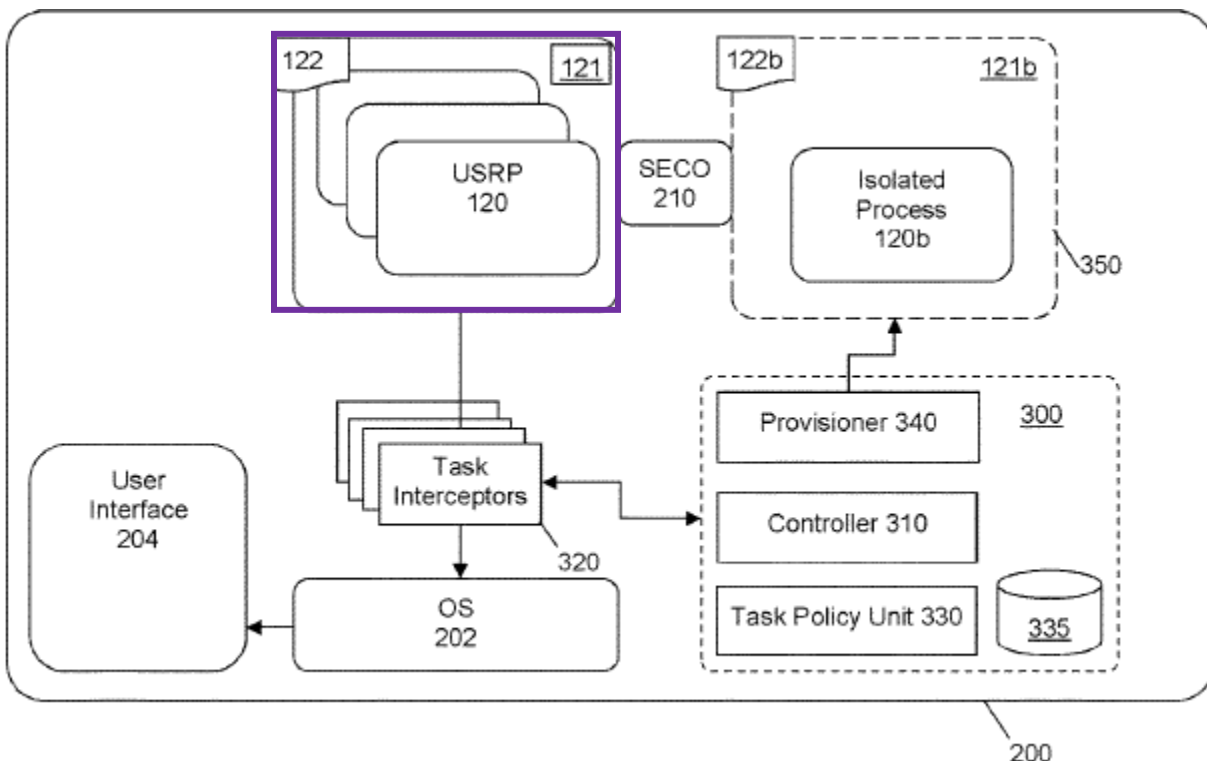
238. Again, Austin’s “computer device 200” includes “physical hardware (HW) 201,” with a processor and memory to support “operating system (OS) 202.”

EX1005, ¶63. The operating system provides “user account 121” (“a workspace that uses a host operating system”). EX1005, ¶87. A POSA would have understood that Austin’s user account encompasses resources available to the user account.

239. For example, Austin discloses that the user account (“workspace”)

uses the operating system to access resources. EX1005, ¶¶67-68; FIG. 1. As noted in Austin, operating system security models assign access privileges to user accounts (“*the workspace*”). EX1005, ¶60. An account could be an ordinary user, super user, or administrator. EX1005, ¶60. Once logged in, the operating system enforces the privileges assigned to the user account. EX1005, ¶60. For example, when a process of the user account attempts to access a resource, the operating system references the assigned privilege level to control access to the resource. EX1005, ¶68.

240. And Austin’s user account 121 (purple below) supports execution “of one or more user processes 120” (“*enables operation of a first set of one or more applications or processes via a first memory space*”). EX1005, ¶70.



EX1005, FIG. 4 (annotated).

241. In particular, Austin’s “user account” has its memory own “namespace” (“*a first memory space*”). EX1005, ¶95. Austin discloses a “primary clipboard . . . accessible from a primary user account.” EX1005, ¶20. The primary clipboard 718 is supported by a primary clipboard memory 819 (i.e., “physical memory within the hardware 201”). EX1005, ¶119, FIGs. 7 (shows applications in the memory space), 8 (shows physical first memory space).

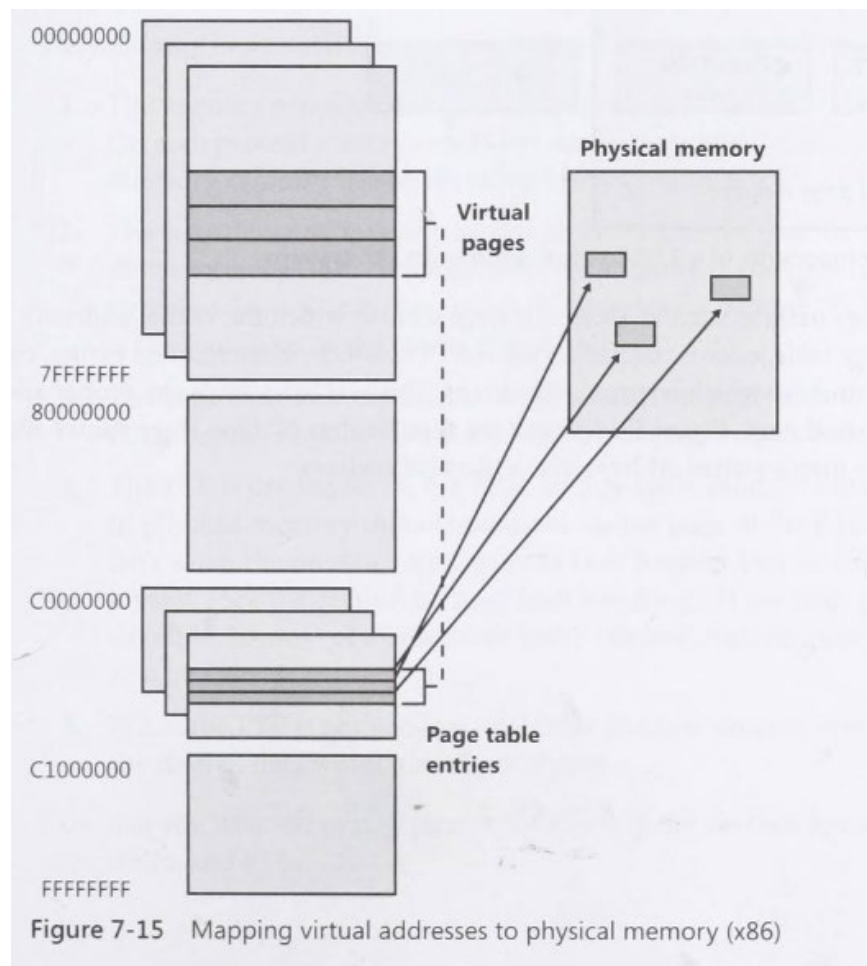
242. Applications and processes execute in their own memory spaces. Modern operating systems use virtual memory, meaning that memory resources are virtualized and processes operate as if they have access to the entire computer’s memory. EX1021, 0023. As a result, the address space of a process could be greater than the physical memory of the computer. EX1021, 0023. Each process is assigned a process address space indicating virtual memory addressable by the process. EX1021, 0023. For example, a 64-bit architecture may assign a process a flat (e.g., single linear range) 64-bit address space. EX1021, 0023. Generally, a flat address space is unique to the process. EX1021, 0023. Processes that access memory addresses outside assigned memory areas are terminated. EX1021, 0024. Virtual memory addresses are mapped to distinct, physical memory addresses using page tables. EX1022, 0051 & 0100.

243. Though described in the context of Linux, these concepts also apply to

the Windows operating system, and therefore apply to Austin's computing device.

For example, the Windows operating system also assigns each process its own private virtual address space. EX1025, 4. Windows also uses page tables to map the process's private virtual address space to a private physical address space.

EX1030, 0073-0075. Additionally, the private address space of a Windows process cannot be accessed by other processes without permission. EX1030, 0061. For example, the image below depicts mapping addresses from a process's virtual memory to the machine's physical memory.

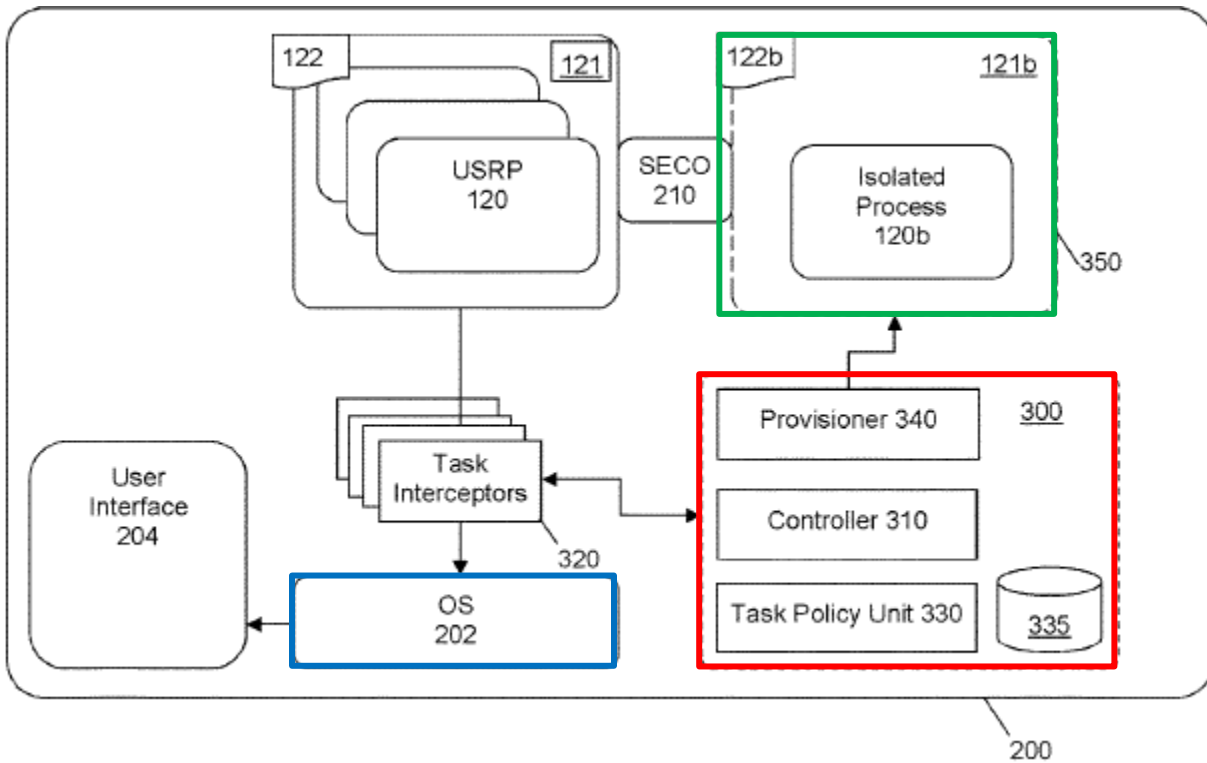


EX1030, 0073 (FIG. 7-15).

5. [1D] and [9B] “[implement/implementing] an isolated computing environment that uses the host operating system”

244. Austin discloses this element. Austin discloses an “isolation environment 350” (“*an isolated computing environment*”) that is implemented by “content isolation agent 300.” EX1005, ¶74. Referring to Austin’s Figure 4 (annotated below), content isolation agent 300 (in red) includes “a controller unit 310” that causes untrusted tasks “to be opened in a task isolation environment or sandbox 350” (in green). EX1005, ¶¶74, 86.

245. Austin’s “isolation environment 350” (“*isolated computing environment*”), which is implemented by content isolation agent 300, “*uses the host operating system,*” as claimed, because Austin discloses that the content isolation agent 300 either “operates in close association with the native operating system 202 [blue below]” or is “integrated within the operating system 202.” EX1005, ¶74.

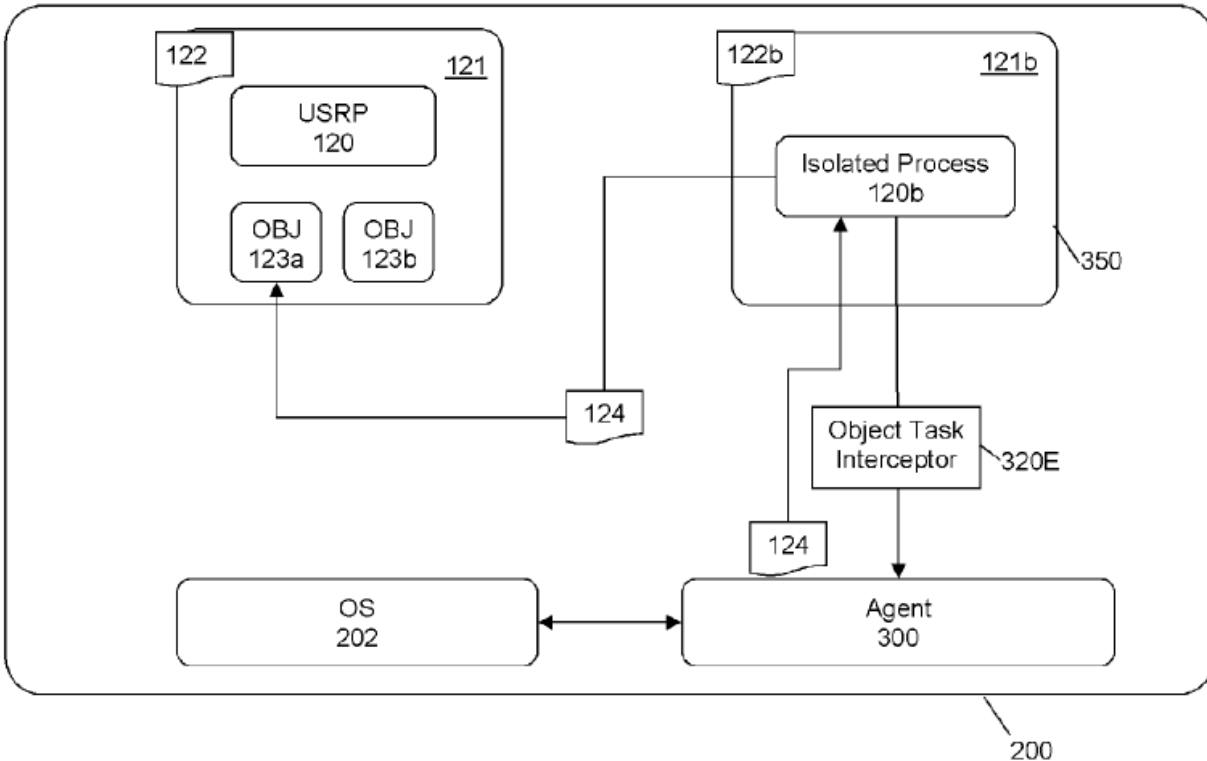


EX1005, FIG. 4 (Annotated).

246. Named objects are an example of Austin’s isolation environment using the host operating system. In Windows, named objects are used by the operating system for inter-process communication and synchronization. EX1005, ¶95. Example named objects include semaphores and mutexes. EX1005, ¶95. In Austin, a “task interceptor 320E” intercepts requests by “isolated process 120b” to access named objects. EX1005, ¶96. Based on a policy associated with the isolated process, access to the named object is controlled. EX1005, ¶96.

247. For example, as depicted in Figure 5 below, “isolated process 120b” requests access to “object 123a.” EX1005, ¶96. The object task interceptor intercepts the request so that “agent 300” can check the policy rules applied to the

process. EX1005, ¶96. Here, the request is granted and the isolated process is allowed to access (e.g., open) the named object. EX1005, ¶96.



EX1005, FIG. 5.

6. [1E] and [9C] “wherein the isolated computing environment enables operation of a second set of one or more applications or processes via a second memory space”

248. Austin discloses this element. Austin’s “isolation environment 350” (“*isolated computing environment*”) enables untrusted tasks to be executed (“*enables operation of a second set of one or more applications or processes*”).

Again, the controller unit 310 causes untrusted tasks “to be opened in a task isolation environment or sandbox 350.” EX1005, ¶86. The untrusted tasks may include “non-executable content (data)” or “executable content (code).” EX1005,

¶72.

249. Austin’s untrusted task (“*a second set of one or more applications or processes*”) execute in the namespace of a “temporary user account” (“*a second memory space*”). According to Austin, the untrusted task “runs in this temporary user account 121b and accesses the profile of the temporary user for both data access and registry access.” EX1005, ¶91. The untrusted task “does not have access to the namespace of the primary user account 121.” EX1005, ¶95. As a result, “[a]ny malicious activity only affects the temporary user account 121b and does not affect the primary user account 121.” EX1005, ¶91. Austin also discloses a “secondary clipboard . . . provisioned and associated with the secondary user account” within the isolated sandbox EX1005, ¶20. “[T]he secondary clipboard 728 is supported by appropriate physical memory 829.” EX1005, ¶119, FIGs. 7 (shows second set of applications), 8 (shows second memory space).

250. As I noted above for [1C]/[9A], applications and processes execute in their own memory spaces. As a result, the “*second memory space*” enabling the “*second set of one or more applications or processes*” is different than the “*first memory space*” because both sets of applications or processes could not execute from the same physical memory space.

7. [1F] and [9D] “wherein the second set of one or more applications or processes comprises a multi-user interactive software application”

251. Austin in view of AAPA discloses this element.

252. Again, Austin’s untrusted tasks are the “*second set of one or more applications or processes.*” Untrusted tasks include executable code, and applications such as web browsers and email clients. EX1005, ¶¶72-73, 81.

253. The ’601 patent admits that the claimed “*multi-user interactive software application*” existed in commercial products such as WebEx or GoToMeeting. EX1001, 4:45-46.

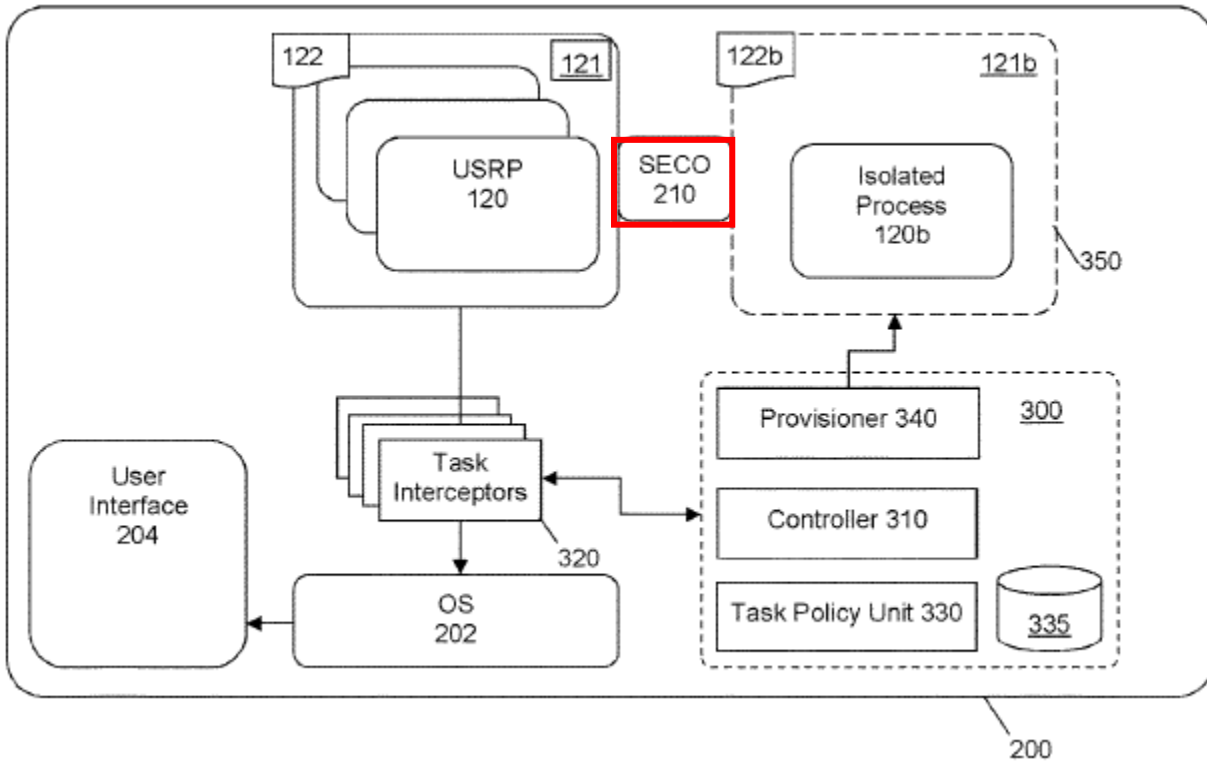
254. A POSA would have been motivated, with a reasonable expectation of success, to execute WebEx or GoToMeeting (“*multi-user interactive software application*”) as one of Austin’s untrusted tasks so that any “malicious activity” on WebEx or GoToMeeting would affect only “the temporary user account 121b and” would “not affect the primary user account 121.” EX1005, ¶91. This combination would have been obvious because it would have amounted to nothing more than combining familiar elements (e.g., Austin’s untrusted tasks and WebEx/GoToMeeting) according to their known methods of use. In the Austin-AAPA combined system, Austin’s untrusted task (“*the second set of one or more applications or processes*”) would be WebEx (“*a multi-user interactive software application*”).

8. [1G] and [9E] “[isolate/isolating] the isolated computing environment from the workspace using an internal isolation firewall”

255. Austin discloses this element.

256. An “internal isolation firewall,” the ’601 patent discloses, enforces “the segregation of the first and second memory spaces.” EX1001, 8:37-38. Like this disclosure, Austin’s “security module 210” (“*internal isolation firewall*”) isolates an untrusted task “from content of the main user account 121.” EX1005, ¶89.

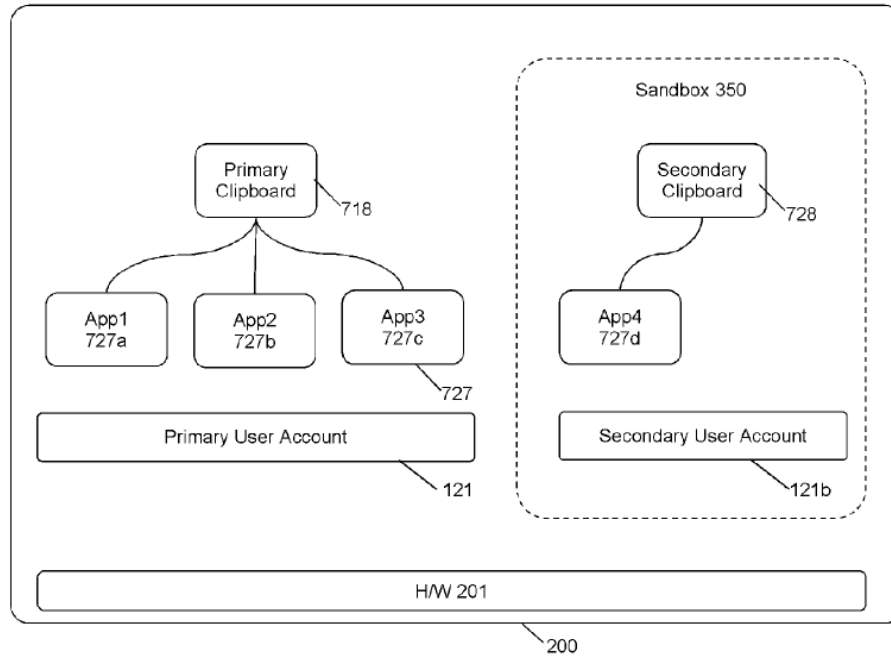
257. “[S]ecurity module 210 is arranged to perform an access check” when an isolated process requests access to a resource, such as “software, system services, drivers, files and/or registry settings.” EX1005, ¶67, ¶63. As a result, the untrusted task is unable to access data or the profile of user account 121. EX1005, ¶90. Additionally, “the untrusted task” is restricted to “temporary user account 121b” for data and registry access. EX1005, ¶91. Thus, “[a]ny malicious activity only affects the temporary user account 121b and does not affect the primary user account 121.” EX1005, ¶91.



EX1005, FIG. 4 (Annotated).

258. An example of the isolation implemented by Austin’s security module is clipboard isolation. Austin notes that in typical systems, even isolated processes would have access to a primary (e.g., unrestricted) clipboard. EX1005, ¶102. This access introduces a possible attack vector for malware or other threats operating within the isolation environment. EX1005, ¶13, ¶102. However, the isolation provided by Austin is implemented to such a degree that even the clipboard mechanism is isolated. As depicted below, the non-isolated primary user account has access to “primary clipboard 718[,]” whereas the “isolated application 727” of the secondary user account accesses “secondary clipboard 728.” EX1005, ¶¶102-03. As a result, clipboard requests from the isolated application are intercepted and

implemented using the secondary clipboard. EX1005, ¶104.



EX1005, FIG. 7.

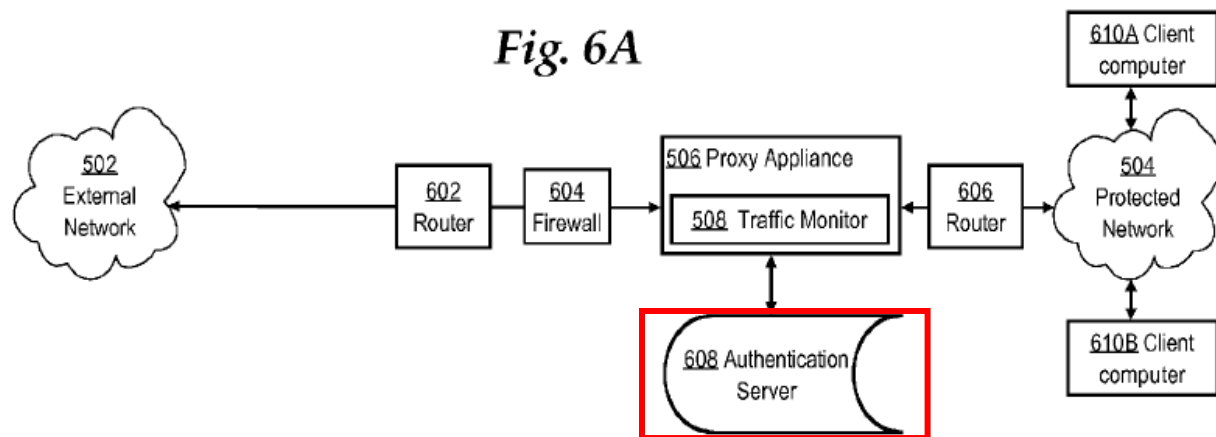
9. [1H] and [9F] “[authenticate/authenticating] the isolated computing environment with an authentication device”

259. The Austin-AAPA-Bloch combination discloses this element.

260. Austin recognizes many operating systems tie access privileges to user accounts. EX1005, ¶60. As noted above, a new user account 121b and credentials may be created and associated with an isolation environment. EX1005, ¶87. The new user account within the isolation environment may be accessed by validating the credentials. EX1005, ¶87. This validation may occur at a local authentication system or via a remote authentication service. EX1005, ¶62. Austin discloses that the remote authentication service may be an “[a]ctive [d]irectory

domain controller.” EX1005, ¶62. To the extent Austin does not provide implementation details about such a remote authentication service, a POSA interested in implementing the remote authentication service within Austin’s system would have been motivated to search for those details. That would have led to Bloch.

261. Bloch discloses an “authentication server 608 [red below]” to provide “integrated user authentication service, for example, using lightweight directory access protocol (LDAP), Microsoft Active Directory, etc.” EX1007, 8:22-24. For example, when a user requests a resource, such as from an HTTP server, the user may first be authenticated.” EX1007, 5:25-28.



EX1007, FIG. 6A (annotated).

262. As I explained above, a POSA would be motivated to implement Austin’s computing device on a network including Bloch’s proxy appliance provide further protection and securely access network content. To provide protection and securely access content Austin’s computing device, including the

isolated task environment (“*the isolated computing environment*”) would authenticate with Bloch’s authentication server (“*an authentication device*”).

263. A POSA would have had a reasonable expectation of success using Austin’s computer device on a network including Bloch’s authentication server because it would amount to nothing more than combining familiar elements according to their known methods of use. Austin discloses communicating over a network via the computer device by, for example, accessing a website via a web browser and accessing attachments received via email. EX1005, ¶8. Bloch discloses client computers “coupled to protected network 504.” EX1007, 8:26-27. Austin further discloses remotely authenticating user credentials, and assigning user credentials to the isolated task environment. EX1005, ¶¶62, 87. Bloch discloses authenticating user credentials. EX1007, 8:20-24.

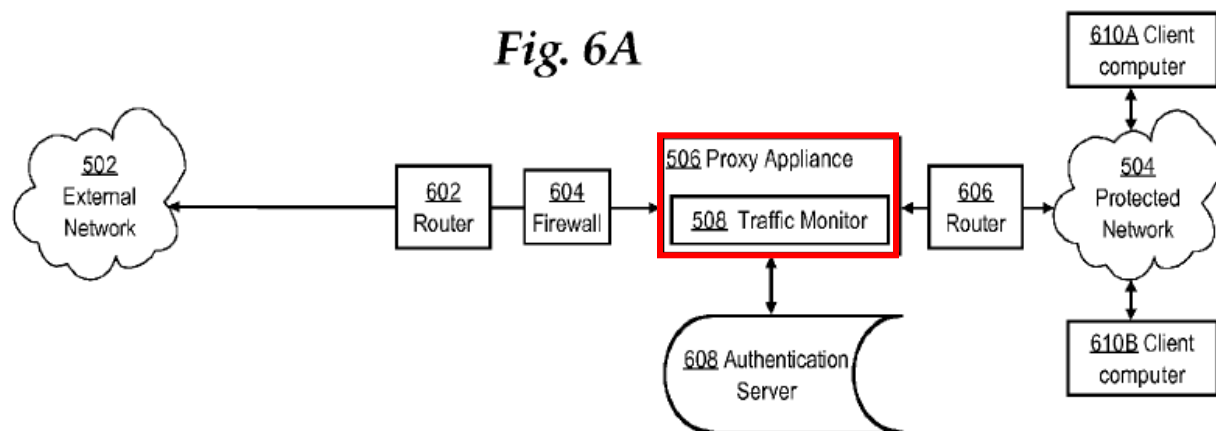
264. In the Austin-Bloch combined system, Austin’s “isolation environment 350” (“*isolated computing environment*”) would be authenticated via Bloch’s “authentication server 608” (“*authentication device*”).

10. [1I] and [9G] “[send/sending] data to an untrusted destination from the multi-user interactive software application via a proxy device when the isolated computing environment has been authenticated.”

265. The Austin-AAPA-Bloch combination discloses this element. Austin recognizes that untrusted tasks (i.e., tasks to isolate) may include forms of web browsing or executing untrusted applications. EX1005, ¶72. For example, a web

browser may be isolated when instructed to access (“*send data to*”) an “untrusted website” (“*an untrusted destination*”). EX1005, ¶73. For example, “[a] trusted Web browser (e.g., Internet Explorer) may be instructed by the user to view an untrusted website.” EX1005, ¶73. WebEx may be used within a web browser. EX1018, 1. GoToMeeting can also be used from within a web browser. EX1024, 12, 131, 204.

266. In the Austin-AAPA-Bloch system, traffic from Austin’s computer device (i.e., WebEx traffic) would pass through Bloch’s proxy appliance 506. EX1007, 7:66-8:8, 8:39-44. Bloch’s proxy appliance is configured to inspect and selectively allow network traffic between client devices and, for example, an HTTP server (i.e., a website). EX1007, 5:22-24, 5:36-37, 5:49-50.



EX1007, FIG. 6A (annotated).

267. A POSA would have been motivated, with a reasonable expectation of success, to implement Austin-AAPA computer device on a network including Bloch’s proxy appliance in order to securely access WebEx content on Austin’s

computing device. Austin discloses communicating over a network via the computer device by, for example, accessing a website via a web browser and accessing attachments received via email. EX1005, ¶8. WebEx was used within a web browser. EX1018, 1; EX1033, 2. GoToMeeting was also used within a web browser. EX1024, 12, 131, 204. Similarly, Bloch discloses client computers “coupled to protected network 504.” EX1007, 8:26-27. Bloch further discloses a proxy appliance configured to receive and inspect “all network traffic.” EX1007, 8:5-8. Moreover, this combination would have been obvious because it would amount to nothing more than combining familiar elements according to their known methods of use.

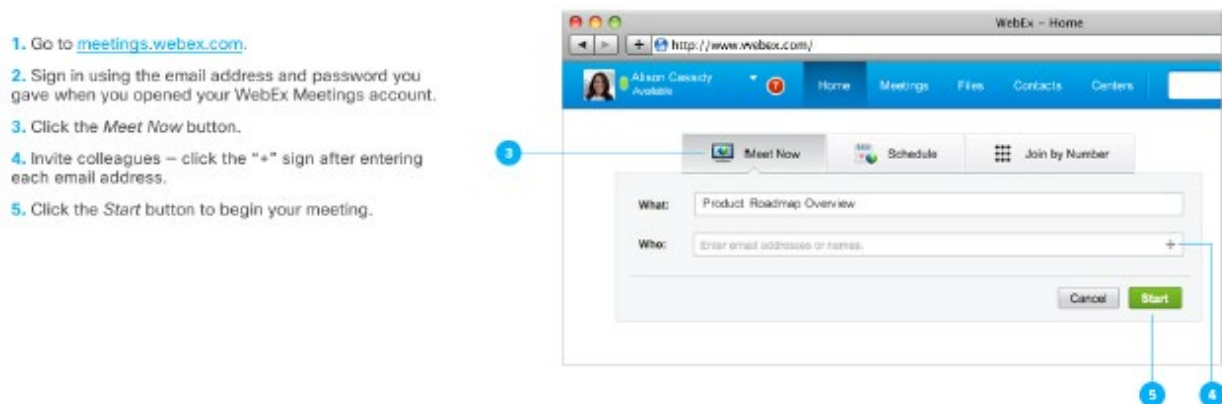
268. In the Austin-AAPA-Bloch combined system, an isolated web browser would access (“*send data to*”) an untrusted website (“*an untrusted destination*”), such as WebEx (“*a multi-user interactive software application*”), via Bloch’s “proxy appliance 506” (“*proxy device*”).

D. Claims 2 and 10

269. Claims 2 and 10 depend from claims 1 and 9, respectively, and recite “*wherein the multi-user interactive software application allows a remote machine to control an application or a process of the second set of one or more applications or processes running in the isolated computing environment, wherein controlling the application or process comprises at least one of viewing, interacting with,*

modifying, uploading data to, or downloading data from the application or the process.” The Austin-AAPA-Bloch combination discloses this element.

270. Again, Austin discloses that “untrusted tasks may include certain forms of Web browsing” EX1005, ¶72. For example, “[a] trusted Web browser (e.g. Internet Explorer) may be instructed by the user to view an untrusted website.” EX1005, ¶73. The AAPA discloses that the multi-user interactive software application is implemented as, for example, WebEx or GoToMeeting. As shown below, WebEx can be used within a browser. EX1018, 1. GoToMeeting was also used within a browser. EX1024, 12, 131, 204.



EX1018, 1.

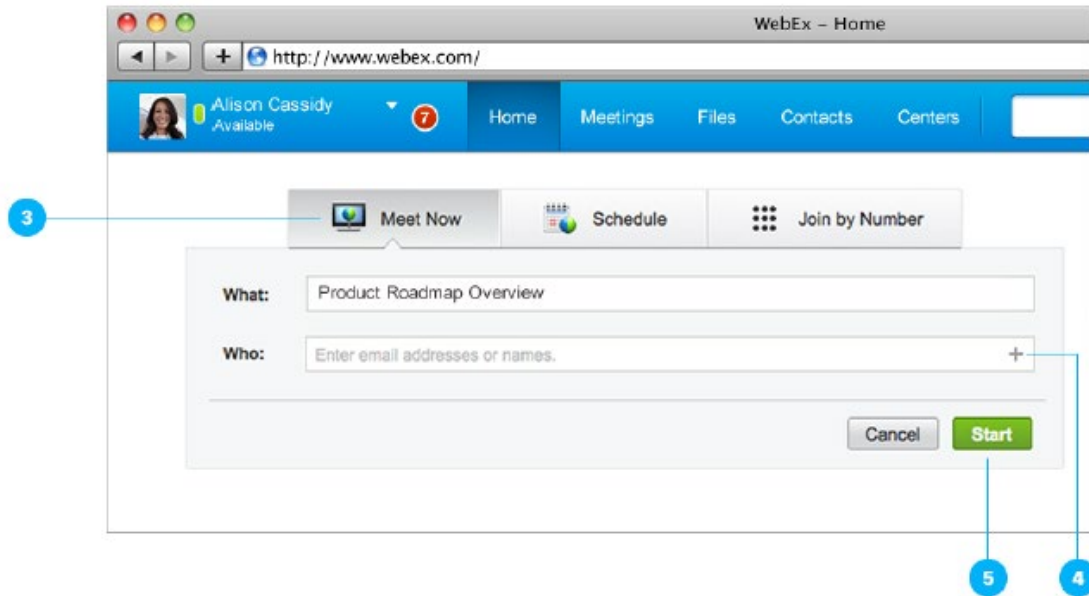
271. Bloch discloses “detection techniques” for when a client is “seeking to retrieve information from an HTTP server” (i.e., web browsing). EX1007, 5:19-24. If the IP address of the requested server is in a whitelist, the request is allowed and the server’s response is provided to the client. EX1007, 5:46-50.

272. A POSA would have understood that providing an HTTP server's response to a client's web browser discloses "*at least one of viewing, interacting with, modifying, uploading data to, or downloading data from the application or the process.*"

E. Claims 3 and 11

273. Claims 3 and 11 depend from claims 1 and 9, respectively, and recite "*wherein the multi-user interactive software application allows a remote device to control one or more of a pointer device, a keyboard input, or a display of the system.*" The Austin-AAPA-Bloch combination discloses this limitation.

274. Again, the '601 patent admits that WebEx discloses the "multi-user interactive software application" element. EX1001, 4:45-47. WebEx would be used on Austin's client computer, either within a browser or as a standalone application. For example, shown below, WebEx is accessed from a web browser.



EX1018, 1.

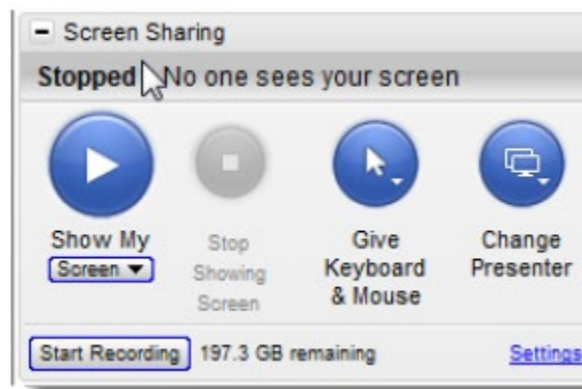
275. GoToMeeting can also be used within a browser. EX1024, 12, 131, 204.

276. A POSA would understand that tools, such as WebEx, allow “a remote device to control . . . a display of the system.” During a WebEx meeting, a presenter (“a remote device”) can share content with participants. For example, the presenter can share their desktop, an application, or a document. EX1011, 0001-02.

277. While sharing, the presenter can control what the other participants see. For example, the presenter can synchronize views of a participant (“control . . . a display of the system”), so the participant’s view is matched to the presenter’s. EX1011, 0001-02. A POSA would have understood that synchronizing the display of a participant controls the display of a system.

278. Noted above, GoToMeeting could also be used within a browser.

EX1024, 12, 131, 204. GoToMeeting also allows “a remote device to control one or more of a pointer device, a keyboard input, or a display of the system.” As I explained above, the Screen Sharing Pane in GoToMeeting is used to: (1) start or pause screen sharing; (2) select which application window or screen to share; (3) give keyboard and mouse control to another meeting participant; and (4) make another meeting participant the presenter. EX1024, 93.



EX1024, 93.

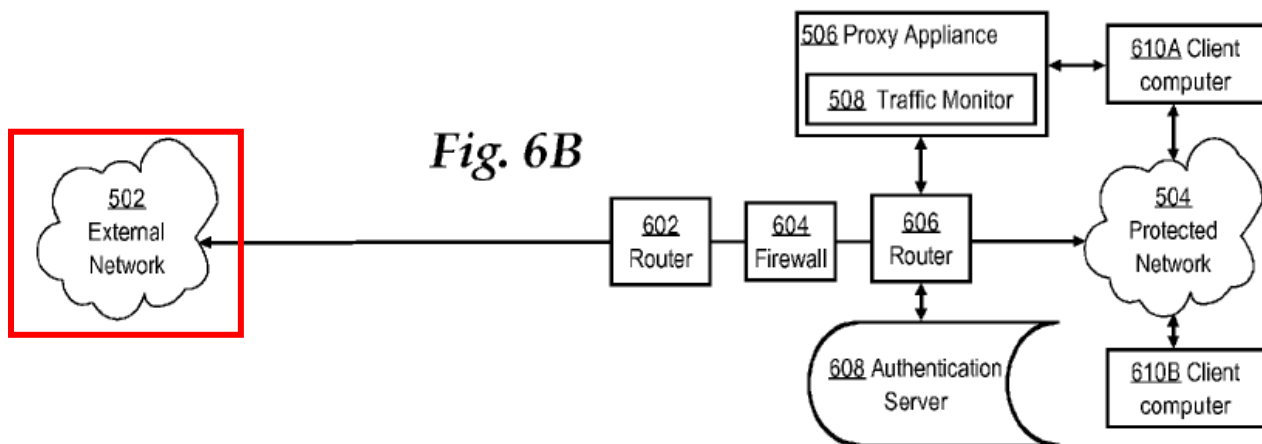
279. In the combined Austin-AAPA-Bloch system, Austin’s web browser that runs a WebEx meeting (“multi-user interactive software application”) would enable a presenter (“a remote device”) to share and control content displayed on the client device (“control . . . a display of the [system/isolated computing environment]”).

F. Claims 4 and 12

280. Claims 4 and 12 depend from claims 1 and 9, respectively, and recite

“[wherein the system is connected/connecting] to a network, the network corresponding to one or more of a local area network (LAN), a wide area network (WAN), a cellular network, a Wi-Fi network, a digital subscribers line (DSL) network, a cloud computing network, a data center, or a satellite network.” The Austin-AAPA-Bloch combination discloses this limitation.

281. Again, Austin discloses that “untrusted tasks may include certain forms of Web browsing.” EX1005, ¶72. Bloch discloses an external network that may include the internet. EX1007, 8:1-3. A POSA would have understood that the internet is a WAN, comprising a collection of LANs. *See* EX1020, 0041 (“an internet is made of up several links (LANs and WANs)”).



EX1007, FIG. 6B (annotated).

G. Claims 7 and 15

282. Claims 7 and 15 depend from claims 4 and 12, respectively, and recite

“[wherein the processor is further configured to: run/running] one or more trusted applications in the first memory space, and wherein the one or more trusted applications communicate with one or more trusted machines via the network.”

The Austin-AAPA-Bloch combination discloses this limitation.

283. As noted above, Austin executes a plurality of processes such as a trusted web browser (*“[run/running] one or more trusted applications”*). EX1005, ¶73. These processes execute under “primary user account 121.” EX1005, ¶¶102, 119.

284. Austin also recognizes that tasks related to trusted content may function without intervention (i.e., isolation). EX1005, ¶78. For example, a web browser may be used to access a webpage. EX1005, ¶73. Bloch discloses “detection techniques” for when a client is “seeking to retrieve information from an HTTP server” (i.e., web browsing). EX1007, 5:19-24. If the IP address of the requested server is in a whitelist (*“one or more trusted machines”*), the request is allowed and the server’s response is provided to the client (*“communicate . . . via the network”*). EX1007, 5:46-50.

285. Thus, in the combined Austin-AAPA-Bloch system, Austin’s computing device would execute a browser (*“[run/running] one or more trusted applications”*) under the namespace of the primary user account (*“in the first memory space”*) to retrieve content from a webpage (*“communicate . . . via the*

network”) hosted by an HTTP server (“*a trusted machine*”).

XIV. GROUND 5: Claims 5, 6, 13, And 14 Are Obvious Over Austin, AAPA, Bloch, and Walker.

A. A POSA would have been motivated to combine Austin, AAPA, and Bloch with Walker based on Austin’s disclosure.

286. Austin discloses embodiments that use the Windows operating system. EX1005, ¶64. For example, Austin states that the security module may be “a security sub-system or security manager” that enacts a Windows security model. EX1005, ¶65. To the extent that Austin does not describe implementation details or functionality of the Windows security model, a POSA would have been motivated to combine the Austin-AAPA-Bloch system with a reference, such as Walker, to identify features of Windows security.

287. Walker is directed to managing a host-based firewall and describes the Windows Filtering Platform within the Windows operating system. EX1008, ¶¶1, 3. For example, Walker discloses that the Windows Filtering Platform is a firewall built into the Windows Operating System. EX1008, ¶70. The firewall is used to “control a flow of information” to and from a device. EX1008, ¶66. The firewall controls information by extracting data from a packet, and comparing the extracted data to a set of rules. EX1008, ¶142. Rules may be based on data from the network layer, transport layer, session layer, presentation layer, or application layer. EX1008, ¶142. For example, a transport layer rule may block packets including the

IP address of a specific website. EX1008, ¶143.

288. A POSA would have had a reasonable expectation of success using Walker's firewall within Austin's computer device because both Austin and Walker discuss protecting a computer using the Windows operating system. This combination would have therefore amounted to nothing more than combining familiar elements according to their known methods of use.

B. Claims 5 and 13

289. Claims 5 and 13 depend from claims 4 and 12, respectively, and recite “[*wherein the processor is further configured to: implement/implementing*] a *host-based firewall, wherein the host-based firewall isolates the [system/first memory space and the second memory space]* from communicating with one or more other devices on the network.” The Austin-Bloch-Walker combination discloses this limitation.

290. Walker discloses that the Windows operating system includes a built-in firewall known as the “Windows Filtering Platform” (“*a host-based firewall*”). EX1008, ¶4. A firewall regulates incoming and outgoing traffic based on a rule set (“*isolates the [system/first memory space and the second memory space]* from communicating”). EX1008, ¶3. Rules may be based on information extracted from a received packet, such as network layer transport layer, session layer, presentation layer, and/or application layer information. EX1008, ¶142. For example, an

incoming packet may be allowed through the firewall based on the source IP address (“*one or more other devices on the network*”) in the packet. EX1008, ¶81.

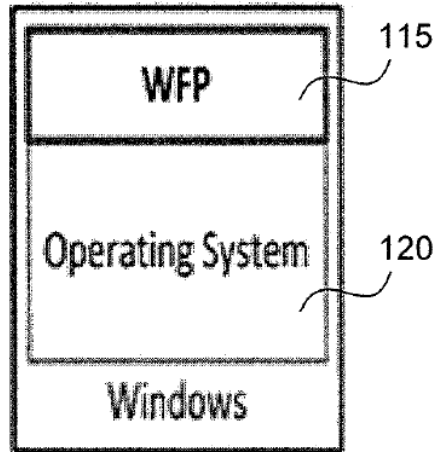


Fig. 1b

EX1008, FIG. 1B.

291. In the Austin-Bloch-Walker combined system, Austin’s “processor” (“*the processor*”) would implement Walker’s Windows Filtering Platform (“*host-based firewall*”) that regulates incoming and outgoing traffic (“*isolates the [system/first memory space and the second memory space] from communicating*”) based on a packet’s source IP address (“*one or more other devices on the network*”).

C. Claims 6 and 14

292. Claims 6 and 14 depend from claims 5 and 13, respectively, and recite “*wherein the host-based firewall blocks incoming communications from the*

network that are sent to the workspace.” The Austin-Bloch-Walker combination discloses this limitation.

293. The ’601 patent explains that the host-based firewall blocks workspace communications based on an application within the workspace, the network device sending the communication, and/or the network port used for the communication. EX1001, 6:52-60.

294. As noted above, Austin’s primary user account (“*the workspace*”) supports execution of a plurality of processes (i.e., applications). EX1005, ¶63. Additionally, Walker discloses a firewall (“*the host-based firewall*”) regulating incoming traffic (“*block[ing] incoming communications from the networks*”) based on a rule set. EX1008, ¶3. The rules may be based on information extracted from a network packet. EX1008, ¶142. A rule may be based on the application associated with the packet (“an application in the workspace”), a source IP address of the packet (“the network device”), and/or a network port (“the port”) specified in the packet. EX1008, ¶¶142-144.

295. For example, a rule can be defined to block traffic “at IP address 1.1.1.1 on port 80.” EX1008, ¶143. Similarly, a rule can be defined to block traffic “destined for a particular application running on the host computing device.” EX1008, ¶144.

296. In the Austin-Bloch-Walker combined system, Austin’s computer

device would implement Walker’s Windows Filtering Platform (“*host-based firewall*”) that regulates incoming and outgoing traffic (“*block[]incoming communications from the network*”) based on an application executing in the workspace, the device sending the communication, or the port used for the communication (“*sent to the workspace*”).

XV. GROUND 6: Claims 8 And 16 Are Obvious Over Austin, AAPA, Bloch, and Franco

A. A POSA would have been motivated to combine Austin, AAPA, and Bloch with Franco based on Austin’s disclosure.

297. As noted above, Austin discloses a “primary clipboard . . . accessible from a primary user account,” and “a secondary clipboard . . . associated with the secondary user account.” EX1005, ¶20. In some instances, when the user initiates requests to access the clipboard, Austin discloses that, the clipboard may provide feedback to the user. For example when the computer is first started and the user attempts to paste data, the user may receive an indication when there is no data available. EX1005, ¶134. Austin also recognizes that data may be transferred between isolated and non-isolated applications via the primary and secondary clipboards. For example, data may be transferred from a non-isolated application to an isolated application, but not from an isolated application to a non-isolated application. EX1005, ¶134. Austin references “additional rules” governing when content may be transferred between the isolated and non-isolated processes

EX1005, ¶137. To the extent Austin does not disclose implementation details about these “additional rules,” a POSA would have been motivated to combine the Austin-AAPA-Bloch system with a reference like Franco.

298. Similar to Austin, Franco is directed at preventing malware or other malicious software from compromising a host computer. EX1009, 1:22-29. To address this issue, Franco discloses executing applications with “reduced privilege” so that they are restricted from accessing portions of the host. EX1009, 2:31-36. The restricted applications are prevented from accessing administrative and user data of the host. EX1009, 1:47-51. As a result, restricted applications can only access special portions of the disk labelled containment zones. EX1009, 1:51-52. When attempts to access a restricted area (e.g., administrative area, user area) of the host are made, Franco discloses capturing explicit user input to authorize the access. EX1009, 2:36-56. For example, when the restricted area is accessed, a broker object prompts the user to interact with a dialog box or input credentials to allow the access attempt. EX1009, 5:4-11.

299. A POSA would have been motivated, with a reasonable expectation of success, to use Franco’s user-prompt mechanism within Austin’s client because it would amount to nothing more than combining familiar elements according to their known methods of use. Both Austin and Franco disclose isolating processes within a computing device. Austin discloses a security module to restrict isolated

applications, and “additional rules” to facilitate communication between isolated and non-isolated applications. EX1005, ¶137. Austin also discusses providing feedback to the user (i.e., indicating there is no data available to paste). EX1005, ¶134. Franco discloses, for example, prompting a user to facilitate communication between an application and a restricted area of the computing device. In the combined system, Austin’s security module would require Franco’s user input when data is transferred between isolated and non-isolated processes.

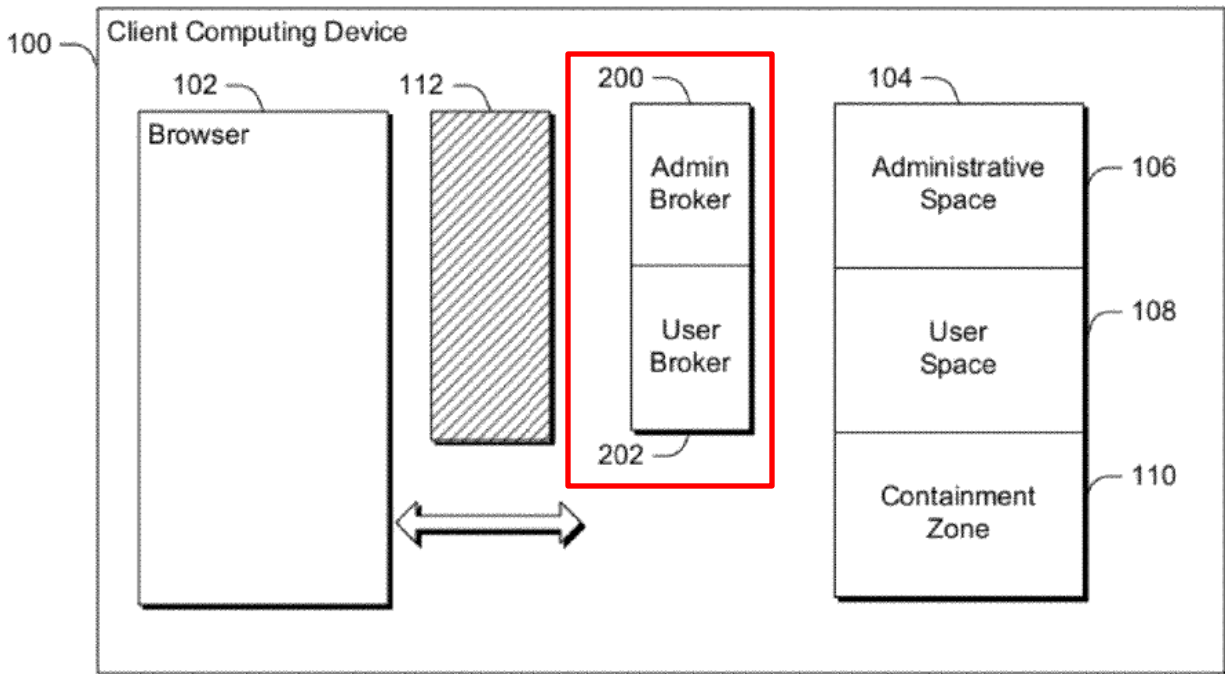
B. Claims 8 and 16

300. Claims 8 and 16 depend from claims 1 and 9, respectively, and recite “*wherein the internal isolation firewall prompts a user of the [system/workspace] to allow communication between the first memory space and the second memory space.*” The Austin-Bloch-Franco combination discloses this limitation.

301. As noted above, Austin’s primary user account has its own “namespace” (“*a first memory space*”). EX1005, ¶95. The primary user account has access to a “primary clipboard. EX1005, ¶20. Similarly, the namespace of the “temporary user account” (“*a second memory space*”) has its own “secondary clipboard.” EX1005, ¶20. Each clipboard is further supported by its own separate physical memory space. EX1005, ¶119. Austin’s “security module” (“*the internal isolation firewall*”) isolates tasks associated with the secondary user account from the primary user account. EX1005, ¶10.

302. The user (“a user”) initiates requests to access the clipboard. EX1005, ¶135. Either the primary or the secondary clipboard is used to transfer data between the isolated and non-isolated applications (“*communication between the first memory space and the second memory space*”). EX1005, ¶¶140-141.

303. Franco discloses preventing applications from accessing restricted areas of a computer and requiring explicit user input when access attempts are made. EX1009, 2:36-56. For example, if a user is attempting to save a file within the My Documents folder, the broker object displays a dialog box prompting (“*prompts a user of the [system/workspace]*”) the user for input (“*to allow communication*”). EX1009, 5:4-7. Similarly, if a process attempts to access the administrative space of the host, the broker object prompts the user (“*prompts a user of the [system/workspace]*”) for administrator credentials to facilitate the request (“*to allow communication*”). EX1009, 5:8-11.



EX1009, FIG. 2 (Annotated).

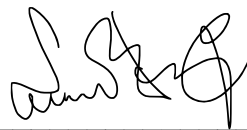
304. In the Austin-Bloch-Franco system, Austin’s “security module” (“*internal isolation firewall*”) would display Franco’s dialog box (“*prompt a user of the [system/workspace]*”) when content is transferred between the isolated and non-isolated applications (“*to allow communication between the first memory space and the second memory space*”).

XVI. CONCLUSION

305. In signing this declaration, I recognize that the declaration will be filed as evidence in an *inter partes* review before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I also recognize that I may be subject to cross-examination in the case and that cross-examination will take place within the United States. If cross-examination is required, I will appear for cross-examination within the United States during the time allotted.

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

Executed on this 14th day of September, 2024, in Silver Spring, Maryland.



Samrat Bhattacharjee, Ph.D.