

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CROWDSTRIKE, INC.,
Petitioner

v.

SKYSONG INNOVATIONS, LLC,
Patent Owner

Inter Partes Review Case No. IPR2025-01400
U.S. Patent No. 10,574,721

**PETITION FOR *INTER PARTES* REVIEW
OF U.S. PATENT NO. 10,574,721**

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
II.	THE '721 PATENT.....	1
	A. The '721 Patent's Alleged Invention	1
	B. The '721 Patent's Prosecution	2
	C. A Person of Ordinary Skill in the Art	3
III.	REQUIREMENTS UNDER 37 C.F.R. § 42.104	3
	A. Standing Under 37 C.F.R. § 42.104(a)	3
	B. Identification of Challenge Under 37 C.F.R. § 42.104(b)	4
	C. Claim Construction Under 37 C.F.R. § 42.104(b)(3)	4
IV.	PRIOR ART SUMMARY	5
	A. Fages	5
	B. Korycki	5
	C. Jai	6
	D. Hayden	7
	E. Russello.....	7
	F. O'Niell	8
	G. Cardona.....	8
	H. Tazuma.....	9
V.	GROUND 1: CLAIMS 1-2 ARE OBVIOUS OVER <i>FAGES</i>, <i>KORYCKI</i>, <i>JAI</i>, AND <i>HAYDEN</i>.....	9
	A. Claim 1.....	10
	1. <i>Claim 1(Pre): "A system, comprising: a processor, operable to:"</i>	10
	a) <i>Korycki teaches a Processor</i>	12
	b) <i>Motivation to Modify Fages per Korycki</i>	13
	2. <i>Claim 1(a): "receive data defining a first web content class and a second web content class, the first web content class defining a first plurality of Uniform Resource Locators (URLs), and the second web content class defining a second plurality of URLs different than the first plurality of URLs, the second web content class further being associated with sensitive information presenting a security risk;"</i>	14

a)	<i>“receive data defining a first web content class and a second web content class...”</i>	14
b)	<i>“the second web content class defining a second plurality of URLs different than the first plurality of URLs”</i>	15
c)	<i>“the second web content class further being associated with sensitive information presenting a security risk”</i>	16
d)	<i>“first web content class defining a first plurality of Uniform Resource Locators (URLs)”</i>	17
3.	<i>Claim 1(b): “generate a first browser instance, the first browser instance defining a browser instance type predetermined to be associated with the first web content class;”</i>	18
a)	<i>’721 Intrinsic Record - “browser instance type”</i>	18
b)	<i>Fages’s “first browser instance” and “browser instance type”</i>	20
4.	<i>Claim 1(c): “intercept a request to access web content from the first browser instance, the request defining a URL of the second plurality of URLs such that the web content is associated with the second web content class;”</i>	23
5.	<i>Claim 1(d): “confirm that the URL is not being accessed by another separate already-running browser instance associated with the second web content class, by”</i>	24
a)	<i>Fages’s Teachings</i>	24
b)	<i>Jai’s URL Comparison</i>	25
c)	<i>Hayden’s Teachings – “another separate already-running browser instance”</i>	28
d)	<i>The Fages-Korycki-Jai-Hayden Combination Teaches “confirm that the URL is not being accessed by another separate already-running browser instance...”</i>	32
e)	<i>Motivation to Modify Fages per Jai and Hayden</i>	32
6.	<i>Claim 1(d)(i): “implementing a daemon outside the first browser instance that tracks active browser instances and launches a new browser in order to manage web access according to the data defining the first web content class and the second web content class,”</i>	35
a)	<i>Fages’s Teachings</i>	36
b)	<i>Korycki’s Teachings</i>	36

c)	Motivation to Modify <i>Fages-Jai-Hayden</i> per <i>Korycki</i>	38
7.	<i>Claim 1(d)(ii): “receiving the request via a browser extension associated with the first browser instance,”</i>	41
a)	<i>Korycki’s Browser Extension</i>	41
b)	Motivation to Modify <i>Fages-Jai-Hayden</i> per <i>Korycki</i>	42
8.	<i>Claim 1(d)(iii): “forwarding the request from the browser extension to the daemon using a native application,”</i>	44
a)	<i>Korycki’s Teachings</i>	44
b)	Motivation to Modify <i>Fages-Jai-Hayden</i> per <i>Korycki</i>	47
9.	<i>Claim 1(d)(iv): “confirming, via the daemon that the web content is not associated with the first web content class and that another browser instance is not already launched and available to access the web content; and”</i>	49
10.	<i>Claim 1(e): “launch a new browser for accessing the web content associated with the second web content class to provide access to the URL of the second plurality of URLs via the new browser.”</i>	51
B.	Claim 2	52
VI.	GROUND 2: CLAIMS 3-5 ARE OBVIOUS OVER <i>FAGES,</i> <i>KORYCKI, JAI, HAYDEN, AND RUSSELLO</i>	53
A.	Claim 3	53
a)	’721 Patent’s API	53
b)	<i>Russello’s API</i>	53
c)	Motivation to Modify <i>Fages-Korycki-Jai-Hayden</i> per <i>Russello</i>	55
B.	Claim 4	56
1.	<i>Claim 4(Pre)</i>	56
2.	<i>Claim 4(a)</i>	56
a)	<i>Russello’s Pausing Operations</i>	56
b)	Motivation to Modify <i>Fages-Korycki-Jai-Hayden</i> per <i>Russello</i>	57
3.	<i>Claim 4(b)</i>	58
a)	Motivation to Combine <i>Fages-Korycki-Jai-</i> <i>Hayden and Russello</i>	59
C.	Claim 5	59

a)	'721 Patent's Tracer	59
b)	<i>Russello's</i> Tracer (Process Monitor Using ptrace).....	60
c)	Motivation to Combine <i>Fages-Korycki-Jai-Hayden-Russello</i>	60
VII.	GROUND 3: CLAIMS 6-8, 12-13, 16-18 ARE OBVIOUS OVER	
	<i>FAGES, KORYCKI, JAI, HAYDEN, O'NIELL, AND CARDONA</i>	62
A.	Claim 6.....	62
a)	<i>O'Niell's</i> C program Daemon Using Unix Domain Sockets	62
b)	<i>Cardona's</i> Multithreaded Daemon.....	63
c)	Motivation to Combine <i>Fages-Korycki-Jai-Hayden</i> per <i>O'Niell</i> and <i>Cardona</i>	64
B.	Claim 7.....	67
1.	<i>Claim 7(Pre)</i>	67
2.	<i>Claim 7(a)</i>	68
3.	<i>Claim 7(a)(i)</i>	68
4.	<i>Claim 7(a)(ii)</i>	68
5.	<i>Claim 7(a)(iii)</i>	68
6.	<i>Claim 7(a)(iii)(1)</i>	68
7.	<i>Claim 7(a)(iii)(2)</i>	69
8.	<i>Claim 7(a)(iii)(3)</i>	69
9.	<i>Claim 7(a)(iv)</i>	69
C.	Claim 8.....	69
D.	Claim 12.....	69
1.	<i>Claim 12(Pre)</i>	69
2.	<i>Claim 12(a)</i>	69
3.	<i>Claim 12(b)</i>	69
4.	<i>Claim 12(c)</i>	69
5.	<i>Claim 12(c)(i)</i>	70
6.	<i>Claim 12(c)(ii)</i>	70
7.	<i>Claim 12(c)(iii)</i>	70
E.	Claim 13.....	70
1.	<i>Claim 13(Pre)</i>	70
2.	<i>Claim 13(a)</i>	70
3.	<i>Claim 13(b)</i>	70
F.	Claim 16.....	70
G.	Claim 17.....	70
H.	Claim 18.....	70

VIII. GROUND 4: CLAIMS 9-10, 14-15 ARE OBVIOUS OVER <i>FAGES, KORYCKI, JAI, HAYDEN, O’NIELL, CARDONA, AND RUSSELLO</i>	71
A. Claim 9	71
B. Claim 10	71
C. Claim 14	71
1. Claim 14(Pre)	71
2. Claim 14(a)	71
3. Claim 14(b)	71
D. Claim 15	71
IX. GROUND 5: CLAIM 11 IS OBVIOUS OVER <i>FAGES, KORYCKI, JAI, HAYDEN, O’NIELL, CARDONA, AND TAZUMA</i>	72
A. Claim 11	72
1. Claim 11(Pre)	72
2. Claim 11(a)	72
a) Tazuma’s DNS Query	72
b) Motivation to Modify the <i>Fages</i> Combination per Tazuma	72
3. Claim 11(b)	73
4. Claim 11(c)	73
a) ’721 Patent DNS Response	73
b) Tazuma’s DNS Response	73
c) Motivation to Modify <i>Fages</i> Combination per Tazuma	74
5. Claim 11(d)	75
a) Tazuma’s Returning a Rewritten IP Address	75
b) Motivation to Modify the <i>Fages</i> Combination per Tazuma	76
X. CONCLUSION	76
XI. MANDATORY NOTICES UNDER 37 C.F.R. § 42.8(A)(1)	77
A. Real Parties-In-Interest	77
B. Related Matters	77
C. Lead and Back-Up Counsel	77
Paul R. Hart, (Reg. No. 59,646) Paul.Hart@eriseip.com	78
D. 37 C.F.R. § 42.8(b)(4) – Service Information	78

TABLE OF AUTHORITIES

Cases

<i>Phillips v. AWH Corp.</i> , 415 F.3d 1303, 1312 (Fed. Cir. 2005).....	5
<i>Skysong Innovations, LLC v. CrowdStrike, Inc. and CrowdStrike Holdings, Inc.</i> , Case No. 7:25-cv-00040-DC-DTG	77
<i>Skysong Innovations, LLC v. Fortinet, Inc.</i> , Case No. 2:25-cv-00098.....	77

Statutes

35 U.S.C. § 102(a)(1)	passim
35 U.S.C. § 102(a)(2)	8, 13

Regulations

37 C.F.R. § 42.100(b).....	4
37 C.F.R. § 42.104(a)	3
37 C.F.R. § 42.104(b).....	4
37 C.F.R. § 42.104(b)(3)	4
37 C.F.R. § 42.8(a)(1)	77
37 C.F.R. §42.24.....	88
37 C.F.R. §42.8.....	88

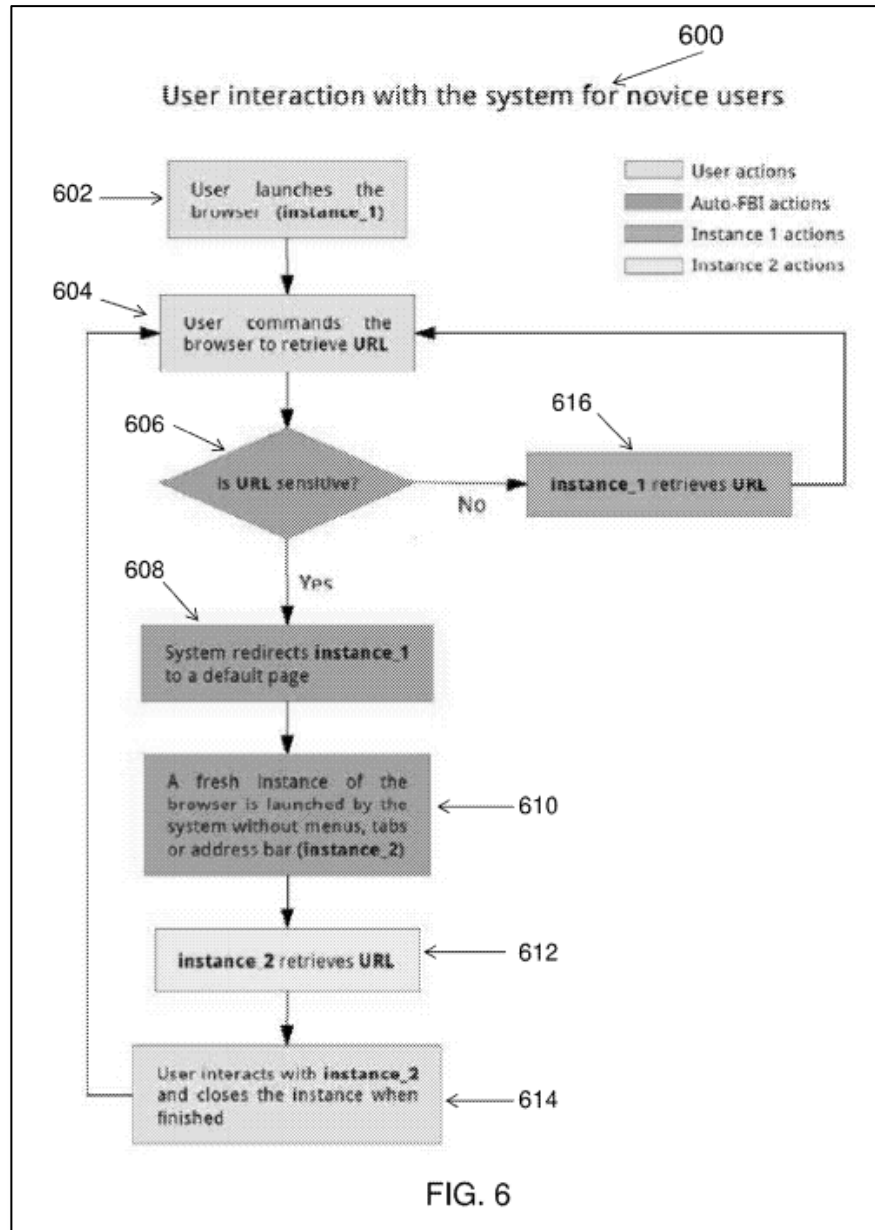
I. INTRODUCTION

CrowdStrike, Inc. (“Petitioner”) respectfully requests *inter partes* review (“IPR”) of claims 1-18 (“Challenged Claims”) of U.S. Patent No. 10,574,721 (“’721 Patent”).

II. THE ’721 PATENT

A. The ’721 Patent’s Alleged Invention

The ’721 Patent describes compartmentalizing “different kinds of content with different browser instances.” ’721 *Patent* (Ex. 1001), Abstract, 1:11-14. To “automate compartmentalization[,]” a “fresh browser instance (‘FBI’)” is automatically launched for sensitive content. *Id.*, Abstract, 1:59-2:3, 2:29-51, 3:24-32, 4:32-53. In embodiments, when an initial web browser sends a request to access a URL containing sensitive data, a browser extension (e.g., JavaScript) associated with that web browser passes the URL to a daemon using a native application. *Id.*, 2:29-51, 6:58-61, 10:8-37, Fig. 6, Claim 1. The daemon “decid[es] how to handle restricted URLs by either dispatching it to an already-running instance that is allowed to access the URL or creating a fresh new instance.” *Id.*, 10:33-50, Fig. 6, Claim 1. Figure 6 depicts this process:



Id., Fig. 6.

B. The '721 Patent's Prosecution

The '721 Patent claims priority to provisional application 61/912,716, filed December 6, 2013. For purposes of this Petition only, Petitioner applies December 6, 2013, as the priority date for the Challenged Claims.

To position the claims for allowance, Applicant amended the claims to recite, in part, a daemon, native application, and browser extension. '721 *File History* ("FH"), 363-368, 370-373 ("Use of the daemon as recited and described is believed to be absent from the cited prior art references...All of the independent claims now recite more clarifying features regarding how the daemon is implemented") (emphasis in original). As set forth below, these concepts were well-known in the art before the earliest priority date of the '721 Patent.

C. A Person of Ordinary Skill in the Art

A POSITA at the time of the '721 Patent would have had a bachelor's degree in computer science, electrical or computer engineering, or an equivalent degree; and 2-3 years of work or research experience in computer security and networking. *Dec.*, ¶¶54-58.¹ Additional education (e.g., MS degree) might substitute for less experience and vice versa. *Id.*

III. REQUIREMENTS UNDER 37 C.F.R. § 42.104

A. Standing Under 37 C.F.R. § 42.104(a)

Petitioner certifies the '721 Patent is eligible for IPR and Petitioner is not barred or estopped from requesting IPR challenging the claims of the '721 Patent. Petitioner is not the owner of the '721 Patent, has not filed a civil action challenging

¹ All citations to "Dec." are to paragraph numbers in Ex. 1003, Declaration of Erez Zadok, Ph.D.

the validity of any claim of the '721 Patent, and this Petition is filed less than one year after Petitioner was served with a complaint alleging infringement of the '721 Patent.

B. Identification of Challenge Under 37 C.F.R. § 42.104(b)

Petitioner requests the Challenged Claims be found unpatentable on the following grounds.

Proposed Grounds of Unpatentability	Exhibits
<u>Ground 1:</u> Claims 1-2 are obvious under AIA 35 U.S.C. § 103 over <i>Fages, Korycki, Jai, and Hayden</i>	Ex. 1004, 1005, 1006, 1050
<u>Ground 2:</u> Claims 3-5 are obvious under AIA 35 U.S.C. § 103 over <i>Fages, Korycki, Jai, Hayden, and Russello</i>	Ex. 1004, 1005, 1006, 1050, 1007
<u>Ground 3:</u> Claims 6-8, 12-13, and 16-18 are obvious under AIA 35 U.S.C. § 103 over <i>Fages, Korycki, Jai, Hayden, O'Niell, and Cardona</i>	Ex. 1004, 1005, 1006, 1050, 1008, 1009
<u>Ground 4:</u> Claims 9-10 and 14-15 are obvious under AIA 35 U.S.C. § 103 over <i>Fages, Korycki, Jai, Hayden, O'Niell, Cardona, and Russello</i>	Ex. 1004, 1005, 1006, 1050, 1008, 1009, 1007
<u>Ground 5:</u> Claim 11 is obvious under AIA 35 U.S.C. § 103 over <i>Fages, Korycki, Jai, Hayden, O'Niell, Cardona, and Tazuma</i>	Ex. 1004, 1005, 1006, 1050, 1008, 1009, 1010

C. Claim Construction Under 37 C.F.R. § 42.104(b)(3)

In this proceeding, claims are interpreted under the *Phillips* standard. 37 C.F.R. § 42.100(b); 83 Fed. Reg. 197 (Oct. 11, 2018); *Phillips v. AWH Corp.*, 415

F.3d 1303, 1312 (Fed. Cir. 2005) (*en banc*). Petitioner applies the ordinary and customary meaning of the claim terms as understood by a POSITA.²

IV. PRIOR ART SUMMARY

A. Fages

U.S. Patent Application Publication No. 2012/0069401 to Fages et al. (“*Fages*”) published March 22, 2012, and qualifies as prior art to the ’721 Patent under at least 35 U.S.C. §§ 102(a)(1)-(2).

Fages teaches “parallel execution of several Internet browsers in execution environments that are independent of one another” such that if a “URL belongs to a pre-recorded list of URLs of sensitive Internet sites to which connection from the first execution environment is not authorized,” *Fages* launches a new browser in parallel in a “second execution environment” “to connect to the sensitive Internet site identified by this URL[.]” *Fages*, [0044]-[0046]; *see also id.*, [0098]-[0099], [0104], [0109]-[0110].

B. Korycki

U.S. Patent No. 9,009,853 to Korycki et al. (“*Korycki*”) was filed March 7, 2012, and qualifies as prior art to the ’721 Patent under at least 35 U.S.C. § 102(a)(2).

² Petitioner’s challenge is not a waiver of any argument in litigation that any claim term in the ’721 Patent is indefinite or invalid. Petitioner does not waive its right to seek to construe claim terms in a different proceeding.

Korycki teaches a computer system enforcing Active Control List (ACL) security to allow or disallow a user to access certain web content. *Korycki*, 8:41-9:8, 9:34-10:22. *Korycki* utilizes a pub-sub.js “widget” 206 (“browser extension”), a browser plugin 208 implemented using native technology (“native application”), and a daemon 218 (“daemon”) that performs background processing including security functions related to the ACL security. *Korycki*, 7:21-33 (“The publish-subscribe daemon application 218 is a native part of the service provider client application and is always running in the background at the device 102.”), 9:60-10:22, 6:40-7:6.

C. Jai

International Publication No. WO 02/37333 to Jai (“*Jai*”) published May 10, 2002, and qualifies as prior art to the ’721 Patent under at least 35 U.S.C. §§ 102(a)(1)-(2).

Jai teaches a computer system that performs a URL comparison to detect if a requested URL is already open in a browser instance. *Jai*, 3:22-4:2, 4:20-21, 12:8-13:7, 14:3-5, Fig. 8. This allows “duplicate applications of the same site [to] be avoided.” *Jai*, 3:22, Abstract. This is because “prior to accessing and loading another site, the present invention searches other open applications and determines whether the new request is actually a duplicate of an already open application. If it is a duplicate, the existing application is displayed and the second requested application of the same site is deleted.” *Jai*, 3:22-4:2.

D. Hayden

U.S. Patent No. 9,509,783 to Hayden was filed January 26, 2012, and qualifies as prior art to the '721 Patent under at least 35 U.S.C. § 102(a)(2).

Hayden teaches determining if a particular browser instance is already launched, and if a particular browser instance is not already launched, *Hayden* teaches launching the particular browser instance. *Hayden*, 35:4-36:32, Fig. 13.

E. Russello

U.S. Patent Application Publication No. 2014/0137184 to Russello et al. ("*Russello*") was filed October 10, 2013, and claims priority to foreign applications dated November 13, 2012, and July 25, 2013, and qualifies as prior art to the '721 Patent under at least 35 U.S.C. § 102(a)(2).

Russello teaches utilizing a process tracer call ("ptrace") to intercept "all or at least a set of system calls executed by the target process[.]" *Russello*, [0026]. In *Russello*, "the tracing system call may be a process trace system call (ptrace)." *Russello*, [0026], [0203], [0250]. *Russello* also implements a "process monitor" in conjunction with ptrace to control system calls and execute security policies related to ptrace operations. *Russello*, [0026]-[0031]. In general, the process monitor is the "master" process associated with a "target process," and system calls are traced such that system calls executed by the target process are intercepted and reported to the master process (process monitor). *Id.*

F. O’Niell

U.S. Patent No. 8,424,017 to O’Niell et al. (“*O’Niell*”) published November 3, 2011, and issued April 16, 2013, and qualifies as prior art to the ’721 Patent under at least 35 U.S.C. §§ 102(a)(1)-(2).

O’Niell teaches a computer system utilizing daemon 235 that uses Unix domain sockets for communication and is capable of utilizing other sockets such as Linux domain sockets. *O’Niell*, 5:28-51, 5:62-6:21. *O’Niell* implements its daemon as a C program. *O’Niell*, 4:26-41.

G. Cardona

U.S. Patent Application Publication No. 2003/0061257 to Cardona (“*Cardona*”) published March 27, 2003, and qualifies as prior art to the ’721 Patent under at least 35 U.S.C. §§ 102(a)(1)-(2).

Cardona teaches daemons are “agent programs that continuously operate, for example, on a server, and provide resources to client systems on the network. These background processes can handle a variety of tasks, including monitoring the network for incoming data.” *Cardona*, [0003], [0006]-[0009], Figs. 1-2. *Cardona*’s daemon is multithreaded to handle multiple threads or tasks concurrently. *Cardona*, [0036] (“The exemplary daemon according to the invention is multithreading, that is, it spawns multiple threads or tasks that exist concurrently.”), [0053].

H. Tazuma

U.S. Patent No. 7,493,414 to Tazuma (“*Tazuma*”) issued February 17, 2009, and qualifies as prior art to the ’721 Patent under at least 35 U.S.C. §§ 102(a)(1)-(2).

Tazuma teaches generating a domain name system (DNS) query to resolve an IP address. *Tazuma*, 8:38-49. *Tazuma* redirects a browser in a network by rewriting an IP address in a DNS response when the requested IP address is not allowed to be accessed. *Tazuma*, 1:14-24, 1:51-58, 2:35-50, 3:13-24, 4:40-52, 6:66-7:16, 7:8-11 (in this example, *Tazuma* teaches using “urlredirector” to redirect URL requests: “Urlredirector also participates in performing access control by using ACL 54 to determine whether to allow a URL to be retrieved by a browser or to redirect the browser instead.”), 9:40-62, 12:1-7, 22:20-27, 22:58-66.

V. GROUND 1: CLAIMS 1-2 ARE OBVIOUS OVER *FAGES, KORYCKI, JAI, AND HAYDEN*

The ’721 Patent attempts to distinguish prior art methods that balance convenience features of web browsers with providing browser security. *’721 Patent*, 1:24-45. Namely, the ’721 Patent highlights the need “to facilitate the organization of content in web browsers into different categories without extra overhead incurred by the user.” *’721 Patent*, 1:46-53. The ’721 Patent’s purported contribution to the art is implementing a daemon to track active browser instances and using a browser extension to communicate with the daemon via a native application. *’721 Patent*,

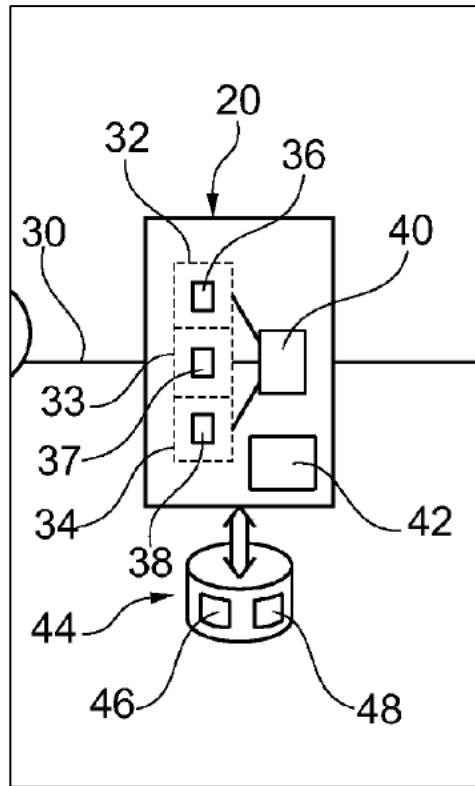
Claim 1; '721 FH, 370-373, 363-368. Critically, these elements were well-known in the prior art.

Petitioner proposes a rejection based on *Fages*, which performs similar browser security to the '721 Patent but does not explicitly discuss the implementation details that *Korycki*, *Jai*, and *Hayden* teach. *Korycki* teaches the claimed daemon, native application, and browser extension. See Claim 1(d)(i)-1(d)(iii). *Jai* teaches comparing a requested URL with all the open URLs in one or more browser instances. *Hayden* teaches confirming if a browser instance is already open and, if not, can launch an instance to handle a requested URL. See Claim 1(d), 1(d)(iv). Together, the *Fages-Korycki-Jai-Hayden* combination teaches all elements of Claim 1.

A. Claim 1

1. Claim 1(Pre): “A system, comprising: a processor, operable to:”

To the extent the preamble is limiting, *Fages* teaches a system for providing browser security and a server computer for implementing its system. *Fages*, [0050] (“FIG. 1 is a schematic illustration of a system for browsing on the Internet[.]”), [0058]-[0063], [0084]-[0089], claims 24-25. As depicted in Figure 1 below, *Fages*’s system 2 comprises server 20 including server module 40:



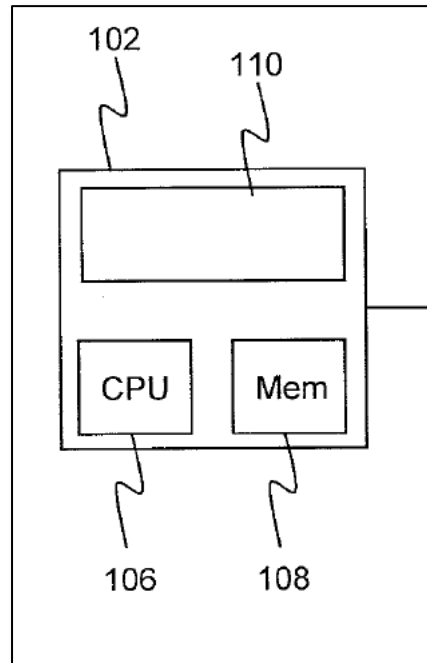
Fages, Fig. 1 (excerpt); *see also id.*, [0055]-[0056]. *Fages*’s server 20 includes an “information-recording medium comprising instructions for executing” the described browsing methods when these instructions are “executed by an electronic computer.” *Fages*, [0044], [0064]-[0065], [0096]. A POSITA would have understood or found obvious that server 20 would have included a processor (e.g., CPU) to execute those instructions. *Dec.*, ¶¶140-142 (describing processor executing instructions to implement server module 40 and anti-virus software program 42, and instructions stored in memory 44).

Notably, *Fages* teaches “[e]ach user station has an electronic computer 50 executing a client module 52 of the software program for taking remote control. For

example, the computer 50 is that of a central processing unit of a desktop computer.” *Fages*, [0065]-[0069]; *see also id.*, [0010] (describing “computer resources” including “processor time”). *Fages*’s server would have had a processor for executing programming instructions similarly to how the user stations include a processor for executing programming instructions. *Dec.*, ¶¶142-143. Further, it was well known that server computers included a processor for executing instructions, as this is a basic function of a server computer. *Dec.*, ¶¶76-80, 143. *Fages* also teaches server 20 executes program instructions stored in memory 44, which a POSITA would have understood or found obvious to have been performed by a processor in the server computer. *Dec.*, ¶143.

a) *Korycki* teaches a Processor

Additionally, or alternatively, *Korycki* teaches “a processor (CPU) 106” in a user station computer (device 102). *Korycki*, 5:32-38, 13:36-38 (“The components 202 to 218 shown in FIG. 2 are preferably implemented in software for execution by the CPU 106 on the device 102”). *Korycki*’s processor performs similar security functions as *Fages*’s server 20. *Korycki*, 3:20-41, 5:46-48, 13:7-34. Processor 106 is depicted below:



Korycki, Fig. 1 (excerpt). A POSITA would have understood or found obvious that server 20 would have natively included a processor to perform its functions in the same way that *Korycki*'s computer includes a processor to perform its functions. *Dec.*, ¶145. A POSITA would have been motivated to modify *Fages*'s server computer to include a processor to execute its programming instructions, consistent with *Korycki*'s teachings of using a processor to execute programming instructions. *Dec.*, ¶¶147-148.

b) Motivation to Modify *Fages* per *Korycki*

A POSITA would have been motivated to modify *Fages* per *Korycki*'s teachings to utilize a processor in a server computer to execute programming instructions. *Dec.*, ¶¶147-148. This modification would have combined prior art elements according to known methods to yield the predictable result of a server

computer performing *Fages*'s functions by executing instructions via a processor. *Dec.*, ¶147. Further, this use of a known technique would have improved the similar computing devices in the same way. *Dec.*, ¶147. This modification would have had a reasonable expectation of success because processors were well known, off-the-shelf components that were ubiquitous in computers before the priority date of the '721 Patent. *Dec.*, ¶148.

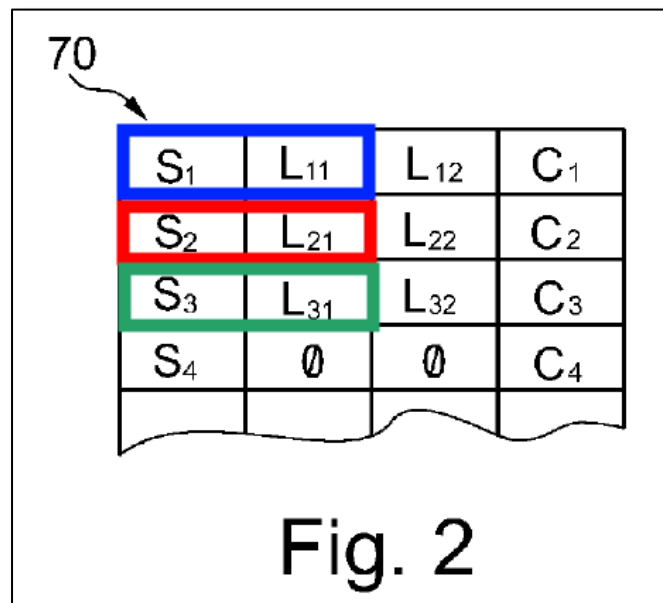
2. Claim 1(a): “receive data defining a first web content class and a second web content class, the first web content class defining a first plurality of Uniform Resource Locators (URLs), and the second web content class defining a second plurality of URLs different than the first plurality of URLs, the second web content class further being associated with sensitive information presenting a security risk;”

a) “receive data defining a first web content class and a second web content class...”

Fages teaches an administrator loads server 20 with data (“receive data”) defining a non-secure content class (“first web content class”) and a secure content class (“second web content class”) such as sensitive/secure websites for online banking. *Fages*, [0084]-[0086]; *see also id.*, [0018]-[0024], [0044]-[0046], [0072]-[0074]. In this way, the administrator loads server 20 with data defining first and second web content classes that each define a plurality of URLs (“first plurality of Uniform Resource Locators (URLs),” “second plurality of URLs different than the first plurality of URLs”) where each content class defines a different category of URLs. *Fages*, [0084]-[0086]; *see also id.*, [0018]-[0024], [0044]-[0046], [0056].

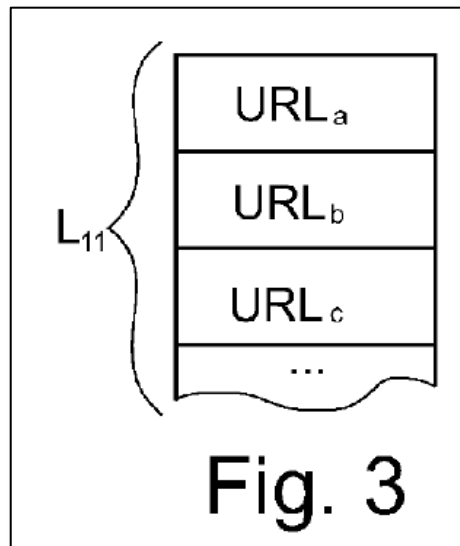
b) “the second web content class defining a second plurality of URLs different than the first plurality of URLs”

Specifically, *Fages* teaches an administrator “gets connected to the server 20” and “uses this connection to define...different lists L_{i1} L_{i2} [.]” *Fages*, [0084]-[0086]. The list of URLs L_{i1} defines “secure” or “sensitive” web content accessible only via a respective “secure sessions” such as session S_1 , S_2 , or S_3 . *Fages*, [0072]-[0074], [0091]-[0093], [0098]-[0100]. *Fages* teaches secure session S_1 has a corresponding sensitive/secure URL list L_{11} , secure session S_2 has a corresponding sensitive/secure URL list L_{21} , and secure session S_3 has a corresponding sensitive/secure URL list L_{31} . *Fages*, [0072]-[0073], Fig. 2; *Dec.*, ¶¶153-154. Fig. 2 below depicts secure session S_1 and URL list L_{11} , secure session S_2 and URL list L_{21} , and secure session S_3 and URL list L_{31} :



Fages, Fig. 2.

Fages's URL lists define a plurality of URLs. List L_{11} "has at least three URLs respectively denoted as URL_a , URL_b , and URL_c . These URLs correspond to URLs of sensitive Internet sites which have to be protected against attacks of the second type. These URLs are accessible only through the session S_1 ." *Fages*, [0073]. A POSITA would have understood that similarly, list L_{21} would be accessible only through session S_2 , and likewise list L_{31} would be accessible only through session S_3 . *Dec.*, ¶156. The URLs in list L_{11} are depicted below:



Fages, Fig. 3.

c) “the second web content class further being associated with sensitive information presenting a security risk”

Fages's data defining sensitive URLs in list L_{i1} are associated with sensitive information presenting a security risk, such as online banking websites. *Fages*, [0056], [0088] (“a sensitive site such as an Internet site enabling the management of

a user's bank accounts"). Indeed, for the same reasons the '721 Patent describes [a URL request to](#) www.bankofamerica.com as "sensitive" such that a new browser handles that request for "security reasons[,] *Fages's* URLs to banking account websites handled by a separate browser instance are "*associated with sensitive information presenting a security risk.*" *Dec.*, ¶¶158-160; compare '721 Patent, 4:65-5:4, Figs. 1-2 with *Fages*, [0056], [0088].

d) "*first web content class defining a first plurality of Uniform Resource Locators (URLs)*"

Fages also teaches an administrator defines "unsecured" web content ("*first web content class*") that may be accessed only via unsecured session S₄ because the "unsecured session enables connection with any Internet site except the sensitive Internet sites." *Fages*, [0072]. Specifically, if a user is browsing "an unsecured Internet session such as for example the session S₄, then [] stage [90]³ of unsecured browsing is carried out on the Internet network 4. At the stage 90, the user can browse on all the Internet sites accessible on the Internet 4, except for those

³ *Fages* has a typographical error "stage 50," which Petitioner corrected to read "stage 90." *Dec.*, ¶161. Element 50 correctly refers to "computer 50," whereas element 90 refers to "stage 90" in the sentence immediately following "stage 50." *Fages*, [0066] ("electronic computer 50"), [0070]-[0071] ("computer 50"), [0091] ("stage 90"), Fig. 1 (depicting computer 50), Fig. 6 (depicting stage 90); *Dec.*, ¶161.

corresponding to URLs listed in the lists L_{i1} .” *Fages*, [0091]. Thus, *Fages* teaches defining a *first plurality of URLs* that include “all the Internet sites” not listed in lists L_{i1} . Then, “at a step 92, whenever the user types out a new URL or whenever he is redirected to a new URL, for example following the selection of an Internet link, this new URL is compared with the URLs contained in all the lists L_{i1} . If a new URL does not belong to any of the lists L_{i1} , then the connection to this Internet site is authorized” and the Internet browser connects to the site at step 94. *Fages*, [0091]. However, if the requested URL is on a list L_{i1} , then “at a step 96, the connection to the Internet site corresponding to this URL is not authorized” which “therefore prevents the user getting connected to a sensitive Internet site during an unsecured Internet session” (session S_4). *Fages*, [0092]. Thus, the *first web content class* (non-secure/non-sensitive) *defines a first plurality of URLs* which include the set of URLs not included in sensitive/secure lists L_{i1} , for example. *Dec.*, ¶¶161-164.

3. Claim 1(b): “generate a first browser instance, the first browser instance defining a browser instance type predetermined to be associated with the first web content class;”

a) ’721 Intrinsic Record - “browser instance type”

Claim 1(b) recites a first “*browser instance type*” is “*associated with the first web content class*[.]” See Claim 1(b).

The ’721 Patent specification describes browser type/browser instance type in relation to content classes, where one browser instance type handles “sensitive”

content and another browser instance type is for “general” browsing. ’721 Patent, Abstract, 4:33-5:21 (describing “novice user” embodiment where one fresh browser instance (FBI) is used to handle sensitive sites and a separate browser instance handles “all other sites[.]”). Thus, there is (1) a secure browser instance/browser instance type and (2) an unsecured browser instance/browser instance type.

This comports with the ’721 Patent’s file history. During prosecution, the Applicant amended the claims to recite “generate a first browser instance, the first browser instance defining a browser instance type predetermined to be associated with the first web content class.” ’721 FH, 280. The Examiner then rejected the amended claims with prior art teaching a private browser instance handling sensitive content and a standard/default browser instance handling general content. ’721 FH, 287 (“web site communication is handled by the user’s standard or default web browser”), *id.* (“the private browser contacts the web site associated with the URL and waits for the remote server to return the requested web page to the private browser.”), 287-288, 339-340. Applicant did not dispute the Examiner’s apparent interpretation of browser instance/browser instance type. Nor did Applicant amend the claims to require different browsers altogether, such as from different companies/vendors, e.g., Chrome and Safari.

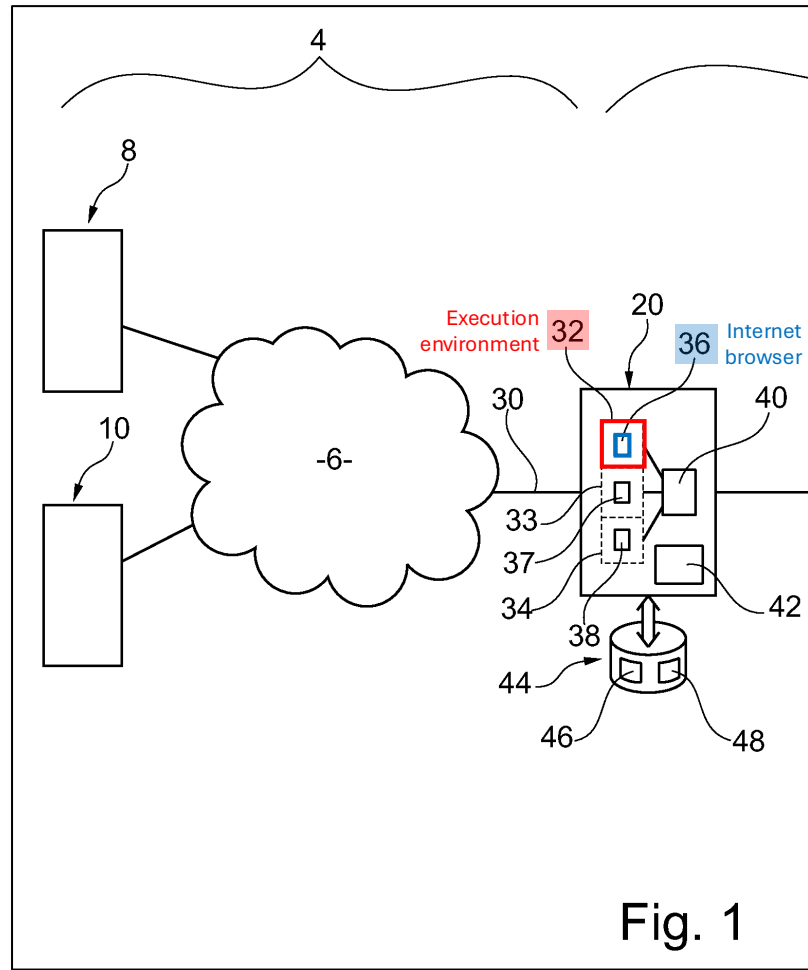
b) *Fages*'s “first browser instance” and “browser instance type”

Fages teaches the same “browser instance”/“browser instance type” as the '721 Patent. *Fages* teaches generating a first browser session/instance (“*first browser instance*”) in an execution environment. *Fages*, [0003] (“an access server connected to the Internet and capable of the parallel execution of **several instances of an Internet browser**”), [0018]-[0020], [0044]-[0048]. A POSITA would have understood that *Fages*'s browser sessions are each a *browser instance* at least because *Fages* uses those terms synonymously. *Fages*, [0003] (“several instances of an Internet browser”), [0072] (table 70 shows “list of Internet sessions configured on the server 20.”), [0072] (“Here, the first column of this table lists identifiers S_i of different Internet sessions which the user of the station 26 can launch.”), [0104]; *Dec.*, ¶¶171-174.

Fages teaches the first browser session/instance defines a *first browser instance type* (e.g., non-secure/non-sensitive) *predetermined to be associated with the first web content class* (e.g., non-sensitive content in URL). *Fages*, [0001]-[0003], [0018]-[0020]. “Secure”/“sensitive” and “non-secure”/“non-sensitive” are “browser instance types” consistent with the '721 Patent's specification. *See* '721 Patent, 4:35-37 (“the FBI system enables the processing of two types of sites for a novice user: sensitive sites and every other type of site”); *Dec.*, ¶¶175-176. When a user in a non-secure browsing session S_4 tries to access a URL not on the

predetermined lists of sensitive/secure URLs, a connection is made to the Internet site using the first Internet browser (e.g., browser 36 of execution environment 32). *Fages*, [0018]-[0020], [0076], [0091]-[0092]. *See* Claim 1(Pre).

In more detail, *Fages* teaches a first browser instance used for session S_4 regarding non-secure web content (e.g., non-sensitive URLs). *Fages*, [0018]-[0020], [0044]-[0046], [0072]-[0074], [0084]-[0093] (“If...the identifier S_i acquired corresponds to an unsecured Internet session such as for example the session S_4 , then a stage [90] of unsecured browsing is carried out on the Internet network 4. At the stage 90, the user can browse on all the Internet sites accessible on the Internet 4, except for those corresponding to URLs listed in the lists L_{i1} .”). As depicted in Fig. 1, Internet browser 36 is executed in execution environment 32:



Fages, Fig. 1 (excerpt), [0063], [0089]-[0093].

A POSITA would have understood browser execution environments 33 and 34 (along with browsers 37 and 38) would likewise have been utilized to access URLs, and *Fages* teaches partitioning browsers into separate execution environments so sensitive/secure URLs are not accessed by a browser accessing non-sensitive/non-secure URLs. *Fages*, [0104]; *Dec.*, ¶179. In other words, *Fages* teaches that each of its execution environments utilizes browsers for sensitive/secure and non-sensitive/non-secure URLs, but browsers are “partitioned” such that a

browser in a single environment (e.g., environment 32) does not access both sensitive/secure and non-sensitive/non-secure URLs at the same time or in a single session. *Dec.*, ¶¶180-182.

4. Claim 1(c): “intercept a request to access web content from the first browser instance, the request defining a URL of the second plurality of URLs such that the web content is associated with the second web content class;”

Fages teaches Internet browser instance 36 in execution environment 32 accesses content from the plurality of URLs of non-sensitive content and “a stage [90] of unsecured browsing is carried out on the Internet network 4.” *Fages*, [0088]-[0091]. When a user submits a request for a new URL from the first browser instance while browsing in non-secure session S₄, “this new URL is compared with the URLs contained in all the lists L_{i1}.” *Fages*, [0091]. If the requested URL is on the list of sensitive/secure URLs, then access to this sensitive/secure URL is blocked because an “unsecured session enables connection with any Internet site except the sensitive Internet sites.” *Fages*, [0088]-[0089], [0091]-[0092], [0072]-[0074]. Another browser instance may be utilized to access a sensitive URL, as detailed in Claim 1(e) below. *Infra*, Claim 1(e); *see also Fages*, [0098]-[0099], [0109]-[0110]. Otherwise, if the URL is non-sensitive (i.e., not contained in a sensitive URL list), “the connection to this Internet site is authorized” and the URL is accessed, for example, using browser 36 in execution environment 32 and session S₄. *Fages*, [0090]-[0093].

Fages server module 40 “*intercept[s]*” requests to access sensitive/secure web content that are sent to server 20 from client module 52 (i.e., *intercept a request to access web content from the first browser instance, the request defining a URL of the second plurality of URLs such that the web content is associated with the second web content class*). *Fages*, [0072]-[0074], [0086]-[0088], [0090]-[0093], [0098]-[0099], [0109]; *Dec.*, ¶¶185-187. In embodiments, instead of simply blocking the attempt, the sensitive URL is automatically sent to a different session (e.g., session S₁-S₃) of a different browser environment (e.g., browser execution environment 33 or 34) rather than a browsing environment being used for non-secure browsing (e.g., browser execution environment 32) of session S₄. *Fages*, [0092], [0109]-[0110]. In either case, server 20 including server module 40 intercepts the request to access web content and either allows, blocks, or opens a new browser instance in another browsing environment in response to the request based on the browser execution environment and list settings. *Dec.*, ¶¶188-190.

5. Claim 1(d): “confirm that the URL is not being accessed by another separate already-running browser instance associated with the second web content class, by”

a) *Fages*’s Teachings

Fages teaches multiple browser instances and browser execution environments and “prevent[ing] the user getting connected to a sensitive Internet site during an unsecured Internet session.” *Fages*, [0063], [0072]-[0073], [0088]-[0092],

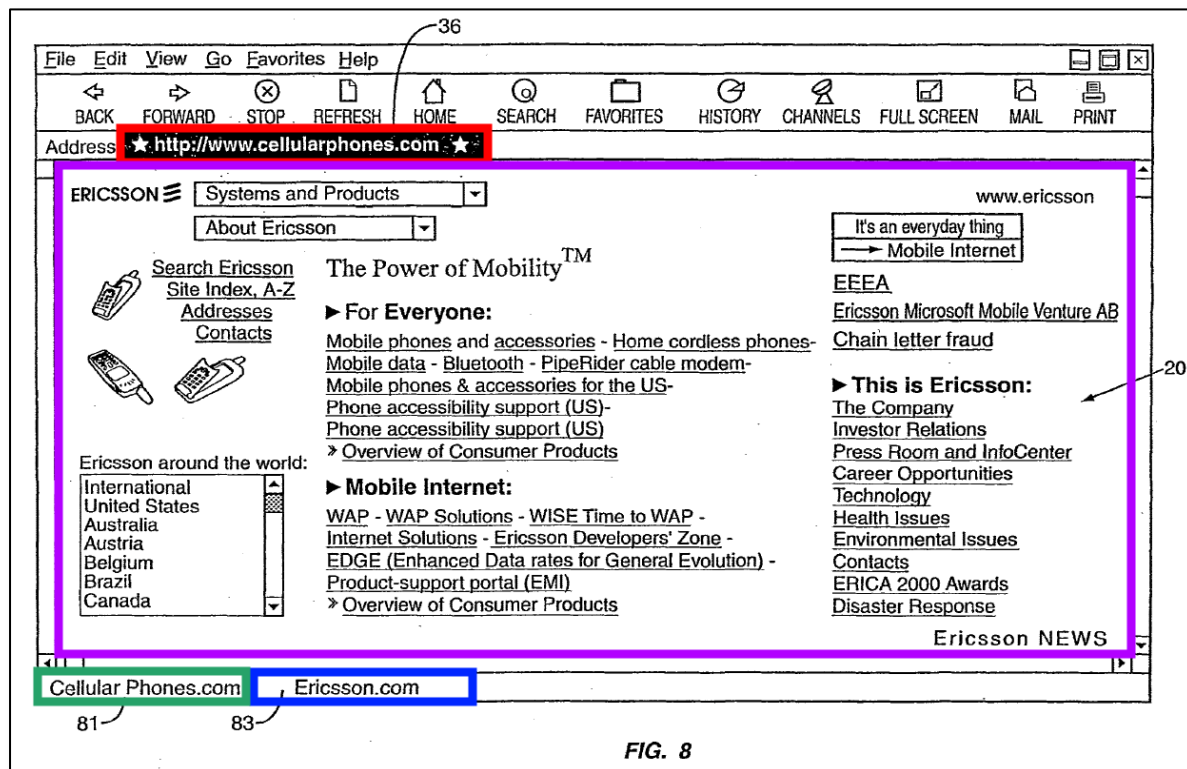
[0104], [0109]-[0110], Fig. 1. *See* Claim 1(a)-(c). *Fages* launches non-secure/non-sensitive URLs in browser 36, for example, and launches secured/sensitive URLs in a different browser instance (e.g., 37 or 38). *Fages*, [0044]-[0046], [0109]-[0110]; *Dec.*, ¶192. A POSITA would have understood that *Fages* utilizes separate browser execution environments and separate browsers such that sensitive URLs are not accessed using a browser that accesses non-sensitive URLs, and therefore “the malicious Internet site cannot affect the working of the sensitive Internet site since the Internet browsers used to connect to these two sites are executed in execution environments that are independent of one another.” *Fages*, [0044]-[0046], [0018]-[0021], [0104], [0109]-[0110]; *Dec.*, ¶193. However, *Fages* does not expressly discuss *confirming that the URL is not being accessed by another separate already-running browser instance associated with the second web content class*.

b) *Jai*’s URL Comparison

Jai teaches “compar[ing] any new URL to be accessed to any other URLs” that are currently open/active “to determine if such new URL is already being accessed by other browsers.” *Jai*, Abstract. *Jai* teaches that in this way, “duplicate applications of the same site may also be avoided.” *Jai*, 3:22. “[P]rior to accessing and loading another site, [*Jai*’s process] searches other open applications and determines whether the new request is actually a duplicate of an already open

application. If it is a duplicate, the existing application is displayed and the second requested application of the same site is deleted.” *Jai*, 3:22-4:2.

Specifically, *Jai*’s Fig. 8 “is a schematic diagram of a web browser illustrating a web site having a minimized application of a second website.” *Jai*, 4:20-21. This schematic is shown below:



Jai, Fig. 8. *Jai*’s Fig. 8 depicts address box 36 (for typing in a URL), website 20 (website a user is currently browsing), location 83 (a “currently loaded website,” specifically www.ericsson.com, which is the same website 20 the user is currently browsing), and location 81 (another “currently loaded website” that is “loaded and

has been minimized,” specifically www.cellularphones.com). *Id.*; *see also id.*, 12:14-18.

Jai teaches that an “aspect of the present invention is illustrated in Figure 8 and prevents redundant applications of the same website 20 from being opened at a given time. When a user inputs a URL and requests that the associated site b[e] opened, a search is made of the other loaded websites 20. If a loaded site having the same URL is already open, it will be displayed to the user without a second application being opened.” *Jai*, 12:8-14. Namely, “[w]hen the user enters the requested URL, the present invention will check the other loaded applications. Upon finding that another application of cellularphones.com is already open, this website [www.cellularphones.com] will be displayed in website 20 without opening a redundant application of cellularphones.com.” *Id.*, 12:17-13:2.

In the proposed combination, *Fages* teaches partitioning sensitive and non-sensitive URLs in different browser instances. *Supra*, Claim 1(a). Modifying *Fages*’s native partitioning functionality with *Jai*’s URL comparison technique allows *Fages*’s system to check whether that URL is already open before opening a duplicative instance. *Dec.*, ¶¶194-196, 200-207 (noting this is simply adding a step to check open URLs before partitioning a browser to open a new URL). For example, if a user requested to launch sensitive URL_a, *Fages-Jai* would confirm whether

URL_a is already open in secure sessions S₁-S₃ before partitioning URL_a to a secure browser instance. *Dec*, ¶¶194-196, 200-207.

In more detail, *Fages* would utilize *Jai*'s URL comparison functionality after a user requests a URL. *Dec*, ¶203. For example, *Fages-Jai* compares requested sensitive URL_a against all the URLs that are currently open (based on *Jai*'s teachings). *Dec*, ¶¶200-207. *Fages-Jai* would thus confirm that URL_a is not already open (i.e., *confirming that the URL is not being accessed by another separate already-running browser instance...*) when URL_a is not among the set of all the currently open URLs. *Dec*, ¶¶200-207. As a result, if the user is using unsecured/non-sensitive session S₄, the *Fages* combination would then launch a new secure browser instance, one of sessions S₁-S₃, to open sensitive URL_a. *See Claim 1(e)*.

c) *Hayden's Teachings* – “*another separate already-running browser instance*”

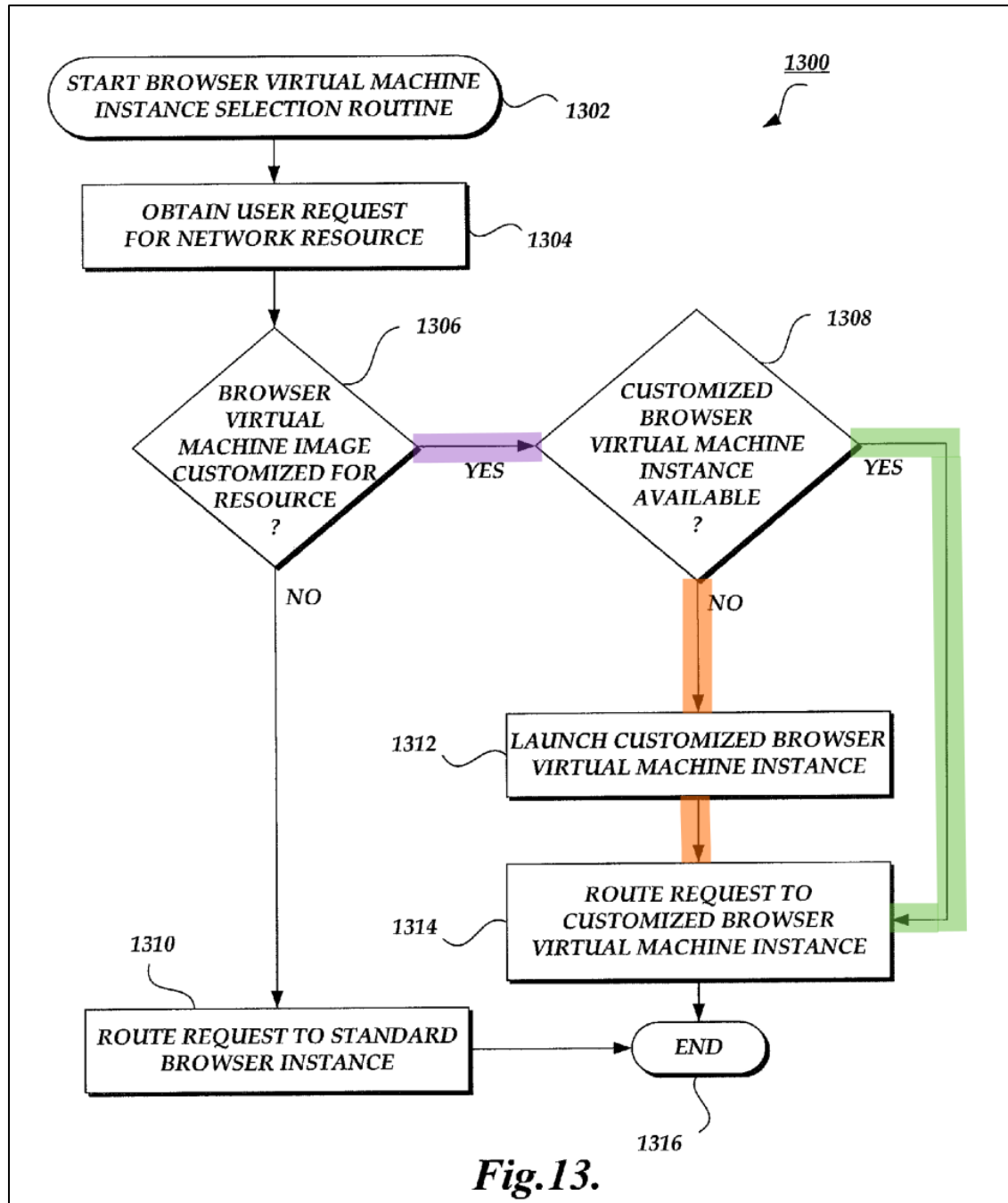
While *Fages-Jai* teaches checking if the requested URL is already open, it does not discuss checking other running browser instances. *Hayden* teaches determining if a particular browser instance is already launched, and if a particular browser instance is not already launched, *Hayden* teaches launching the particular browser instance. *Hayden*, 35:4-36:32, Fig. 13. In *Hayden*, “FIG. 13 [reproduced

later below] illustrates an example routine 1300 implemented by a virtual machine manager 182 or other component of the network computing provider 107 for routing requests to browser instances that have been customized to handle the request.” *Hayden*, 35:4-8. “[I]f a user of a client computing device 102 submits a request for <http://www.xyzwebsite.com>, the instance selection module 196 of the virtual machine manager 182 can determine which of the available network computing components to route the request to, based on whether a customized browser virtual machine image 186 has been created for the requested site and, if so, whether a browser has been instantiated from the image 186.” *Hayden*, 35:8-15.

At block 1306, “if a browser virtual machine image 186 containing a customized browser corresponding to the requested web site has already been created, the routine 1300 can proceed to decision block 1308.” *Hayden*, 35:38-62. At “decision block 1308, the instance selection module 196 can determine whether a browser, customized for the requested web site, has been instantiated from a browser virtual machine image 186 and is ready to process a new request.” *Hayden*, 36:7-11. “[I]n response to receiving the request, the instance selection module 196 can consult a database, query a service, or perform some other lookup of currently running customized browser instances to determine whether a browser customized for the <http://www.xyzwebsite.com>” “has been instantiated and is available.” *Hayden*, 36:11-18. If a browser customized for the requested website “has been

instantiated and is available,” *Hayden* teaches “the routine can proceed to block 1314. If none is available, the routine 1300 can proceed to block 1312.” *Hayden*, 36:18-20.

“[A]t block 1314, the instance selection module 196 can route the request to the customized browser for processing.” *Hayden*, 36:26-32. Otherwise, if no customized browser instance was available at block 1308, then at block 1312, “the instance selection module 192 or another component of virtual machine manager 182 or the network computing provider 107 can launch a customized browser from a browser virtual machine image 186 to process the request.” *Hayden*, 36:19-32. Afterwards, the process proceeds to block 1314. *Id.* Fig. 13 below depicts this process:



Hayden, Fig. 13, see also *id.*, 35:4-62, 7:32-43, 7:58-8:17, 8:45-9:1; *Dec.*, ¶199. A POSITA would have understood that *Hayden's* process would have confirmed whether “another separate already-running browser instance” was available, and if one existed but was not currently running, *Hayden's* technique would have launched

an instance. *Dec.*, ¶¶197-199. Indeed, Fig. 13 above shows a “yes” at blocks 1306 and 1308 lead to block 1314, and a “no” at block 1308 leads to block 1312. *Supra*, Fig. 13. In the latter case, a customized browser instance is launched and the process proceeds to block 1314. *Id.*

d) The *Fages-Korycki-Jai-Hayden* Combination Teaches “confirm that the URL is not being accessed by another separate already-running browser instance...”

It would have been obvious to modify the *Fages* combination to include URL and browser instance comparison steps to check the requested URL against all open browser instances (e.g., browser instances in every one of *Fages*’s execution environments). *Dec.*, ¶¶200-208. As modified, *Fages-Korycki-Jai-Hayden* confirms which browser instances and URLs within browser instances are already open and therefore confirms the requested URL is *not* already open before partitioning a new URL to a specific browser instance. *Fages-Korycki-Jai-Hayden* teaches the negative limitation of confirming that the URL is ***not*** being accessed by another separately already-running browser instance. *Dec.*, ¶¶200-208.

e) Motivation to Modify *Fages* per *Jai* and *Hayden*

A POSITA would have been motivated to modify *Fages* per *Hayden* to confirm if another separate already-running browser instance associated with the second web content class (e.g., sensitive content) is being accessed. *Dec.*, ¶¶209-217. Further, a POSITA would have been motivated to modify *Fages* per *Jai* to

confirm whether a requested URL is already open in a browser instance. *Dec.*, ¶¶209-217. Thus, the *Fages* combination would have confirmed whether a requested URL is already open (per *Jai*) in another browser instance (per *Hayden*) to determine if another browser instance should be opened to launch the requested URL. *Dec.*, ¶¶209-217.

That is, *Fages* teaches separate execution environments and separate browsers and partitioning the execution environments so that sensitive/secure content is not accessed from a non-sensitive/non-secure browser instance. *Fages*, [0063], [0072]-[0074], [0104], [0109]-[0110], [0019]-[0021]; *Dec.*, ¶¶210-212. Modifying *Fages* with *Jai* and *Hayden* would have allowed *Fages* to determine which browser instances and URLs are currently open to more efficiently manage execution environments and browser instances. *Dec.*, ¶213. In this manner, the *Fages* combination would have advantageously determined whether a currently running sensitive/secure browser instance is open to handle a requested sensitive URL and further whether the requested sensitive URL is already open and may be utilized without needing to launch another instance. *Dec.*, ¶213.

This modification would reduce the computing resources required for browser partitioning by obviating launching a separate browser instance for each sensitive URL when a currently running sensitive browser instance can handle that URL request. *Dec.*, ¶214. Indeed, *Fages*'s modified system would reuse a cached or

already open browser instance rather than open a repetitive URL, allowing a user to rejoin an existing session (e.g., existing sensitive banking session) while preserving session state. *Dec.*, ¶¶214-215. Similarly, if no sensitive/secure browser instance is currently running, *Fages's* modified system would determine that a new sensitive/secure browser instance should be launched to handle a requested URL. *Dec.*, ¶¶214-215.

This use of a known technique would have improved *Fages's* similar computer system in the same way as *Jai's* and *Hayden's*. *Dec.*, ¶216. Combining prior art elements such as *Jai's* URL comparison functionality (e.g., software) and *Hayden's* browser instance comparison functionality (e.g., software) with *Fages's* computer system according to known methods would have predictably yielded an improved *Fages* computer system. *Dec.*, ¶¶216 *Hayden*, 38:32-55 (“Any process descriptions, elements, or blocks in the flow diagrams described herein and/or depicted in the attached FIGURES should be understood as potentially representing modules, segments, or **portions of code which include one or more executable instructions** for implementing specific logical functions or steps in the process.”); *Jai*, 12:8-23 (“This redundancy checking feature may be **user programmable**.”).

A POSITA would have had a reasonable expectation of success modifying *Fages* for several reasons. *Dec.*, ¶¶217-224. *Fages*, *Jai*, and *Hayden* teach similar computer systems, such that the same hardware in *Fages* would implement *Jai's*

modification and *Hayden*'s modification. *Dec.*, ¶¶218-219. Further, *Jai* teaches the “present invention may be used in a variety of computing devices. A ‘computer device’ is defined herein as any data processing device including microprocessors, conventional personal computers...and the like. The term is meant to be construed broadly.” *Jai*, 13:3-7; *Dec.*, ¶220. Likewise, *Hayden* teaches “general purpose computing devices may be configured to implement the processes, algorithms, and methodology of the present disclosure with the processing and/or execution of the various data and/or components described above.” *Hayden*, 38:51-55, 7:43-47 (“The physical computing systems 184 and storage modes 188 can be computer servers.”).

A POSITA would have understood that modifying *Fages* would thus have required only straightforward modifications to programming instructions on *Fages*'s computing device, which would have been well within the skillset of a POSITA. *Dec.*, ¶¶221-222. For example, *Fages* already teaches a process by which an administrator can load/define/customize URL lists to server 20. *Fages*, [0085]. A POSITA would have understood that *Fages*'s native method for modifying server 20 would have been one method to modify *Fages* per *Jai*'s teachings and *Hayden*'s teachings with a reasonable expectation of success. *Dec.*, ¶¶223-224.

6. Claim 1(d)(i): “implementing a daemon outside the first browser instance that tracks active browser instances and launches a new browser in order to manage web access according to the data defining the first web content class and the second web content class,”

Petitioner notes that limitations 1(d)(i)-1(d)(iv) relate to claim elements including a daemon, a native application, and a browser extension. Petitioner's combination relies on *Fages*'s overall system (as modified by *Jai* and *Hayden*) and implements a daemon, a native application, and a browser extension as taught by *Korycki*.

a) *Fages*'s Teachings

Fages teaches a first browser instance, and *Fages-Jai-Hayden* teaches tracking active browser instances and URLs and launching a new browser instance to manage web access (e.g., by utilizing "partitioning") according to data defining the first web content class and second web content class. *See* Claim 1(a)-1(d) (describing *Jai*'s URL comparison and *Hayden*'s browser instance detection functionality to confirm an instance is not already running a requested URL). *Fages-Jai-Hayden* does not expressly discuss utilizing a daemon to perform this functionality, but a POSITA would have been motivated to modify *Fages-Jai-Hayden* to use a daemon, per *Korycki*.

b) *Korycki*'s Teachings

Korycki teaches performing computer security by checking whether a user is allowed to access certain content, such as a web application, which is "executed in a first web browser instance[.]" *Korycki*, 1:10-21, 1:66-2:31. Specifically, *Korycki* teaches a "**daemon**" 218 that accesses a list of approved partners for a service web

application and determines “whether the domain of the partner associated with the partner web application 212 (which was indicated in the subscribe request message) is present in the ACL, i.e. whether the service provider has approved the partner web application 212 for communication with the service provider web application 204.” *Korycki*, 8:41-9:8, 9:60-10:22. *Korycki* teaches “only desired partner web applications are allowed to communicate with the service provider web application[,]” and “[a]ny other web application (other than those on the list of partners approved by the service provider) may be prevented from participating in communication with the service provider web application.” *Korycki*, 1:66-2:31. In one example, *Korycki* teaches “comparing a domain associated with the partner web application with the entries of the list of partners approved by the service provider” to determine “whether the partner web application is approved for communication.” *Korycki*, 3:9-13.

Korycki’s daemon 218 performs background processes related to the device security described above. *Korycki*, 7:21-33 (“The publish-subscribe daemon application 218 is a native part of the service provider client application and is always running in the background at the device 102.”), 9:60-10:22, 6:66-7:6.

Fages-Jai-Hayden-Korycki thus teaches a “daemon” that “*confirm[s]* that the URL is not being accessed by another separate already-running browser instance associated with the second web content class” by “implementing a daemon outside

the first browser instance that tracks active browser instances and launches a new browser in order to manage web access according to the data defining the first web content class and the second web content class.” Dec., ¶¶228-234.

c) Motivation to Modify *Fages-Jai-Hayden* per *Korycki*

A POSITA would have been motivated to utilize a daemon to control tracking active browser instances and launching a new browser to manage web access according to the data defining the first and second web content classes in *Fages-Jai-Hayden*’s modified system because daemons were a well-known, commonly utilized piece of software/set of programming instructions implemented to perform background tasks that a user would not have needed to actively control. *Dec.*, ¶¶102-108, 235-242. *Fages* already teaches automatically performing its functionality but does not expressly teach the implementation details of what programming technique is utilized for performing its background tasks. *Dec.*, ¶236. A POSITA would have been motivated to modify *Fages* to implement its functionality using a daemon to advantageously monitor web browser instances outside the web browsers because it is easier to monitor one process from another, rather than from inside the same process. *Dec.* ¶237. Moreover, external monitoring renders the system more robust, providing safeguards if the monitored process terminates abruptly (e.g., from a bug or attack), the monitoring process still remains to alert an administrator. *Dec.* ¶238.

Further, external monitoring is computationally easier, including using system call traces and resource usage calls. *Dec.* ¶¶238-239.

A POSITA would have also been motivated to modify *Fages-Jai-Hayden* per *Korycki* to achieve *Korycki*'s benefits. *Dec.*, ¶¶240-242. For example, *Korycki* teaches utilizing daemon 218 to achieve “[h]igher performance[,]” “[l]ower delay[,]” and “[l]ower cost” in its computing environment. *Korycki*, 11:24-12:30; *Dec.*, ¶240. Modifying *Fages* per *Korycki* would have appreciated these same benefits. *Dec.*, ¶240. Further, combining prior art elements such as *Fages*'s computer system and server with well-known computer programming techniques (e.g., daemon) according to known methods would have predictably improved *Fages-Jai-Hayden-Korycki* computer system. *Dec.*, ¶¶241-242.

A POSITA would have had a reasonable expectation of success modifying *Fages-Jai-Hayden* per *Korycki*. *Dec.*, ¶¶243. *Korycki* teaches implementing daemon 218 “in software for execution by the CPU 106 on the device 102” such that daemon 218 “may be provided by way of a computer program product embodied on a non-transient computer-readable medium which is configured so as when executed on the CPU 106 of the device 102 to perform the method steps described herein[.]” *Korycki*, 13:36-61. Implementing this known software process in *Fages-Jai-Hayden* would have required nothing more than the ordinary skill of a POSITA. *Dec.*, ¶244.

Further, *Fages*, *Jai*, and *Korycki* teach similar computer systems, such that *Fages*'s hardware would not have needed to be modified to implement *Korycki*'s modification. *Dec.*, ¶¶59-74, 245 (opining that *Fages*, *Korycki*, *Jai*, and *Hayden* are compatible with Linux-based operating systems). Modifying *Fages-Jai-Hayden* per *Korycki* would have utilized *Fages*'s native server modification process, where an administrator would have modified server 20 to utilize daemon 218. *Dec.*, ¶245. This modification would have required only straightforward modifications to programming instructions, which would have been well-within the skillset of a POSITA. *Dec.*, ¶246.

While *Korycki* teaches implementing its functionality in a “client-side” setup, *Korycki* is compatible with *Fages* because *Fages*'s setup involves a server executing browsers and communicating with web servers, and a user at a client device remotely accesses the browser via a remote control program such as “NX protocol” from “NOMACHINE (www.nomachine.com)[,]” “RDP (Remote Desktop Protocol) by Microsoft© or ICA (Independent Computing Architecture) by Citrix©[,]” for example. *Fages*, [0064], [0108]; *Dec.*, ¶247. *Fages* remote-control setup allows for the system to function “as if the user were in front of the screen of the machine on which the Internet browser is executed.” *Fages*, [0009], [0004]-[0005], [0011], [0048], [0064]-[0071] (“The remote-controlled Internet browsers are the same as those that could be directly executed on the user stations without the assistance of

the modules 40 and 52.”). Thus, *Fages* when modified by *Korycki* would have functioned similarly to how *Korycki* functions on a client computer that executes Internet browsers locally. *Dec.*, ¶¶248-249.

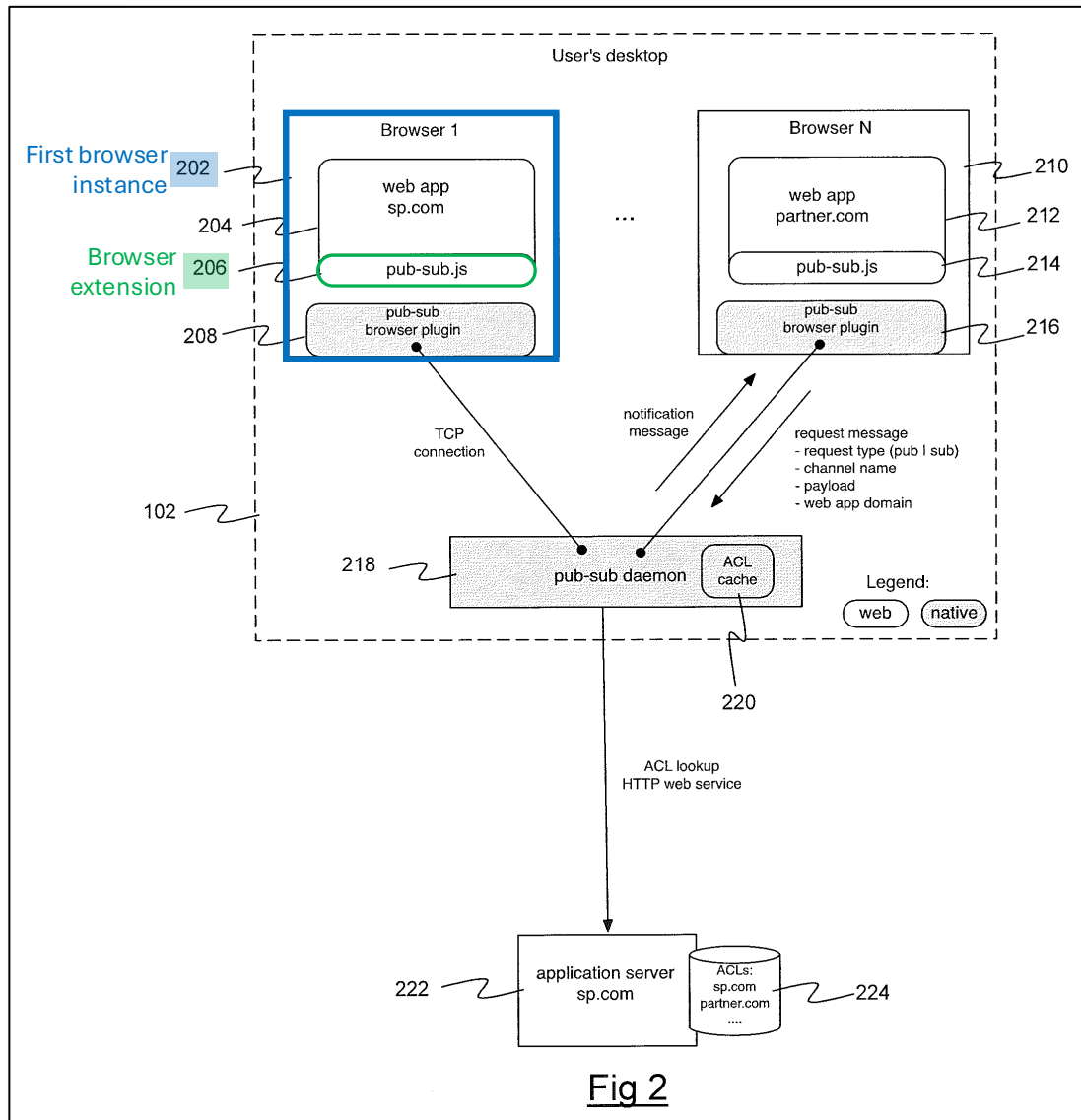
7. Claim 1(d)(ii): “receiving the request via a browser extension associated with the first browser instance,”

a) *Korycki*’s Browser Extension

Fages does not expressly discuss a browser extension for its browsers. However, browser extensions were well known in the prior art. For example, *Korycki* teaches utilizing “JavaScript” “pub-sub.js 206” (also referred to as “widget 206”), which a POSITA would have understood is a “*browser extension*” of browser 202. *Korycki*, 6:40-65; *Dec.*, ¶250. This is consistent with the disclosures of the ’721 Patent. ’721 Patent, 10:9-31 (describing “JavaScript” browser extension for Firefox and Chrome).

Korycki teaches pub-sub.js 206 communicates with pub-sub browser plugin 208 (“*native application*[.]” see Claim 1(d)(iii)), which in turn communicates with daemon 218 via a TCP connection, for example. *Korycki*, 6:40-65, 7:21-49, 8:41-9:8, Fig. 2. Specifically, *Korycki* teaches “widget 206 which is a JavaScript Application Programming Interface (API). The widget 206 allows the service provider web application 204 to communicate using the publish-subscribe signaling scheme.” *Korycki*, 6:40-65, 7:21-49. In this way, *Fages*’s modified system “*receiv[es] the request [to access web content] via a browser extension [pub-sub.js]*

associated with the first browser instance.” Dec., ¶¶251-254. Korycki’s Fig. 2 below depicts browser 202 including pub-sub.js 206:



Korycki, Fig. 2.

b) Motivation to Modify *Fages-Jai-Hayden* per Korycki

A POSITA would have been motivated to modify *Fages-Jai-Hayden* per Korycki’s teachings to include a browser extension such as pub-sub.js 206 in the *Fages-Jai-Hayden-Korycki* system. Dec., ¶¶256-264. Similar to the daemon 218

above, *Fages-Jai-Hayden* already teaches performing its browser security functionality but does not expressly teach the implementation details of what programming technique/component is utilized for extending a browser's functionality and providing communication with other processes. *Dec.*, ¶258. Utilizing a browser extension in *Fages-Jai-Hayden* per *Korycki* would have provided *Fages-Jai-Hayden* with a well-known and widely utilized component for communicating with a native application and daemon to extend the browser's functionality. *Dec.*, ¶259.

Browser extensions were a well-known, commonly utilized piece of software/set of programming instructions implemented to extend browser functionality and provide communication with other processes/components in the computer system (e.g., native application). *Dec.*, ¶81-87. A POSITA would have been motivated to modify *Fages-Jai-Hayden* to incorporate a browser extension like *pub-sub.js* 206 to advantageously customize their browser and improve functionality across web pages for tasks important to the user, such as security. *Dec.*, ¶260; Ex. 1017 (*Google Team*), 20. Moreover, a POSITA would have been motivated to modify *Fages-Jai-Hayden* with *Korycki* for the same reasons above. *See* Claim 1(d)(i); *Dec.*, ¶235-249, 261-264.

A POSITA would have had a reasonable expectation of success modifying *Fages-Jai-Hayden* to include *Korycki*'s browser extension. *Dec.*, ¶¶265-267. For

example, *Korycki* teaches implementing its components including pub-sub.js 206 “in software for execution by the CPU 106 on the device 102” and specifically pub-sub.js 206 “may be provided by way of a computer program product embodied on a non-transient computer-readable medium which is configured so as when executed on the CPU 106 of the device 102 to perform the method steps described herein.” *Korycki*, 13:36-61. Implementing this known software process in *Fages-Jai-Hayden* would have required nothing more than the ordinary skill of a POSITA. *Dec.*, ¶265.

Further, *Fages*, *Jai*, *Hayden*, and *Korycki* teach similar computer systems, such that *Fages*’s hardware would not have needed to be modified to implement *Korycki*’s modification. *Dec.*, ¶266. Similar to above when modifying *Fages* per *Jai*, modifying *Fages-Jai-Hayden* per *Korycki* would have utilized *Fages*’s native server modification process, where an administrator would have modified server 20 to utilize pub-sub.js 206. *Dec.*, ¶267. This modification would have required only straightforward modifications to programming instructions, which would have been well-within the skillset of a POSITA. *Dec.*, ¶267.

8. Claim 1(d)(iii): “forwarding the request from the browser extension to the daemon using a native application,”

a) *Korycki*’s Teachings

As explained above, *Fages-Jai-Hayden-Korycki* teaches using daemon 218 and browser extension pub-sub.js 206. *Supra*, Claims 1(d)(i)-1(d)(ii). *Fages-Jai-Hayden* does not expressly discuss utilizing a “native application” in its system to

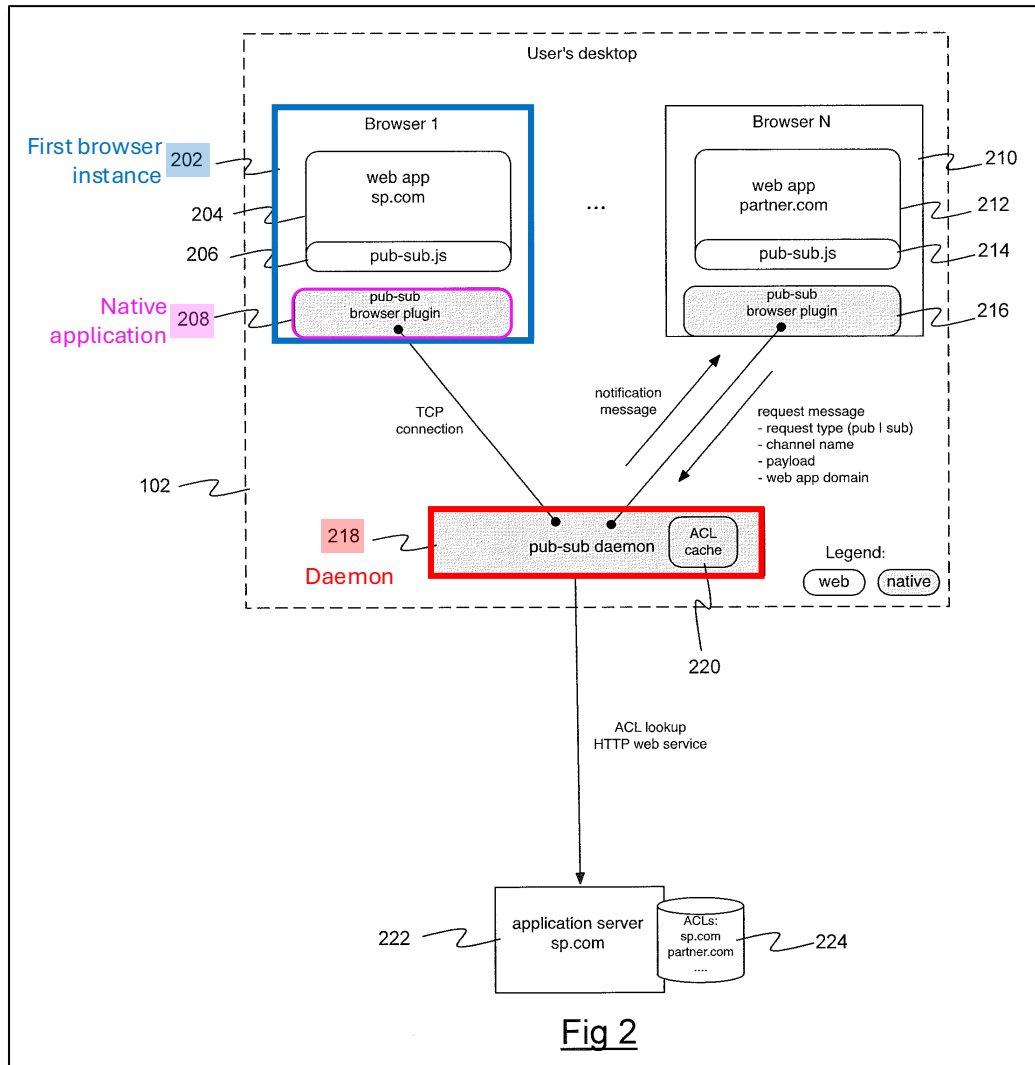
facilitate communication between a browser extension and daemon. *Korycki* teaches the claimed native application that communicates between pub-sub.js 206 (“*browser extension*”) and *daemon* 218.

Korycki’s pub-sub.js 206 browser extension is a JavaScript Application Programming Interface (API) implemented using “web-based technology (e.g. in accordance with the HTML 5 standard).” *Korycki*, 6:40-55, 7:21-33. *Korycki* teaches daemon 218 and publish-subscribe browser plugins 208 and 216 are “implemented using a native technology, that is, a technology that is native to the device platform of the device 102 (e.g. C++).” *Id.* Thus, a POSITA would have understood that pub-sub browser plugin 208 is a *native application*. *Dec.*, ¶270.

Korycki teaches pub-sub browser plugin 208 performs the same functionality as the claimed *native application*. *Korycki* teaches pub-sub.js 206 (“*browser extension*”) communicates with pub-sub browser plugin 208 (“*native application*”) because “widget 206 allows the service provider web application 204 to communicate using the publish-subscribe signaling scheme[,]” and pub-sub browser plugin 208 communicates with daemon 218, for example, “by establishing respective Transport Control Protocol (TCP) connections[,]” where plugin 208 and daemon 218 are both implemented using a C-based language, e.g., C++ (“*forwarding the request from the browser extension to the daemon using a native application*”). *Korycki*, 6:40-65, 7:34-48, 8:44-9:8 (teaching similar communication

between widget 214 and browser plugin 216 where “**widget 214 delegates the subscribe request to the native publish-subscribe browser plugin 216.**”); *Dec.*, ¶271.

A POSITA would have understood that pub-sub.js 206 facilitates communication between web application 204, browser plugin 208, and daemon 218, such that “publish-subscribe browser plugins 208 and 216 provide bridges between the respective web applications 204 and 212 and the publish-subscribe daemon application 218[,]” where pub-sub.js 206 “allows the service provider web application 204 to communicate[.]” *Korycki*, 7:44-46, 6:51-55. *Korycki*’s Fig. 2 below depicts browser 202 includes pub-sub browser plugin 208 (“*native application*”), where both plugin 208 and daemon 218 are shaded to indicate they are “native” technology.



Korycki, Fig. 2, 1:22-32, 1:66-2:19, 7:21-33, 8:54-9:8, 11:53-12:12; Dec., ¶¶273-274.

b) Motivation to Modify *Fages-Jai-Hayden* per Korycki

A POSITA would have been motivated to modify *Fages-Jai-Hayden* per Korycki's teachings to include a native application such as browser plugin 208 in the *Fages-Jai-Hayden-Korycki* system. Dec., ¶¶277-283. Similar to the daemon 218 and browser extension pub-sub.js 206 above, *Fages-Jai-Hayden* already teaches

performing its URL comparison functionality but does not expressly teach the implementation details of what programming technique/component is utilized for communicating between a browser extension and daemon. *Dec.*, ¶278. Utilizing a browser plugin (“*native application*”) in *Fages-Jai-Hayden* per *Korycki* would have provided *Fages-Jai-Hayden* with a well-known and widely utilized component for communicating with a browser extension and daemon and would have extended the browser’s functionality. *Dec.*, ¶279. Native applications were a well-known, commonly utilized piece of software/set of programming instructions implemented to extend browser functionality and provide communication with other processes/components in the computer system (e.g., native application). *Dec.*, ¶¶111, 279. A POSITA would have understood and been motivated to modify *Fages-Jai-Hayden* to implement its functionality using the well-known technique of implementing a native application like browser plugin 208 which is advantageously browser-agnostic and can access features of the native device. *Id.* Further, the native application can interface with available APIs within the operating system. *Dec.*, ¶¶111-113, 279 (citing *Kirkham* (Ex. 1028), [0003]; *IBM* (Ex. 1029), 2). A POSITA would have been motivated to modify *Fages-Jai-Hayden* for the same reasons discussed above, including applying similar motivation for including a browser extension in addition to a daemon. *See* Claim 1(d)(i)-1(d)(ii); *Dec.*, ¶¶235-249, 256-267, 280.

A POSITA would have had a reasonable expectation of success modifying *Fages-Jai-Hayden* per *Korycki* for the same reasons above. *Dec.*, ¶¶284-285 (describing *Korycki* implementing its components including browser plugin 208 “in software for execution by the CPU 106 on the device 102”). Further, *Fages*, *Jai*, *Hayden*, and *Korycki* teach similar computer systems, such that *Fages* would utilize the same hardware to implement *Korycki*’s modification. *Dec.*, ¶285. Similar to above when modifying *Fages* per *Jai* and *Hayden*, modifying *Fages-Jai-Hayden* per *Korycki* would have utilized *Fages*’s native server modification process, where an administrator would have modified server 20 to utilize browser plugin 208. *Dec.*, ¶285. This modification would have required only straightforward modifications to programming instructions, which would have been well-within the skillset of a POSITA. *Dec.*, ¶285.

9. Claim 1(d)(iv): “confirming, via the daemon that the web content is not associated with the first web content class and that another browser instance is not already launched and available to access the web content; and”

Fages as modified by *Jai* and *Hayden* teaches determining which browser instances are active and/or inactive (“confirming...another browser instance is not already launched and available to access the web content”). See Claim 1(d). *Fages* teaches determining whether a requested URL is associated with a first web content class. See Claim 1(a). *Fages* does not expressly discuss utilizing a daemon for this

process. *Fages* as modified by *Korycki* teaches utilizing a daemon to perform this task. *See* Claim 1(d)(i)-1(d)(iii).

Korycki teaches performing security functions with daemon 218 including accessing an Access Control List (ACL) to determine if a browser can access a particular website. *Korycki*, 7:34-49, 9:34-11:5 (including teaching daemon 218 performing security tasks such as “access[ing] the list (i.e. the ACL) of approved partners” to determine if a browser may access said partner (e.g., website/web application)), 13:7-35, 6:16-21 (“web applications may be web pages downloaded over the network”). In the proposed combination, *Fages*’s security process would have been modified to include a daemon to perform such background security processes. *Dec.*, ¶¶286-289.

Thus, in the *Fages-Jai-Hayden-Korycki* combination, a daemon would have been utilized to perform background processes such as confirming that web content is associated with a first or second web content class. *Dec.*, ¶¶286-289. A POSITA would have been motivated to modify *Fages-Jai-Hayden-Korycki* and would have had a reasonable expectation of success for the same reasons discussed above. *See* Claim 1(d)(i)-1(d)(iii).

10. Claim 1(e): “launch a new browser for accessing the web content associated with the second web content class to provide access to the URL of the second plurality of URLs via the new browser.”

Fages teaches “parallel execution of several Internet browsers in execution environments that are independent of one another” such that if a “URL belongs to a pre-recorded list of URLs of sensitive Internet sites to which connection from the first execution environment is not authorized,” *Fages* launches a new browser in parallel in a “second execution environment” “to connect to the sensitive Internet site identified by this URL[.]” *Fages*, [0044]-[0046]; *see also id.*, [0098]-[0099], [0104], [0109]-[0110] (“When several Internet sessions are executed simultaneously, they are displayed in distinct windows on the screen 58” or “in different tabs of a same window.”), *id.* (“server 20 automatically launches the execution of another Internet browser in another execution environment”); *supra*, Claim 1(a). A POSITA would have understood that if a user attempts to access non-sensitive/non-secure sites using browser 36 via session S₄ in execution environment 32, for example, then another browser instance (e.g., 37, 38) would be utilized for secure session S₁ to access a sensitive/secure URL in list L₁₁ in execution environment 33 or 34. *Fages*, Fig. 3; *Dec.*, ¶¶291-294; *see* Claim 1(a). In this way, *Fages* “partition[s]” non-sensitive/non-secure and sensitive/secure browser instances/sessions. *Fages*, [0104], [0109]-[0110]; *Dec.*, ¶¶291-294. *See* Claim 1(b).

B. Claim 2⁴

Fages teaches “rather than blocking a URL with which it is not possible to get connected through the Internet browser executed in the current execution environment, the server 20 **automatically launches the execution of another Internet browser** in another execution environment from which the connection to this URL is possible.” *Fages*, [0109]-[0110]; *see also id.*, [0024]-[0025], [0035], [0038]-[0039], [0092], [0100].

Fages also teaches the web content is associated with a web server. For example, *Fages* teaches Internet servers 8 and 10, where in one example, Internet server 8 “is an Internet server that enables online handling of bank accounts.” *Fages*, [0056], [0088] (“Thus, a sensitive site such as an Internet site enabling the management of a user’s bank accounts will be accessible only by means of an Internet browser whose configuration defined in the list C_i is appropriate to this use.”), [0091]-[0093], [0098]-[0100], [0008]. Thus, a POSITA would have understood that *Fages*’s web content is associated with a web server such as servers 8 and/or 10. *Dec.*, ¶¶296-298.

⁴ *See* Claims Appendix for Claims 2-18.

VI. GROUND 2: CLAIMS 3-5 ARE OBVIOUS OVER *FAGES, KORYCKI, JAI, HAYDEN, AND RUSSELLO*

A. Claim 3

a) '721 Patent's API

The '721 Patent describes intercepting system calls using a ptrace API. '721 *Patent*, 7:16-31 (“This API allows one program (tracer) to be notified and take control of every system call made by another program (browser).”), 8:62-65, 9:22-26. The '721 Patent claims the API intercepts system calls (claim 3) and controls system calls made by other programs and other browser instances (claim 4). '721 *Patent*, Claims 3-4.

b) *Russello's API*

Russello teaches utilizing “a process trace system call (ptrace)” to intercept “all or at least a set of system calls executed by the target process[.]” *Russello*, [0026], [0036], [0047]-[0048], [0092]. For example, *Russello* teaches its ptrace intercepts system calls “if a parent process launches a new child process” where “any system calls invoked by the new child process are intercepted and redirected to a monitoring entity associated with the parent process to implement a security action.” *Id.*, [0127]; *see also id.*, [0196], [0203], [0250]; *Dec.*, ¶¶305-311.

Russello also implements a “process monitor” in conjunction with ptrace to control system calls and execute security policies related to ptrace operations. *Russello*, [0026]-[0031], [0036] (“Each process monitor may be configured to

enforce security policies based on parameters extracted from the detected system calls.”), [0040]-[0049] (“a process monitor attached by the main monitor process to each new process created by the mother process and each process monitor being configured to monitor the system calls made by its attached process to the kernel[.]”), [0052]-[0056], [0064], [0092], [0196], [0203]-[0204], [0249]-[0252], [0256], [0258], [0293]-[0294], [0296].

Russello’s ptrace functionality is an *API* that controls system calls and enables *Russello*’s process monitor to control system calls. A POSITA would have understood that the process monitor is a tracer and the ptrace functionality is an *API* Dec., ¶¶305-311. The ’721 Patent’s description of a tracer program and an *API* overlaps with *Russello*’s teachings. ’721 Patent, 7:16-31, 8:62-9:26; Dec., ¶¶305-311. Thus, *Russello*’s process monitor is a tracer using a ptrace *API* that performs a *call configured to intercept a system call*. Dec., ¶¶305-311. In combination with *Fages* modified system, *Russello*’s ptrace performs *Fages*’s interception functionality such that it “*intercept[s]...the system call to the web server intended to provide access of the URL to the first browser instance.*” Dec., ¶¶305-311.

Russello teaches implementing its process monitoring and ptrace functionality in a web browser embodiment. *Russello*, [0406]-[0415]. In this example, *Russello* teaches utilizing a “monitoring entity 812” to perform ptrace functionality. *Russello*, [0203], [0406]-[0415].

c) Motivation to Modify *Fages-Korycki-Jai-Hayden* per *Russello*

A POSITA would have been motivated to modify *Fages-Korycki-Jai-Hayden* per *Russello* to achieve *Russello*'s security benefits. *Dec.*, ¶¶313-324. A POSITA would have understood that *Russello* teaches improving system security, such as using a process trace (ptrace) to intercept system calls and using process monitoring to enforce security policies (among other functionality), including in a web browsing example. *Russello*, [0015]-[0016], [0036], [0047]-[0048], [0406]-[0415] (teaching utilizing the security system for Chromium, including “code downloaded visiting a web page, executing a web application, and other code that it will be rendered and executed by Chromium processes.”); *Dec.*, ¶¶313-324, 109-110. This use of a known technique would have improved the similar *Fages-Korycki-Jai-Hayden* system in the same way as *Russello*'s computer system. *Dec.*, ¶¶313-324. Further, combining prior art elements such as *Russello*'s ptrace and process monitor with the *Fages-Korycki-Jai-Hayden* computer system's elements according to known methods would have yielded the predictable result of increased system security. *Dec.*, ¶¶313-324. Further, this modification would have informed a POSITA how to achieve additional security benefits by providing low-level implementation details not expressly taught in *Fages*. *Dec.*, ¶¶313-324.

A POSITA would have had a reasonable expectation of success modifying *Fages-Korycki-Jai-Hayden* per *Russello*. *Dec.*, ¶¶313-324. *Russello* teaches the

well-known ptrace and process monitoring functionality that were present in prior art computer systems, such as Linux. *Dec.*, ¶¶313-324. *Fages-Korycki-Jai-Hayden* does not expressly teach a specific operating system is required, such that a POSITA would have understood that using techniques for Linux would also have functioned in the modified *Fages-Korycki-Jai-Hayden-Russello* system. *Dec.*, ¶¶313-324. Modifying *Fages-Korycki-Jai-Hayden* per *Russello* would have required only straightforward changes to programming instructions, which would have been well-within the skillset of a POSITA. *Dec.*, ¶¶313-324.

To the extent it is argued that modifying *Fages-Korycki-Jai-Hayden* per *Russello* would have required utilizing a specific operating system environment such as Unix or Linux, a POSITA would have been motivated to modify *Fages-Korycki-Jai-Hayden-Russello* to utilize Linux to achieve *Russello*'s security benefits and because Linux was a well-known, well-understood, and commonly utilized operating system choice in the prior art. *Dec.*, ¶¶313-324.

B. Claim 4

1. Claim 4(Pre)

See Claim 3.

2. Claim 4(a)

a) Russello's Pausing Operations

Russello teaches using ptrace and processor monitor to pause operations associated with a system call. *Russello*, [0026] ("the kernel is configured to report

to the master process system calls made by the target process both prior to and after the execution of the system call, and in some embodiments, before the kernel actually resumes the target process execution.”), [0250] (teaching the kernel utilizes process monitoring and ptrace functionality to “**suspend the target process**” and “await a security action from that process monitor” to resume the target process), [0343]-[0346] (“The child process 375 also raises a SIGSTOP 380 to be **stopped** until the parent process 370 allows it to resume its execution.”). Operations would have been paused before resuming target process execution. *Dec.*, ¶¶327-329. Thus, when modifying *Fages-Korycki-Jai-Hayden* per *Russello*, the processor would pause operations associated with the first browser instance upon interception of the system call by utilizing *Russello*’s process monitoring functionality and ptrace API. *Id.*

b) Motivation to Modify *Fages-Korycki-Jai-Hayden* per *Russello*

A POSITA would have been motivated to modify *Fages-Korycki-Jai-Hayden* per *Russello* for the reasons discussed above. *See* Claim 3. Further, a POSITA would have been motivated to modify the *Fages* combination to achieve *Russello*’s security benefits by pausing browser operations upon interception of the system call to allow sufficient time to perform security operations and to prevent processes from occurring until the security procedures have completed. *Dec.*, ¶¶331-334.

Modifying *Fages-Korycki-Jai-Hayden* per *Russello* would have had a reasonable expectation of success for the reasons discussed above. See Claim 3.

3. Claim 4(b)

Fages-Korycki-Jai-Hayden does not expressly discuss the API controls system calls *made by other programs and other browser instances*. Specifically, *Korycki* teaches an API widget 206 for each browsing instance rather than a single API. *Russello* teaches *the API controls system calls made by other programs and other browser instances*.

Russello teaches its ptrace API controls system calls for all programs/processes set as master/target when invoked by a process monitor, and all programs/processes set as master/target would have included other programs and other browser instances. *Dec.*, ¶¶336-338; *Russello*, [0036], [0047]-[0048], [0092] (“any system calls invoked by the new child process are intercepted and directed to the embedded process monitor of the parent process to implement a security action.”), [0196], [0203]-[0204], [0267]-[0273] (“As the process monitor 72 is attached to the new process 74, it can intercept and control all the system calls that this process executes.”), [0406]-[0415] (“each monitoring entity retrieves and enforces security policies specific to the application corresponding to its process.”). Therefore, a POSITA would have understood that any other program or other browser instance set as a master/target to utilize *Russello*’s process monitoring and

ptrace functionality would thus have benefited from *Russello*'s improved security functionality by controlling system calls to these other programs and other browser instances in the same way as described above for the first browser instance. *Dec.*, ¶¶336-338.

a) Motivation to Combine *Fages-Korycki-Jai-Hayden* and *Russello*

A POSITA would have been motivated to modify *Fages-Korycki-Jai-Hayden* per *Russello* for the same reasons discussed above to achieve *Russello*'s improved computer security and benefits associated with such security. *Dec.*, ¶¶340-344. *See* Claims 3, 4(a). A POSITA would have had a reasonable expectation of success because *Russello* teaches applying its functionality to a first browser instance and utilizing its process monitoring and ptrace functionality for other programs. *Dec.*, ¶¶340-344.

C. Claim 5

a) '721 Patent's Tracer

The '721 Patent describes a tracer as a program enabled by a ptrace API to take control of system calls of another program. *'721 Patent*, 7:16-31 ("To intercept system calls, the system includes a ptrace API available in Linux. This API allows one program (tracer) to be notified and take control of every system call made by another program (browser)."), 8:62-65, 9:22-26.

b) *Russello's Tracer (Process Monitor Using ptrace)*

Fages-Korycki-Jai-Hayden does not expressly discuss implementing a tracer. *Russello* teaches implementing a tracer program, such as a process monitor, enabled by “ptrace” that can, for example, “intercept all or at least a set of system calls executed by the target process and report those to the master process prior to execution of such system calls.” *Russello*, [0026]. *Russello*, [0196] (“a process trace system call, e.g. **ptrace**, or any other mechanism that creates a link or attachment between processes which enables one process to control one or more aspects or functions of the other process.”), [0203]-[0204], [0250] (“When a process monitor invokes ptrace to attach to a target process, each time the target process invokes a system call the gateway will suspend the target process[.]”), [0258]. *See* Claims 3-4.

In the *Fages* combination, *Russello's* ptrace would have allowed *Fages's* system to utilize a tracer to track system calls and pause processes to supplement *Fages's* native security functionality used to determine whether a requested network object is allowed to be accessed by a first browser instance. *See* Claims 3-4.

c) Motivation to Combine *Fages-Korycki-Jai-Hayden-Russello*

A POSITA would have been motivated to modify *Fages-Korycki-Jai-Hayden* per *Russello* to implement a tracer such as a process monitor enabled by ptrace to track system calls, for example, from a first browser instance/session. *Dec.*, ¶¶347-

348, 350-353. *Fages* already teaches checking URL lists to determine if a browser instance can access sensitive URLs. *See* Claim 1. *Dec.*, ¶350. However, *Fages* does not expressly discuss utilizing a tracer to track system calls, which *Russello* teaches. *Dec.*, ¶350. Modifying the *Fages* combination per *Russello* would have been use of a known technique to implement a well-known tracer such as a process monitor enabled by ptrace to track system calls and enhance *Fages*'s security. *Dec.*, ¶¶350-353. This combination of prior art elements according to known methods would have yielded the predictable result of an improved *Fages* computer system, including because *Russello* provides low-level implementation details that *Fages* does not expressly describe. *Id.*

A POSITA would have had a reasonable expectation of success modifying the *Fages* combination because *Fages* already teaches a similar computing system and utilizing security such as URL lists. *Dec.*, ¶¶350-353. Implementing *Russello*'s process monitor tracer would have provided a well-known prior art solution for implementing additional security and would have enhanced the *Fages* combination in the same way as *Russello*. *Id.* Further, modifying *Fages-Korycki-Jai-Hayden* per *Russello* would have required only straightforward modifications to programming instructions, which would have been well-within the skillset of a POSITA. *Id.*

VII. GROUND 3: CLAIMS 6-8, 12-13, 16-18 ARE OBVIOUS OVER *FAGES*, *KORYCKI*, *JAI*, *HAYDEN*, *O'NIELL*, AND *CARDONA*

A. Claim 6

Fages-Korycki-Jai-Hayden does not expressly discuss the daemon comprises a multi-threaded C program that uses Unix domain sockets. *O'Niell* teaches a daemon implemented as a C program that uses Unix domain sockets for communication. *Cardona* teaches a C program daemon is multi-threaded.

a) *O'Niell's* C program Daemon Using Unix Domain Sockets

O'Niell teaches a computer system utilizing daemon 235 that uses Unix domain sockets for communication and is capable of utilizing other sockets such as Linux domain sockets. *O'Niell*, 5:28-54 (“Network stack 240 includes a domain socket handler 245 that creates a socket, such as a Unix domain socket or Linux domain socket, in response to a request from application 230 or daemon process 235.”), 5:62-6:30 (“process ID embedding handler 205 embeds the process name for the particular daemon process 235 in the socket pathname for the Unix domain socket that corresponds to the particular daemon process.”).

O'Niell utilizes computer program code for carrying out operations where the computer program code “may be written in any combination of one or more programming languages, including...Java, Smalltalk, C++ or the like and conventional procedural programming languages, such as the ‘C’ programming language or similar programming languages.” *O'Niell*, 4:26-41.

Therefore, *O’Niell’s* daemon is programmed in the C programming language. *Dec.*, ¶¶362-366. Further, *O’Niell’s* daemon would have utilized Unix domain sockets in embodiments where the daemon is a C program. *Dec.*, ¶¶362-366.

b) Cardona’s Multithreaded Daemon

Cardona teaches implementing a multithreaded daemon in a computer system. *Cardona*, Abstract (“A system and method for a multithreaded universal daemon (MUD) is described.”). *Cardona* teaches daemons are “agent programs that continuously operate, for example, on a server, and provide resources to client systems on the network. These background processes can handle a variety of tasks, including monitoring the network for incoming data.” *Cardona*, [0003], [0006]-[0009], Figs. 1-2.

Specifically, *Cardona* teaches its daemon is a “multi threaded daemon” to handle multiple threads or tasks concurrently. *Cardona*, [0036] (“The exemplary daemon according to the invention is **multithreading**, that is, it spawns multiple threads or tasks that exist concurrently.”), [0037] (“a single multi threaded daemon can service multiple connections.”), [0053] (“The resulting daemon is referred to as a multithreaded universal daemon (MUD), because it is capable of simultaneously monitoring many connections, up to the number that was specified as a parameter in the create function, and the same library functions are used to create and edit daemons that work with any type of data, using any protocol.”), [0055], Fig. 3.

**c) Motivation to Combine *Fages-Korycki-Jai-Hayden* per
O’Niell and *Cardona***

(1) O’Niell

A POSITA would have been motivated to modify *Fages-Korycki-Jai-Hayden* per *O’Niell* to advantageously utilize a daemon written in the C programming language (“*C program*”) and where the daemon uses Unix domain sockets. *Dec.*, ¶¶364-366, 370-377. The C programming language was well known and widely utilized in the prior art and would have offered widespread compatibility for a daemon when the daemon is written as a C program. *Dec.*, ¶¶364-366, 370-377, 102-108, 116-117. Implementing the daemon as a C program would have provided a standard programming language which combines features of both high-level and low-level languages and is a structured programming language that allows for complex programs to be broken into simpler programs (i.e., functions). *Id.* Further, C is a general-purpose programming language that can work efficiently in various environments including enterprise and home use. *Id.* Modifying *Fages-Korycki-Jai-Hayden* per *O’Niell* would have been use of a known technique that would have improved the similar *Fages-Korycki-Jai-Hayden* daemon and computer system in the same way as *O’Niell* by capturing the benefits of using the C programming language. *Id.*

Regarding using Unix domain sockets, a POSITA would have been motivated to modify *Fages-Korycki-Jai-Hayden* to utilize Unix domain sockets for daemon

communication to capture the advantages of Unix domain sockets. *Dec.*, ¶¶373-377. For example, using Unix domain sockets would have allowed the *Fages-Korycki-Jai-Hayden* daemon to use local files to send and receive data instead of having to rely on network interfaces. *Dec.*, ¶¶373-377. A POSITA would have understood that programs that run on the same server can communicate with each other using Unix domain sockets to advantageously bypass network-based sockets and protocols. *Dec.*, ¶¶373-377. This use of a known technique would have improved the similar *Fages-Korycki-Jai-Hayden* computer system and daemon in the same way as *O’Niell’s* computer system and daemon. *Dec.*, ¶¶373-377.

This modification would have had a reasonable expectation of success. *Dec.*, ¶¶375-377. C was a well-known and widely utilized programming language, and these modifications would have required only straightforward changes to programming instructions, which would have been well-within the skillset of a POSITA. *Dec.*, ¶¶375-377. Further, *Fages-Korycki-Jai-Hayden* does not provide express requirements for a particular programming language to be used, which would have led a POSITA to choose a widely utilized and powerful language such as C for programming a daemon. *Id.* Further, programming a daemon in languages such as C was well known in the prior art, further leading a POSITA to a reasonable expectation of success. *Id.*

Further, utilizing Unix domain sockets was a well-known and widely utilized technique which was compatible with the C programming language. *Dec.*, ¶¶375-377. Thus, when modifying *Fages-Korycki-Jai-Hayden* per *O’Niell*, the computer system and daemon would have been able to utilize Unix domain sockets for communication (e.g., to/from the daemon) without having to rely on networking protocols, although such native functionality in *Korycki* would have been maintained as optional. *Id.* This modification would have required only straightforward changes to programming instructions, which would have been well-within the skillset of a POSITA. *Id.*

(2) Cardona

A POSITA would have been motivated to modify *Fages-Korycki-Jai-Hayden-O’Niell* per *Cardona’s* teachings of implementing a daemon as a multithreaded program. *Dec.*, ¶¶379-386. *Cardona* teaches benefits of implementing a daemon as a multithreaded program including handling multiple threads or tasks concurrently. *Id.*; *Cardona*, [0036]-[0037]. A POSITA would have been motivated to modify the *Fages* combination to allow a daemon to handle multiple threads and/or tasks at the same time. *Dec.*, ¶¶379-386. This use of a known technique of implementing a daemon as multithreaded would have improved the similar *Fages-Korycki-Jai-Hayden-O’Niell* daemon in the same way as *Cardona’s* daemon. *Id.* Further, *Cardona’s* express teaching, suggestion, and/or motivation to utilize a

multithreaded daemon would have motivated a POSITA to modify the daemon in the *Fages* combination to be multithreaded to capture the benefits of a multithreaded daemon. *Id.*

A POSITA would have had a reasonable expectation of success modifying the *Fages* combination's daemon to be multithreaded. *Dec.*, ¶¶379-386. *Korycki* already teaches its daemon 218 is connected to multiple browsers, but *Korycki* does not expressly teach its daemon is multithreaded. *Id.* A POSITA would have understood that modifying a daemon that handles multiple tasks to be multithreaded would have required only straightforward programming changes. *Id.* Further, a POSITA would have understood that *Cardona*'s daemon is compatible with a server, such as *Fages*'s server. *Cardona*, [0012], [0015]; *Dec.*, ¶¶379-386. Thus, modifying the *Fages* combination would have required only straightforward changes to programming instructions, which would have been well-within the skillset of a POSITA. *Dec.*, ¶¶379-386.

B. Claim 7

1. Claim 7(Pre)

To the extent the preamble is limiting, *Fages* teaches a method for facilitating different categories of content (e.g., sensitive, non-sensitive) to be displayed in different independent browser instances. *See* Claim 1.

2. Claim 7(a)

Fages alone or in view of *Korycki* teaches a processor and non-transitory tangible storage medium storing programming instructions executed by a processor. *See* Claim 1(Pre).

3. Claim 7(a)(i)

See Claim 1(a). A POSITA would have understood the data defining which browser instances access sensitive versus non-sensitive web content are “*rules[,]*” as claimed. *Dec.*, ¶389.

4. Claim 7(a)(ii)

Fages teaches lists defining URLs. *See* Claim 1(b). A POSITA would have understood that a URL would have specified or been related to a particular IP address, such that identifying a request for a particular URL would have also identified a request to access an IP address related to said URL. *Dec.*, ¶¶390, 94-101 (explaining how a URL is converted into a corresponding IP address via a Domain Name System (DNS) query). Therefore, *Fages* teaches *identifying a request to access an IP address*. *See* Claim 1(b). Further, *Fages* teaches a first browser instance of the first browser type. *Id.*

5. Claim 7(a)(iii)

See Claims 1(c)-1(d), 7(a)(ii).

6. Claim 7(a)(iii)(1)

See Claims 1(d)(i), 6 (*multi-threaded program*).

7. Claim 7(a)(iii)(2)

See Claim 1(d)(ii).

8. Claim 7(a)(iii)(3)

See Claim 1(d)(ii), 1(d)(iii), 6 (*domain sockets*).

9. Claim 7(a)(iv)

See Claims 1(e), 7(a)(ii). A POSITA would have understood that *Fages's* “new browser” (Claim 1(e)) is *a second browser instance of the second browser type*. Dec., ¶¶395-396.

C. Claim 8

See Claims 2 and 7.

D. Claim 12

1. Claim 12(Pre)

See Claims 1(Pre), 7(Pre)-(a). *Fages* alone or in view of *Korycki* teaches a processor and non-transitory computer readable medium encoded with a set of instructions.

2. Claim 12(a)

See Claim 7(a)(i).

3. Claim 12(b)

See Claim 7(a)(ii).

4. Claim 12(c)

See Claims 1(c), 7(a)(iii).

5. Claim 12(c)(i)

See Claim 7(a)(i), 7(a)(iii)(1).

6. Claim 12(c)(ii)

See Claim 7(a)(iii)(2).

7. Claim 12(c)(iii)

See Claim 7(a)(iii)(3).

E. Claim 13

1. Claim 13(Pre)

See Claim 12(Pre).

2. Claim 13(a)

See Claims 1(e), 7(a)(iv).

3. Claim 13(b)

See Claims 2, 8.

F. Claim 16

See Claims 1(a), 12.

G. Claim 17

See Claims 1(a), 12.

H. Claim 18

See Claims 7(a)(i), 12.

**VIII. GROUND 4: CLAIMS 9-10, 14-15 ARE OBVIOUS OVER *FAGES*,
*KORYCKI, JAI, HAYDEN, O'NIELL, CARDONA, AND RUSSELLO***

A. Claim 9

See Claims 3 (discussing how the *Fages* combination teaches the claimed API call functionality), 7(a)(ii) (teaching DNS query concerting URL to IP address). A POSITA would have been motivated to further modify the *Fages* combination with *O'Niell*, *Cardona*, and *Russello* for the same reasons above in Grounds 2 and 3. *Dec.*, ¶¶313-324, 370-386; *Supra*, Ground 2 (*Russello* motivation and reasonable expectation of success); Ground 3 (*O'Niell* and *Cardona* motivation and reasonable expectation of success).

B. Claim 10

See Claims 5 (discussing tracer in the *Fages* combination) and 7.

C. Claim 14

1. Claim 14(Pre)

See Claim 12(Pre).

2. Claim 14(a)

See Claims 3 and 12.

3. Claim 14(b)

See Claims 3 and 12.

D. Claim 15

See Claims 5, 10, 12.

IX. GROUND 5: CLAIM 11 IS OBVIOUS OVER *FAGES*, *KORYCKI*, *JAI*, *HAYDEN*, *O’NIELL*, *CARDONA*, AND *TAZUMA*

A. Claim 11

1. Claim 11(Pre)

See Claim 7(Pre).

2. Claim 11(a)

See Claim 7(a)(ii).

a) *Tazuma*’s DNS Query

To the extent Patent Owner argues *Fages* does not teach generating a DNS query to resolve the IP address, *Tazuma* teaches this. *Tazuma* teaches a computer “performing a Domain Name Service (DNS) query to determine the IP address of the destination server.” *Tazuma*, 8:38-49. “DNS is a standard IP protocol and service that can be used to translate domain names (e.g., www.yahoo.com, etc.) to IP addresses (e.g. 200.35.53.2, etc.).” *Id.*; *Dec.*, ¶424.

b) Motivation to Modify the *Fages* Combination per *Tazuma*

A POSITA would have been motivated to modify *Fages-Korycki-Jai-Hayden-O’Niell-Cardona* per *Tazuma* to advantageously generate a DNS query to resolve an IP address. *Dec.*, ¶¶426-429. This use of a known technique would have improved the modified *Fages*’s similar computer system in the same way as *Tazuma*’s computer system. *Dec.*, ¶¶426-427. Further, this use of a known technique would have resulted in functional DNS queries to translate URLs including domain

names into IP addresses, resulting in a functional computer system to access websites on the Internet, for example. *Dec.*, ¶427.

A POSITA would have had a reasonable expectation of success modifying the *Fages* combination per *Tazuma* because *Tazuma* teaches a standard DNS query technique and because the *Fages* combination's computer system is similar to and compatible with *Tazuma*'s. *Dec.*, ¶¶428-429. A POSITA would have understood that modifying the *Fages* combination would have required only straightforward modifications to programming instructions, which would have been well-within the skillset of a POSITA. *Dec.*, ¶429.

3. Claim 11(b)

See Claim 7(a)(iii).

4. Claim 11(c)

See Claim 7(a)(iii). *Fages* teaches utilizing multiple independent browsers where different browsers and sessions are related to first and second content classes. *See* Claims 1, 7.

a) '721 Patent DNS Response

The '721 Patent describes rewriting an IP address in a DNS response when the requested IP address is not allowed to be accessed. *'721 Patent*, 7:36-44.

b) *Tazuma*'s DNS Response

Tazuma teaches redirecting a browser in a network by “chang[ing] an original destination address to another address in such a way that the client is not aware of

the change.” *Tazuma*, 4:40-52. Thus, *Tazuma* teaches redirecting/modifying (*rewriting*) an IP address in a DNS response when the requested IP address is not allowed to be accessed. *Tazuma*, 1:14-24, 1:51-58, 2:35-50, 3:13-24, 4:40-52, 7:8-11 (“Urlredirector also participates in performing access control by using ACL 54 to determine whether to allow a URL to be retrieved by a browser or to redirect the browser instead.”), 9:40-62, 12:1-7 (teaching Access Control List (ACL)-based redirection), 13:10-44 (“a redirect of the web request...can result in a modified URL...”), 22:20-27, 22:58-66; *Dec.*, ¶¶433-434. Thus, a POSITA would have understood that *Tazuma* teaches redirection at least by rewriting the URL or IP address. *Dec.*, ¶¶433-434. A POSITA would have understood that this redirection prevents a browser from accessing the requested IP address or URL. *Dec.*, ¶¶433-434.

c) Motivation to Modify *Fages* Combination per *Tazuma*

A POSITA would have been motivated to modify the *Fages* combination per *Tazuma* to advantageously perform redirection. *Dec.*, ¶¶436-442. Redirection would have been use of a known technique of returning a rewritten IP address or URL in response to a request being unauthorized or unable to be fulfilled instead of simply returning no response at all. *Dec.*, ¶¶436-442. This use of a known technique would have improved the similar *Fages* combination’s computer system and *Tazuma*’s computer system in the same way. *Dec.*, ¶¶436-442. *See* Claims 1, 7. Using

Tazuma's redirection technique would have informed a user instead of simply denying the request. *Dec.*, ¶¶436-442.

A POSITA would have had a reasonable expectation of success modifying the *Fages* combination per *Tazuma*. *Dec.*, ¶¶436-442. This modification would have required only straightforward changes to programming instructions, which would have been well-within the skillset of a POSITA. *Id.* Further, *Korycki* teaches ACL and *Fages* teaches security similar to ACL by utilizing authorized URL lists for particular browser instances and sessions. *Dec.*, ¶¶436-442; *see* Claim 1. Thus, modifying the *Fages* combination would have required only adding redirection functionality per *Tazuma*. *Dec.*, ¶¶436-442.

5. *Claim 11(d)*

a) *Tazuma's* Returning a Rewritten IP Address

The *Fages* combination as further modified by *Tazuma* teaches *returning the IP address as rewritten to the first browser instance*. *Dec.*, ¶¶443-445. For example, *Tazuma* teaches “in operation 456 the URL requested is checked against the ACL. If the URL passes the ACL check, in operation 460 WRH 42 performs the request on behalf of the browser, returning any reply back to the browser. **If the URL is determined to be restricted, in operation 464 a reply is sent to the browser causing it to be redirected to a pre-designated redirection URL**, e.g., a splash or login page.”). *Tazuma*, 22:21-27, 22:58-66, 4:40-52, 1:21-24; *Dec.*, ¶¶443-445.

**b) Motivation to Modify the *Fages* Combination per
*Tazuma***

A POSITA would have been motivated to modify the *Fages* combination and would have had a reasonable expectation of success in doing so for the same reasons discussed above and applied herein. *See* Claim 11(c). *Dec.*, ¶447.

X. CONCLUSION

For the foregoing reasons, Petitioner respectfully requests *inter partes* review of the Challenged Claims.

Respectfully submitted,

By: /s/ Paul R. Hart
Paul R. Hart, Reg. No. 59,646

COUNSEL FOR PETITIONER

XI. MANDATORY NOTICES UNDER 37 C.F.R. § 42.8(A)(1)

A. Real Parties-In-Interest

Petitioner and CrowdStrike Holdings, Inc. are the real parties-in-interest. 37 C.F.R. § 42.8(b)(1).

B. Related Matters

The '721 Patent is presently the subject of a patent infringement lawsuit filed against Petitioner in the United States District Court for the Western District of Texas, *Skysong Innovations, LLC v. CrowdStrike, Inc. and CrowdStrike Holdings, Inc.*, Case No. 7:25-cv-00040-DC-DTG. The '721 Patent has also been asserted against Fortinet, Inc. in the United States District Court for the Eastern District of Texas, *Skysong Innovations, LLC v. Fortinet, Inc.*, Case No. 2:25-cv-00098.

C. Lead and Back-Up Counsel

Petitioner provides the following designation and service information for lead and back-up counsel. 37 C.F.R. § 42.8(b)(3) and (b)(4).

Lead Counsel	Back-Up Counsel
Paul R. Hart, (Reg. No. 59,646) Paul.Hart@eriseip.com <u>PTAB@eriseip.com</u> <u>Postal and Hand-Delivery Address:</u> ERISE IP, P.A. 717 17 th St., Suite 1400 Denver, Colorado 80202 Telephone: (913) 777-5600 Fax: (913) 777-5601	Adam P. Seitz (Reg. No. 52,206) Adam.Seitz@eriseip.com Hunter A. Horton (<i>pro hac vice</i> to be submitted) hunter.horton@eriseip.com Kevin J. Rongish (Reg. No. 78,716) kevin.rongish@eriseip.com <u>Postal and Hand-Delivery Address:</u> ERISE IP, P.A. 7015 College Blvd., Suite 700 Overland Park, Kansas 66211 Telephone: (913) 777-5600 Fax: (913) 777-5601

D. 37 C.F.R. § 42.8(b)(4) – Service Information

Please address all correspondence to the lead and back-up counsel as shown above. Petitioner consents to electronic service by e-mail at the e-mail addresses provided above.

CLAIM LISTING APPENDIX

Claim Designation	Claim Language
Claim 1(Pre)	A system, comprising: a processor, operable to:
Claim 1(a)	receive data defining a first web content class and a second web content class, the first web content class defining a first plurality of Uniform Resource Locators (URLs), and the second web content class defining a second plurality of URLs different than the first plurality of URLs, the second web content class further being associated with sensitive information presenting a security risk;
Claim 1(b)	generate a first browser instance, the first browser instance defining a browser instance type predetermined to be associated with the first web content class;
Claim 1(c)	intercept a request to access web content from the first browser instance, the request defining a URL of the second plurality of URLs such that the web content is associated with the second web content class;
Claim 1(d)	confirm that the URL is not being accessed by another separate already-running browser instance associated with the second web content class, by
Claim 1(d)(i)	implementing a daemon outside the first browser instance that tracks active browser instances and launches a new browser in order to manage web access according to the data defining the first web content class and the second web content class,
Claim 1(d)(ii)	receiving the request via a browser extension associated with the first browser instance,
Claim 1(d)(iii)	forwarding the request from the browser extension to the daemon using a native application,
Claim 1(d)(iv)	confirming, via the daemon that the web content is not associated with the first web content class and that another browser instance is not already launched and available to access the web content; and
Claim 1(e)	launch a new browser for accessing the web content associated with the second web content class to provide access to the URL of the second plurality of URLs via the new browser.

Claim Designation	Claim Language
Claim 2	The system of claim 1, wherein the new browser instance is launched automatically for display, and the web content is associated with a web server.
Claim 3	The system of claim 2, wherein the processor is operable to execute an API call configured to intercept a system call to the web server, the system call to the web server intended to provide access of the URL to the first browser instance.
Claim 4(Pre)	The system of claim 3, further comprising:
Claim 4(a)	wherein the processor is operable to pause operations associated with the first browser instance upon interception of the system call; and
Claim 4(b)	wherein the API controls system calls made by other programs and other browser instances.
Claim 5	The system of claim 1, wherein the processor is operable to implement a tracer to determine whether or not a requested network object is allowed to be accessed by the first browser instance.
Claim 6	The system of claim 1, wherein the daemon comprises a multi-threaded C program that uses Unix domain sockets to communicate with the browser extension and other extensions
Claim 7(Pre)	A method for facilitating different categories of content to be displayed in different independent browser instances, the method comprising:
Claim 7(a)	utilizing a processor in communication with a non-transitory tangible storage medium storing instructions that are executed by the processor to perform operations comprising:
Claim 7(a)(i)	accessing data defining rules for delegating web access associated with a first content class to a first browser type and delegating web access associated with a second content class to a second browser type;
Claim 7(a)(ii)	identifying a request to access an IP address by a first browser instance, the first browser instance being of the first browser type;

Claim Designation	Claim Language
Claim 7(a)(iii)	determining that the IP address is associated with the second content class such that the IP address should not be accessed by the first browser instance and confirming that a separate browser instance is not already in use and available to access the request, by:
Claim 7(a)(iii)(1)	implementing a daemon, the daemon comprising a multi-threaded program executed outside the first browser instance that tracks active browser instances and launches a new browser instance in order to manage according to the rules,
Claim 7(a)(iii)(2)	receiving the request via a browser extension associated with the first browser instance,
Claim 7(a)(iii)(3)	forwarding the request from the browser extension to the daemon using domain sockets; and
Claim 7(a)(iv)	generating a second browser instance to access the IP address, the second browser instance being of the second browser type.
Claim 8	The method of claim 7, wherein the second browser instance is launched automatically to a client device for display.
Claim 9	The method of claim 7, further comprising: executing an API call configured to intercept a system call to a web server associated with the IP address, the system call initiated by the first browser instance.
Claim 10	The method of claim 7, further comprising: executing a tracer to determine whether or not a requested network object is allowed to be accessed by the first browser instance.
Claim 11(Pre)	The method of claim 7, further comprising:
Claim 11(a)	generating a domain name system (DNS) query to resolve the IP address;
Claim 11(b)	determining that the IP address is associated with the second content class; and
Claim 11(c)	issuing a DNS response, the IP address being rewritten in the DNS response to prevent the first browser instance from accessing the IP address and information of the second content class; and
Claim 11(d)	returning the IP address as rewritten to the first browser instance.

Claim Designation	Claim Language
Claim 12(Pre)	A non-transitory computer readable medium encoded with a set of instructions for facilitating different categories of content to be displayed in different independent browser instances, the set of instructions, executable by a processor, comprising:
Claim 12(a)	accessing data associating a first web content class with a first browser type and a second web content class with a second browser type;
Claim 12(b)	accessing a request to access web content via a first browser instance, the first browser instance being of the first browser type; and
Claim 12(c)	comparing the web content to the first web content class and the second web content class to determine whether the web content is accessible to the first browser instance,
Claim 12(c)(i)	by implementing a daemon, the daemon comprising a multi-threaded program executed outside the first browser instance that tracks active browser instances and launches a new browser instance in order to manage according to the rules,
Claim 12(c)(ii)	receiving the request via a browser extension associated with the first browser instance,
Claim 12(c)(iii)	forwarding the request from the browser extension to the daemon using domain sockets.
Claim 13(Pre)	The non-transitory computer readable medium of claim 12, further comprising:
Claim 13(a)	generating a second browser instance to access the web content,
Claim 13(b)	wherein the second browser instance is launched automatically to a client device for display.
Claim 14(Pre)	The non-transitory computer readable medium of claim 12, further comprising:
Claim 14(a)	executing an API call configured to intercept a system call to a web server,
Claim 14(b)	the system call associated with the first browser instance.
Claim 15	The non-transitory computer readable medium of claim 12, further comprising: executing a tracer to determine whether or not the first browser instance is authorized to access a requested network.

Claim Designation	Claim Language
Claim 16	The non-transitory computer readable medium of claim 12, wherein the second web content class is associated with sensitive information indicating that accessing the web content by the first browser instance presents a security risk.
Claim 17	The non-transitory computer readable medium of claim 12, wherein the first web content class is associated with a first plurality of URLs and the second web content class is associated with a second plurality of URLs, the first plurality of URLs being different from the second plurality of URLs.
Claim 18	The non-transitory computer readable medium of claim 12, wherein the data further defines rules limiting web access by the first browser instance to URLs defined by the first web content class.

APPENDIX OF EXHIBITS

Exhibit 1001	U.S. Patent No. 10,574,721 (“’721 Patent”)
Exhibit 1002	File History of the ’721 Patent (“’721 File History”)
Exhibit 1003	Declaration of Erez Zadok, Ph.D. (“Dec.”)
Exhibit 1004	U.S. Patent Application Publication No. 2012/0069401 to Fages et al. (“Fages”)
Exhibit 1005	U.S. Patent No. 9,009,853 to Korycki et al. (“Korycki”)
Exhibit 1006	International Publication No. WO 02/37333 to Jai (“Jai”)
Exhibit 1007	U.S. Patent Application Publication No. 2014/0137184 to Russello et al. (“Russello”)
Exhibit 1008	U.S. Patent No. 8,424,017 to O’Niell et al. (“O’Niell”)
Exhibit 1009	U.S. Patent Application Publication No. 2003/0061257 to Cardona (“Cardona”)
Exhibit 1010	U.S. Patent No. 7,493,414 to Tazuma (“Tazuma”)
Exhibit 1011	<i>Curriculum vitae</i> of Erez Zadok, Ph.D.
Exhibit 1012	<i>Dictionary of Computer and Internet Terms</i> . 10 th Edition. Barron’s Educational Series, Inc. (2009). (“Barron’s”)
Exhibit 1013	<i>PC Basics Get a Great Start</i> . Gateway, Inc. (2002). (“Gateway”)
Exhibit 1014	Dell, Inc. <i>Dell PowerEdge R620</i> . (2012). (“Dell”)
Exhibit 1015	U.S. Patent No. 7,689,666 to Commons (“Commons”)
Exhibit 1016	U.S. Patent No. 5,787,254 to Maddalozzo, Jr. et al. (“Maddalozzo, Jr.”)
Exhibit 1017	Google Chrome Team. <i>20 Things I Learned About Browsers and the Web</i> . (2010). Available at: https://web.archive.org/web/20120127221853/https://dusk.geo.st.edu/gis/Google_Book_20Things.pdf (“Google Team”)
Exhibit 1018	U.S. Patent No. 8,914,813 to Sigurdsson et al. (“Sigurdsson”)
Exhibit 1019	U.S. Patent No. 8,990,710 to Warner et al. (“Warner”)
Exhibit 1020	US-CERT. <i>Banking Securely Online</i> . (2008). (“US-CERT”)
Exhibit 1021	Pope, M.B. et al. <i>The Domain Name System—Past, Present, and Future</i> . Communications of the Association for Information Systems: Vol. 30, Article 21. (2012). Available at: http://aisel.aisnet.org/cais/vol30/iss1/21 (“Pope”)
Exhibit 1022	Microsoft Press. <i>Microsoft Exchange 2000 Server Resource Kit</i> . Pgs. 1-62. (2000). (“Microsoft Exchange”)
Exhibit 1023	<i>Processes: A Brief Introduction</i> . (2006). Available at: https://www.linfo.org/process.html (retrieved August 13, 2025). (“Linux Information Project”)

Exhibit 1024	Bradford Nichols, et. al. <i>Pthreads Programming: A POSIX Standard for Better Multiprocessing</i> . O'Reilly & Associates, Inc. (1996) (“O’Reilly”)
Exhibit 1025	Devin Watson. <i>Linux Daemon Writing HOWTO</i> . (2004). Available at: https://homes.cs.aau.dk/~adavid/teaching/MTP-05/exercises/10/Linux-Daemon_Writing.pdf . Accessed via the Web Archive at https://web.archive.org/web/20120217144006/https://www.netzmafia.de/skripten/unix/linux-daemon-howto.html (“Watson”)
Exhibit 1026	U.S. Patent No. 6,845,505 to Adunuthula et al. (“Adunuthula”)
Exhibit 1027	U.S. Patent No. 8,230,399 to Vertes (“Vertes”)
Exhibit 1028	U.S. Patent Application Publication No. 2014/0052683 to Kirkham et al. (“Kirkham”)
Exhibit 1029	IBM Software Thought Leadership Whitepaper. <i>Native, web or hybrid mobile-app development</i> . (2012). Available at: https://public.dhe.ibm.com/software/sk/websphere/pdf/Native_Web_or_Hybrid_Mobile_Development.pdf (“IBM”)
Exhibit 1030	Oracle Corporation. <i>DNS Basics</i> . Solaris Naming Administration Guide. (2010). Available at: https://docs.oracle.com/cd/E19455-01/806-1387/6jam692f2/index.html (retrieved August 20, 2025). (“Oracle – DNS”)
Exhibit 1031	Martin Robillard. <i>What Makes APIs Hard to Learn? Answers from Developers</i> . IEEE Software. (November/December 2009). (“Robillard”)
Exhibit 1032	Larry L.P. and Bruce S.D. <i>In The Morgan Kaufmann Series in Networking, Computer Networks</i> . 5 th Edition. (2012). Pgs. 1-69. (“Computer Networks”)
Exhibit 1033	Alok Sinha. <i>Client Server Computing Time Shared Computing</i> . Communications of the ACM. Vol. 35, No. 7. (1992). (“Sinha”)
Exhibit 1034	Paul Clark. <i>Introduction to Linux on IBM Power Systems</i> . IBM. (2021). Available at: https://developer.ibm.com/articles/intro-to-lop/ (retrieved August 18, 2025). (“Clarke”)
Exhibit 1035	Stephen G.K. <i>Programing in C</i> . 3 rd Edition. Sams Publishing. (2005). (“Kochan”)
Exhibit 1036	The Bell System Technical Journal. Vol. 57, No. 6, Part 2. Pgs. 1897-1094, 2807-2101. (“The Bell System Technical Journal”)
Exhibit 1037	<i>Sockets Tutorial</i> . (2007). Available at: https://web.archive.org/web/20070120054454/https://www.linuxh

	owtos.org/C_C++/socket.htm (retrieved August 19, 2025). (“ <i>linuxhowtos</i> ”)
Exhibit 1038	U.S. Patent Application Publication No. 2012/0317363 to Uola et al. (“ <i>Uola</i> ”)
Exhibit 1039	Scott M.L. <i>Interprocess Communication in Unix and Windows NT</i> . Department of Computer Science, Brown University. (1997). Available at: https://cs.brown.edu/people/slewando/files/IPCWinNTUNIX.pdf (“ <i>Lewandowski</i> ”)
Exhibit 1040	Oracle Corporation. <i>Appendix A Union Domain Sockets</i> . Network Interface Guide. (2010). Available at: https://docs.oracle.com/cd/E19455-01/806-1017/6jab5di3d/index.html#:~:text=UNIX%20domain%20sockets%20communicate%20only,properties%20visible%20to%20a%20user (retrieved August 20, 2025) (“ <i>Oracle – UNIX Domain Sockets</i> ”)
Exhibit 1041	NoMachine. <i>NoMachine 4 Release Announcement</i> . (September 25, 2013). Available at: https://kb.nomachine.com/SU10K00101?s=nx+protocol (retrieved August 20, 2025)
Exhibit 1042	NoMachine. <i>Using NX Client on a Linux desktop</i> . (February 5, 2012). Available at: https://web.archive.org/web/20120205022257/http://www.nomachine.com/screenshot/linux-client.php (retrieved August 20, 2025)
Exhibit 1043	NoMachine. <i>Article #AR10B00012</i> . (January 12, 2010). Available at: https://web.archive.org/web/20120114040432/http://www.nomachine.com/ar/view.php?ar_id=AR10B00012 (retrieved August 20, 2025)
Exhibit 1044	U.S. Patent Application Publication No. 2012/0023207 to Palchenko (“ <i>Palchenko</i> ”)
Exhibit 1045	Richard Peterson. <i>Linux: The Complete Reference Fourth Edition</i> . The McGraw-Hill Companies. (2001).
Exhibit 1046	U.S. Patent Application Publication No. 2007/0136579 to Levy et al. (“ <i>Levy</i> ”)
Exhibit 1047	Conny Franke et al. “Isolated Web Browsing Enforcing OS-like Isolation for Browsers.” December 16, 2005. https://www.researchgate.net/publication/277288272_Isolated_W

	eb_Browsing_Enforcing_OS-like_Isolation_for_Browsers (“ <i>Franke</i> ”)
Exhibit 1048	U.S. Patent Application Publication No. 2010/0218106 to Chen et al. (“ <i>Chen</i> ”)
Exhibit 1049	Mohsen Zohrevandi et al. “Auto-FBI: A User-friendly Approach for Secure Access to Sensitive Content on the Web.” ACSAC ’13. December 9-13, 2013. New Orleans, LA. (“ <i>Auto-FBI</i> ”)
Exhibit 1050	U.S. Patent No. 9,509,783 to Hayden (“ <i>Hayden</i> ”)
Exhibit 1051	U.S. Patent No. 9,015,226 to Sullivan et al. (“ <i>Sullivan</i> ”)
Exhibit 1052	U.S. Patent Application Publication No. 2005/0283767 to Blaku (“ <i>Blaku</i> ”)

CERTIFICATION OF WORD COUNT

The undersigned certifies pursuant to 37 C.F.R. §42.24 that the foregoing Petition for *Inter Partes* Review, excluding any table of contents, mandatory notices under 37 C.F.R. §42.8, certificates of service or word count, or appendix of exhibits, contains 13,965 words according to the word-processing program used to prepare this document (Microsoft Word).

Dated: September 12, 2025

Respectfully submitted,

By: /s/ Paul R. Hart
Paul R. Hart, Reg. No. 59,646

COUNSEL FOR PETITIONER

**CERTIFICATE OF SERVICE ON PATENT OWNER
UNDER 37 C.F.R. § 42.105**

Pursuant to 37 C.F.R. §§ 42.6(e) and 42.105(b), the undersigned certifies that on September 12, 2025, a complete and entire copy of this Petition for *Inter Partes* Review and Exhibits were provided via Federal Express to the Patent Owner by serving the correspondence address of record for the '721 Patent:

POLSINELLI PC
PO Box 140310
Kansas City, MO 64114-0310

Further, a courtesy copy of this Petition for *Inter Partes* Review was sent via email to Patent Owner's litigation counsel:

Jay P. Kesan (jay.kesan@kkd-law.com)
Cecil E. Key (cecil.key@kkd-law.com)
Alfonso Garcia Chan (achan@cahill.com)
Britton F. Davis (bdavis@cahill.com)
Christopher C. Campbell (ccampbell@cahill.com)
Halima Shukri Ndai (hndai@cahill.com)

Respectfully submitted,

By: /s/ Paul R. Hart
Paul R. Hart, Reg. No. 59,646

COUNSEL FOR PETITIONER