

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CISCO SYSTEMS INC.,

Petitioner

IPR2024-01283

U.S. Patent No. 11,223,601

**DECLARATION OF PRASHANT SHENOY, PH.D.,
UNDER 37 C.F.R. § 1.68 IN SUPPORT OF PETITION
FOR *INTER PARTES* REVIEW**

TABLE OF CONTENTS

I.	INTRODUCTION	4
II.	QUALIFICATIONS AND PROFESSIONAL EXPERIENCE	7
III.	LEVEL OF ORDINARY SKILL IN THE ART	11
IV.	RELEVANT LEGAL STANDARDS	12
V.	BACKGROUND	14
VI.	OVERVIEW OF THE '601 PATENT	16
VII.	CLAIM CONSTRUCTION	18
VIII.	IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE....	18
IX.	GROUND 1: THE COMBINATION OF JEFFRIES, LABINE, AND ISHAYA RENDER OBVIOUS CLAIMS 1-4, 7, 9-12, AND 15.....	19
A.	Summary of Jeffries	19
B.	Summary of LaBine	23
C.	Summary of Ishaya.....	26
D.	Reasons to Combine Jeffries and LaBine	31
E.	Reasons to Combine Jeffries and Ishaya.....	37
F.	Reasons to Combine Jeffries, LaBine, and Ishaya.....	43
G.	Detailed Analysis of Claims.....	46
1.	Claim 1	46
2.	Claim 2	89
3.	Claim 3	96
4.	Claim 4	97
5.	Claim 7	98
6.	Claim 9	102
7.	Claim 10	104
8.	Claim 11	105
9.	Claim 12	107

10.	Claim 15.....	108
X.	GROUND 2: THE COMBINATION OF JEFFRIES, LABINE, ISHAYA, AND CUCINOTTA RENDERS OBVIOUS CLAIMS 5-6 AND 13-14....	108
A.	Summary of Cucinotta	108
B.	Reasons to Combine Jeffries, LaBine, Ishaya, and Cucinotta	110
C.	Detailed Analysis of Claims.....	114
1.	Claim 5.....	114
2.	Claim 6.....	117
3.	Claim 13.....	118
4.	Claim 14.....	120
XI.	GROUND 3: THE COMBINATION OF JEFFRIES, LABINE, ISHAYA, LEVY RENDERS OBVIOUS CLAIMS 8 AND 16.....	121
A.	Summary of Levy	121
B.	Reasons to Combine Jeffries, LaBine, Ishaya, and Levy.....	122
C.	Detailed Analysis of Claims.....	126
1.	Claim 8.....	126
2.	Claim 16.....	129
XII.	CONCLUSION.....	130

I, Prashant Shenoy, Ph.D., do hereby declare as follows:

I. INTRODUCTION

1. I am making this declaration at the request of Cisco Systems, Inc. in the matter of the *Inter Partes* Review of U.S. Patent 11,223,601 (“’601 patent”) to Coleman, et al.

2. I am being compensated for my work in this matter at my standard hourly rate. I am also being reimbursed for reasonable and customary expenses associated with my work and testimony in this proceeding. My compensation is not contingent on the outcome of this matter or the specifics of my testimony.

3. I have been asked to provide my opinions regarding whether the subject matter of claims 1-16 (the “Challenged Claims”) of the ’601 patent would have been obvious to a person having ordinary skill in the art (“POSITA”) at the time of the alleged invention, in light of the prior art. It is my opinion that the Challenged Claims would have been obvious to a POSITA.

4. In the preparation of this declaration, I have studied:

- a. the ’601 patent, Ex.1001;
- b. the file history of the ’601 patent (“’601 File History”), Ex.1002;
- c. U.S. Patent No. 10,885,189 to Jeffries, et al. (“Jeffries”), Ex.1005;

d. U.S. Patent Application Publication 2009/0292999 to LaBine, et al. (“LaBine”), Ex.1006;

e. U.S. Patent Application Publication 2014/0282889 to Ishaya, et al. (“Ishaya”), Ex.1007;

f. U.S. Patent Publication No. 2016/0048406 to Cucinotta, et al. (“Cucinotta”), Ex.1008; and

g. U.S. Patent Application Publication 2007/0136579 to Levy, et al. (“Levy”), Ex.1009.

5. In forming the opinions expressed below, I have considered: the documents listed above; the relevant legal standards, including the standard for obviousness; and my own knowledge and experience based upon my work in the field of network communications and security as described below, as well as portions of the following additional materials:

a. U.S. Patent No. 6,167,432 to Jiang (“Jiang”), Ex.1010;

b. Prevelakis, Sandboxing Applications (“Prevelakis”), Ex.1011;

c. Goldberg, A Secure Environment for Untrusted Helper Applications (Confining the Wily Hacker), Proceedings of the Sixth USENIX UNIX Security Symposium (July 1996) (“Goldberg”), Ex.1012;

d. U.S. Patent No. 5,974,549 to Golan (“Golan”), Ex.1013;

e. U.S. Patent No. 5,987,611 to Freund (“Freund”), Ex.1014;

- f. U.S. Patent Publication 2002/0178375 to Whittaker, et al.
 (“Whittaker”), Ex.1015;
- g. U.S. Patent No. 6,292,874 to Barnett (“Barnett”), Ex.1016;
- h. U.S. Patent No. 5,067,072 to Talati, et al. (“Talati”), Ex.1017;
- i. U.S. Patent Publication 2006/0112384 to Frank, et al. (“Frank”),
 Ex.1018;
- j. U.S. Patent No. 8,326,872 to Zwilling, et al. (“Zwilling”), Ex.1019;
- k. U.S. Patent Publication 2009/0259720 to Heins, et al. (“Heins”),
 Ex.1020;
- l. U.S. Patent No. 6,226,677 to Slemmer (“Slemmer”), Ex.1021;
- m. U.S. Patent Publication 2013/0191880 to Conlan, et al. (“Conlan”),
 Ex.1022;
- n. U.S. Patent Publication 2013/0097706 to Titonis, et al. (“Titonis”),
 Ex.1023;
- o. U.S. Patent Publication 2015/0113544 to Yu (“Yu”), Ex.1024;
- p. U.S. Patent Publication 2009/0225767 to Song, et al. (“Song”),
 Ex.1025;
- q. U.S. Patent Publication 2005/0223118 to Tucker, et al. (“Tucker”),
 Ex.1026;

- r. U.S. Patent Publication 2002/0010800 to Riley, et al. (“Riley”), Ex.1027;
- s. U.S. Statutory Invention Registration H1944 H to Cheswick, et al. (“Cheswick”), Ex.1028;
- t. U.S. Patent Publication 2011/0078550 to Nabutovsky (“Nabutovsky”), Ex.1029;
- u. Azab, SICE: A Hardware-Level Strongly Isolated Computing Environment for x86 Multi-core Platforms (October 2011), (“Azab”), Ex.1034;
- v. U.S. Patent Publication No. 2016/0088155 to Patel (“Patel”), Ex.1035;
- w. U.S. Patent Publication No. 2013/0110999 to Kopasz, et al. (“Kopasz”), Ex.1036; and
- x. U.S. Patent Publication No. 2016/0381072 to Todd, et al. (“Todd”), Ex.1037.

6. Unless otherwise noted, all emphasis in any quoted material has been added. Claim terms are italicized.

II. QUALIFICATIONS AND PROFESSIONAL EXPERIENCE

7. My qualifications for forming the opinions set forth in this declaration are summarized here and explained in more detail in my curriculum vitae, which is

attached as Exhibit 1004. Exhibit 1004 also includes a list of my publications and the cases in which I have testified at deposition, hearing, or trial during the past seven years.

8. I am a tenured professor in the College of Information and Computer Sciences at the University of Massachusetts Amherst ("UMass"), where I currently hold the position of Distinguished Professor of Computer Science. I am also an Associate Dean in the College of Information and Computer Sciences. I currently work in both these capacities. I have been a faculty member at UMass since 1998.

9. I received my Bachelor of Technology (B.Tech.) degree in 1993 from the Department of Computer Science and Engineering at The Indian Institute of Technology ("IIT"), Mumbai, India. IIT awarded me the Institute Medal for being the top ranking undergraduate student. In 1994, I received my Master of Science (M.S.) degree in Computer Sciences from The University of Texas at Austin. In 1998, I received my Doctor of Philosophy (Ph.D.) also from the Department of Computer Sciences at The University of Texas at Austin. My dissertation, which was in the area of file systems, was recognized by the department as the Best Doctoral Dissertation of 1998-1999.

10. I have been involved in the research and development of distributed computing systems for more than 30 years. I have been a faculty member at UMass since 1998 where I perform research and teach in the areas of distributed systems,

operating systems, computer networking, data management, and multimedia systems, among others.

11. A major focus of my research has been the design of large-scale distributed computing systems. Over the years, I have led research and industry collaboration projects in areas such as computer networking, distributed systems, storage and file systems, Web and Internet systems, content distribution, streaming media, database systems, cloud computing, virtualization, RFID and sensor networks, and energy systems. During that time, I have consulted with both large companies and startups in a technical advisory role related to product architecture, IP issues, and applied research.

12. For example, starting in the early 1990s, my research focused on the development of storage and file systems. During that time, I developed multimedia file system and data management technologies that were designed for efficient storage and retrieval of large amounts of video and audio data. I collaborated with AT&T Bell Laboratories to design a new media server for video streaming. These efforts led to resulted in U.S. Patent No. 6,079,028 on fault-tolerant architectures for continuous media servers.

13. In the late 1990s, I led the design of a new operating system called QLinux, that was based on the open-source Linux operating system. As part of this research, I developed resource management and scheduling techniques that provide

isolation and service guarantees to applications. This technology provided the underpinnings for fair resource management and isolation techniques used in OS virtualization. It was adopted by a startup company for providing resource isolation in its web hosting products.

14. In the early 2000s, I led several projects on web systems and cloud computing. For example, I led research projects on web technologies such as web caching and scale web applications. I also led research projects on cloud computing platforms (which were known as web hosting platforms at the time) involving virtualization, dynamic resource management, replication, and storage. My group at UMass collaborated with technology companies such as IBM, AT&T, Sprint, Intel, and many others as part of this research.

15. My research on virtualization has involved design of technologies for virtual machines and container resource management and their deployment in cloud computing systems and data centers. I am co-inventor on US patent 8,473,557, jointly assigned to AT&T and UMass, on virtual machine migration.

16. Over the years, I have authored or co-authored more than 300 articles in peer-reviewed journals and conference proceedings that have been cited over 25,000 times. Seventeen papers authored by myself and members of my research group on this project won awards for their technical contributions. Two papers have won Test-of-Time awards, an award given to a publication that has had the most

impact over a ten-year period since its publication.

17. I have served as an Editor for six leading journals and have served as the Technical Program Chair for twenty conferences and workshops in my field. I founded the ACM Special Interest Group on Energy Systems and also served on steering committee of conferences and journals.

18. I am also a Fellow of the ACM, IEEE, AAAS, and AAIA, four professional societies in computing and science.

19. I am a named inventor on four U.S. patents (U.S. Patent Nos. 10,152,398; 8,510,596; 8,473,557; and 6,079,028). These patents are generally related to distributed and operating systems, virtualization, and cloud data centers. As a faculty member at UMass, I have also taught undergraduate and graduate courses in operating systems, distributed systems, networking, and related topics for more than twenty years. These courses teach concepts related to operating systems, virtual machines, containers, cloud computing, and distributed systems. They also involve hands-on course projects on these topics.

III. LEVEL OF ORDINARY SKILL IN THE ART

20. I understand there are multiple factors relevant to determining the level of ordinary skill in the pertinent art, including (1) the levels of education and experience of persons working in the field at the time of the invention; (2) the sophistication of the technology; (3) the types of problems encountered in the field;

and (4) the prior art solutions to those problems.

21. A POSITA in the field of the '601 patent, as of its priority date of September 28, 2017, would have been someone knowledgeable and familiar with network security and application isolation techniques as those are applied in the context of network security available in the 2010s. Such a POSITA would have a bachelor's degree in computer science, computer engineering, electrical engineering, or equivalent training, and approximately two years of experience working in the field of network security. Additional work experience can substitute for specific educational background, and vice versa.

22. For purposes of this Declaration, in general, and unless otherwise noted, my statements and opinions, such as those regarding my own experience and what a POSITA would have understood or known generally (and specifically related to the references I consulted herein), reflect the knowledge that existed in the relevant field as of the priority date of the '601 patent.

IV. RELEVANT LEGAL STANDARDS

23. I am not an attorney. In preparing and expressing my opinions and considering the subject matter of the '601 patent, I am relying on certain basic legal principles that Cisco's counsel has explained to me.

24. I understand that prior art to the '601 patent includes patents and printed publications in the relevant art that predate the priority date of the '601 patent. For

purposes of this Declaration, I am applying September 28, 2017, as the priority date of the '601 patent.

25. I have been informed by Cisco's counsel that a claimed invention is unpatentable under 35 U.S.C. § 103 if the differences between the claimed invention and the prior art are such that the subject matter as a whole would have been obvious to a POSITA at the time of the invention. I have also been informed by Cisco's counsel that the obviousness analysis considers factual inquiries, including the level of ordinary skill in the art, the scope and content of the prior art, and the differences between the prior art and the claimed subject matter.

26. I have been further informed by Cisco's counsel that there are several recognized rationales for combining references or modifying a reference to show obviousness. These rationales include: (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 a known technique to improve a similar device (method, or product) 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) some teaching, suggestion, or motivation in the prior art that would have led a POSITA to modify the prior art or to combine prior art teachings to arrive at the claimed invention.

V. BACKGROUND

27. The '601 patent claims *a multi-user interactive software application* executed within *an isolated computing environment*. '601 patent, 22:49-55. Neither of these concepts were novel at the priority date of the '601 patent.

28. The specification acknowledges that a number of *multi-user interactive software application[s]* were “commercially available” before the priority date, including “WebEx.” '601 patent, 4:47. WebEx is a video conferencing application that allows users at remote locations to communicate and collaborate over the internet. Jiang, Abstract. The technologies underlying WebEx date from at least as early as 1996, more than 20 years before the '601 patent's priority date. See Jiang, [22], [73] (application filed February 29, 1996, originally assigned to WebEx Communications, Inc.).

29. The '601 patent proposes executing a *multi-user interactive software application* in an *isolated computing environment*, also describe as a “sandboxed computing environment.” '601 patent, 5:33-35. But executing applications in an isolated or sandboxed environment was also not new. The computer security industry has used this technique to secure computers from potential malware for decades. See, e.g., Goldberg, Abstract (published July 1996) (describing “creat[ing] a secure environment to contain untrusted helper applications” and “restricting the program's access to the operating system”); Golan, Abstract (published Oct. 26,

1999) (describing “creating a secure sandbox within which a plurality of downloaded software components can execute in a secure manner”); Freund, ¶76 (published November 16, 1999) (describing “redirect[ing] all requests to a ‘Sandbox’ server application”); Whittaker, ¶11 (published November 28, 2002) (“protecting a host computer from malicious mobile code” with “sandbox-based protection”); Patel, [0044]. By 2001, “[t]he concept of creating restricted environments for programs that are considered unsafe” was considered “by no means new.” Prevelakis, 1.

30. Further, the other ancillary features recited in the ’601 patent claims (e.g., a *workspace*, separating a *first memory space* from a *second memory space*, an *authentication device*, a *proxy device*) were also known in the art. The ’601 patent refers to a *workspace* as computing resources that support the host operating system including a *first memory space* that is isolated from a *second memory space*. See, e.g., ’601 patent, 5:19-20 (“The sandbox container process may be a security mechanism used to separate the sandboxed computing environment from other applications and/or processes that may be running on a workspace of the host computer system.”); 1:62-63 (“One or more applications may run in the workspace via a first memory space.”), 2:40-41 (“The host computer system may run one or more trusted applications in the workspace.”). Partitioning and isolating memory of a host computer into a *first memory space* and a *second memory space* is the standard

method of isolating computing environments. *See, e.g.*, Barnett, Abstract (describing “partition[ing] a homogenous memory” and “restrict[ing] access of installed applications”); Talati, 6:11 (describing a “partitioned storage area 38 in memory”).

31. In addition, the inclusion of *authentication devices* and *proxy devices* with isolated computing environments constitutes routine implementation details. A POSITA at the time would have recognized container authentication as routine. *See, e.g.*, Frank, ¶3 (“an isolated computing environment” that “uses cryptographic methods for authentication”); Zwilling, ¶18 (describing “a sandbox component 10 that includes an authentication component 215”); Heins, ¶180 (describing “‘execution environment’ authentication and execution sandbox protections”). Similarly, isolating computing environments were commonly used with *proxy devices*. *See, e.g.*, Slemmer, 3:51-64 (“selective ‘sandboxing’” via a “forced proxy server 130”); Conlan, ¶48 (“Service providers 628 may interact with sandbox manager 616 through provider proxies 619.”); Titonis, ¶296 (using a “proxy for the analysis and execution of a Sandbox Request”).

VI. OVERVIEW OF THE '601 PATENT

32. The '601 patent describes methods and systems for isolating collaboration software on a host computer system. '601 patent, 4:11-12. This isolation “prevent[s] malware from effectively exfiltrating data from the host computer system.” '601 patent, 4:36-42.

33. The collaboration software includes applications that allow one or more remote machines “to control (e.g., access, view, interact with, download from, upload to, and/or modify) an application running on [a] host computer system.” ’601 patent, 2:19-23. Examples of such applications include “WebEx, GoToMeeting, etc.” ’601 patent, 4:56-60, 4:45-47. The ’601 patent contemplates isolating other types of software as well, such as a “browser application.” ’601 patent, 19:32.

34. The claimed system executes the collaboration software in a sandboxed computing environment, which corresponds to a separate memory space that limits the software’s access to the host computer system. ’601 patent, 3:9-15, 11:65-12:3. An internal isolation firewall enforces the isolation. ’601 patent, 1:67-2:6. To further protect both the host computer system as well as the isolated computing environment, communication with untrusted sources may be routed through a web proxy. *See, e.g.*, ’601 patent, 14:46-50 (“The network 330 may send web traffic...to one or more untrusted network destinations using one or more of a border firewall 338 or a web proxy 336.”). In addition, before the isolated computing environment may communicate with untrusted sources, it may be authenticated “via an authentication device.” ’601 patent, 2:8-17, 6:14-18.

35. As I will explain below, all of these concepts were known—and their combination would have been obvious—as of the priority date of the ’601 patent.

VII. CLAIM CONSTRUCTION

36. It is my understanding that to properly evaluate the '601 patent, the terms of the claims must first be interpreted. It is my understanding that for the purposes of this *inter partes* review, the claims are to be construed under the so-called *Phillips* standard, under which claim terms are given their ordinary and customary meaning as would have been understood by a POSITA in light of the specification and prosecution history, unless the inventor has set forth a special meaning for a term. I have also been informed that claim terms only need to be construed to the extent necessary to resolve the obviousness inquiry.

37. I have reviewed the entirety of the '601 patent, as well as its prosecution history. It is my opinion that for purposes of applying the asserted prior art to evaluate the patentability of the Challenged Claims, no terms require express construction.

VIII. IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE

38. The discussion in this Declaration provides a detailed analysis of how the asserted prior art references teach each limitation of the Challenged Claims.

39. As part of my analysis, I have considered, and discuss in detail, the scope and content of the prior art and any differences between the alleged invention and the prior art.

40. It is my opinion that the alleged invention recited in the Challenged

Claims would have been obvious in view of the prior art and the knowledge of a POSITA.

IX. GROUND 1: THE COMBINATION OF JEFFRIES, LABINE, AND ISHAYA RENDER OBVIOUS CLAIMS 1-4, 7, 9-12, AND 15

A. Summary of Jeffries

41. Like the '601 patent, Jeffries is directed to isolating applications that access untrusted sources. *See* Jeffries, Abstract. In Jeffries “a computing device monitors resource access by an application running in a container that is isolated from the host operating system.” Jeffries, Abstract. As previously explained, sandboxing is another term for the isolation described in Jeffries. *See* Prevelakis, 1.

42. Jeffries' isolated containers are illustrated in Figure 1 as containers 122, highlighted below.

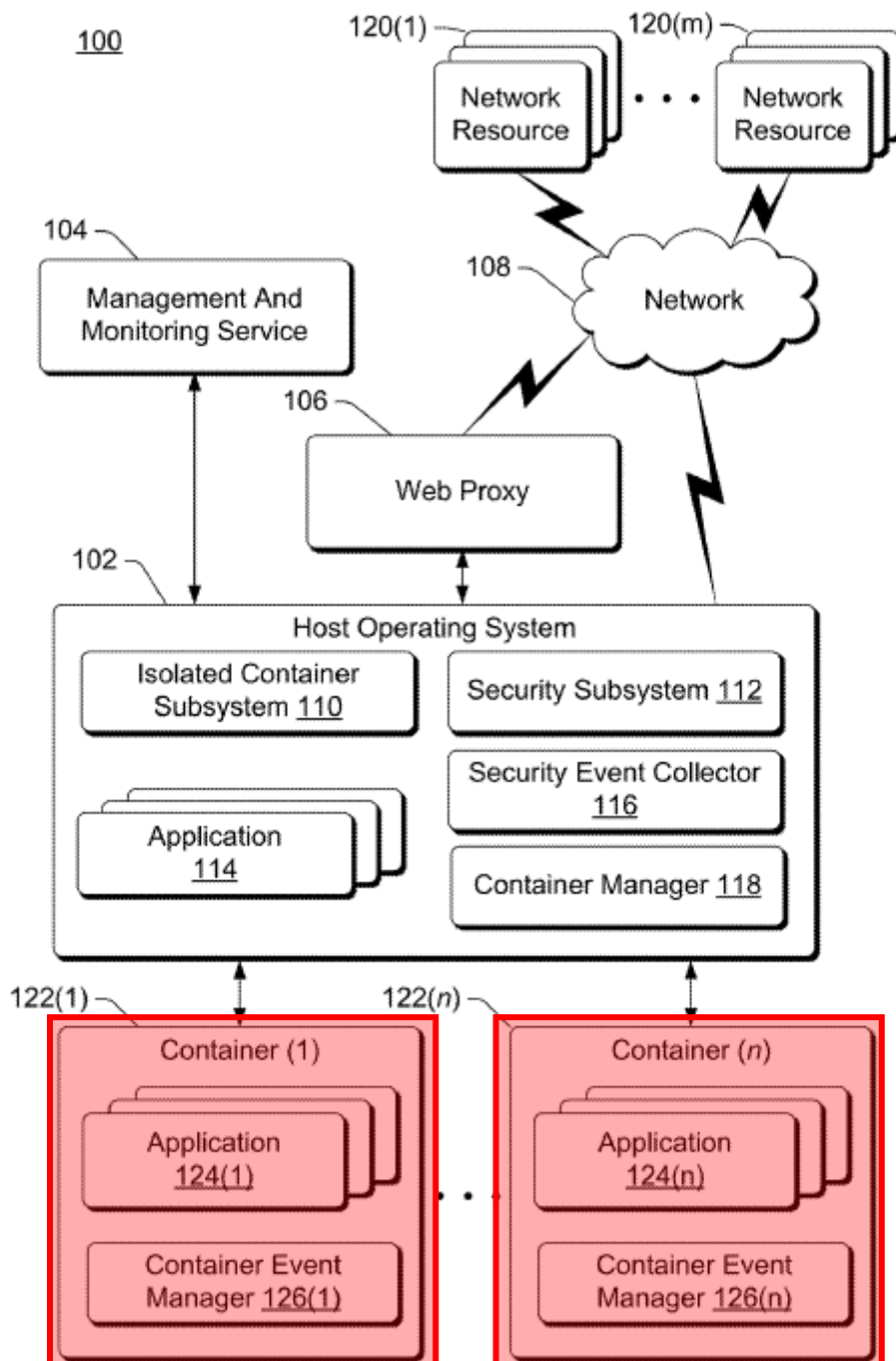


Fig. 1

Jeffries, Fig. 1 (annotated)

43. Jeffries’ isolated containers 122 “include one or more applications 124.” Jeffries, 10:55-56. These applications 124 may be “web browsers, related

applications, and operating system components” and are “isolat[ed]” to “protect[] the computing device from attacks.” Jeffries, 2:46-49.

44. Jeffries’ system includes a “host operating system 102” that isolates containers 122. Jeffries, 6:44. “The host operating system 102 also includes...one or more applications 114.” Jeffries, 8:55-57. The “[a]pplications 114 may include a web browser application...such as e-mail applications, word processing applications, spreadsheet applications, visual presentation applications, and the like.” Jeffries, 9:29-33.

45. In Jeffries’ system, the containers 122 (and applications 124 executing within them) are isolated from the host operating system by a container manager 118. For example, the “container manager 118 is configured to quarantine the container 122 and prevent the container 122 from further accessing resources. By isolating the container 122 from further accessing resources, the container manager 118 prevents malicious activity occurring in the container from gaining file system access.” Jeffries, 16:29-34. The container manager 118 enforces this isolation in conjunction with a “secure kernel”:

The communicative connection between the host operating system 102 and the container 122 implements a secure kernel to maintain integrity of information passed from the container 122 to the host operating system 102.... Accordingly, when malicious software [executing in

container 122] attempts to access off-limit data or memory locations, the secure kernel causes an exception or interrupt to occur.

Jeffries, 12:7-19.

46. Communications between applications 124 and untrusted sources may be permitted, despite potential risks, because the applications 124 are isolated within isolated containers 122, thus protecting the host system. For example, Jeffries explains that “applications 124(1),..., 124(n) are thus useable to access untrusted resources in one or more of containers 122(1),..., 122(n).” Jeffries, 10:58-61; *see also* 3:48-51 (“the host operating system considers every resource to be untrusted and forces resources to be accessed within an isolated container.”).

47. Jeffries’ system also includes a web proxy 106 positioned between the host computer (including the host operating system 102 and the isolated containers 122) and external network resources 120. *See* Jeffries, Fig. 1. The web proxy 106 “controls access to one or more network resources 120(1),...,120(m) accessed remotely from the computing device implementing system 102.” Jeffries, 6:45-54.

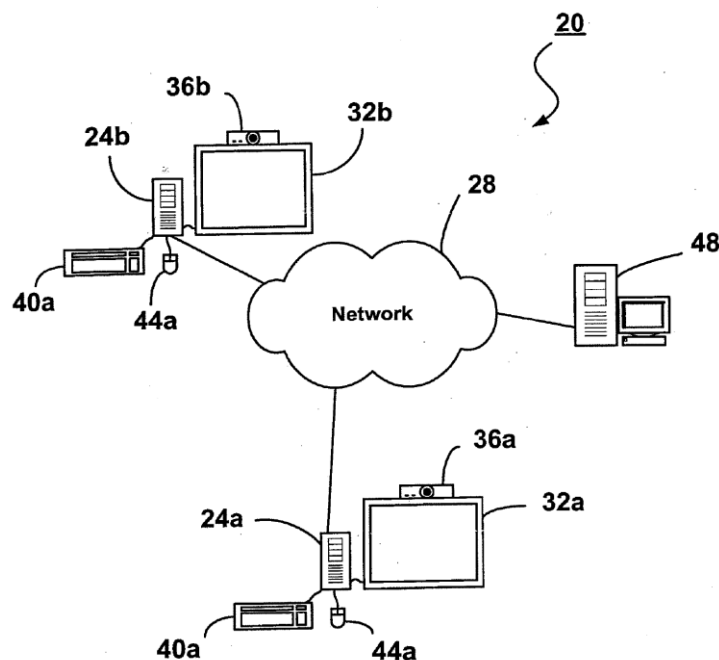
48. Jeffries is directed to the same technical field as the ’601 patent. The ’601 patent relates to “isolat[ing] applications” that “have access to one or more Internet destinations (e.g., untrusted Internet destinations).” ’601 patent, 2:10-11. Jeffries is within the same field, as it pertains to “an application running in a

container that is isolated from the host operating system,” Jeffries, Abstract, that accesses “untrusted resources,” Jeffries, 2:48-49.

49. In addition, Jeffries addresses the same problem as the ’601 patent, which is improving the security of a host computer and local intranet against potential security threats received by an application with access to untrusted sources. See ’601 patent, 2:8-11 (“One or more isolated applications may run in the isolated computed environment” which have “access to one or more Internet destinations.”); Jeffries, 1:6-7 (Background) (describing protecting against “attacks from malicious software”).

B. Summary of LaBine

50. LaBine describes the “SMART Bridgit Conferencing Software,” which is video conferencing software. LaBine, ¶45; see LaBine, ¶43, Fig. 1. In Figure 1, LaBine shows a “computing environment 20” including “a pair of computers 24a, 24b communicating over a network 28.” LaBine, ¶43.



LaBine, Fig. 1.

51. The “Bridgit Conferencing Software...allows a conference to be set up between personal computers with one personal computer at a time being designated as a host, or ‘presenter’ computer and the remaining personal computers being designated as client, or ‘participant’ computers.” LaBine, ¶45.

52. During a conferencing session, a client computer may request remote control of the host computer with a “REMOTE CONTROL request.” LaBine, ¶78. If approved, the client computer may “input events on the client computer..., such as mouse, keyboard and pen events” and the host computer “treats the remote user input events as though they were input locally.” LaBine, ¶84.

53. LaBine is also directed to the same technical field as the ’601 patent. The ’601 patent describes isolating *a multi-user interactive software application*,

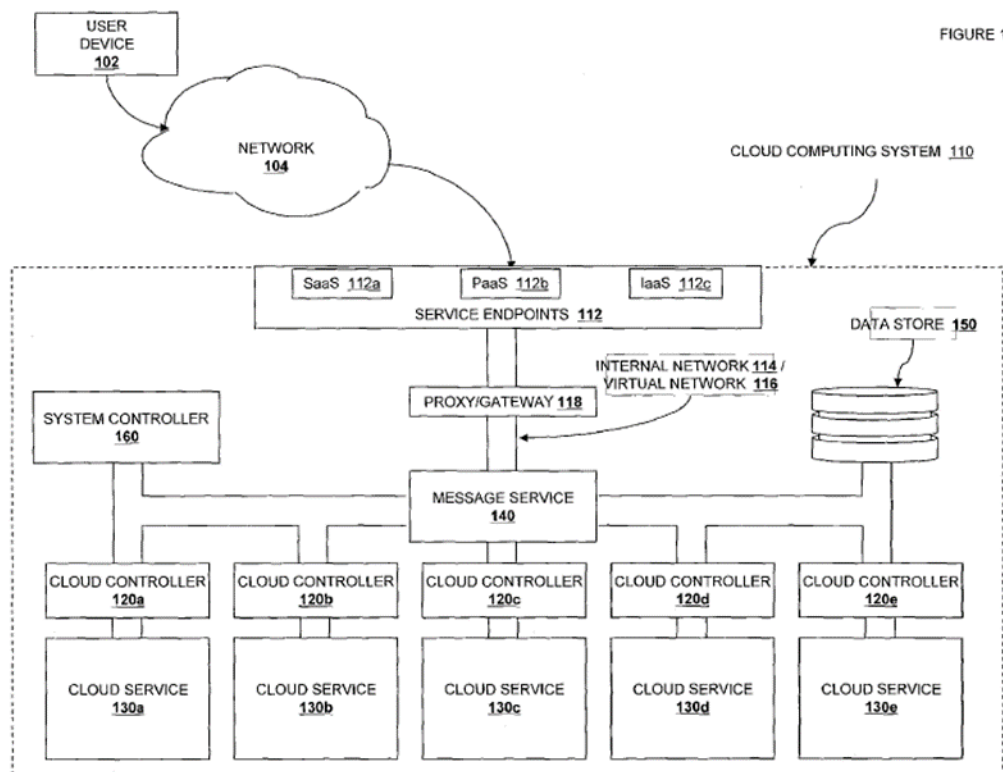
'601 patent, 22:55 (Claim 1), that allows *a remote machine to control an application*, '601 patent, 22:65-66 (Claim 2), to improve security. The '601 patent describes the multi-user interactive software application as “collaboration software” that “enables interaction with one or more other host computer systems” and allows the other computer systems to “access meeting data.” '601 patent, Abstract. Similarly, LaBine relates to improving interactive conferencing software by granting a client computer “remote control of the host computer” during a web conference, such that the “remote user input[s] events on the client computer having remote control” and the “[h]ost computer 48 treats the remote user input events as though they were input locally.” LaBine, ¶¶78-84. Like the '601 patent, LaBine recognizes security risks of its conferencing software. For instance, LaBine prevents transmitted data from being intercepted and tampered with through “SSL connections.” LaBine, ¶46. LaBine also describes positioning a “frame 120 to exclude sensitive controls from the shared region, thereby ensuring that other participants are not granted access to them.” LaBine, ¶48.

54. In addition, LaBine addresses the same problem of the '601 patent which is improving collaborative software applications by allowing users of client computers to remotely control host computers while providing security mechanisms to prevent malicious attacks and preserve confidential data. See '601 patent, 2:20-23 (“allow[ing] one or more remote machines to control...an application running on the

host computer system”); LaBine, Abstract (“enabling remote control of a host computer”).

C. Summary of Ishaya

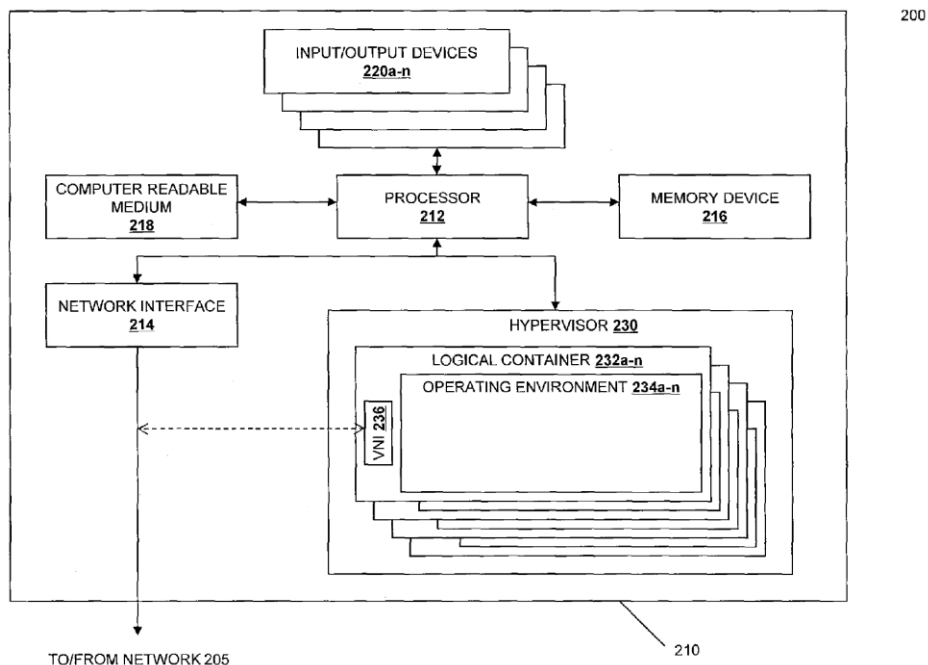
55. Ishaya describes a process for requesting, instantiating, and authenticating “virtual machine instances.” Ishaya, Abstract. This process is explained in the context of a “cloud computing system 110,” shown below. Ishaya, ¶25.



Ishaya, Fig. 1Jeffries, Fig. 1

Ishaya illustrates in Figure 2 an “information processing system 210 configured to host one or more virtual machines.” Ishaya, ¶35. The processing system 210 can

include “a personal computer (e.g., a desktop computer or a portable computer such as, for example, a laptop computer).” Ishaya, ¶35.



Ishaya, Fig. 2.

Ishaya describes different ways of configuring its “operating environment[s].”

Ishaya, [0041].

In one embodiment, a logical container 232 implements a hardware-like interface, such that the associated operating environment 234 appears to be running on or within an information processing system such as the information processing system 210.... In another embodiment, a logical container 232 implements an operating system-like interface, such that the associated operating environment 234 appears to be running on or within an operating system.... In a third environment, a logical container 232

implements a software-defined interface, such a language runtime or logical process that the associated operating environment 234 can use to run and interact with its environment.... Adding, removing, or modifying a logical container 232 may or may not also involve adding, removing, or modifying an associated operating environment 234.

Ishaya, ¶41.

56. Ishaya primarily uses the term “virtual machines” to refer to any of these logical containers.

For ease of explanation below, these operating environments will be described in terms of an embodiment as “Virtual Machines,” or “VMs,” but this is simply one implementation among the options listed above.

Ishaya, ¶41.

57. Ishaya also explains that the containers or virtual machines it creates may be isolated from other environments. *See* Ishaya, ¶41 (“A corresponding operating environment 234 could enforce separation between users and processes such that each process or group of processes appeared to have sole access to the resources of the operating system.”).

58. In Ishaya, each virtual machine or container is assigned its own “identification data,” such as a “Media Access Control (MAC) address,” or “Internet Protocol (IP) address.” Ishaya, ¶43; *see also* Ishaya, ¶53. Ishaya explains the process

for creating and authenticating a virtual machine in the context of Figure 7, shown below. See Ishaya, ¶¶177-187.

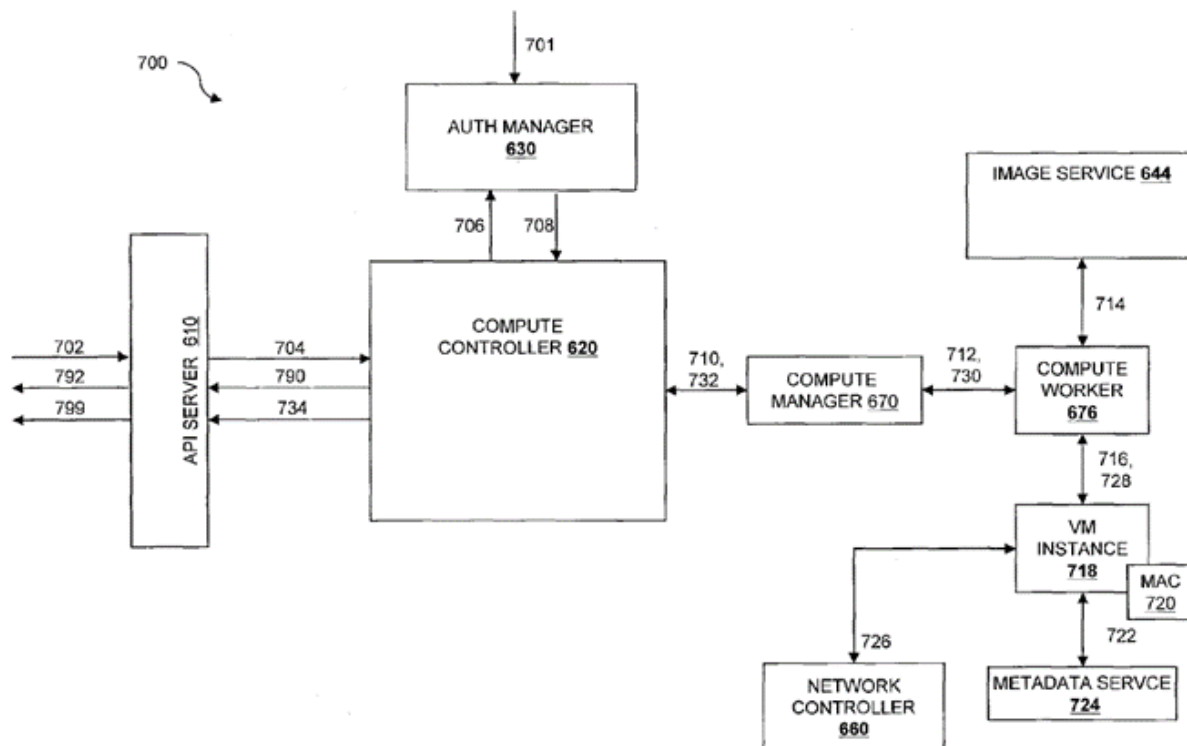


FIGURE 7

Ishaya, Fig. 7

59. At times 702 and 704, a user requests the creation of a virtual machine. See Ishaya, ¶¶177-178. At times 706 and 708, the “Auth Manager 630” determines whether the user has permission to create the type of virtual machine requested. Ishaya, ¶¶179-180. At time 716, the virtual machine is instantiated and assigned “a MAC address 720” which may be “selected from a pool of Mac addresses.” Ishaya, ¶181. At time 722, “the VM instance 718 authenticates to the Metadata Service 724 via an identity,” such as the MAC address previously assigned. Ishaya, ¶184. At time

726, the virtual machine authenticates itself again “to the Network Controller 660” to “obtain [a] VLAN and IP address.” Ishaya, ¶185. Using various identification numbers, such as the assigned MAC address or IP address described, the VM can authenticate itself to many “different components in a cloud computing system...using an identity-based approach.” Ishaya, ¶187.

60. Ishaya is analogous art to the '601 patent. The '601 patent describes “an authentication device” that “prevent[s] the multi-user interactive software application from communicating with the Internet destinations until after the isolated computing environment has authenticated with the authentication device.” '601 patent, 2:13-17. The authentication device described by the '601 patent improves security because it only allows “communicat[ion] with the untrusted network destinations...after the isolated computing environment has [been] authenticated.” '601 patent, 6:17-21. Ishaya similarly describes authenticating virtual machines, a term used to also describe containers. *See, e.g.*, Ishaya, ¶1 (“using identity information for authentication of virtual machines”), ¶41. Like the '601 patent, virtual machines/containers in Ishaya may be isolated, by “enforc[ing] separation between users and processes.” Ishaya, ¶41. Just as the '601 patent is concerned primarily with improving computer and network security that implement isolated computing environments, one of Ishaya’s stated goals is “increased security” through “identity-based authentication” of virtual machines. Ishaya, ¶194. Thus,

Ishaya is in the same field of endeavor as the '601 patent and is reasonably pertinent to a particular problem with which the inventor of the '601 patent was involved (i.e., enhancing computer security for isolated containers through container authentication).

D. Reasons to Combine Jeffries and LaBine

61. A POSITA would have combined Jeffries with LaBine because both of these references describe improving system security while executing applications with access to untrusted resources. See Jeffries 2:46-49 (“[U]sing containers...protects the computing device from attacks by isolating” the computing device “from untrusted resources.”), 10:59-60 (“Applications 124” within isolated containers “are thus useable to access untrusted resources.”); LaBine, ¶4, ¶50 (describing “[w]eb conferencing software” in which “viewers are given remote control access” of the host computer).

62. A POSITA would have been motivated to combine Jeffries and LaBine for at least the reasons discussed below and would have had a reasonable expectation of success in the combination. A POSITA would have been motivated to execute a web conferencing application like LaBine’s, within an isolated container, as described by Jeffries. Doing so would isolate the host computer and local intranet from any malicious code potentially introduced by the web conferencing application, as described by Jeffries, increasing the security of the computer and local intranet.

See Jeffries 2:46-49. A POSITA would also have understood that other implementations of the combination of Jeffries and LaBine are possible and that the specific details of such implementation would be left to a POSITA's design choice.

63. Jeffries describes running an application “in a container that is isolated from the host operating system.” Jeffries, Abstract. Because isolating applications in containers “protects the computing device from attacks,” Jeffries, 2:46-49, the “[a]pplications 124” within isolated containers “are thus useable to access untrusted resources,” Jeffries, 10:59-60.

64. The applications 124 isolated within the containers 122 in Jeffries can be of any type, including any “applications that are executable by one or more processors of the computing device.” Jeffries, 9:26-27. Jeffries discloses some examples of applications including “a web browser application” as well as “email applications, spreadsheet applications, visual presentation applications, and the like.” Jeffries, 9:29-33 (listing types of applications 114); see 10:54-58 (“applications 124(1),..., 124(n) correspond to instances of individual ones of the applications 114”).

65. Jeffries does not list every kind of application that may be executed in its isolated containers in detail. However, based on the categories of applications Jeffries does disclose, a POSITA would have understood that Jeffries' techniques of isolating applications within isolated containers would be applied to any type of

application to improve computer and network security. *See* Jeffries, 9:26-33 (“[A]pplications 114 may include a web browser application[,]... email applications, word processing applications, spreadsheet applications, visual presentation applications and the like.”), 10:56-58 (“Individual ones of the one or more applications 124(1),...,124(n) correspond to instances of individual ones of the application 114 on host operating system 102.”); *see also* Kopasz, [0004].

66. LaBine provides greater detail on a particular application that would be executed within an isolated container as described by Jeffries. In particular, LaBine describes a video conferencing software that allows participants of the video conference to remotely access and control the host user’s computer. *See generally* LaBine, Abstract, ¶¶78-84. As explained above, *see* §IX.B, when a remote user has obtained remote control of the host computer, the remote user may provide “user input events on the client computer..., such as mouse, keyboard and pen events” which are “treat[ed]” by the host computer “as though they were input locally.” LaBine, ¶84. LaBine explains that its system is “intuitive for users collaborating in a conferencing or other environment” but also recognizes potential security risks involved with a remote user having access to input commands as if the user were at the computer locally. LaBine, ¶18. For example, LaBine recognizes that this collaborative software may transmit or receive “sensitive information,” LaBine, ¶46, as well as expose sensitive and/or personal information of the host computer to

viewers with remote access control. LaBine, ¶50.

67. In addition, a POSITA would have recognized LaBine’s collaborative software as software that accesses an untrusted source. An untrusted source may be any unknown source. *See* Jeffries, 3:48-49 (“[T]he host operating system considers every resource to be untrusted.”). LaBine’s software allows a remote user, including unknown users, to control the host computer. LaBine, ¶84. Because the remote user with access to the host computer could attempt to control the host computer to download and execute malicious content, access sensitive or confidential data, or perform any number of nefarious acts while controlling the host computer, a POSITA would have recognized that LaBine’s software was software that accesses an untrusted source.

68. To alleviate the security concerns posed by running LaBine’s web conferencing software in an uncontrolled environment, a POSITA would have found it obvious to implement LaBine’s web conferencing application within an isolated container as described by Jeffries. LaBine simply describes a specific implementation of the same type of applications Jeffries describes as being isolated within its containers 122. *See* Jeffries, 9:26-33; LaBine, Abstract, ¶¶78-84. Just as Jeffries’ applications 124 contained in isolated containers 122 are designed to “access untrusted resources,” *see* Jeffries, 10:59-60, LaBine’s web conferencing application is designed for communication with untrusted resources, including any

remote computer, *see* LaBine, ¶84. Thus, LaBine's web conferencing is exactly the type of application Jeffries' anticipates as being executed within its isolated containers 122 to protect against potentially malicious content that may be received from untrusted resources.

69. The combination of Jeffries and LaBine would be simply combining prior art elements according to known methods to yield predictable results. The known methods would involve isolating the web conferencing application described in LaBine, *see* LaBine, ¶¶45, 84, within an isolated container as described in Jeffries, *see* Jeffries, Abstract, 2:46-49. The results of the combination would have been predictable because Jeffries explains how this isolation protects the host computer from any viruses or malware or other nefarious acts as described by LaBine that may be introduced through the application. *See* Jeffries, 10:58-61; *see also* 3:48-51.

70. In addition, a POSITA would have recognized multiple benefits of the combination. As explained above, the combination of Jeffries with LaBine would advantageously improve the security of a computer system that participates in web conferencing that allows for remote control of participating computers. In addition, a POSITA would have recognized that implementing a web conferencing application, such as the one described by LaBine, within a computer system would provide multiple desirable results, such as improving the ability of users to collaborate with team members within an organization. With the added layer of

security provided by Jeffries' isolated containers, users could use these same collaborative techniques with less familiar participants to lessen the risk of exposing the host computer to malicious code. For example, users of the combined system could share files, present media content, or even grant remote access to users outside of their organization, while protecting the host computer and network.

71. Further, isolating LaBine's interactive remote-control software in a container 122 would prevent it "from accessing any resources or data outside the container." *See* Jeffries, 5:52-54; *see also* Jeffries, 12:17-19 ("[W]hen malicious software attempts to access off-limit data or memory locations, the secure kernel causes an exception or interrupt to occur."). Thus, the effect of the executed code would be contained within the isolated container and prevented from affecting any other part of the host computer or local network. *See* Jeffries, 16:29-31 ("[C]ontainer manager 118 is configured to quarantine the container 122 and prevent the container 122 from further accessing resources" thus "prevent[ing] malicious activity occurring in the container from gaining file system access."). In this event, "corrective action" may include "terminating the container 122 and rebuilding a new container." Jeffries, 16:39-42.

72. A POSITA would have had a reasonable expectation of success in implementing the web conferencing software application described by LaBine in an isolated container described by Jeffries. Even though Jeffries' techniques of

container isolation apply to any kind of application, *see* Jeffries, 9:26-33, Jeffries specifically contemplates isolating applications that access untrusted resources, *see* Jeffries, 10:58-61, 3:48-51. LaBine’s web conferencing application is exactly the type of application Jeffries contemplates because it allows remote access from untrusted users. LaBine, ¶84. In addition, Jeffries explains that the policies of container isolation it implements are flexible enough to ensure that any application has access to the necessary resources to run as expected. *See* Jeffries, 15:26-30 (describing “allocat[ing] resources that may be required by container subsystems 206, such as memory, processors, disks, or networks”). Running LaBine’s web conferencing application in Jeffries’ isolated containers would therefore have been simple and within the skillset of a POSITA.

73. For all these reasons, a POSITA would be motivated to implement a multi-user interactive software application, like LaBine’s web conferencing software, in an isolated container, as described by Jeffries. Doing so would predictably improve the security of the host computer executing the multi-user interactive software application for the reasons described above.

E. Reasons to Combine Jeffries and Ishaya

74. A POSITA would have combined Jeffries with Ishaya because both of these references describe systems and methods for improving security of computers that implement containers. *See* Jeffries, Abstract, 2:46-49 (“protect[ing] the

computing device from attacks by isolating” applications in containers); Ishaya, ¶¶1, 194 (describing “increased security” through “identity-based authentication” of virtual machines/containers).

75. A POSITA would have been motivated to combine the references for at least the reasons discussed below and would have had a reasonable expectation of success in the combination. One possible combination of Jeffries and Ishaya, as described below, would include implementing the details of virtual machine and container authentication described in Ishaya in a device in communication with the host operating system 102 of Jeffries, such as the management and monitoring service 104. Containers, such as those described in Jeffries, run in environments (e.g., enterprise networks, cloud) where ensuring security is important. A POSITA would have looked to Ishaya for methods of improving the security of the environments described by Jeffries. Ishaya describes one such technique of authentication and a POSITA would be motivated to incorporate these techniques into the system of Jeffries to improve security. A POSITA would additionally have had the necessary knowledge to implement this combination and would have had a reasonable expectation of success. A POSITA also would have understood that other implementations of the combination of Jeffries and Ishaya are possible and that the specific details of such implementation would be left to a POSITA’s design choice.

76. Jeffries describes running an application “in a container that is isolated

from the host operating system.” Jeffries, Abstract. Because isolating applications in containers “protects the computing device from attacks,” Jeffries, 2:46-49, the “[a]pplications 124” within isolated containers “are thus useable to access untrusted resources,” Jeffries, 10:59-60.

77. Jeffries also generally describes the importance of authentication. *See, e.g.*, Jeffries, 6:55-57 (describing “requiring authentication from host operating system 102 before granting access to the one or more resources.”); Jeffries, 14:47-54 (describing user authentication). Jeffries mentions the process of “creat[ing] and activat[ing]” a new virtual machine or container using “[a] container image base.” Jeffries, 15:14-15. However, Jeffries provides limited details regarding (1) specific steps of container creation, (2) how the identity of the newly created container may differ from the host computer, or (2) how the container is permitted to communicate with other network destinations upon creation.

78. Ishaya provides these implementation details regarding the process of container creation. Given Jeffries’ limited details regarding specific steps of creating an isolated container on a computer system, a POSITA seeking to implement an isolated container as described by Jeffries, would have found it obvious to consult an additional reference explaining this creation process. Ishaya provides these details including authenticating a container as part of this process by assigning it a MAC and/or IP address. Ishaya, ¶¶174-186. Ishaya further explains multiple advantages

its container creation process including authentication, including ensuring that the container can be trusted, Ishaya, ¶189, the ability to authenticate with controllers of other vendors, Ishaya, ¶190, as well as other advantages. *See* Ishaya, ¶¶191-196.

79. In particular, Ishaya explains that when a new container, referred to as a virtual machine, is created, it is assigned a new MAC address different from the host machine. Ishaya, ¶43. The new MAC address may be selected from a “pool” of available MAC addresses. Ishaya, ¶181. The MAC address may be used to obtain a new IP address that is different from the host machine such that the container appears on the network to be a different machine than the host computer. Ishaya, ¶185.

80. Ishaya explains that with this new IP address, the container may authenticate itself to other devices on the network. For example, a newly created container may authenticate itself to the “Metadata Service 724” and “Network Controller 660.” Ishaya, ¶¶184-185. In addition, the container may authenticate to a variety of “different components in a cloud computing system.” Ishaya, ¶187.

81. Ishaya also explains multiple advantages of using its identity-based authentication methods to authenticate containers. For example, Ishaya’s “identity-based authentication...allows interaction between...a cloud computing system and VMs based upon APIs” which “increases...flexibility and capability.” Ishaya, ¶193. In addition, Ishaya’s “identity-based authentication...increase[s] security and privacy for users.” Ishaya, ¶194.

82. Ishaya explains that its authentication method also “addresses the difficult problem of authentication during initial setup of a virtual machine instance” by “remov[ing] the need for storing user credentials or encryption keys with a virtual machine for authentication.” Ishaya, ¶195. Ishaya’s authentication method also advantageously “provide[s] secure authentication and authorization of components within a multi-vendor cloud computing system.” Ishaya, ¶196.

83. The combination of Jeffries and Ishaya simply represents combining prior art elements according to known methods to yield predictable results. The known methods would involve authenticating a newly created isolated container, such as a container 122 described in Jeffries, *see* Jeffries, 16:14-30, by assigning it a new MAC/IP address and using that address to authenticate to other networked devices, as described by Ishaya, *see* Ishaya, ¶¶43, 53, 181. This authentication technique was common before the ’601 patent. *See* Rangarajan, 7:41-44 (“For example, if compute resource 50 were a container, the trust level of the container might additionally depend on the trust level of the Docker image associated with the container.”). The results of the combination would have been predictable because Ishaya explains how this process of container isolation improves security with identity-based authentication as an improvement to simple user authentication with usernames and passwords. *See* Ishaya, ¶194.

84. In addition, a POSITA would have had a reasonable expectation of

success of combining Jeffries and Ishaya. A POSITA at the time would have had a sufficient understanding of creating, isolating, and authenticating virtual machines and containers, as each of these steps were well known at the time. In addition, both Jeffries and Ishaya refer to containers and virtual machines. *See* Jeffries, 10:22-53; Ishaya, ¶¶41-43. Both Jeffries and Ishaya describe isolating these virtual machines and containers. *See* Jeffries, Abstract, Fig. 1, 2:39-60; Ishaya, ¶41 (enforc[ing] separation between users and processes”). Both Jeffries and Ishaya describe the importance of authentication in the context of virtual machines and containers, including authenticating operating systems and users, *see* Jeffries, 6:55-57, 14:47-54, as well as authenticating virtual machines and containers, *see* Ishaya, Abstract, ¶1. Therefore, authenticating containers in Jeffries’ system would simply include assigning a MAC or IP address to the container after creation, and authenticating the container to any other network device, such as a metadata service, network controller, or any other authentication device, as taught by Ishaya. In the combination, Jeffries’ containers 122 would authenticate to the management and monitoring service 104 which “provides one or more administrative policies for the computing device” and/or to the web proxy 106 which “requir[es] authentication from host operating system 102 before granting access to the one or more resources.” Jeffries, 6:44-57; *see also* Jeffries 7:1-18 (“In one or more embodiments,...management and monitoring service 104, and web proxy 106 are

implemented as part of the same computing device.”). Accordingly, a POSITA would have reasonably expected success in applying aspects or features of container authentication described by Ishaya in reference to Jeffries.

85. For all these reasons, a POSITA would have been motivated to authenticate the containers described by Jeffries as part of the initial creation and processing of the containers 122. Doing so would predictably improve the security of the host computer through identity-based authentication as described by Ishaya.

F. Reasons to Combine Jeffries, LaBine, and Ishaya

86. A POSITA would have been further motivated to combine Jeffries and LaBine with Ishaya because the references all describe complimentary techniques for improving security when executing applications that access untrusted sources. Jeffries improves security by using “isolated containers” to “protect[] the computing device from attacks.” Jeffries 2:39, 45-46. LaBine improves security by masking sensitive data when allowing remote users to access and control a host computer. LaBine, ¶¶50, 84. Ishaya describes isolating and authenticating containers. Ishaya, ¶¶41, 43, 45, 177-187.

87. A POSITA would have been motivated to combine the combination of Jeffries and LaBine with Ishaya and doing so would have been within the POSITA’s skillset. The combination would have been obvious because it represents merely the use of a known technique (authenticating a container) to improve a similar process

(executing a web conferencing application that grants unknown remote users the ability to access and modify the host computer within an isolated container) in the same way to yield predictable results. Specifically, Ishaya provides implementation details regarding the process of container creation, including assigning a new container a new MAC address different from the host machine. Ishaya, ¶43. The MAC address may be used to authenticate the container after it is instantiated. *See* Ishaya, ¶¶184-185.

88. A POSITA also would have been motivated to combine the combination of Jeffries and LaBine with Ishaya because Ishaya provides additional details regarding the instantiation and configuration of containers that the combination of Jeffries and LaBine does not explicitly disclose. Ishaya, Fig. 7, ¶¶177-187. For example, Ishaya explains the process for creating a new container including a user requesting the creation of a container, Ishaya, ¶¶177-178, determining whether the user has permission to create the type of container requested, Ishaya, ¶¶179-180, and assigning the container a MAC address, Ishaya, ¶181. Ishaya further explains how the container is given access to the resources it needs and restricted from resources it shouldn't have access to. Ishaya, ¶183. Ishaya then explains how the container can use the MAC address assigned to it to authenticate itself to the "Metadata service 724" and "Network Controller 660." Ishaya, ¶¶184-185.

89. Ishaya further explains multiple benefits of implementing its disclosed methods of container authentication. For example, Ishaya’s method of container authentication ensures that various network devices “can determine whether or not to trust information from” the container. Ishaya, ¶189. Further, Ishaya’s method allows “compute controllers from different vendors or entities” to “authenticate to each other.” Ishaya, ¶190. Ishaya’s method “allows interaction between a cloud computing system and” the container, Ishaya, ¶193, and offers “increased security and privacy for users,” Ishaya, ¶194; *see also* Ishaya, ¶¶195-96.

90. A POSITA would have had a reasonable expectation of success in combining Jeffries and LaBine with Ishaya. Neither Jeffries nor Ishaya limits the applications that may be executed within the isolated containers they describe. In addition, both Jeffries and Ishaya describe the importance of authentication in the context of virtual machines and containers, including authenticating operating systems and users, *see* Jeffries, 6:55-57, 14:47-54, as well as authenticating virtual machines and containers, *see* Ishaya, Abstract, ¶1. And a POSITA would have understood that Ishaya’s methods of container authentication would be equally applicable to containers specifically containing a web conferencing application, like the one described by LaBine. Ishaya, Abstract, ¶25; LaBine, ¶45. The proposed combination would include simply creating one of Jeffries’ containers implementing the steps of creation and authentication described by Ishaya. After the container is

created, a video conferencing application, such as the one described in LaBine, would easily be installed and executed within the isolated container.

91. For all of these reasons, a POSITA would have been motivated to authenticate the containers described by Jeffries as part of the initial creation and processing of the containers 122 and to execute a web conferencing application that permits unknown users to remotely control a host computer, as described by LaBine, in the authenticated container, as described by Ishaya. Doing so would predictably improve the security of the host computer through identity-based authentication as described by Ishaya.

G. Detailed Analysis of Claims

1. Claim 1

[1.0] A system comprising:

92. To the extent the preamble is limiting, Jeffries discloses it.

93. Jeffries Figure 1 “illustrates an example **system 100**,” shown below.
Jeffries, 6:20.

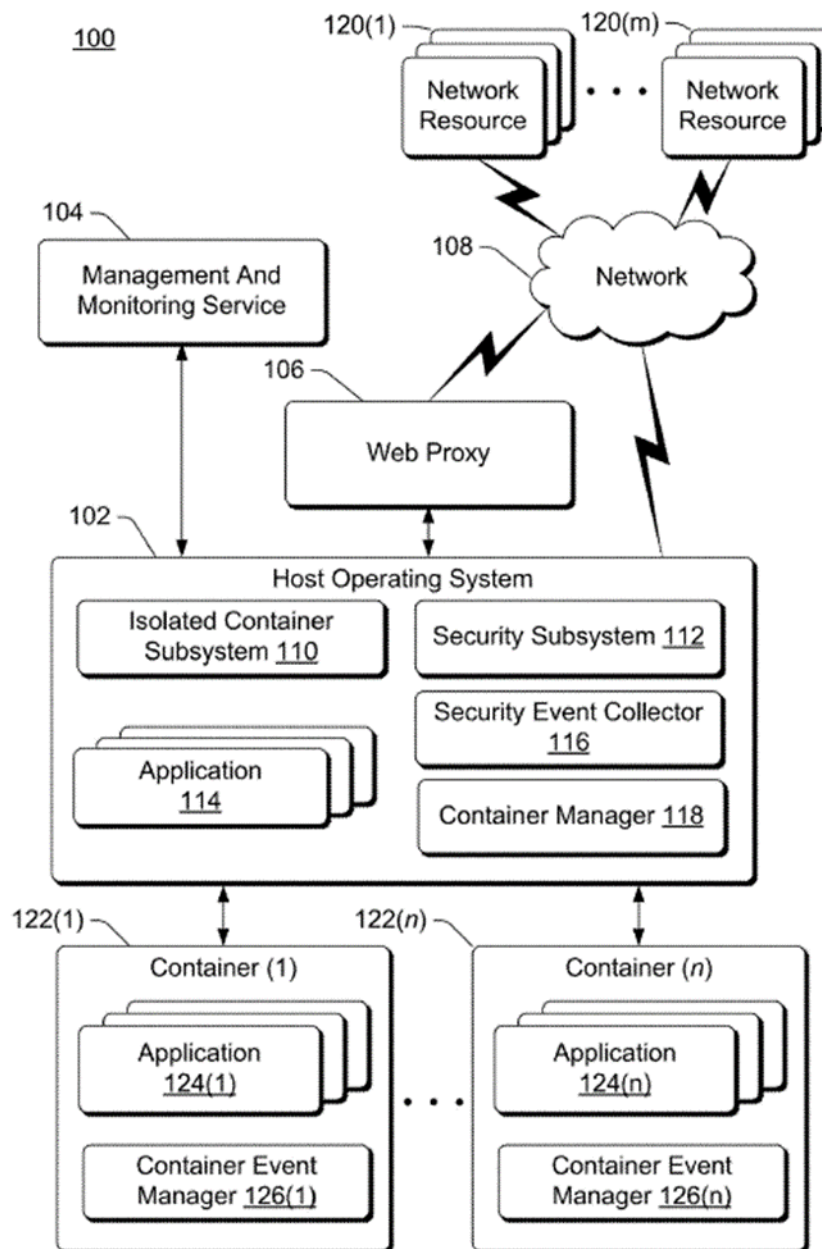


Fig. 1

Jeffries, Fig. 1

94. Therefore, Jeffries teaches “a system,” as recited in the preamble.

[1.1] a memory; and

95. Jeffries also discloses this limitation.

96. Jeffries explains that the device that implements the system 100 illustrated in Figure 1, above, includes a memory:

Thus, the computing device implementing system 100 may range from a full resource device with **substantial memory** and processor resources (e.g., personal computers, game consoles) to a low-resource device with **limited memory** and/or processing resources (e.g., traditional set-top boxes, hand-held game consoles).

Jeffries, 6:37-43.

97. In addition, Jeffries Figure 6 illustrates “an example computing device 602 that is representative of one or more systems and/or devices that may implement the various techniques described herein.” Jeffries, 22:39-42. Figure 6 is shown below.

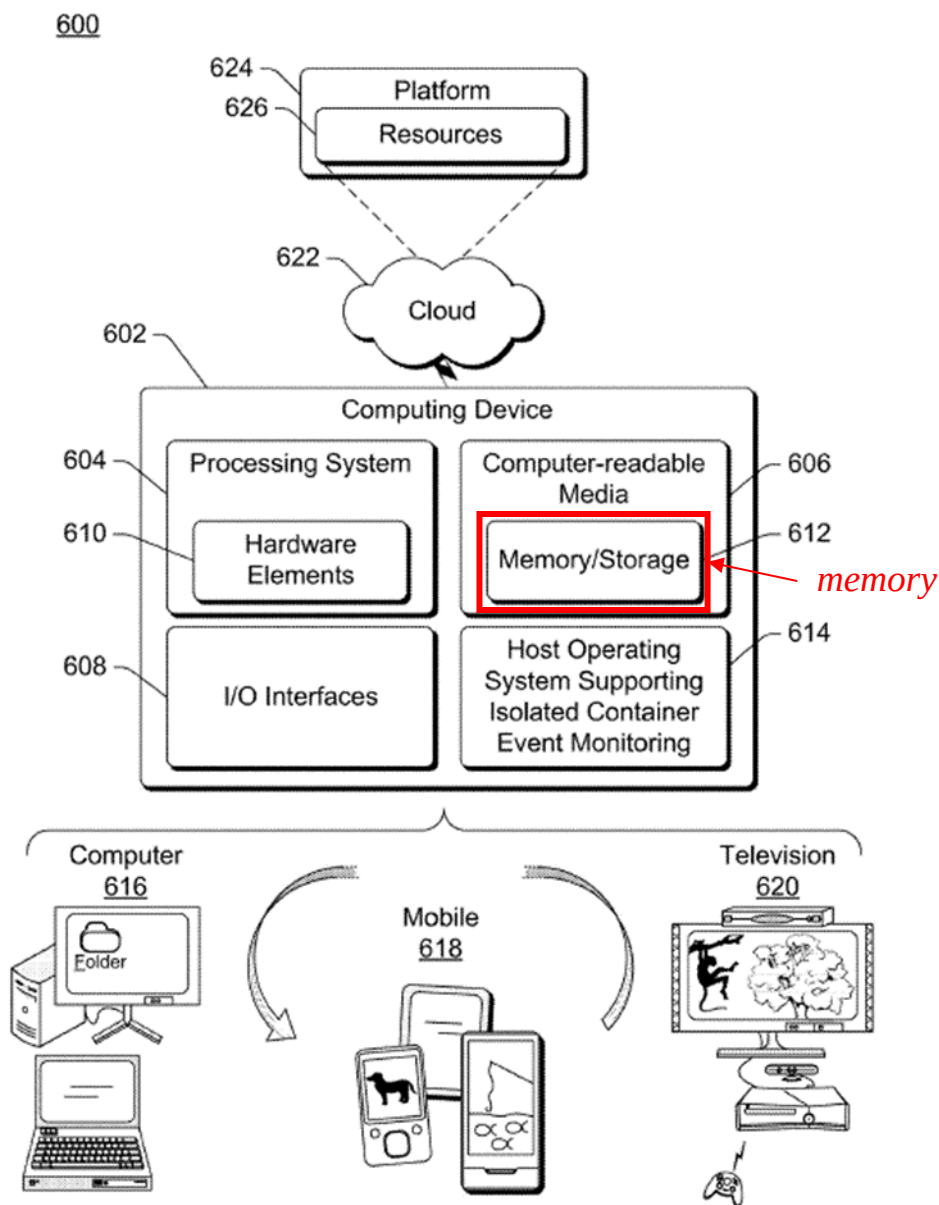


Fig. 6

Jeffries, Fig. 6 (annotated)

98. The computing device 602 includes a “memory/storage 612.” Jeffries 23:8. Figure 6 is directed to the same disclosure as the system 100 described in Figure 1 above and merely provides further details regarding the physical

components of devices that implement the alleged invention.

99. Therefore, Jeffries's system includes a "memory/storage 612" which corresponds to "*a memory*" claimed.

[1.2] a processor configured to:

100. Jeffries discloses this limitation.

101. The example computing device 602 illustrated in Figure 6 also includes a "processing system 604," shown below. Jeffries, 22:47-48.

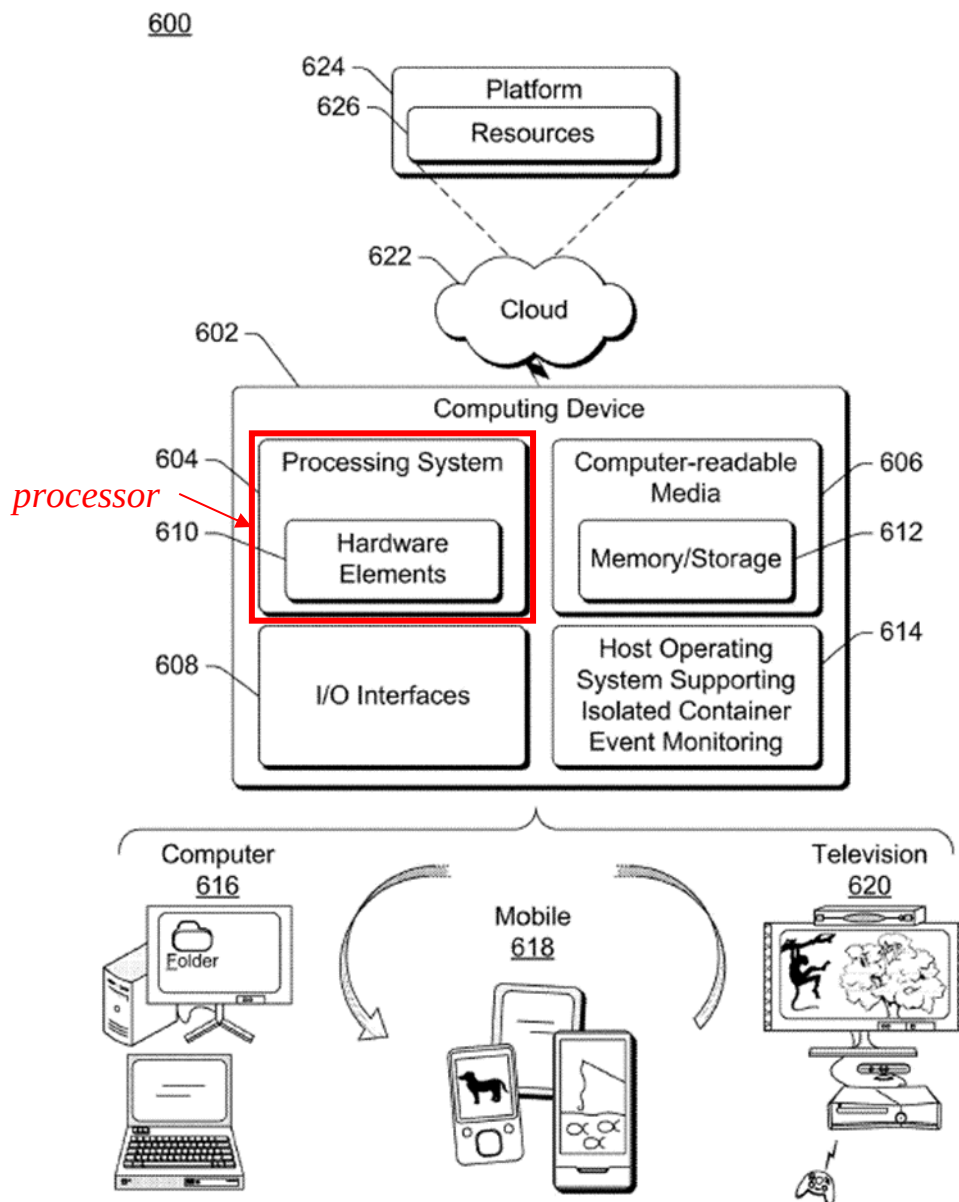


Fig. 6

Jeffries, Fig. 6 (annotated)

102. Jeffries explains that the processing system 604 “may be configured as **processors**” and “is representative of functionality to perform one or more operations using hardware.” Jeffries, 22:60-64.

103. Thus, Jeffries teaches a processing system 604 that includes a processor, which corresponds to “*a processor*,” as claimed.

[1.3] 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;

104. Jeffries renders this limitation obvious.

105. First, Jeffries renders obvious “*a host operating system*.” Jeffries’ “system 100 includes a host operating system 102.” Jeffries, 6:44. The host operating system 102 is illustrated in Figure 1, highlighted in blue below.

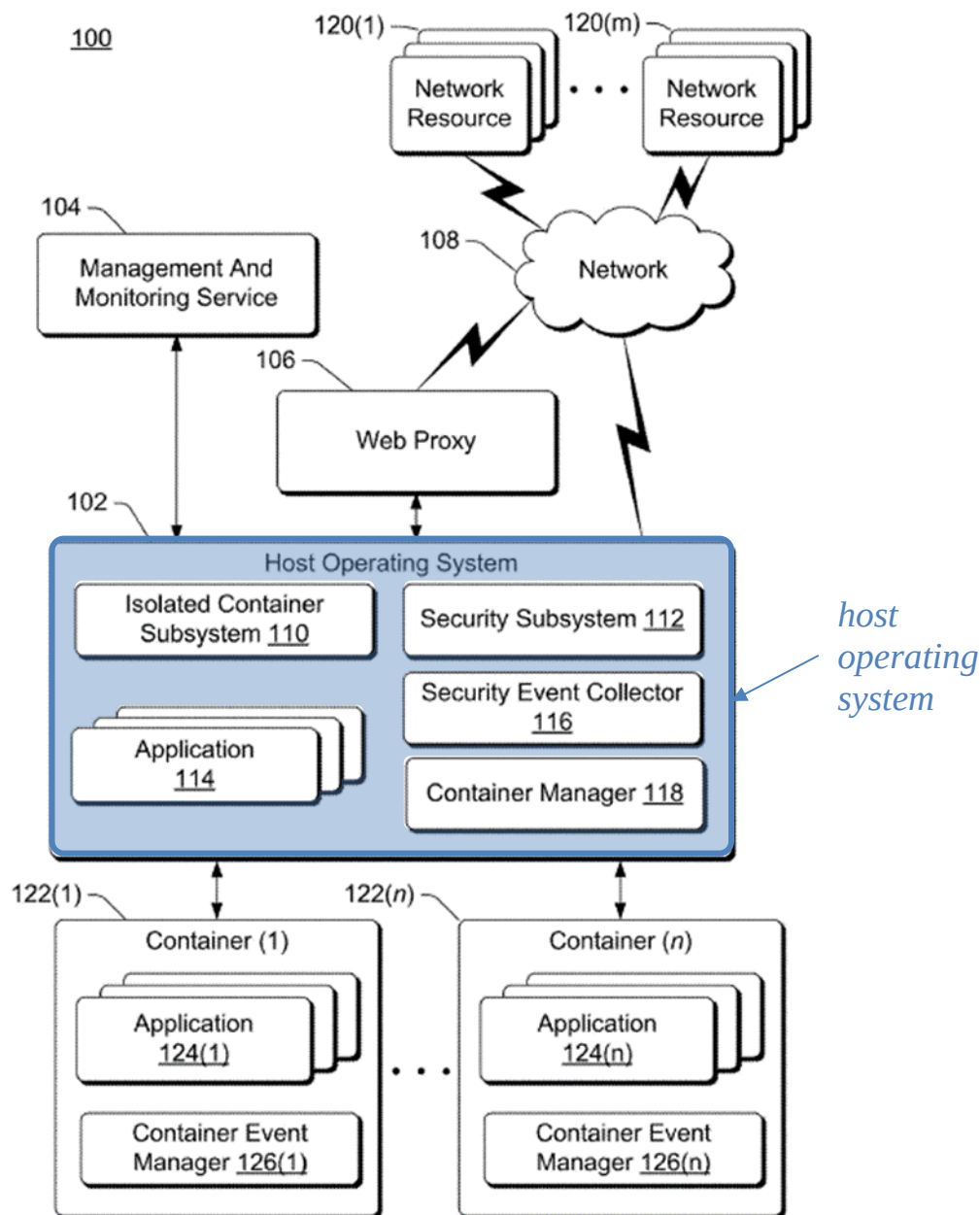


Fig. 1

Jeffries, Fig. 1 (annotated)

106. Second, a POSITA would have found it obvious for Jeffries’ system to “implement a workspace that uses” the host operating system 102, in two ways.

107. Jeffries renders obvious a “workspace” by describing resources that are

not isolated within an isolated computing environment. For example, and as will be explained in more detail with reference to claim limitation [1.4.1], Figure 1 illustrates multiple components of the system 100, highlighted in dark blue below.

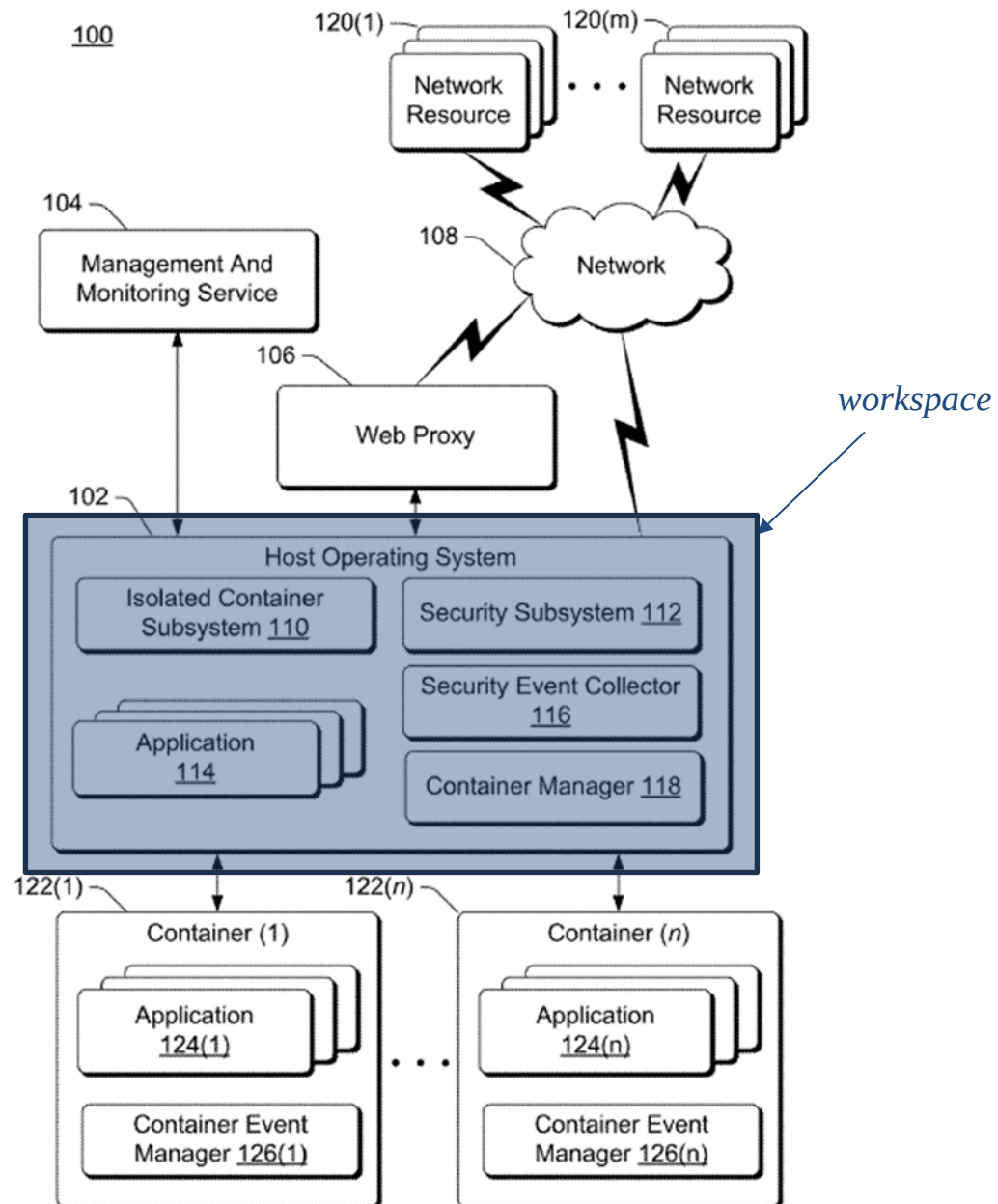


Fig. 1

Jeffries, Fig. 1 (annotated)

108. The host operating system 102, isolated container subsystem 110, applications 114, security subsystem 112, security event collector 116, and container manager 118 shown above are all not contained within the isolated computing environments (containers 122) in Jeffries and would therefore collectively be a “workspace,” as claimed and illustrated above. To the extent components such as the web proxy 106 or management and monitoring service 104 are “implemented as part of the same computing device,” Jeffries, 7:1-4, these components would also be part of the “workspace” in the context of Jeffries’ system. Jeffries’ disclosure of a workspace is consistent with the ’601 patent’s description of the term “workspace.” The ’601 patent does not explicitly define the term “workspace.” Instead, it generally contrasts the workspace with the isolated computing environment (discussed in more detail at limitation [1.4.1] below). For example, the ’601 patent describes “prevent[ing] communication between the isolated computing environment and the workspace.” ’601 patent, 1:65-2:2. Similarly, the ’601 patent describes the workspace as including “one or more processes operating within the operating system that are not restricted by the sandbox container process.” ’601 patent, 5:44-46.

109. Alternatively, Jeffries also describes “memory locations” associated with the components depicted in Figure 1, above. Jeffries, 12:13. Specifically, Jeffries describes “memory locations associated with the container event manager

126” within isolated containers 122, as well as memory locations associated with “the security event collector 116, the container manager 118, or **any other component of host operating system 102.**” Jeffries, 12:13-16.

110. Jeffries explains that the memory associated with the host operating system 102 is “off-limits” to the “container[s] 122.” Jeffries, 12:10, 12:16.

[W]hen malicious software attempts to access **off-limit data or memory locations**, the secure kernel causes an exception or interrupt to occur, which is monitored by the container event manager 126 and included in a security event passed to the security event collector for determining a threat level associated with the off-limit access attempt.

Jeffries, 12:17-22.

111. What Jeffries refers to as “off-limit data or memory locations” are not contained within Jeffries’ isolated computing environments. Therefore, these “off-limit data or memory locations” correspond to a “workspace” as claimed.

112. Similarly, Jeffries explains that its invention is directed to preventing untrusted sources from “writ[ing] illicit information to a computing device’s operating system or access[ing] **privileged memory locations.**” Jeffries, 1:10-11. Like the “off-limit...memory locations” described above, these privileged memory locations are locations that are isolated from the isolated containers described in Jeffries. Therefore, Jeffries’ off-limit data or memory locations and/or privileged

memory locations would constitute a “workspace” as claimed.

113. This alternative mapping of certain memory locations to the claimed “workspace” is also consistent with the ’601 patent. For example, the ’601 patent implies that the workspace is itself memory, including all memory that is not dedicated to the isolated computing environment:

The terms ‘sandboxed computing environment’ and ‘isolated computing environment’ may be used interchangeably herein. For example, they may refer to a memory space that is isolated (e.g., by use of an internal isolation firewall) from **another memory space** (e.g., a **workspace**).

’601 patent, 5:33-37.

114. In summary, Jeffries describes a “workspace” as claimed because it describes both resources and memory locations that are not isolated within its isolated containers 122.

115. Third, Jeffries also explains how the resources and memory associated with the host operating system it describes (the “workspace”) “*enables operation of a first set of one or more applications or processes.*”

116. Jeffries illustrates a set of applications 114 within the host operating system 102 in Figure 1, highlighted in green below.

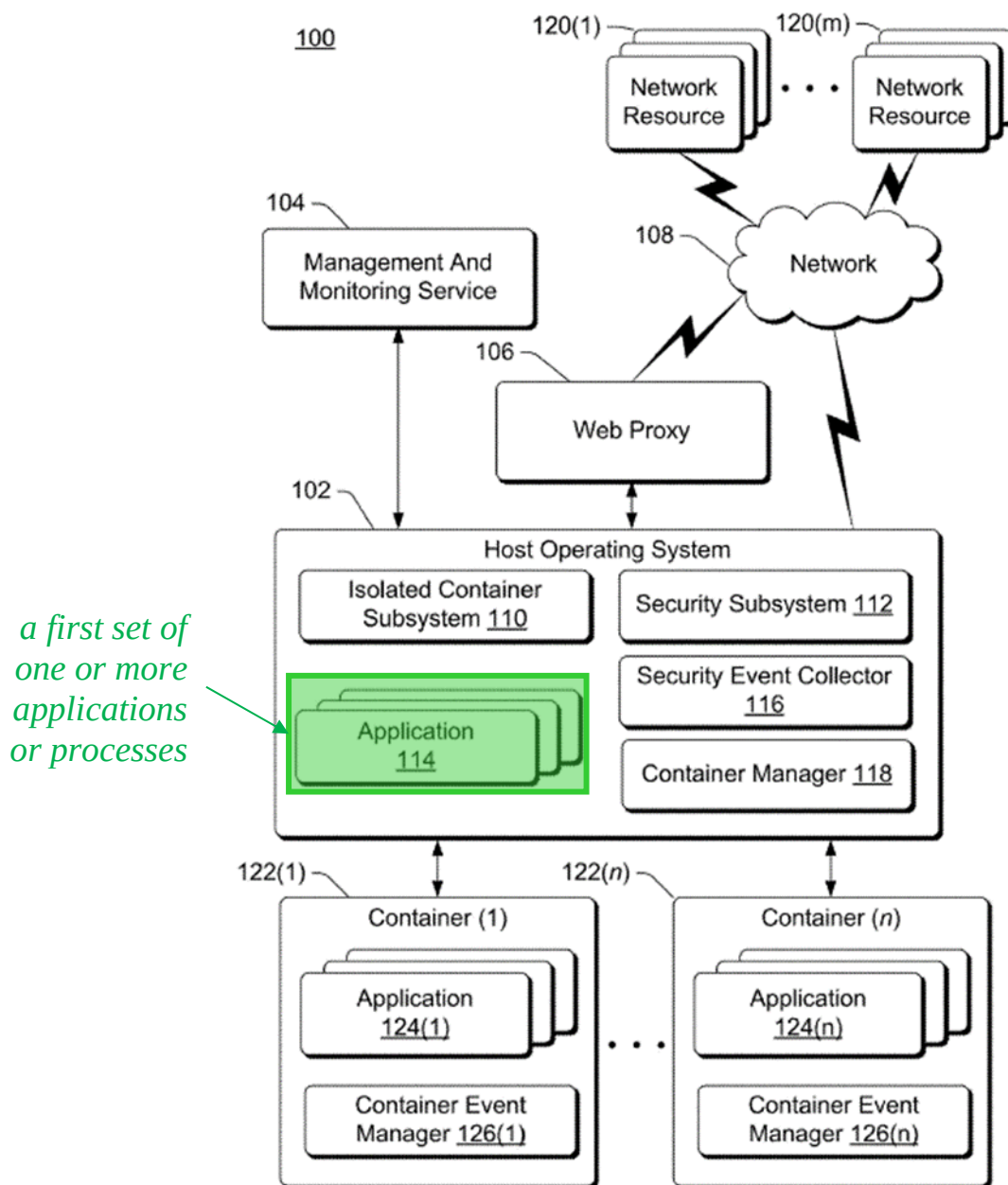


Fig. 1

Jeffries, Fig. 1 (annotated)

117. Jeffries explains that these applications 114 are “one or more applications that are executable by one or more processors of the computing device implementing the host operating system 102.” Jeffries, 9:26-28; *see also* Jeffries,

8:55-57 (“The host operating system 102 also includes...**one or more applications 114.**”). The “[a]pplications 114 may include a web browser application...such as e-mail applications, word processing applications, spreadsheet applications, visual presentation applications, and the like.” Jeffries, 9:29-33.

118. Because the applications 114 are included within the host operating system 102, they are *enabled* by the workspace corresponding to the host operating system 102, including by general resources or actual memory not contained within the isolated containers 122.

119. Fourth, Jeffries explains that the applications 114 are enabled by the workspace “*via a first memory space,*” as claimed.

120. As explained above, Jeffries describes “memory locations” associated with the host operating system 102. Jeffries, 12:12-16. Specifically, Jeffries explains that each component of the host operating system 102 is associated with a corresponding memory location.

[T]he VMBUS 208 implements a secure kernel that specifies any data or **memory locations associated with** the container event manager 126, the security event collector 116, the container manager 118, or **any other component of host operating system 102.**

Jeffries, 12:12-16.

121. The applications 114 are depicted as a “component of the host operating

system 102” in Figure 1. Jeffries, 12:16. Based on this disclosure, a POSITA would have understood that Jeffries’ system includes “memory locations” associated with the applications 114, which correspond to “*a first memory space*” that enable the operation of the applications 114.

122. This understanding is consistent with basic principles of computer architecture. A POSITA at the time would have understood that an application running on a host computer would be supported by an associated memory space. *See, e.g.,* Yu, ¶3 (“Typically, each application executes in a corresponding memory space, which may be referred to as application memory space.”); Song, Abstract (describing “a memory space of a corresponding application”); Tucker, Abstract (describing “a contiguous region of virtual memory space of a corresponding endpoint computer application”).

123. Thus, Jeffries describes separating and isolating resources and/or a memory space from its isolated containers 122 (“*implement a workspace*”) and a host operating system 102 (“*that uses a host operating system*”). In addition, Jeffries explains that its memory space isolated from the containers 122 supports applications 114 via associated memory locations (“*wherein the workspace enables operation of a first set of one or more applications or processes via a first memory space*”).

[1.4.1] 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

124. Jeffries renders this limitation obvious.

125. First, Jeffries describes “*implementing an isolated computing environment.*”

126. As mentioned previously with regards to the analysis of a “workspace” at claim limitation [1.3], Jeffries describes creating “containers 122(1),..., 122(n)” which are “*isolated computing environments.*” Jeffries, 8:17. Jeffries’ containers 122 are illustrated in Fig. 1, with container 122(1) highlighted in orange below.

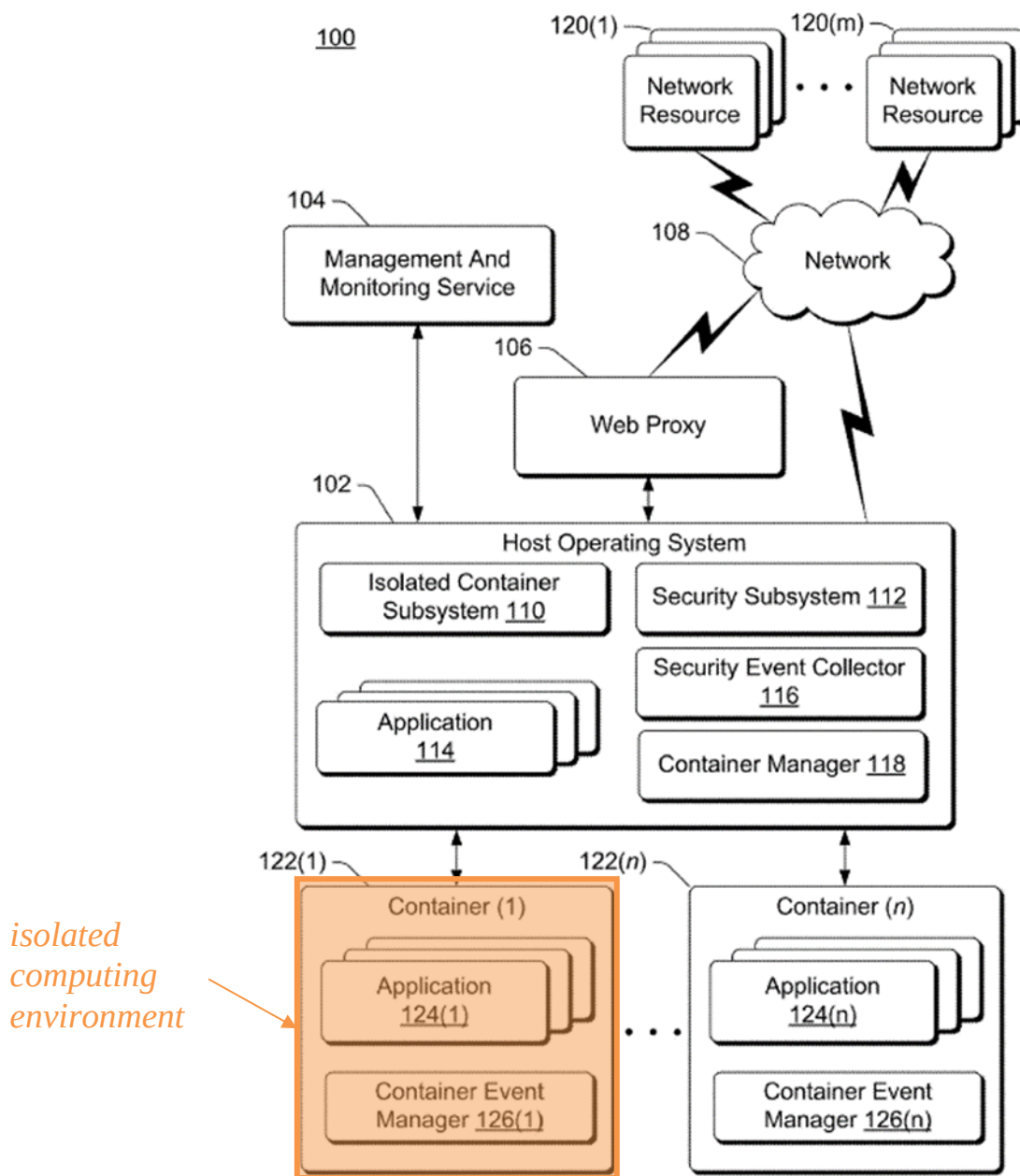


Fig. 1

Jeffries, Fig. 1 (annotated)

127. Jeffries containers 122 are “*isolated computing environment[s]*” because Jeffries describes the containers as “isolated containers” and explains that the “one or more containers...provide full kernel isolation from the host operating

system.” Jeffries, 2:39, 54-55. Jeffries further explains that each container 122 is “isolated from the host operating system” to “prevent...malicious activity from spreading beyond the isolated container.” Jeffries, Abstract. The ’601 patent also acknowledges that containers are isolated computing environments, or “sandboxed computing environment[s],” describing a “sandbox container process...permit[ting] isolation of the sandboxed computing environment.” ’601 patent, 6:4-5. Jeffries’ isolated containers 122 are precisely the same type of containers described by the ’601 patent.

128. Second, Jeffries describes how an isolated container 122 (an “*isolated computing environment*”) “*uses the host operating system.*”

129. Jeffries describes implementing “a container 122...as a process container” that includes a view of certain parts of the host system 102’s operation system. Jeffries, 10:23-25.

For a process container, the application processes within the container run as if they were operating on their own individual system (e.g., computing device), which is accomplished using namespace isolation. Host operation system 102 implements namespace isolation. Namespace isolation provides processes in a container a composed view consisting of **the shared parts of host operating system 102 and the isolated parts of the operating system** that are **specific to each container** such as

filesystem, configuration, network, and so forth.

Jeffries, 10:25-34.

130. Because Jeffries describes “shared parts of host operating system 102” that are used by the containers 122, as well as “parts of the operating system that are specific to each container,” it shows that containers 122 “*use the host operating system*” 102, as claimed. Jeffries, 10:31-33.

131. In addition, Jeffries describes the same method of isolating containers as the ’601 patent, specifically using namespace isolation. The ’601 patent describes “enforc[ing] the segregation using techniques such as namespace isolation such that processes running in the sandboxed computing environment in the second memory space are restricted from accessing the first memory space including the workspace” while allowing “the sandboxed computing environment to share access to the host operating system and host operating system resources.” ’601 patent, 5:57-67. Similarly, Jeffries explains that the “[h]ost operation system 102 implements namespace isolation” to isolate containers 122. Jeffries, 10:28-29. Jeffries’ isolated containers 122 access “shared parts of host operating system 102” and are enforced via “namespace isolation.” Jeffries, 10:28-31. Therefore, the isolated containers 122 use the host operating system 102 just as the ’601 patent’s “sandboxed computing environment” has access to the host operating system, ’601 patent, 5:66-67.

132. In addition, the isolated containers 122 use the host operating system in

other more specific ways. For example, “the host operating system monitors application activity in the container while the application is accessing a resource.” Jeffries, 1:30-32. Also, the “container manager 118” which is illustrated within the host operating system 102 in Jeffries’ Figure 1, “is responsible for activating one or more containers 122(1),..., 122(n).” Jeffries, 9:67-10:1, 14:40-42.

133. The patent owner added the phrase “*uses the host operating system*” during prosecution to overcome a non-final office action citing Azab (Ex.1034). In an office action response dated September 14, 2021, the patent owner argued that Azab did not render the claims obvious because it “does not rely on any software component in the host environment (i.e., an OS or a hypervisor).” ’601 File History, 217 (quoting Azab, Abstract). In addition, the patent owner argued that Azab’s “security manager for the isolated environment is separate and independent from the host OS.” ’601 File History, 217.

134. Unlike Azab, Jeffries’ isolated containers use the host operating system in the same way as the ’601 patent describes and as the patent owner characterized during prosecution. While Azab’s “security manager is separate and independent from the host OS,” Jeffries’ container manager 118 is an integral part of the host operating system 102, as shown in Fig. 1 below.

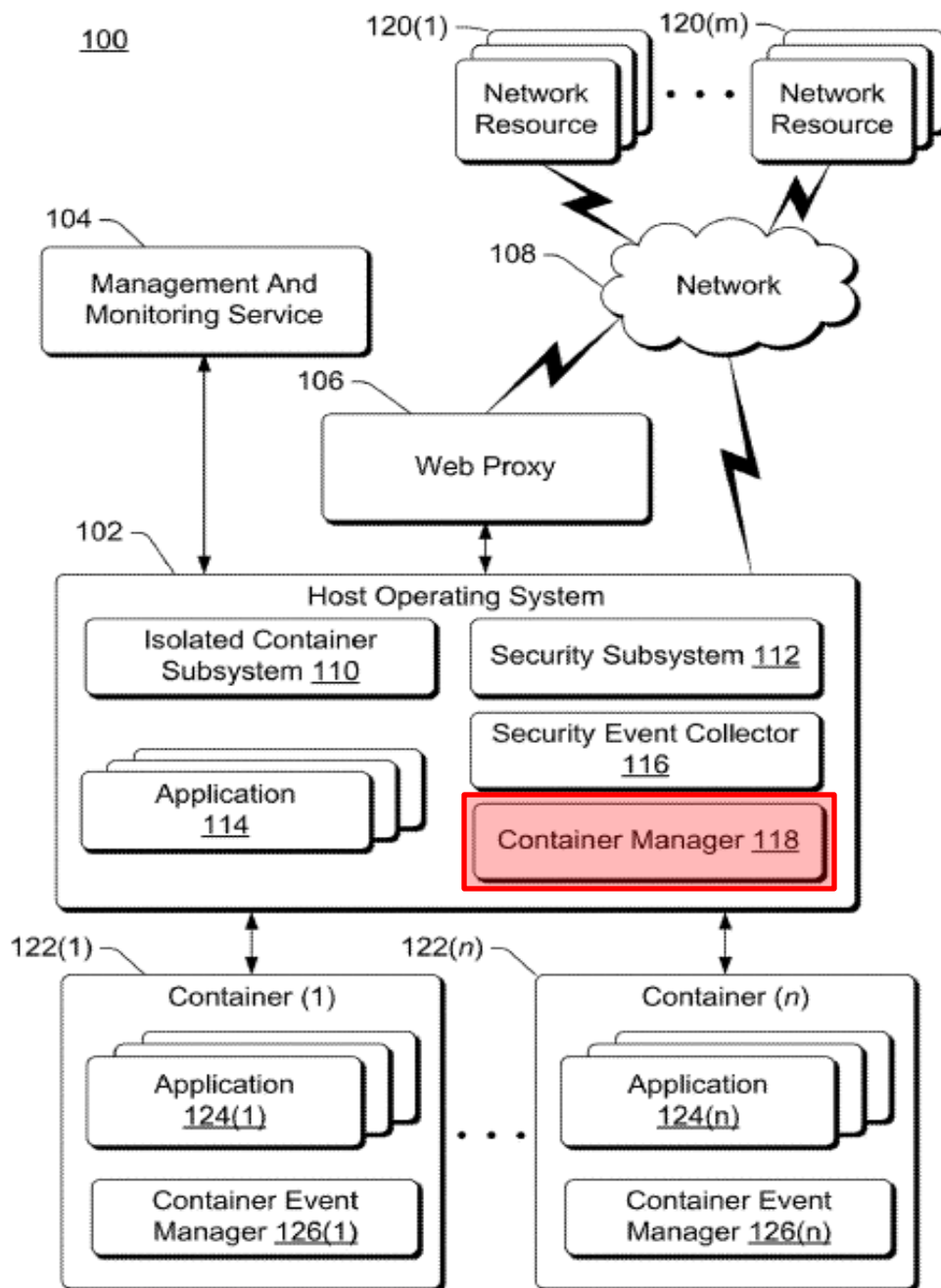


Fig. 1

Jeffries, Fig. 1 (annotated)

135. The container manager 118 “manages the scheduling of containers 122(1), . . . , 122(n) in the system 100 and determines which containers 122(1), . . .

, 122(n) are run **on the host operating system 102 at what times.**” Jeffries, 9:63-67. Thus, unlike Azab’s isolated computing environments which do not “rely on any software component of the host environment,” Jeffries’ containers 122 use the host operating system 102 as claimed.

136. The containers 122 in Jeffries are also managed by several other components within the host operating system 102. The isolated container subsystem 110 “causes the host operating system 102 to activate one or more of containers 122(1), . . . , 122(n) and allow the one or more activated containers to access the untrusted network resource.” Jeffries, 9:2-6. In addition, the “security subsystem 112” and “security event collector 116” monitor “container activity resulting from resource access in one or more of containers 122(1), . . . , 122(n).” Jeffries, 9:7-15.

137. Thus, the containers 122 use the host operating system for both container monitoring and initial activation and “run on the host operating system 102” while protecting the host operating system 102 from malware through isolation.

138. Third, Jeffries describes “*the isolated computing environment enables operation of a second set of one or more applications or processes via a second memory space.*”

139. Jeffries illustrates a set of applications 124(1) running in container 122(1), as highlighted in purple below.

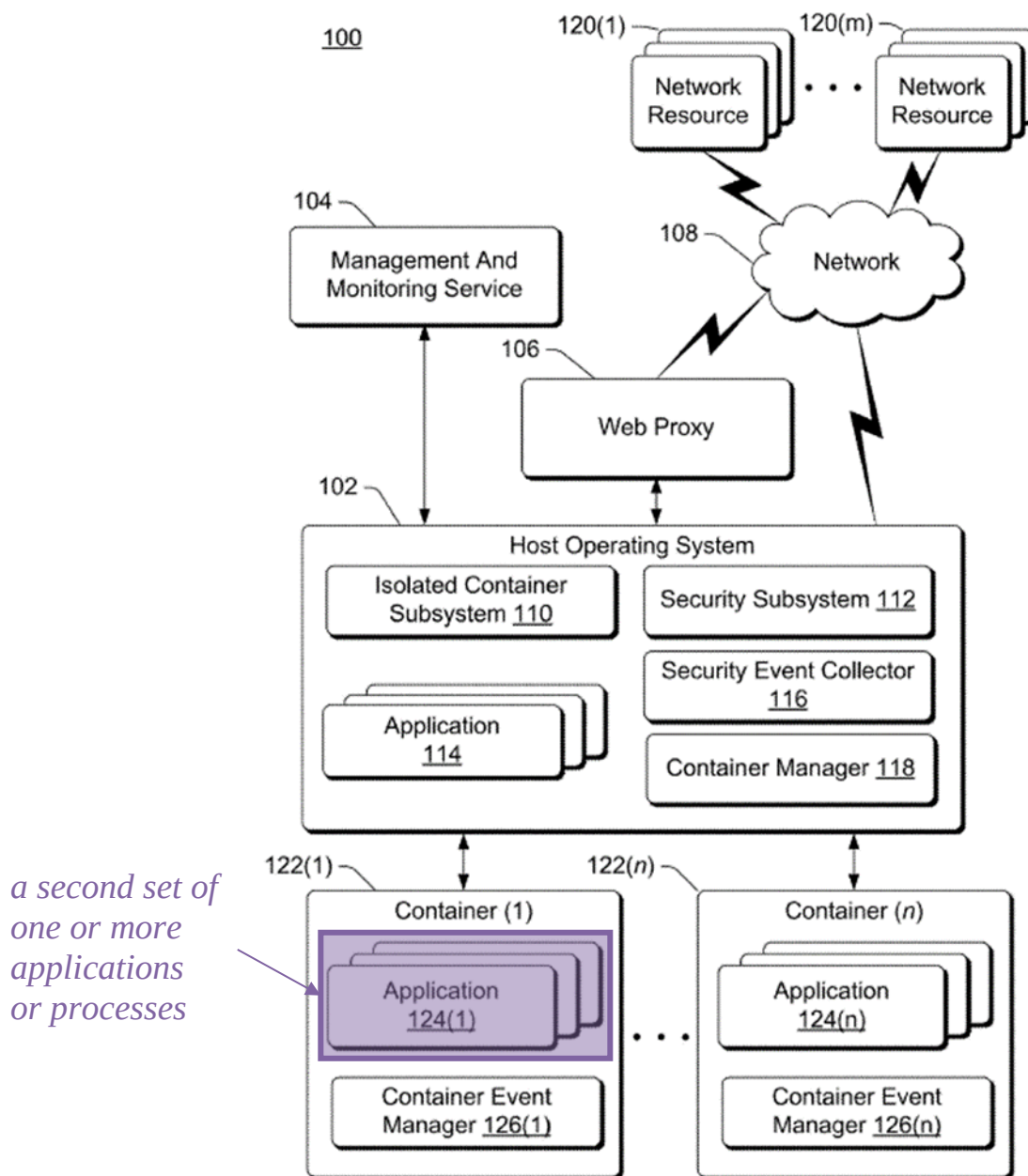


Fig. 1

Jeffries, Fig. 1 (annotated)

140. The purpose of containers 122 is to isolate applications 124 from the remaining portions of the host operating system 102. See Jeffries, 2:46-49 (“[U]sing containers in conjunction with the event monitoring techniques described herein

protects the computing device from attacks by isolating web browsers, related applications, and operating system components such as the kernel, the file system and the network from untrusted resources.”), 1:52-54 (“an application is running in a container that is isolated from a host operating system of a device.”), 10:25 (“For a process container, the application processes within the container run as if they were operating on their own individual system (e.g., computing device).”).

141. The applications 124 within containers 122 are associated with separate memory space isolated from the memory space of the host operating system 102, including applications 114. Jeffries describes “isolated parts of the operating system that are specific to each container” which “may also be referred to as container subsystems.” Jeffries, 10:34-36.

142. Jeffries also explains that “[a] container base image contains information required to create and activate an isolated container.” Jeffries, 15:14-15. The base image “includes information regarding an amount of allocated resources that may be required by container subsystems 206, **such as memory**, processors, disks, or networks, when container 122 is active.” Jeffries, 15:27-30. This memory allocated to an isolated container, or to the applications 124 within an isolated container, corresponds to “*a second memory space*” isolated from the remainder of the memory.

143. Jeffries also describes “container configuration files” associated

specifically with a container 122 as well as “**container memory**.” See Jeffries, 17:7-20, 21:6-7. These disclosures show that Jeffries allocates specific memory (“*a second memory space*”) for the operation of applications 124 within a container 122.

144. As I discussed previously for [1.3], a POSITA at the time would have understood that an application running in an isolated container would have an associated memory space. See, e.g., Yu, ¶3 (“Typically, each application executes in a corresponding memory space, which may be referred to as application memory space.”); Song, Abstract (describing “a memory space of a corresponding application”); Tucker, Abstract (describing “a continuous region of virtual memory space of a corresponding endpoint computer application”).

145. Thus, Jeffries describes creating and isolating containers 122 (“*implement[ing] an isolated computing environment*”) that access portions of the host operating system 102 and use the host operating system 102 for monitoring and activation (“*that uses the host operating system*”). Jeffries also describes a set of applications 124 within the isolated containers 122 (“*wherein the isolated computing environment enables operation of a second set of one or more applications or processes*”) and explains that container memory is allocated to these applications 124 (“*via a second memory space*”) as claimed.

[1.4.2] wherein the second set of one or more applications or processes comprises a multi-user interactive software application;

146. The combination of Jeffries and LaBine renders this limitation obvious.

147. As described above, *see* limitation [1.4.1], Jeffries describes a set of applications 124 that are executed within the isolated containers 122. These “applications 124(1),..., 124(n) are thus useable to access untrusted resources in one or more of containers 122(1),..., 122(n).” Jeffries, 10:58-61; *see also* Jeffries, 3:48-51 (“the host operating system considers every resource to be untrusted and forces resources to be accessed within an isolated container.”).

148. Jeffries does not list every type of application 124 that may be executed in the isolated containers 122 in detail. Rather, it provides a non-limiting list of broad categories of the types applications 124 which may be executed in the isolated containers 122, emphasizing the broad applicability of its techniques. *See* Jeffries, 9:29-33 (listing types of applications 114, including “a web browser application” in addition to “e-mail applications, word processing applications, spreadsheet applications, visual presentation applications, and the like”); *see also* Jeffries, 10:54-58 (explaining that applications 124 “correspond to instances of individual ones of the applications 114.”). A POSITA would have understood that Jeffries’ techniques of executing applications within isolated containers would be applied to any type of application to predictably improve computer and network security. *See supra* §IX.D.

149. LaBine describes a “*multi-user interactive software application.*” LaBine describes a video conferencing software that allows participants of the video conference to remotely access and control the host user’s computer. *See generally*

LaBine, Abstract, ¶¶78-84.

150. LaBine's software is referred to as the "SMART Bridgit Conferencing Software," and is video conferencing software implemented in a "computing environment 20" including "a pair of computers 24a, 24b communicating over a network 28." LaBine, ¶¶43, 45. The computing environment 20 is illustrated in Figure 1, below.

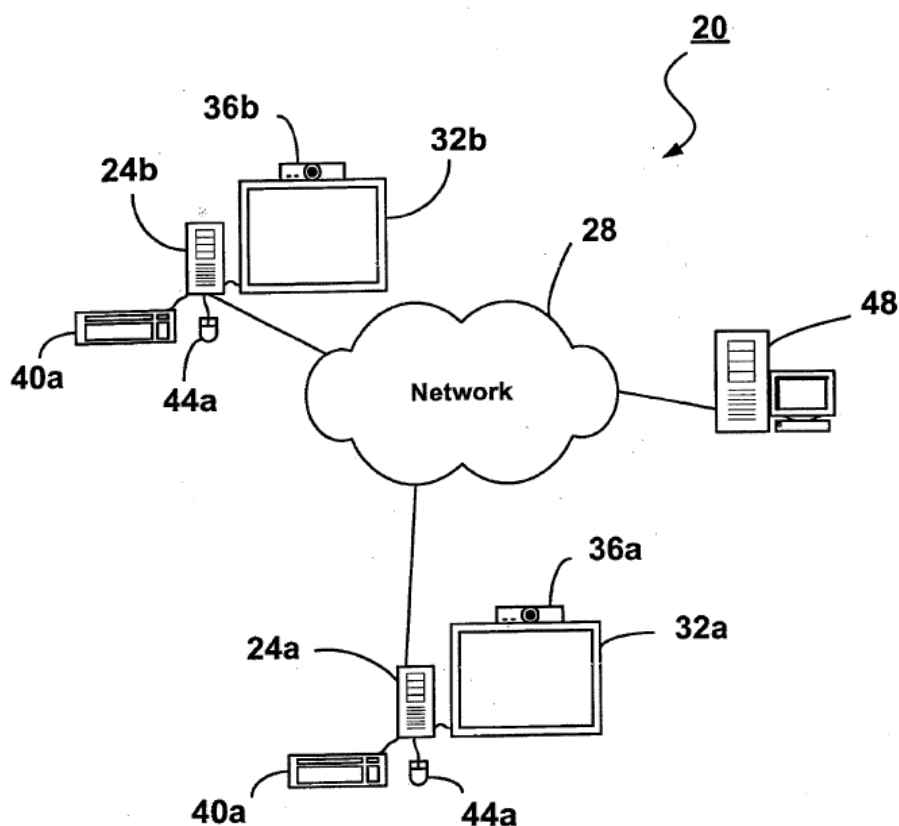


Fig. 1

LaBine, Fig. 1

151. "The computers 24a, 24b...communicate with web cameras 36a,

36b..., keyboards 40a, 40b, and mice 44a, 44b.” LaBine, ¶43. The “Bridgit Conferencing Software...allows a conference to be set up between personal computers with one personal computer at a time being designated as a host, or ‘presenter’ computer and the remaining personal computers being designated as client, or ‘participant’ computers.” LaBine, ¶45. In one scenario, “the host computer desktop is presented on the touch screens 32a and 32b” and “[t]he client computers can be enabled to control the host computer to a degree.” LaBine, ¶45.

152. LaBine explains how, during a conferencing session, a client computer may send a “REMOTE CONTROL request” to the conferencing server 48. LaBine, ¶78. If “[t]he host computer immediately approves the request,” it sends “a REMOTE CONTROL RESPONSE...to the conferencing server 48.” LaBine, ¶80. The conferencing server 48 then “forwards the REMOTE CONTROL RESPONSE message to the requesting client computer to complete the enablement of remote control by the requesting client computer.” LaBine, ¶81. With remote control of the host computer, “remote user input events on the client computer..., such as mouse, keyboard and pen events, are received at the conferencing server 48 from the remote controller and routed to the host computer.” LaBine, ¶84. The host computer “treats the remote user input events as though they were input locally.” LaBine, ¶84.

153. LaBine illustrates the graphical user interface displayed to a remote-control user of LaBine’s system with Figure 22, shown below.

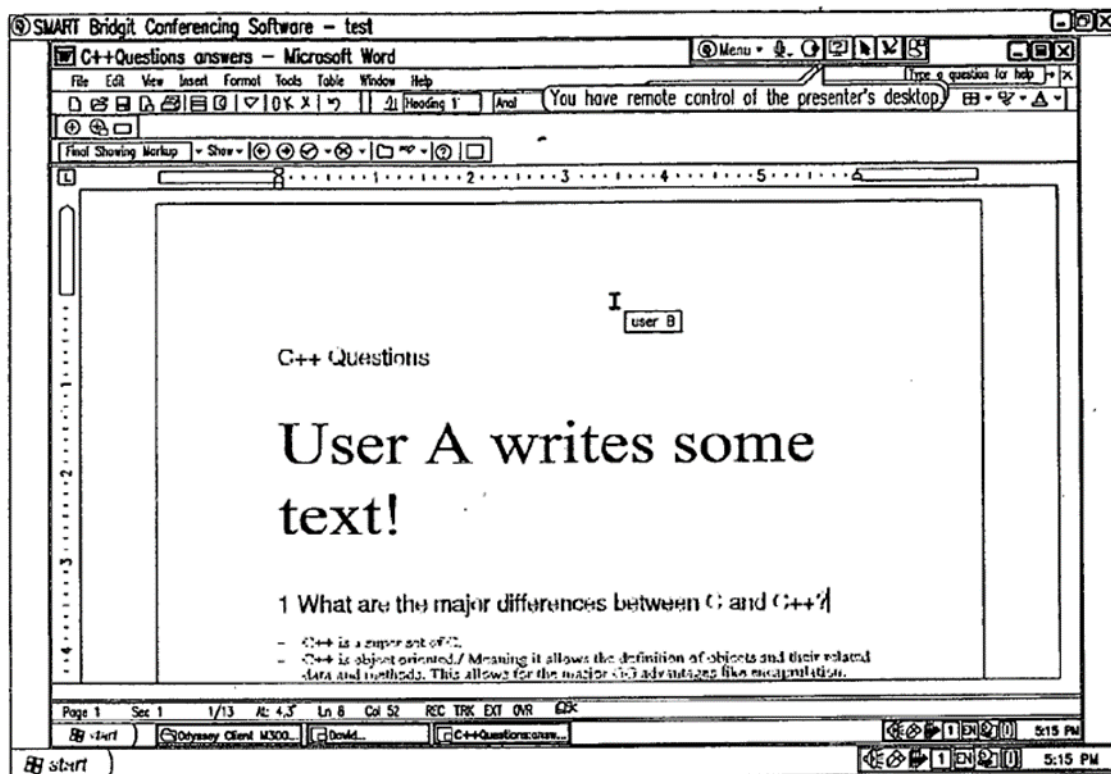


FIG. 22

LaBine, Fig. 22

154. LaBine explains that “In Fig. 22, there are two indications to indicate to the viewer ‘User B’ that that user has control of the host or presenter’s desktop...shown isolated and magnified in FIGS. 23 and 24.” LaBine, ¶81. Figure 23 shows a message that says, “You have remote control of the presenter’s desktop” as shown below.

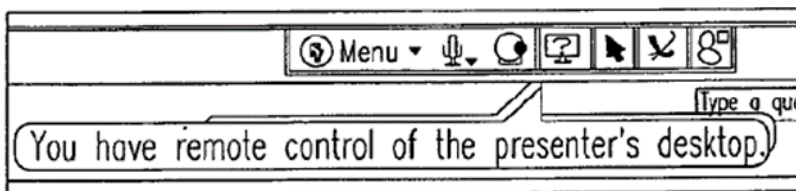


FIG. 23

LaBine, Fig. 23

Figure 24 shows the remote user that has remote control of the presenter's desktop, in this example, "user B."

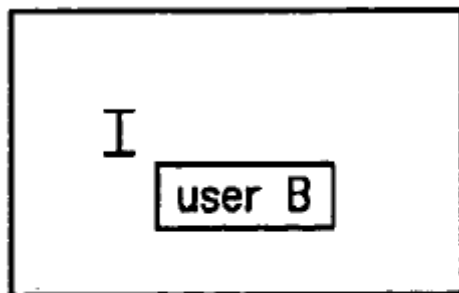


FIG. 24

LaBine, Fig. 24

155. Therefore, LaBine's video conferencing application allows one user to access and control another user's computer remotely.

156. LaBine's video conferencing software is consistent with the '601 patent's description of "*multi-user interactive software application*" because it is "an application that allows one or more remote machines to control (e.g., access, view, interact with, download from, upload to, and/or modify) an application running on

the host computer system.” ’601 patent, 2:18-23; *see also* §IX.G.2, [2.1]-[2.2].

157. As described previously, *see* §IX.D, a POSITA would have been motivated to implement a software application such as the one described by LaBine within an isolated container as described by Jeffries. In the combination, LaBine’s application would be installed on the computer system and executed within Jeffries’ isolated container. LaBine’s interactive software allows unknown users, i.e., untrusted sources, to access and control a host computer. LaBine, ¶84. Jeffries’ system is designed to increase security when executing just such applications via container isolation. Jeffries, Abstract, 2:46-49.

158. Thus, the combination of Jeffries and LaBine describes a software application within an isolated container (“*the second set of one or more applications or processes*”) that allows interactions with other users, including allowing a remote user to access and control the host computer (“*a multi-user interactive software application*”).

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

159. Jeffries renders this limitation obvious.

160. Jeffries describes a container manager 118 which, in conjunction with an operating system secure kernel, enforces isolation between the container 122 and the remainder of the system, as explained below. The container manager 118 is shown in Figure 2, below, and highlighted in dark green.

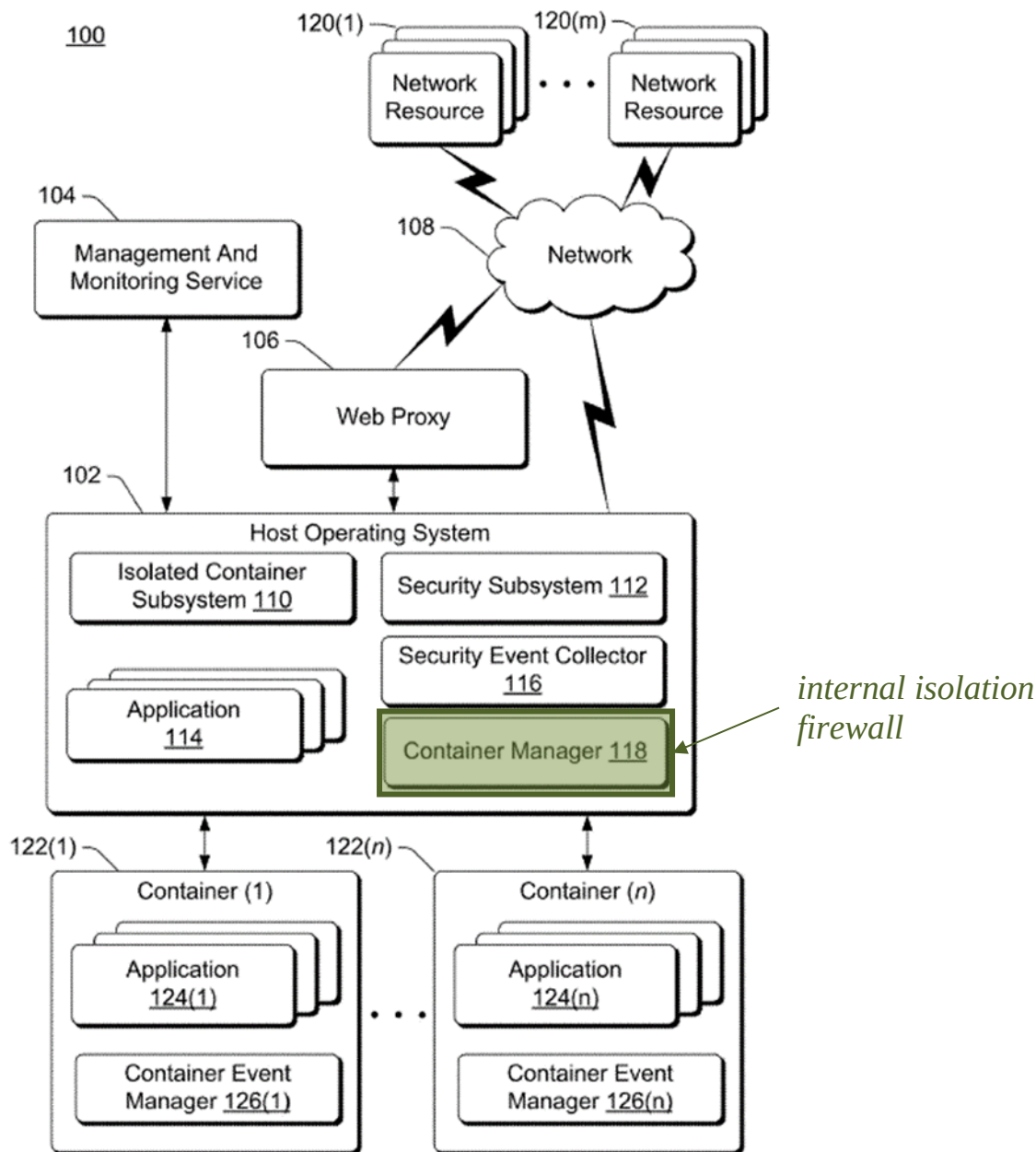


Fig. 1

Jeffries, Fig. 1 (annotated)

[C]ontainer manager 118 is configured to **quarantine the container 122** and **prevent the container 122 from further accessing resources**. By blocking the container

122 from further accessing resources, the container manager 118 prevents malicious activity occurring in the container from gaining file system access.

Jeffries, 16:29-34.

161. In addition, a secure kernel helps to enforce the separation between the host operating system 102 and the container 122.

The communicative connection between the host operating system 102 and the container 122 implements a secure kernel to maintain integrity of information passed from the container 122 to the host operating system 102.... Accordingly, **when malicious software attempts to access off-limit data or memory locations, the secure kernel causes an exception or interrupt to occur.**

Jeffries, 12:7-19.

162. Jeffries' description of the functions performed by the container manager 118 and secure kernel are consistent with the '601 patent's description of an "*internal isolation firewall*." While, the '601 patent does not define the term "*internal isolation firewall*," it describes it in terms of the functions it performs. For example, "[t]he internal isolation firewall may prevent communication between the isolated computing environment and the workspace." '601 patent, 1:67-2:2. Similarly, "[t]he segregation of the trusted memory space and the untrusted memory space may be enforced by the internal isolation firewall." '601 patent, 3:38-40; see also '601 patent, 6:63-66, 13:15-23, 14:8-14.

163. The container 118 and secure kernel described in Jeffries corresponds to an “*internal isolation firewall*” as claimed because they perform the same functions of preventing communication between the isolated computing environment and the workspace, as explained above.

164. In summary, Jeffries describes isolating the containers 122 from the remainder of the host operating system 102 (“*isolat[ing] the isolated computing environment from the workspace*”) with a container manager 118 in conjunction with the secure kernel (“*using an internal isolation firewall*”).

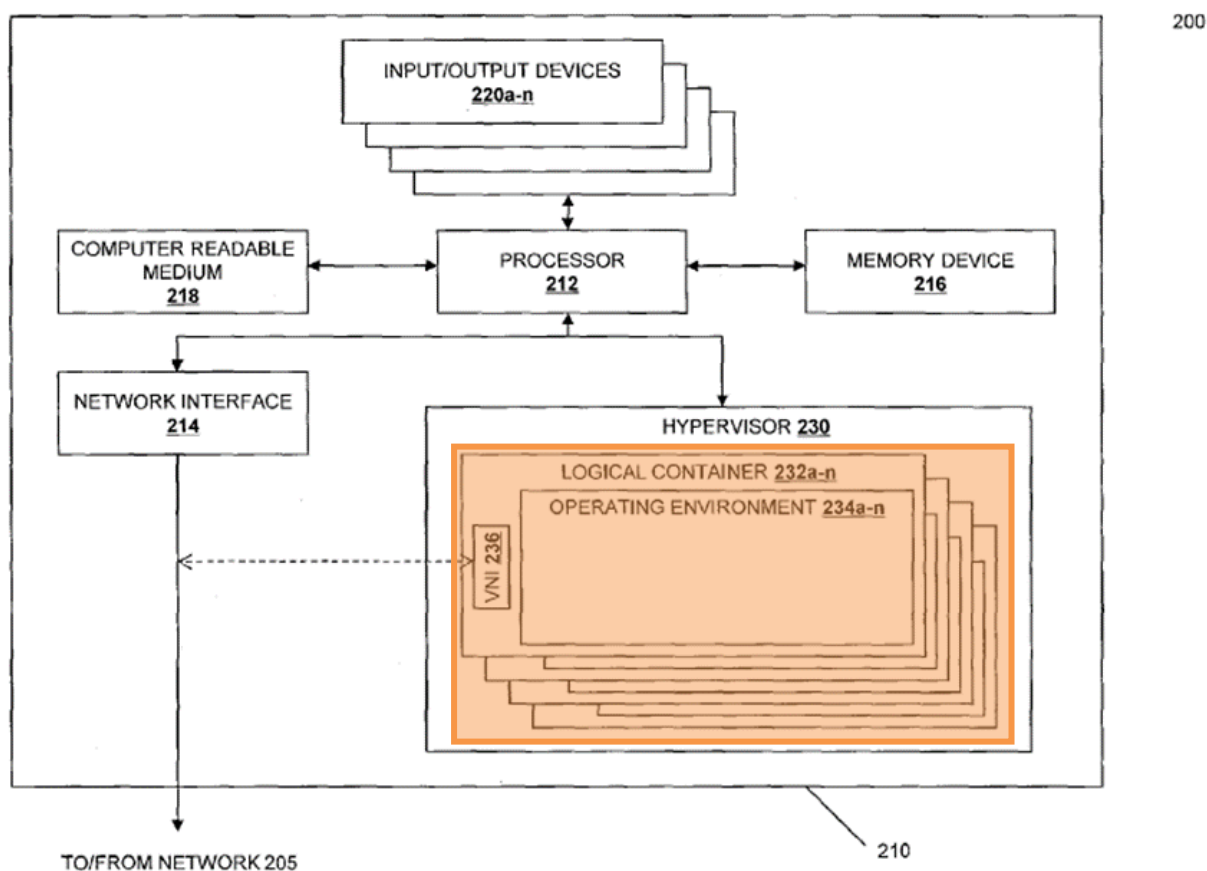
[1.6] authenticate the isolated computing environment with an authentication device; and

165. The combination of Jeffries and Ishaya renders this limitation obvious.

166. Jeffries generally describes the importance of authenticating the host operating system 102 as well as users of isolated containers 122. *See, e.g.*, Jeffries, 6:55-57 (“requiring authentication from host operating system 102”), 14:47-52, 14:57-58 (describing “an individual user logged into host operating system 102” and configuring containers based on the “logged-on user’s identity.”), 15:12-13, 15:25-26.

167. Ishaya describes authenticating isolated computing environments. Ishaya “relates generally to using identity information for authentication of virtual machines.” Ishaya, ¶1. Figure 2, below, illustrates “an information processing system 210” that may be “a personal computer (e.g., a desktop computer or a

portable computer such as, for example, a laptop computer).” Ishaya, ¶35. As shown in Figure 2, the personal computer includes “a number of logical containers 232a-n” that “contain associated operating environments 234a-n.” Ishaya, ¶41 (highlighted in orange below).



Ishaya, Fig. 2 (annotated)

168. Like the '601 patent and Jeffries, the containers 232 described in Ishaya may be isolated from the remainder of the system. See Ishaya, ¶41 (“A corresponding operating environment 234 could enforce separation between users and processes such that each process or group of processes appeared to have sole

access to the resources of the operating system.”). Ishaya further explains that it refers to the containers as “‘Virtual Machines,’ or ‘VMs,’ but this is simply one implementation.” Ishaya, ¶41. Ishaya explains that isolated containers may communicate with a network via a “virtual network interface 236” which “appears as one or more virtual network interface cards.” Ishaya, ¶42. As a result, “each VM,” the term Ishaya uses to refer to containers as well, *see* Ishaya, ¶41, “includes identification data for use naming, interacting, or referring to the VM” and which may “include the Media Access Control (MAC) address, the Internet Protocol (IP) address, and on or more unambiguous names or identifiers.” Ishaya, ¶43.

169. Ishaya further explains the process by which a container is instantiated, which includes verification based on the MAC address or IP address of the container, with reference to Figure 7, shown below.

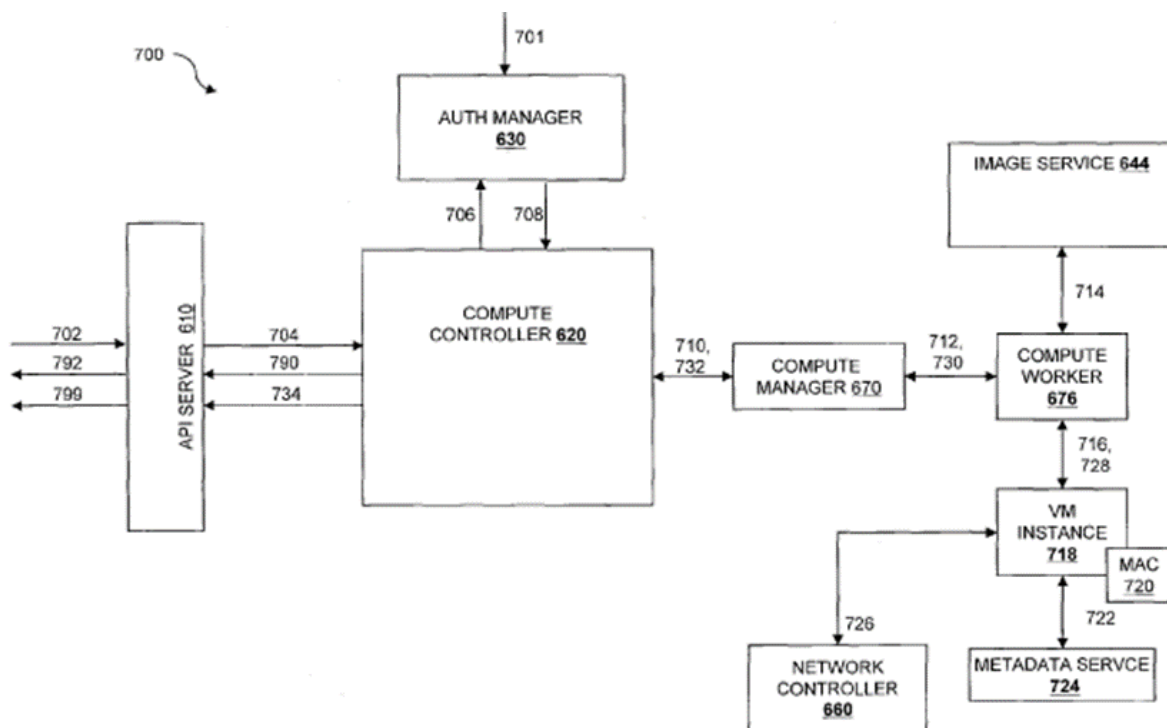


FIGURE 7

Ishaya, Fig. 7

170. At times 702 and 704, a user requests that a container is created. Ishaya, ¶¶177-178. At time 706, the “Auth Manager 630” determines whether the user has permission to create the type of container requested. Ishaya, ¶¶179-180. If the user has the required permission, at time 710, “a message is sent to the Compute Manager 670 to instantiate the requested virtual machine at time 710,” Ishaya, ¶180, and the VM is instantiated, Ishaya, ¶181. “Also at time 716, the Computer Worker assigns the VM instance 718 a MAC address 720” which is “selected from a pool of MAC addresses.” Ishaya, ¶181. The MAC address may be “replaced with...an IP address.” Ishaya, ¶181. Then, “[a]t time 722, the newly instantiated VM instance 718 begins the process of obtaining its initial configuration by interacting with Metadata Service

724,” Ishaya, ¶183, and “authenticates to the Metadata Service 724 via an identity” such as its MAC address or IP address “as described above,” Ishaya, ¶184. After this initial authentication, the VM instance receives “possession-based credentials” that it can use “for subsequent authentication to other resources.” Ishaya, ¶184. For example, at time 726, “[t]he VM instance 718 again provides its MAC address 720 in order to authenticate itself to the Network Controller 660.” Ishaya, ¶185.

171. This process of configuring a container so that it can authenticate itself to other devices on the network, such as the Network Controller 660, provides multiple advantages, as explained by Ishaya. For example, authenticating an isolated container as described in Ishaya allows different devices, including other virtual machines or containers to “determine whether or not to trust information from [the container] based on the identity value of” the container. Ishaya, ¶189. In addition, it increases security even while interoperating with different devices by allowing “controllers from different vendors or entities” to “authenticate to each other.” Ishaya, ¶190. Ishaya lists multiple other advantages stemming from container authentication. *See, e.g.*, Ishaya, ¶¶193-196.

172. For at least all these reasons, *see* §IX.E, a POSITA would have been motivated to implement container isolation within Jeffries’ system by authenticating containers 122 with an authentication device, such as a device similar to the Network Controller 660 described in Ishaya, using a unique device identifier, such as a MAC

address or IP address. In the combination, Jeffries' isolated containers 122 would be instantiated using the steps described in Ishaya, see Ishaya, ¶¶177-187, including assigning each container 122 a MAC or IP address, after which the container 122 would be authenticated to an authentication device, such as a metadata service or network controller.

173. The implementation of an authentication server such as the one described by Ishaya with Jeffries' system would be an obvious design choice to a POSITA. In one implementation, the management and monitoring service 104 of Jeffries would include functionality for maintaining a list of available MAC addresses, assigning a MAC address to an isolated container after it is created, and assigning an IP address to the isolated container. The isolated container in Jeffries may then authenticate itself to the management and monitoring service using the MAC address or IP address as described in Ishaya. Alternatively, these functions may be implemented in any of the web proxy 106, security subsystem 112, or isolated container subsystem 110 of Jeffries' system. In addition, a POSITA may implement an additional server or network device dedicated to authentication of isolated containers.

174. Therefore, the combination of Jeffries and Ishaya describes authenticating an isolated container to a network controller using, for example, a MAC or IP address of the isolated container (*authenticat[ing] the isolated computing*

environment with an authentication device).

[1.7] 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.

175. The combination of Jeffries, LaBine, and Ishaya renders this limitation obvious.

176. In the combined system, a multi-user interactive software application, such as the video conferencing application described by LaBine, *see* LaBine, ¶¶45, 78-84, is executed within an isolated container, as described in Jeffries, *see* Jeffries, 10:55-56, 2:46-49.

177. LaBine's software, which is a multi-user interface software application, *see* §IX.G.1 (limitation [1.4.2]), is designed to *send data to an untrusted destination*. Specifically, LaBine describes a video conferencing software that allows participants of the video conference to remotely access and control the host user's computer. LaBine, Abstract, ¶¶78-84. After receiving a "REMOTE CONTROL RESPONSE message," LaBine, ¶80, a remote user may "input events on the client computer" and the host computer "treats the remote user input events as though they were input locally," LaBine, ¶84. The users of client computers in communication with the host may be any user, known or unknown, and yet "the host computer enables **any** client computer during the conference to be granted remote control of the host computer." LaBine, ¶75. Therefore, LaBine's application allows

communication with untrusted destinations (e.g., remote client computers and their associated users), and anticipates various security risks this presents. *See* LaBine, ¶¶46, 48, 50.

178. Similarly, applications within isolated containers in Jeffries send data to an untrusted destination. For example, Jeffries explains that “applications 124(1),..., 124(n) are thus useable to access untrusted resources in one or more of containers 122(1),..., 122(n).” Jeffries, 10:58-61; *see also* 3:48-51 (“the host operating system considers every resource to be untrusted and forces resources to be accessed within an isolated container.”).

179. Jeffries further explains that this data communication with an untrusted destination may occur via the proxy device. The web proxy 106 is positioned between the host computer (including the host operating system 102 and the isolated containers 122) and external network resources 120, as shown in Figure 1 below, highlighted in dark red.

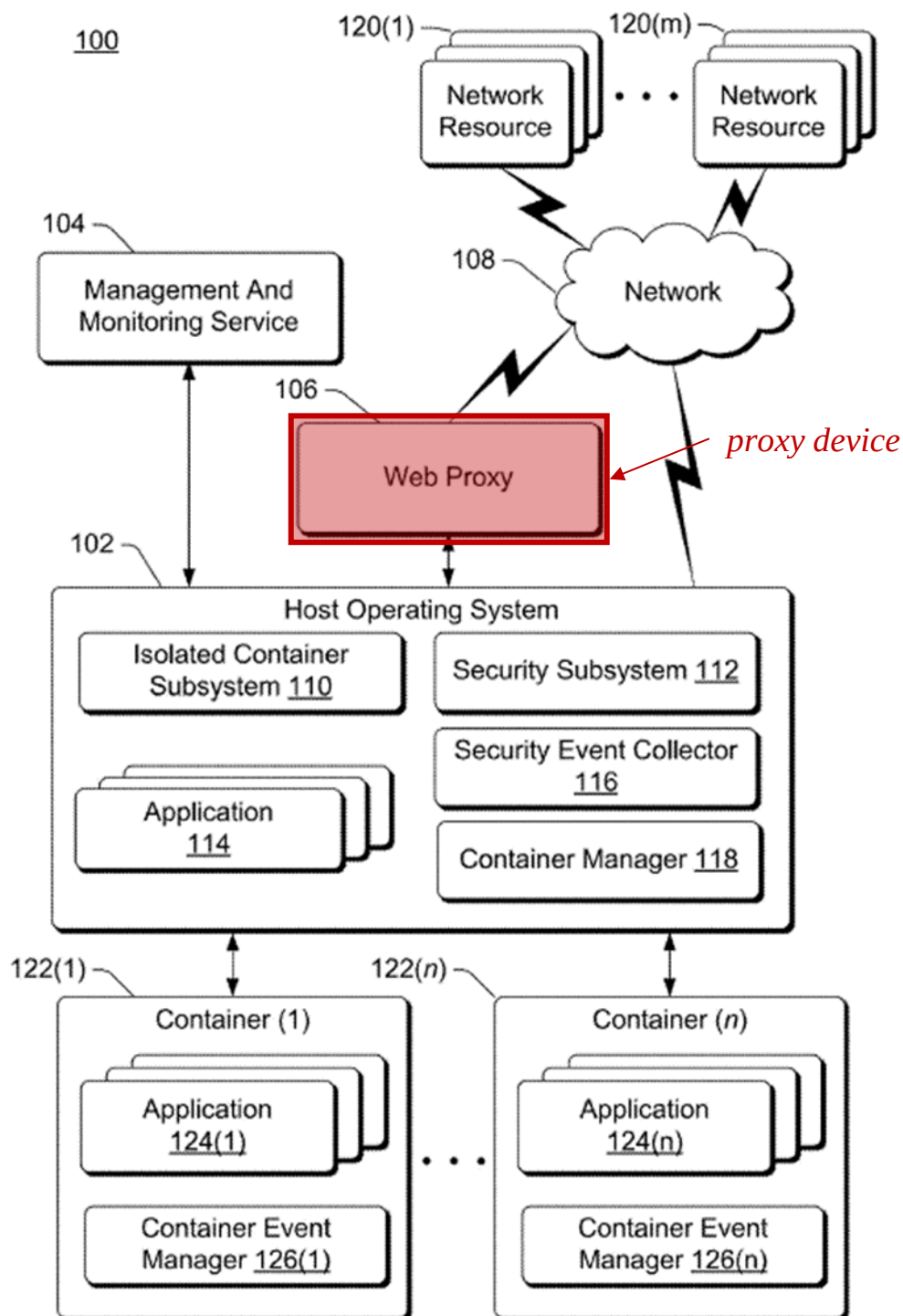


Fig. 1

Jeffries, Fig. 1 (annotated)

180. Jeffries explains that the web proxy 106 “controls access to one or more

network resources 120(1),..., 120(m) accessed remotely from the computing device implementing the host operating system 102.” Jeffries, 6:50-54. In one embodiment, the network resources 120 may be resources of a local network requiring authentication prior to access. See Jeffries 6:53-57 (“For example, in one or more embodiments web proxy 106 controls access to one or more resources accessed from network 108 by requiring authentication from host operating system 102.”), 16:36-37 (describing “network resources 120(1),..., 120(m)” as “shared network resources”). But in other embodiments, the network resources 120 alternatively represent “web sites, web applications, emails, [or] documents” (Jeffries, 6:66-67) that are treated as untrusted resources that are “access[ed] . . . in a container 122(1),..., 122(n)” (Jeffries, 7:53-54). Therefore, Jeffries explains that applications 124 within isolated containers 122 would transmit data to the network resources 120, which may be untrusted, via the web proxy 106. For example, a browser application within an isolated container 122 may send a website request to an untrusted website of the network resources 120 via the web proxy 106.

181. In addition, according to the combination, the isolated container 122 is authenticated, as described by Ishaya. See §IX.G.1 (limitation [1.6]). Ishaya further explains that the isolated container is not permitted to communicate with any source, including an untrusted source, until after it is authenticated. For example, at Figure 8, Ishaya describes “a method instantiating, launching and authenticating a VM

instance.” Ishaya, ¶20. At step 820, the penultimate step, the “VM uses the MAC address to authenticate itself to various configuration resources.” Ishaya, Fig. 8, ¶192]. After this authentication step, at step 822, the final step, the “VM is configured and ready for regular use.” Ishaya, Fig. 8, ¶192]. In the combination, the isolated and authenticated container is then used for access untrusted sources. See §IX.G.1 (limitation [1.4.1]). Therefore, Ishaya explains that an application within a container may communicate with an untrusted source after the container is authenticated.

182. Thus, the combination renders obvious transmitting data from a video conferencing application to unknown destinations, as described by Jeffries and LaBine (*send data to an untrusted destination from the multi-user interactive software application*), transmitting data to the untrusted source through the web proxy 106, as described by Jeffries (*via a proxy device*), after the container 122 has been authenticated, as described by Ishaya (*when the isolated computing environment has been authenticated*).

2. Claim 2

[2.0] The system of claim 1,

183. The combination of Jeffries, LaBine, and Ishaya renders obvious *the system of claim 1*.

[2.1] 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,

184. The combination of Jeffries and LaBine renders this limitation obvious.

185. As described above with respect to limitation [1.7], in the combined system, the video conferencing application described by LaBine, *see* LaBine, ¶¶45, 78-84, is executed within an isolated container, as described in Jeffries, *see* Jeffries, 10:55-56, 2:46-49. As previously explained, LaBine’s video conferencing application corresponds to the *multi-user interactive software application* claimed. *See* limitation [1.4.2].

186. LaBine further explains that its system allows *a remote* machine to control the applications running on the host machine, as explained below. LaBine’s “Bridgit Conferencing Software” “allows a conference to be set up between personal computers with one personal computer at a time being designated as a host, or ‘presenter’ computer and the remaining personal computers being designated as client, or ‘participant’ computers.” LaBine, ¶45. The client computer may “input events on the client computer..., such as mouse, keyboard and pen events” and the host computer “treats the remote user input events as though they were input locally.” LaBine, ¶84. Thus, LaBine’s “client, or ‘participant’ computers” are *remote machine[s]* that have remote control over the host computer.

187. LaBine’s client computer (*a remote machine*) is also allowed to control

functions of applications running on the host computer in LaBine via LaBine's "Bridgit Conferencing Software." LaBine, ¶45.

188. For example, LaBine describes allowing remote viewers to access the Microsoft Word application of the host computer. *See, e.g.*, LaBine, Figs. 20-25. At Figure 22, LaBine shows a "GUI available to a viewer that has remote control of a presenter's shared desktop." LaBine, ¶81.

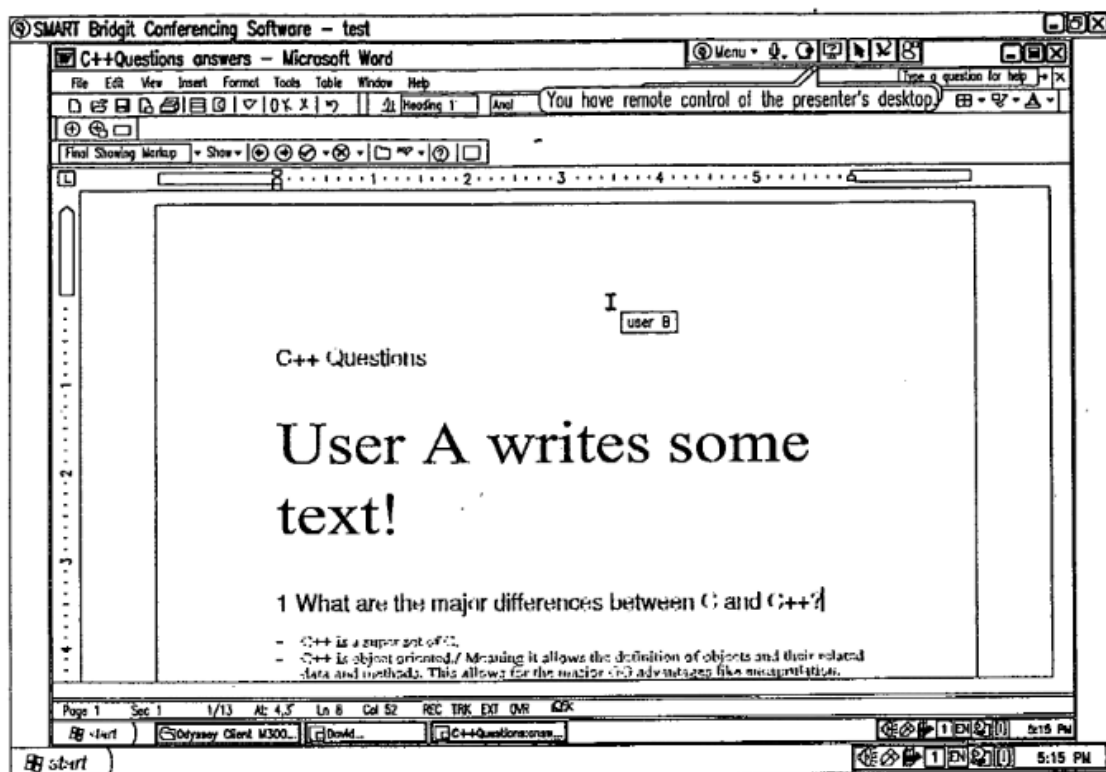


FIG. 22
LaBine, Fig. 22

189. Figures 23 and 24 show that that the remote user has "control of the presenter's desktop." LaBine, Fig. 23. While the remote user has control of the presenter's computer, he or she may control the cursor of the presenter's computer

and the “[h]ost computer displays to its user a cursor label with the name of the user on the client computer than [sic] has remote control.” LaBine, ¶81. In the example shown in Figures 22-25, the remote user may also control the host computer by adding text within the Microsoft Word application of the host computer. *See* LaBine, Figs. 22-25.

190. In the combination, the Microsoft Word application would correspond to an application running within the isolated containers 122 of Jeffries because it is accessed by untrusted sources. *See* Jeffries, 10:59-60 (Applications within isolated containers 122 “are thus useable to access untrusted resources.”). In addition, Jeffries’ system enforces isolation between its containers. *See* Jeffries, 9:67-10:5 (describing preventing one container 122 from accessing another). However, Jeffries explains that multiple applications may be run in a single container, as shown below. *See* Jeffries, Fig. 1.

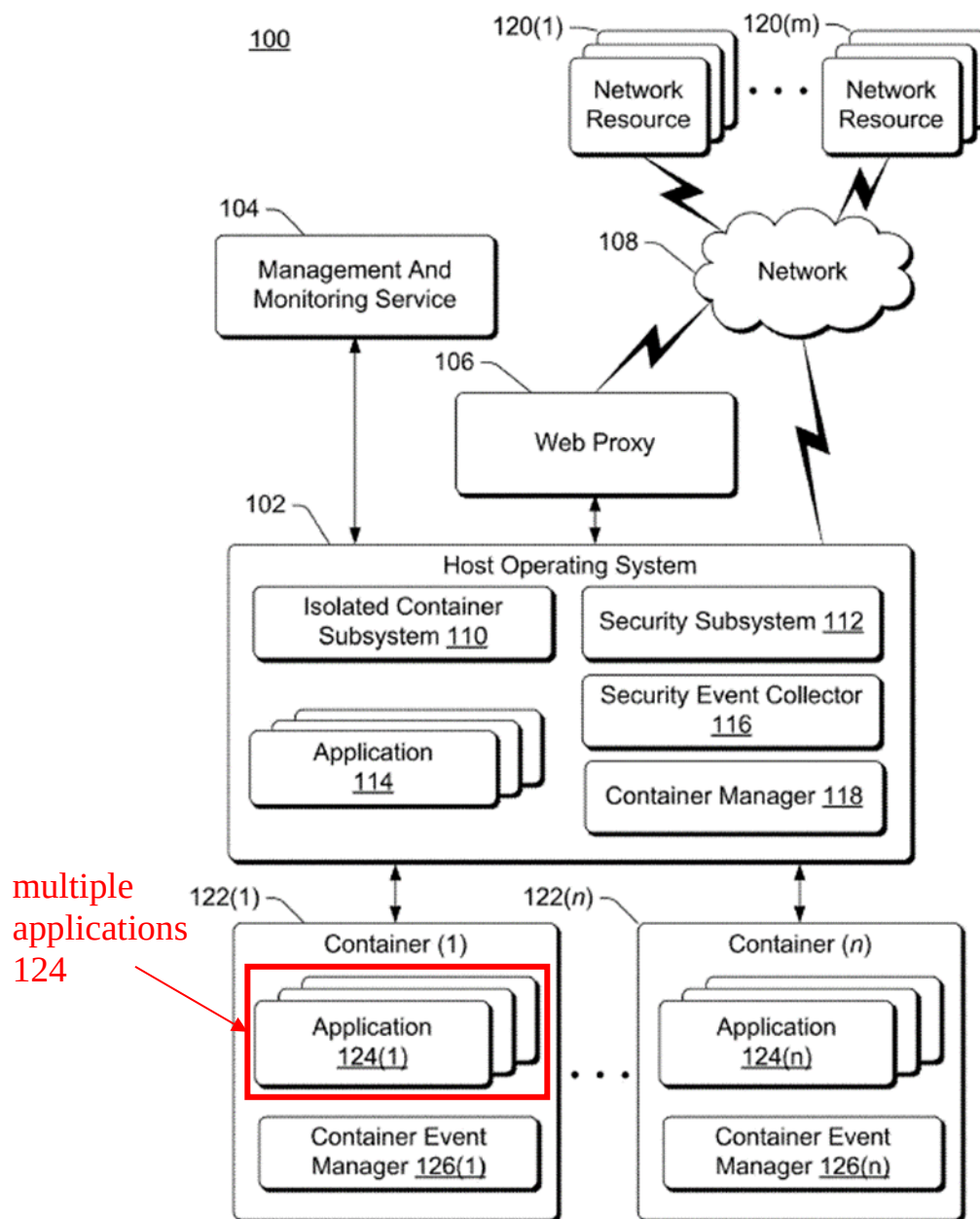


Fig. 1

Jeffries, Fig. 1 (annotated)

191. Accordingly, the Microsoft Word application accessed via LaBine's Bridgit Conferencing Software would be run in the same container as LaBine's Bridgit Conferencing Software.

192. Thus, LaBine's Bridgit Conferencing Software (*the multi-user interactive software application*) allows a viewer (*allows a remote machine*), to control the host computer's cursor or add text to a Microsoft Word application (*to control an application or a process of the second set of one or more applications or processes*). In addition, in the combination as described at limitation [1.4.1] above, LaBine's Bridgit Conferencing Software would be executed in an isolated container, as described by Jeffries (*running in the isolated computing environment*).

[2.2] 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.

193. The combination of Jeffries and LaBine renders obvious this limitation.

194. LaBine describes *viewing...data from the* applications on the host computer, such as a Microsoft Word application. For example, LaBine's "GUI 100" allows a viewer to view data from the conferencing software because it "shows the shared region 116 of the presenter's desktop." LaBine, ¶49. In the example in which a Microsoft Word application is running and displayed on the host computer, client users may view the Microsoft Word application, as shown in Figure 22 of LaBine, below.

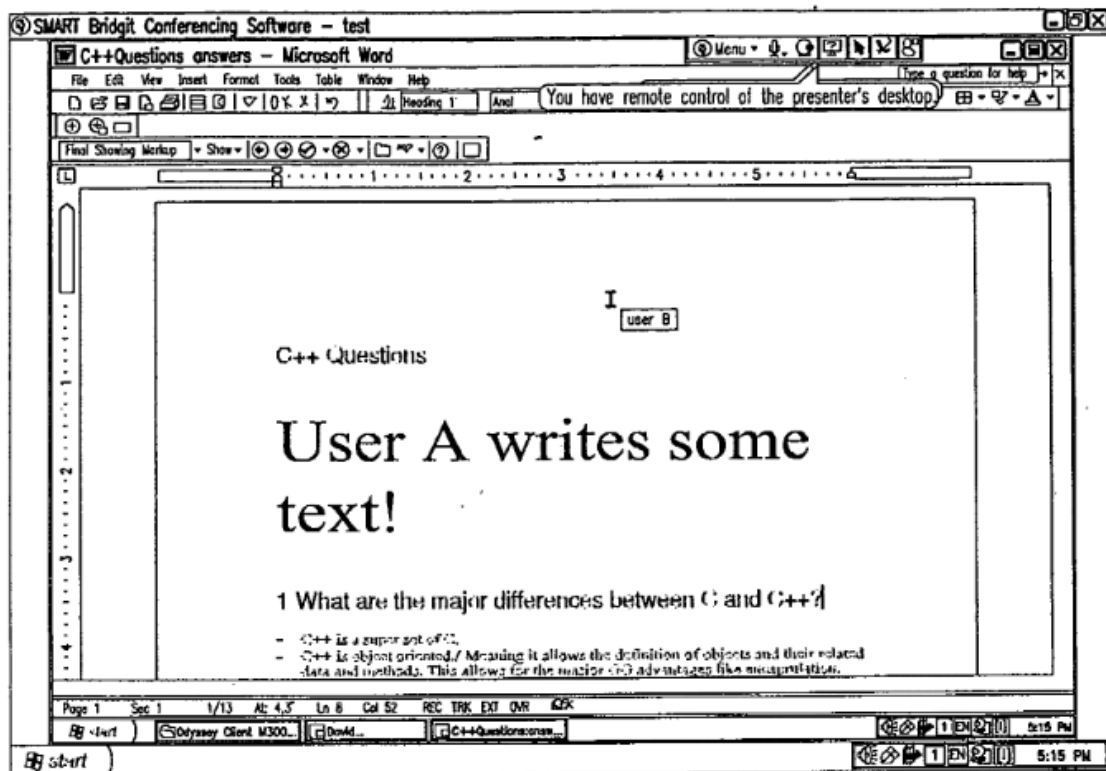


FIG. 22

LaBine, Fig. 22

195. As shown above, the Bridgit Data Conferencing software window allows a viewer, i.e., a user of a client computer or host machine, to view[]...data from the application or the process.

196. In addition, LaBine describes allowing a user of a client computer to interact[] with...data from the Microsoft Word application. As described above at limitation [2.1], LaBine's Software allows a user to move and control a cursor of the host computer, and enter text into the Microsoft Word application.

197. Thus, LaBine describes allowing a user of a client computer to control data of the Microsoft Word application, including viewing the host computer screen

(wherein controlling the application or process comprises at least one of viewing...data from the application or the process) and editing text within the Microsoft Word application (interacting with...data from the application or the process).

3. Claim 3

[3.0] The system of claim 1,

198. The combination of Jeffries, LaBine, and Ishaya renders obvious *the system of claim 1.*

[3.1] 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.

199. The combination of Jeffries and LaBine renders this limitation obvious.

200. LaBine's Bridgit Conferencing software allows viewers to remotely access and control various functionality of the host computer. As described at limitation [2.1] above, LaBine's viewers, which are users of computers participating in a video conference, correspond to *remote machine[s]*. A remote machine is also a remote device. Therefore, for the same reasons as explained above, the combination of Jeffries and LaBine render obvious *a remote device.*

201. Further, LaBine describes allowing the viewer's computer (*remote device*) to control the mouse, keyboard, and display of the system. In LaBine, "[t]he viewer's mouse events, including button clicking and movement, are handled as if they occurred locally on the presenters [sic] computer." LaBine, ¶74. A mouse is a

pointer device. LaBine also describes allowing a remote user to control “keyboard and pen events.” LaBine, ¶84. These “keyboard...events” are described as “remote user input events” and are therefore *keyboard input[s]*. As explained above at limitation [2.1], a user of a client computer may also control *a display* by annotating the desktop of the host computer.

202. Therefore, Jeffries and LaBine explain how the Bridgit Conferencing Software (*the multi-user interactive software application*) allows a viewer (*allows a remote device*) to control mouse events (*to control one or more of a pointer device*), keyboard events (*a keyboard input*), and annotations to the desktop display of the host computer (*or a display of the system*).

4. Claim 4

[4.0] The system of claim 1,

203. The combination of Jeffries, LaBine, and Ishaya renders obvious *the system of claim 1*.

[4.1] wherein the system is connected 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.

204. Jeffries renders this limitation obvious.

205. Jeffries explains that “the computing device implementing host operating system 102 accesses network resources 120(1),...,120(m) **via network 108.**” Jeffries, 6:59-60. The “[n]etwork 108 represents functionality of a data

network, such as the Internet, [or] **a local area network (LAN).**” Jeffries, 6:58-62.

206. Jeffries also describes “a **cellular** or **Wi-Fi network.**” Jeffries, 25:67.

207. Jeffries also describes “a **cloud computing environment** where multiple computing devices are connected and share access to common resources.” Jeffries, 3:24-26.

208. Thus, Jeffries describes a system that is connected to a LAN network (*wherein the system is connected to a network, the network corresponding to one or more of a local area network (LAN)*), a cellular network (*a cellular network*), a Wi-Fi network (*a Wi-Fi network*), or a cloud computing environment (*cloud computing network*).

5. Claim 7

[7.0] The system of claim 4, wherein the processor is further configured to:

209. The combination of Jeffries, LaBine, and Ishaya renders obvious *the system of claim 4*, as well as a processor.

[7.1] run one or more trusted applications in the first memory space, and

210. Jeffries renders this limitation obvious.

211. As described at limitation [1.3] above, Jeffries describes “one or more applications 114” which correspond to *a first set of one or more applications or processes via a first memory space*. Jeffries additionally explains that these one or more applications 114 are trusted applications. For example, Jeffries explains that “one of applications 114...**access[es] a trusted resource** on the host operating

system 102.” Jeffries, 10:61-63. For example, an email client of the host operating system 102 may access a trusted email server or a web browser on the host operating system 102 may access a trusted intranet web server. *See, e.g.*, Jeffries, 6:60-63 (describing accessing network resources such as “a local area network (LAN)” or “intranet” “independent of a web proxy”). In contrast, applications 124 within isolated containers are used to access untrusted resources. Jeffries, 10:55-60. Thus, a primary reason an application is isolated from the remainder of a computer system is because it accesses untrusted resources. Accordingly, an application that is not isolated within an isolated container in Jeffries, but allowed unrestricted access to the host computer, would be a trusted application or an application that accesses trusted resources. Because the applications 114 that are not contained within isolated containers are used to access trusted resources, they are *trusted* applications.

212. Thus, Jeffries describes applications 114 that are not contained in isolated containers and are used to access trusted resources (*run one or more trusted applications in the first memory space*).

[7.2] wherein the one or more trusted applications communicate with one or more trusted machines via the network.

213. Jeffries renders this limitation obvious.

214. As previously explained at limitation [1.3], Jeffries’ applications 114 are executed via the host operating system 102 and not contained in isolated containers. *See* Jeffries, Fig. 1 (applications 114 outside of containers 122), 9:26-28.

These applications 114 are trusted because they are used “to access a trusted resource on the host operating system 102.” Jeffries, 10:61-63. By contrast, Jeffries’ applications 124 that access untrusted resources are isolated in isolated containers. See Jeffries, Fig. 1, 1:27-29, 10:59-60 (Application 124(1),...,124(n) are thus useable to access untrusted resources.”). Thus, Jeffries’ applications 114 are *trusted applications* that access trusted resources on the host computer. But Jeffries also explains that these applications 114 communicate with other *trusted machines via the network*.

215. At Figure 6, Jeffries explains an additional embodiment of the system 100, in the context of an “example system generally at 600” including a “computing device 602.” Jeffries, 22:39-40. “The platform 624 may abstract resources and functions to connect the computing device 602 with other computing devices.” Jeffries, 26:1-3. Jeffries further explains that connection with these other computing devices may occur “through a network” such as any of the networks described previously, such as a LAN network, Jeffries, 6:58-62, a “cellular or Wi-Fi network,” Jeffries, 25:67, or “a cloud computing environment.” Jeffries, 6:58-62, 25:67, 3:24-26, 25:15-19. Thus, Jeffries’ system is configured to *communicate with one or more...machines via the network*.

216. A POSITA would have found it obvious for any of these machines—with which Jeffries’ host computer communicates via the network—to be *trusted*

machines, as claimed. For example, Jeffries describes “an enterprise environment where multiple users of the enterprise are connected and share access to common enterprise data and/or networks.” Jeffries, 3:19-21. Policies are applied to all users of the enterprise environment “to ensure that individual users as well as the shared data and networks of the enterprise remain protected.” Jeffries, 3:22-24. These devices that are part of an enterprise environment are trusted devices to the host machine because they are known devices that share common resources, are subject to the same security policies, and are protected as part of the same group from outside attacks.

217. In addition, Jeffries explains that applications of machines within this enterprise environment access these shared, trusted resources. For example, the administrators of a system may maintain a list of “resources that are expected to be used by the installed applications.” Jeffries, 3:30-31. In one example, therefore, an application 114 of the host operating system 102 would be used to access another machine, such as an intranet server of a local network.

218. Thus, Jeffries describes applications 114 executed outside of isolated containers 122 and used to access trusted resources (*the one or more trusted applications*) that communicate with resources that are expected to be used by the applications 114, those expected resources being within a local enterprise environment of devices protected by the same security policies from outside attacks

(communicate with one or more trusted machines via the network).

6. Claim 9

[9.0] A method comprising:

219. To the extent the preamble is limiting, Jeffries discloses it.

220. At Figure 3, Jeffries depicts “a flowchart illustrating **an example process** for implementing isolated container event monitoring.” Jeffries, 2:23-24.

Figure 3 is shown below.

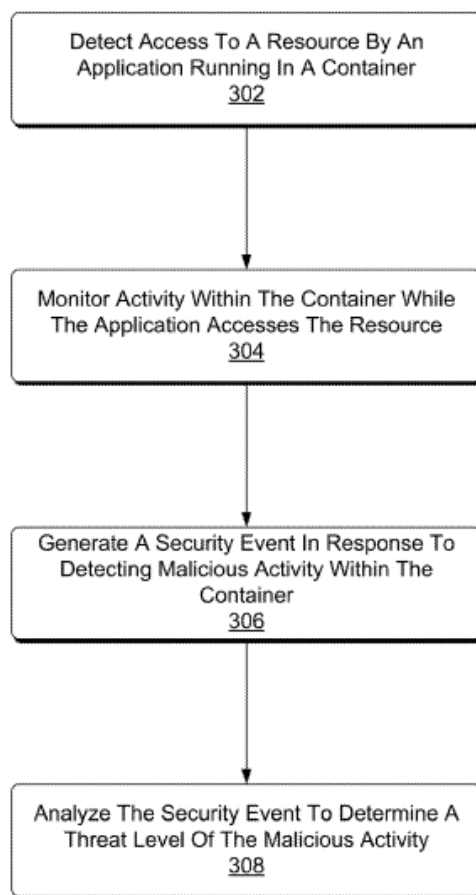


Fig. 3

Jeffries, Fig. 3

221. Additional processes related to implementing isolated containers are illustrated in Figures 4 and 5. See, e.g., Jeffries, Figs. 4-5, 2:26-31. Jeffries characterizes these processes as methods at claims 1-19. See Jeffries, 26:25-30:31.

222. Thus, Jeffries discloses [a] *method*.

[9.1] 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;

223. Limitation [9.1] is the same as limitation [1.3] except limitation [9.1] recites “implementing” while limitation [1.3] recites “implement.” Jeffries renders limitation [9.1] obvious for the same reasons described at limitation [1.3]. See §IX.G.1, [1.3].

[9.2.1] 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,

224. Limitation [9.2.1] is the same as limitation [1.4.1] except limitation [9.2.1] recites “implementing” and does not recite the term “and” while limitation [1.4.1] recites “implement” and recites the term “and.” Jeffries renders limitation [9.2.1] obvious for the same reasons described at limitation [1.4.1]. See §IX.G.1, [1.4.1].

[9.2.2] wherein the second set of one or more applications or processes comprises a multi-user interactive software application;

225. Limitation [9.2.2] is the same as limitation [1.4.2] and the combination of Jeffries and LaBine renders limitation [9.2.2] obvious for the same reasons

described at limitation [1.4.2]. See §IX.G.1, [1.4.2].

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

226. Limitation [9.3] is the same as limitation [1.5] except limitation [9.3] recites “isolating” while limitation [1.5] recites “isolate.” Jeffries renders limitation [9.3] obvious for the same reasons described at limitation [1.5]. See §IX.G.1, [1.5].

[9.4] authenticating the isolated computing environment with an authentication device; and

227. Limitation [9.4] is the same as limitation [1.6] except limitation [9.4] recites “authenticating” while limitation [1.6] recites “authenticate.” The combination of Jeffries and Ishaya renders limitation [9.4] obvious for the same reasons described at limitation [1.6]. See §IX.G.1, [1.6].

[9.5] 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.

228. Limitation [9.5] is the same as limitation [1.7] except limitation [9.5] recites “sending” while limitation [1.7] recites “send.” The combination of Jeffries, LaBine, and Ishaya renders limitation [9.5] obvious for the same reasons described at limitation [1.7]. See §IX.G.1, [1.7].

7. Claim 10

[10.0] The method of claim 9,

229. The combination of Jeffries, LaBine, and Ishaya renders obvious the *method of claim 9.*

[10.1] 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,

230. Limitation [10.1] is the same as limitation [2.1] and the combination of Jeffries and LaBine renders limitation [10.1] obvious for the same reasons described at limitation [2.1]. See §IX.G.2, [2.1].

[10.2] 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.

231. Limitation [10.2] is the same as limitation [2.2] and the combination of Jeffries and LaBine renders limitation [10.2] obvious for the same reasons described at limitation [2.2]. See §IX.G.2, [2.2].

8. Claim 11

[11.0] The method of claim 9,

232. The combination of Jeffries, LaBine, and Ishaya renders obvious the *method of claim 9.*

[11.1] 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 isolated computing environment.

233. Limitation [11.1] is similar to limitation [3.1]. Limitation [3.1] recites *allow[ing] a remote device to control one or more of a pointer device, a keyboard input, or a display of the system.* Limitation [11.1] recites *allow[ing] a remote device to control one or more of a pointer device, a keyboard input, or a display of the isolated computing environment.* The combination of Jeffries and LaBine renders

limitation [11.1] obvious for the same reasons described at limitation [3.1] as well as those explained below. See §IX.G.3, [3.1].

234. In the '601 patent, the mouse, keyboard, and display of *the system* is the same as the mouse, keyboard, and display of *the isolated computing environment*. “The host computer” of the '601 patent “include[s] [the] isolated computing environment.” '601 patent, 1:64-65. Similarly, the isolated computing environment is described by the '601 patent as a “virtual machine.” '601 patent, 5:43. As a virtual machine, the isolated computing environment does not use its own input hardware devices (mouse, keyboard, and display) separate from the host computer. The same hardware is used, but the isolated computing environment uses its own “memory associated with the sandboxed computing environment...separate from memory that is configured to enable storage and operation of the workspace.” '601 patent, 5:28-32.

235. Jeffries describes the same scenario, in which isolated containers 122 act as virtual machines that operate using a separate memory space but within the same host computer. See Jeffries, 10:38-41 (describing containers 122 as “virtual machine[s]”); § IX.G.1, [1.3]-[1.4.1] (explaining how Jeffries containers use the same operating system as the host computer but separate memory space).

236. In addition, LaBine expressly contemplates limiting its conferencing software to the “shared region” of the desktop and limiting access to selected

applications. LaBine, ¶¶74, 50. Thus, in the combination, it would have been obvious to limit LaBine's conferencing software to sharing only itself and other applications executing within the same isolated container.

237. Therefore, in both the '601 patent and in Jeffries, the mouse, keyboard, and display of the host system are the same devices as the mouse, keyboard, and display of the isolated computing environment. In the combination, therefore, LaBine's disclosure of allowing inputs from a viewer to be transmitted to the host machine and treated as inputs of the host machine renders obvious *allow[ing] a remote device to control one or more of a pointer device, a keyboard input, or a display of the isolated computing environment.*

9. Claim 12

[12.0] The method of claim 9, further comprising

238. The combination of Jeffries, LaBine, and Ishaya renders obvious the *method of claim 9.*

[12.1] 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.

239. Limitation [12.1] is the same as limitation [4.1] except limitation [12.1] recites "connecting to a network" while limitation [4.1] recites "wherein the system is connected to a network." Jeffries renders limitation [12.1] obvious for the same reasons described at limitation [4.1]. See §IX.G.4, [4.1].

10. Claim 15

[15.0] The method of claim 12, further comprising:

240. The combination of Jeffries, LaBine, and Ishaya renders obvious the *method of claim 12.*

[15.1] running one or more trusted applications in the first memory space,

241. Limitation [15.1] is the same as limitation [7.1] except limitation [15.1] recites “running” and does not recite the term “and,” while limitation [7.1] recites “run” and the term “and.” Jeffries renders limitation [15.1] obvious for the same reasons described at limitation [7.1]. *See §IX.G.5, [7.1].*

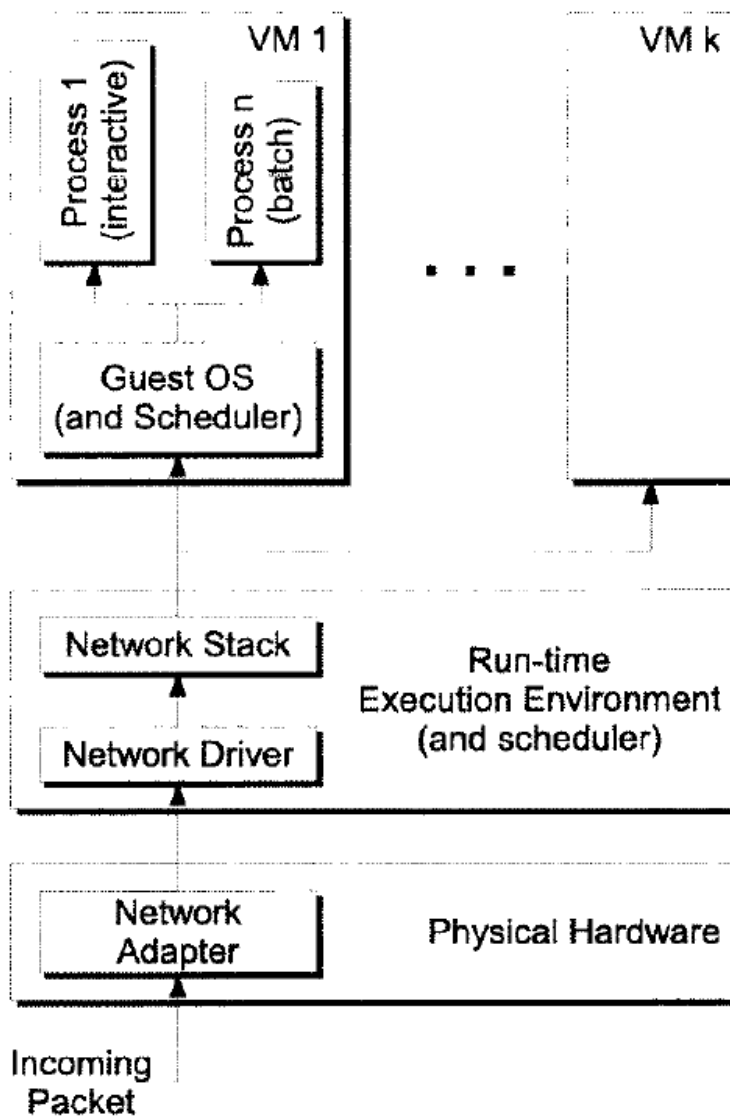
[15.2] wherein the one or more trusted applications communicate with one or more trusted machines via the network.

242. Limitation [15.2] is the same as limitation [7.2] and Jeffries renders limitation [15.2] obvious for the same reasons described at limitation [7.2]. *See §IX.G.5, [7.2].*

X. GROUND 2: THE COMBINATION OF JEFFRIES, LABINE, ISHAYA, AND CUCINOTTA RENDERS OBVIOUS CLAIMS 5-6 AND 13-14

A. Summary of Cucinotta

243. Like Jeffries and the '601 patent, Cucinotta describes a computer system that implements multiple isolated computing environments, referred to as “multiple virtual machines.” Cucinotta, [0002]. Cucinotta’s system is illustrated in Figure 1 below.



Ex.1008, Fig. 1

244. Cucinotta's system includes a "firewall component." Cucinotta, [0039]. The firewall component "inspects each and every incoming and outgoing packet" for the host system as well as "traffic...directed towards KVM handling VMs." Cucinotta, [0039], [0049].

245. Thus, because Cucinotta implements multiple isolated computing

environments, it is in the same field of endeavor and is reasonably pertinent to a particular problem with which the inventor of the '601 patent was involved (i.e., system security).

B. Reasons to Combine Jeffries, LaBine, Ishaya, and Cucinotta

246. A POSITA would have further combined Jeffries, LaBine, and Ishaya with Cucinotta. A POSITA would have been motivated to combine Jeffries with LaBine and Ishaya as previously explained. *See* § IX.D-F. This combined system would also be readily combined with Cucinotta because both the combination described above as well as Cucinotta describe improving system security and running applications in isolated computing environments. Jeffries isolates applications that access untrusted resources in a container. *See* Jeffries 2:46-49 (“[U]sing containers...protects the computing device from attacks by isolating” the computing device “from untrusted resources.”), 10:59-60 (“Applications 124” within isolated containers “are thus useable to access untrusted resources.”). Cucinotta runs “processes” in virtual machines. Ex.1008, Figs. 1 (illustrating VM 1-VM k). Jeffries also describes virtual machines. *See* Jeffries, 6:10-11 (“containers and virtual machines are used to isolate and contain malicious software.”).

247. A POSITA would have been motivated to combine Jeffries, LaBine, and Ishaya with Cucinotta for at least the reasons discussed below and would have had a reasonable expectation of success in the combination. A POSITA would have

been motivated to implement a host-based firewall, described in Cucinotta as a “firewall component,” to protect the system as a whole, including any virtual machines, from potential attacks. Cucinotta, [0039],[0049]. As explained above, isolation of applications that access untrusted resources in containers or virtual machines advantageously prevents the effects of malware or a virus from spreading outside the isolated container or virtual machines. *See* Jeffries 2:46-49, 10:59-60. However, this method of protection typically protects the host system from malware that is received by the application within the container or virtual machine, not applications within containers or virtual machines. Typical host systems include applications that are executed outside of the containers or virtual machines. *See, e.g.,* Jeffries, Fig. 1 (application 114), 9:26-33. These applications may still be vulnerable to potential attacks. For this reason, a POSITA would be motivated to additionally implement a host-based firewall to inspect packets received by the host system to determine whether they contain potentially harmful data or files.

248. In addition, a POSITA would further be motivated to implement a host-based firewall, such as the firewall component described in Cucinotta, to prevent malicious files from being received by the applications within the isolated containers or virtual machines. Jeffries explains why isolating applications is advantageous. *See* Jeffries 2:46-49, 10:59-60. However, using an isolated container or virtual machine to protect against received malware typically requires terminating the container or

machine, along with its contents, entirely to prevent the spread of malware. *See* Jeffries, 16: 41-42 (“terminating the container 122 and rebuilding a new container”), 27:34-35. Doing so results in the loss of unsaved or unfinished work and is a lengthy process leading to frustration of the user. A POSITA would have recognized the advantage of additionally filtering packets received by containers/virtual machines to minimize the instances in which an isolated container or virtual machine must be terminated and rebuilt.

249. The combination of Jeffries and Cucinotta would be simply combining prior art elements according to known methods to yield predictable results. The known methods would simply involve implementing a host-based firewall component as described in Cucinotta, *see* Cucinotta, [0039], [0049], within a system that utilizes isolated containers or virtual machines as described in Jeffries, *see* Jeffries, Abstract, 2:46-49. The results of the combination would have been predictable because Cucinotta’s system includes the firewall component to isolate the system, including the VMs, from other computers on the network, and would accomplish the same purpose when implemented in Jeffries’ system. *See* Jeffries, Fig. 1; Cucinotta, Fig. 1.

250. In addition, a POSITA would have recognized multiple benefits of the combination. As explained above, the combination of Jeffries with Cucinotta would advantageously improve the security of a computer system by protecting all

components of a host system, both within and outside of isolated environments. In addition, a POSITA would have recognized that implementing a host-based filter, such as the one described by Cucinotta, within a computer system would enhance efficiency by filtering packets on behalf of the entire host system. A host-based filter may also advantageously maintain consistency and uniformity of packet filtering policies across multiple isolated environments.

251. A POSITA would have had a reasonable expectation of success in implementing the host-based firewall component described by Cucinotta with the system described by Jeffries. The firewall component in Cucinotta is implemented to inspect and filter packets destined either for locations on the virtual machines as well as the host system. Cucinotta, [0049] (The firewall component “operates to intercept any incoming, outgoing, or forwarded traffic...including instances when such traffic is directed towards KVM handling VMs.”). Such a host-based filter would be readily implemented to similarly inspect and filter packets destined for locations on the multiple containers 122 as well as the host system 102 in Jeffries. See Jeffries, Fig. 1.

252. For all these reasons, a POSITA would be motivated to implement a host-based firewall, like Cucinotta’s host-based filter component, in the system described by Jeffries. Doing so would predictably improve the security of the host computer by identifying and blocking packets that may potentially include nefarious

data or files.

C. Detailed Analysis of Claims

1. Claim 5

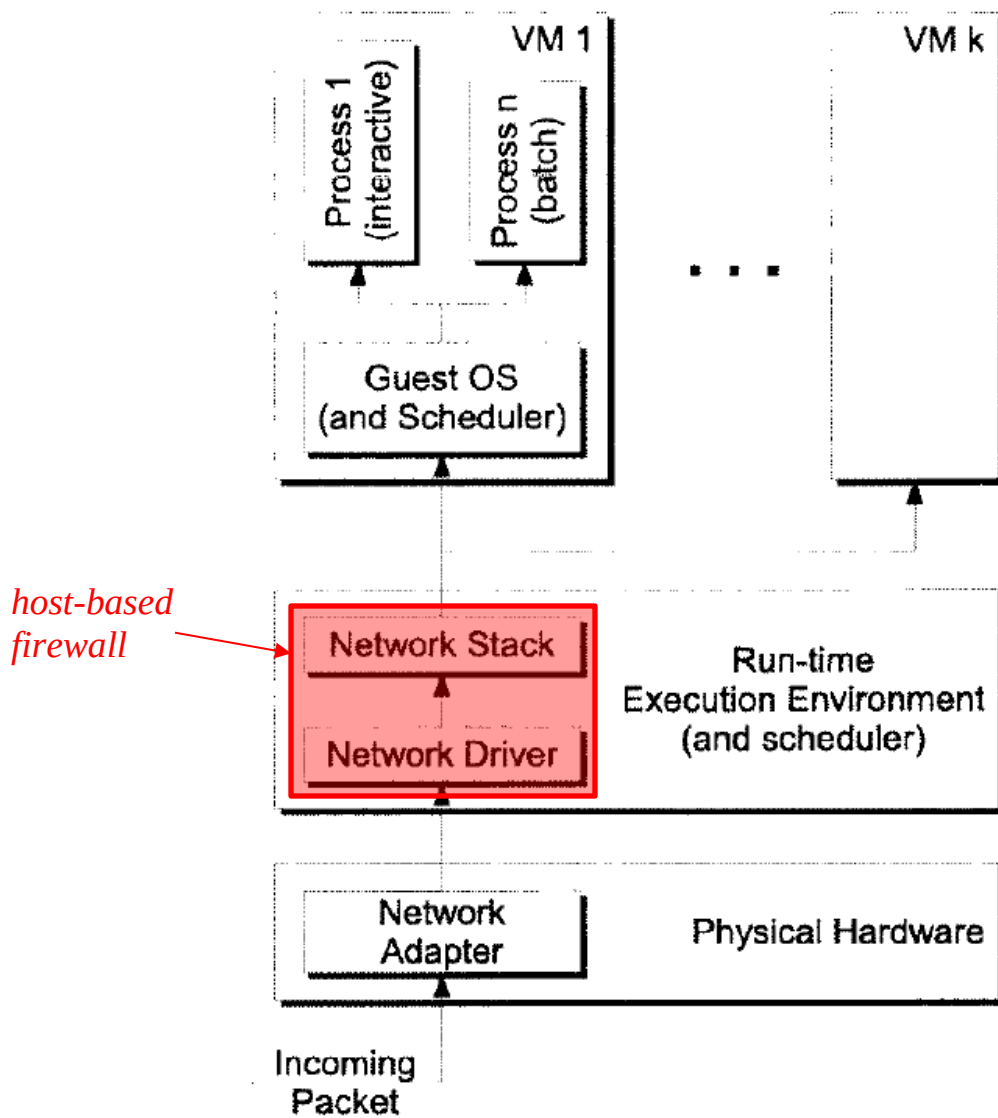
[5.0] The system of claim 4, wherein the processor is further configured to:

253. The combination of Jeffries, LaBine, and Ishaya renders obvious the *system of claim 4.*

[5.1] implement a host-based firewall, wherein the host-based firewall isolates the system from communicating with one or more other devices on the network.

254. The combination of Jeffries and Cucinotta renders this limitation obvious.

255. Cucinotta describes “a software component” that “comprise[s] a firewall component.” Ex.1008, [0039]. This firewall is part of “a multi-programmed computing system.” Ex.1008, [0001]. Figure 1, below, illustrates the path of a network packet received by the system. Ex.1008, [0032]. Cucinotta’s “firewall component” may “belong to the network stack of the runtime, or may be a network driver,” highlighted below. Ex.1008, [0039].



Ex.1008, Fig. 1 (annotated)

256. Because Cucinotta’s software component resides within the computing system shown in Figure 1, it is a *host-based firewall*. In addition, Cucinotta’s software component isolates the computer system as a whole from communicating with other devices on the network because it “inspect[s] each and every incoming and outgoing packet” Ex.1008, [0039]. Similarly, Cucinotta explains that the

“firewall software is typically operable to intercept all incoming and outgoing traffic travelling across [the] computing system.” Ex.1008, [0049].

257. Intercepting traffic in this way isolates the host computer from external resources. In addition, filters at the time commonly used whitelist and blacklist features to isolate protected machines or regions of memory. *See, e.g.*, Titonis, ¶143 (“blacklisting of new malicious sites”); Nabutovsky, Abstract (describing “creat[ing] and manag[ing] whitelists 111 and blacklists 116 which are used as filters”). The ’601 patent describes isolating systems using similar whitelist and blacklist features. *See* ’601 patent, 6:29-33 (describing “maintain[ing] a list of untrusted network destination” as “a blacklist”).

258. In the combination, a host-based filter, such as the host-based firewall component described in Cucinotta, would be implemented with Jeffries’ host operating system 102. As described in Cucinotta, this host-based firewall component would inspect all packets received by the host operating system 102 and selectively allow the packets to proceed to their destination in the computer system based on configured security characteristics or policies. *See* Cucinotta, [0049].

259. Thus, the combination of Jeffries and Cucinotta describes implementing a host-based firewall component (*implement a host-based firewall*) that isolates the host computer system from other machines on the network by intercepting all incoming and outgoing packets (*wherein the host-based firewall*

isolates the system from communicating with one or more other devices on the network).

2. Claim 6

[6.0] The system of claim 5,

260. The combination of Jeffries, LaBine, Ishaya, and Cucinotta renders obvious the *system of claim 5*.

[6.1] wherein the host-based firewall blocks incoming communications from the network that are sent to the workspace.

261. The combination of Jeffries and Cucinotta renders this limitation obvious.

262. As explained previously, Jeffries describes a *workspace*. See § IX.G.1, [1.3]. In the combination, a host-based filter, such as the firewall component described in Cucinotta, would be implemented in conjunction with the host computer of Jeffries and would protect the workspace. See § X.C.1, [5.1]. A host-based packet filter is used to block incoming communications. See Riley, ¶15 (“Additionally, the packet filter firewall could block incoming packets for all IP addresses combined with one of these ports.”); Cheswick, 6:39-45 (“For example, firewall security device 200 may deliver a packet or circuit filter which...blocks all incoming TCP connections (except electronic mail) and also blocks all conventional User Datagram Protocol (‘UDP’) packets except for well-known Domain Name Service (‘DNS’) packets.”). In addition, Cucinotta’s firewall component is configured to “intercept

all incoming and outgoing traffic travelling across [the] computing system.” Cucinotta, [0049]. In the combination, a POSITA would have recognized that a host-based firewall component positioned between the host computer and other machines would selectively block packets to or from the host computer. Because the filter is host-based, such as the firewall component described in Cucinotta, *see* Cucinotta, Fig. 1, this selective packet filtering would apply to packets intended to be sent either to the workspace or the isolated containers 122 of Jeffries. More broadly, this limitation would have been obvious to a POSITA because blocking potentially harmful incoming communications was a well-known and common function of host-based firewall software. *See, e.g.*, Cheswick, 6:41-42 (“block[ing] all incoming TCP connections”); Riley, ¶12 (“firewalls block traffic from the untrusted network”).

263. Thus, the combination of Jeffries and Cucinotta describes a host-based firewall component (*the host-based firewall*) that intercepts (*blocks*) network packets from external devices (*incoming communications from the network*) from reaching the workspace and/or isolated containers 122 (*that are sent to the workspace*).

3. Claim 13

[13.0] The method of claim 12, further comprising:

264. The combination of Jeffries, LaBine, Ishaya, and Cucinotta renders obvious the *method of claim 12*.

[13.1] implementing a host-based firewall, wherein the host-based firewall isolates the first memory space and the second memory space from communicating with one or more devices on the network.

265. Limitation [13.1] is similar to limitation [5.1]. Limitation [13.1] recites “implementing” while limitation [5.1] recites “implement.” In addition, limitation [13.1] recites *wherein the host-based firewall isolates **the first memory space and the second memory space***, whereas limitation [5.1] recites *wherein the host-based firewall isolates **the system***.

266. As explained at limitation [1.3], Jeffries describes “memory locations” associated with applications 114, which correspond to the *first memory space*. Jeffries, 12:13-16; *see* § IX.G.1, [1.3]. As explained at limitation [1.4.1], separate “resources” are “allocated” to isolated containers 122 in Jeffries, “such as memory, processors, disks, or networks.” Jeffries, 15:27-30; *see* § IX.G.1, [1.4.1]. This memory allocated to isolated containers 122 corresponds to *the second memory space*. *See* § IX.G.1, [1.4.1].

267. Because the “memory locations” (*the first memory space*) and the “memory” allocated to the isolated containers 122 (*the second memory space*) are part of the same host computer in Jeffries, the host-based firewall described above at [5.1] isolates both the first memory space and the second memory space for the same reasons it isolates the system as a whole. But, in addition, Cucinotta specifically describes intercepting packets addressed to the VMs of its system.

Specifically, Cucinotta explains that its “firewall software...can also be used in combination with a KVM hypervisor.” Cucinotta, [0049]. In such case, “iptables operate to intercept any incoming, outgoing, or forwards traffic travelling across a Linux system, including instances when such traffic is **directed towards KVM handling VMs.**” Cucinotta, [0049].

268. In the combination, the firewall component described by Cucinotta and implemented within Jeffries’ system would similarly intercept packets destined for the host computer, corresponding to the first memory space, as well as the isolated containers 122, corresponding to the second memory space.

269. In Cucinotta, the firewall component isolates the host system as well as virtual machines. See Cucinotta, Fig. 1, [0049]. Based on this description, a POSITA would have implemented such a host-based firewall component in the combination to isolate both the trusted applications 114 (within the *first memory space*) as well as the untrusted applications 124 (within the *second memory space*).

4. Claim 14

[14.0] The method of claim 13,

270. The combination of Jeffries, LaBine, Ishaya, and Cucinotta renders obvious the *method of claim 13*.

[14.1] wherein the host-based firewall blocks incoming communications from the network that are sent to the workspace.

271. Limitation [14.1] is the same as limitation [6.1] and the combination of

Jeffries and Cucinotta renders limitation [14.1] obvious for the same reasons described at limitation [6.1]. *See* §X.C.2, [6.1].

XI. GROUND 3: THE COMBINATION OF JEFFRIES, LABINE, ISHAYA, LEVY RENDERS OBVIOUS CLAIMS 8 AND 16

A. Summary of Levy

272. Levy describes a “a browser operating system (BOS)” that “runs the client-side component of each Web application (e.g., on-line banking, and Web mail) in its own virtual machine, which provides strong isolation between Web services and the user’s local resources.” Levy, Abstract. Like the ’601 patent and Jeffries, this isolation “protect[s] other applications and client resources from malicious downloaded code.” Levy, [0057]. Because “each Web application is isolated within its own virtual machine sandbox,” Levy’s system “remov[es] the need to trust Web browsers and the services they access.” Levy, [0130]. “Virtual machine sandboxes contain the damage that can be caused by malicious or vulnerable browsers.” Levy, [0130]. If malicious code is downloaded or installed, “the user can easily remove all local effects of a Web application by simply deleting its virtual machine.” Levy, [0046]; *see also* Levy, [0072].

273. Levy also explains that security of the host computer can be increased through “empowerment of users.” Levy, [0057]. In Levy’s system, users have “explicit knowledge about and control over downloaded content and code” and can dictate when downloaded code or objects may be released from the memory of the

isolated virtual machine to the remainder of the system. Levy, Abstract.

274. In more detail, the operating system associates “a private temporary storage for each virtual machine.” Levy, [0020]. “This private temporary storage is used for temporarily holding an object that is to be transferred into or out of the virtual machine.” Levy, [0020]. “A user is enabled to explicitly initiate a transfer of an object into or out of the temporary storage.” Levy, [0020].

B. Reasons to Combine Jeffries, LaBine, Ishaya, and Levy

275. A POSITA would have combined Jeffries, LaBine, and Ishaya with Levy because both the combination of Jeffries, LaBine, and Ishaya as well as Levy describe improving system security by isolating applications that access untrusted resources in an isolated environment. Jeffries’ isolated environments are containers. *See* Jeffries 2:46-49 (“[U]sing containers...protects the computing device from attacks by isolating” the computing device “from untrusted resources.”), 10:59-60 (“Applications 124” within isolated containers “are thus useable to access untrusted resources.”). Levy’s isolated environments are private virtual machines. Levy, ¶14 (“Active Web content and the browser that interprets and renders it are thereby isolated in a private virtual machine, protecting the user’s desktop from side-effects, malicious or otherwise.”), ¶¶21, 39, 43.

276. A POSITA would have been motivated to combine Levy with the combination of Jeffries, LaBine, and Ishaya for at least the reasons discussed below

and would have had a reasonable expectation of success in the combination. A POSITA would have been motivated to permit a user to designate which files may be permitted to pass from an isolated environment to the remainder of the host computer. Levy's system "provides strong safety guarantees both to users and Web services" while also "enabl[ing] users to gain knowledge and control of the active Web content being downloaded and executed." Levy, ¶14. Levy's system provides the user with more information about accessed resources to allow the user to make an informed decision about whether to access a resource. *See* Levy, ¶52 ("[T]he BOS provides trusted information to the user, such as the name and credentials of the Web application with which the screen is associated.").

277. A POSITA would be motivated to implement this technique in the combined system described above. For example, while running a video conferencing software in an isolated container, the user of the host computer may receive a link from a client computer through the video conferencing software. Because this link is untrusted, it may be opened in an "instance[]" of a web browser within the isolated container. Jeffries, 10:57; *see supra* §X.B.7 (limitation [2.1]) (describing multiple applications in a single isolated container). If, however, the user determines that the content received is trustworthy, the user may designate that it be transferred out of the isolated container to the remainder of the system, as taught by Levy. Levy, [0020].

278. A POSITA would have been motivated to implement Levy's teachings of allowing data or files to be moved from the isolated environment to the remainder of the computer based on a user input. Doing so would advantageously reduce false positives. In some scenarios, for example, a system which automatically filters packets from the isolated environment may flag an item as untrusted that the user knows is actually trusted. For example, if the item were sent by the same user from another machine to the host computer, the host computer may not recognize the sending machine or recognize that this transfer was actually initiated by the same user. In such a scenario, the system may flag it as an untrusted resource that remains quarantined in the isolated system. Therefore, it would be advantageous to allow the user to designate when objects may be transferred out of the isolated container.

279. The combination of Jeffries, Ishaya, LaBine, and Levy would be simply combining prior art elements according to known methods to yield predictable results. The known methods would simply involve implementing a function of allowing a user to designate which data objects may be permitted to be transferred from the container to the remainder of the host system. The results of the combination would have been predictable because Levy's system "enable[es] a user to explicitly initiate a transfer of an object into or out of the temporary storage." Levy, [0020].

280. In addition, a POSITA would have recognized multiple benefits of the

combination. As explained above, the combination would improve methods of conveying information regarding the security state of a user's computer, including additional information regarding the source of received data. The combination would additionally help to eliminate false positives by allowing data known to be trusted by the user to be designated as such. In addition, a POSITA would have recognized that implementing Levy's techniques of allowing a user to designate files to be transferred from the isolated container to the host system would serve to optimize resource management, ensuring that only the necessary files are transferred from the isolated container to the remainder of the host system, thus preserving memory availability. This process also raises awareness of users to be more vigilant as to which resources are accessed. The process of allowing users to identify files that may be transferred also adds greater flexibility and customization. For example, administrators of a local network may inspect files to be transferred when a user is unsure of whether the particular file should be trusted. Various policies could be implemented to further enhance the security of the host computer and/or the local network more broadly.

281. A POSITA would have had a reasonable expectation of success in implementing the techniques described by Levy with the combined system. Levy's system is designed to work with conventional window systems. See Levy, ¶52 ("The BOS routes input events to an appropriate browser instance as indicated by the user,

similar to the way conventional window systems operate.”). Levy also emphasizes its interoperability by explaining that “the BOS is designed to be implemented in different ways.” Levy, ¶56. Further, Levy’s system is designed to operate with “a conventional Web browser.” Levy, ¶82. Levy’s virtual machines are also “language-independent sandbox[es].” Levy, ¶101. Therefore, a POSITA at the time would have recognized that Levy’s teachings could be readily incorporated with Jeffries’ system.

282. For all these reasons, a POSITA would be motivated to implement the functionality of allowing a user to select which files may be transferred out of an isolated container, as taught by Levy, in the combined system. Doing so would predictably improve the security of the host computer by providing the user with additional information as to the source of downloaded files, as well as reduce false positives that may arise from automated or absolute isolation.

C. Detailed Analysis of Claims

1. Claim 8

[8.0] The system of claim 1,

283. The combination of Jeffries, LaBine, and Ishaya renders obvious the *system of claim 1*.

[8.1] wherein the internal isolation firewall prompts a user of the system to allow communication between the first memory space and the second memory space.

284. The combination of Jeffries and Levy renders this limitation obvious.

285. As described above, Levy describes a system that is remarkably similar

to both the '601 patent's and Jeffries' systems. *See* § XI.A. Levy's system similarly uses isolated computing environments, which it refers to as private virtual machines, to isolate applications that access untrusted resources. *See* Levy, Abstract, [0057], [0130].

286. In Levy, each virtual machine is associated with a “temporary storage” also referred to as a “holding bin.” Levy, [0020], [0067]. “This private temporary storage is used for temporarily holding an object that is to be transferred into or out of the virtual machine.” Levy, [0020]. “A user is enabled to explicitly initiate a transfer of an object into or out of the temporary storage.” Levy, [0020].

287. In more detail, Levy describes a system which achieves both “strong VM-based isolation” to improve security while still “permit[ting] the controlled transfer of objects outside of a VM.” Levy, [0067].

The BOS supports strong VM-based isolation between browser instances. However, it must also **permit the controlled transfer of objects outside of a VM**. For example, a user should be able to **copy a photo** from a Web-mail application into a photo album, or vice versa. For the purpose of enabling such controlled transfers, the BOS kernel implements a private “holding bin” or temporary storage for each browser instance. To manipulate the holding bin, the kernel provides two browser-calls—BinStore and BinFetch. A browser instance copies an object to its holding bin by invoking the

BinStore browser-call, specifying the objects URL, an object name, and a MIME type. Similarly, the BinFetch browser-call lets the browser instance find and retrieve an object from the holding bin. However, **a transfer between the holding bin and the host OS must be initiated explicitly by a user through a trusted tool**; it cannot be initiated by the browser instance. In this way, controlled transfers are permitted, but code in the browser instance is prohibited from directly manipulating host OS resources.

Levy, [0067].

288. In the combination, a POSITA would have been motivated to similarly allow a user to control the transfer of objects, as taught by Levy, between the memory associated with the isolated containers 122 and the remaining, unrestricted memory of the host operating system 102 in Jeffries. This functionality would be implemented within container manager 118 described in Jeffries. The container manager 118 is responsible for monitoring activities within containers as well as implementing “policy describing what types of access authorized or unauthorized.” Jeffries, 17:39-40. A POSITA would have easily implemented a policy carried out by such components that allows a user to dictate which files or data may be transferred from the isolated containers 122 to the remainder of the computer system, as described by Levy. Such a combination would advantageously allow users to obtain files or data objects as needed from the isolated container while still ensuring

the security of the host system. See § XI.B.

289. Thus, the combination of Jeffries and Levy describes the container manager 118 (*internal isolation firewall*) allowing a user to explicitly initiate a transfer of the data (*prompts a user of the system to allow communication*) from memory associated with the isolated containers 122 (*between the first memory space*) to the memory of the remainder of the system (*and the second memory space*).

2. Claim 16

[16.0] The method of claim 9,

290. The combination of Jeffries, LaBine, and Ishaya renders obvious the *method of claim 9*.

[16.1] wherein the internal isolation firewall prompts a user of the workspace to allow communication between the first memory space and the second memory space.

291. Limitation [16.1] is similar to limitation [8.1]. While limitation [16.1] recites *wherein the internal isolation firewall prompts a user of the **workspace***, while limitation [8.1] recites *wherein the internal isolation firewall prompts a user of the **system***. The combination of Jeffries and Levy renders this limitation obvious for the same reasons as described at limitation [8.1]. See § XI.C.1, [8.1].

292. In addition, the combination of Jeffries and Levy renders obvious *prompt[ing] a user of the **workspace***. As described above at limitation [1.3], the claimed system comprises the *workspace*. See § IX.G.1, [1.3]. Therefore, a *user of the system*, as explained with regards to limitation [8.1], is also the *user of the*

workspace.

XII. CONCLUSION

I 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 that these statements were made with knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under section 1001 of Title 18 of the United States Code.

Dated: Aug 11, 2024



Prashant Shenoy, Ph.D.