Filed on behalf of: Blue Coat Systems LLC
By: Michael T. Rosato (mrosato@wsgr.com)
Andrew S. Brown (asbrown@wsgr.com)
WILSON SONSINI GOODRICH & ROSATI
701 Fifth Avenue, Suite 5100
Seattle, WA 98104-7036

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

BLUE COAT SYSTEMS LLC,
Petitioner,

v.

FINJAN, INC.,
Patent Owner.

Patent No. 9,219,755
Case No. IPR2017-00997

PETITION FOR INTER PARTES REVIEW
OF U.S. PATENT NO. 9,219,755

TABLE OF CONTENTS

I. INTRODUCTION

Petitioner Blue Coat Systems LLC ("Blue Coat") hereby requests *inter partes* review of United States Patent No. 9,219,755 to Touboul ("the '755 patent," EX1001), which issued on December 22, 2015, and is currently assigned to Finjan, Inc. ("Patent Owner"). This Petition demonstrates by a preponderance of the evidence that claims 1-8 of the '755 patent are unpatentable for obviousness under 35 U.S.C. § 103(a).[1] Thus, claims 1-8 of the '755 patent should be held unpatentable and canceled.

The challenged claims generally recite systems for protecting one or more personal computers or other network accessible devices or processes from undesirable or otherwise malicious operations from a downloaded executable application program, referred to as a "downloadable." Each independent claim recites a processor and memory storing elements including instructions associated with the elements, the elements including: (1) a downloadable engine; (2) certain operating system probes; (3) a runtime environment monitor for comparing messages relating to operating system and extension calls against a security policy; and (4) a response engine for allowing or blocking calls based on whether they violate a security policy.

[1] Citations to 35 U.S.C. §§ 102 and 103 are to the pre-AIA statute.

Prior to the alleged invention, however, the increasing threat of malware had already led to the development of new protection systems for reviewing operating system ("OS") calls to control access to system objects. Such systems were known in the art prior to the alleged invention of the '755 patent.

A. The '755 Patent

The '755 patent is entitled "MALICIOUS MOBILE CODE RUNTIME MONITORING SYSTEM AND METHODS." Generally, the '755 patent is directed to systems for reviewing OS calls issued by a downloadable including probes associated with requests to various OS components, comparing the various requests to a security policy, and either allowing or blocking the requests based on the comparison. *See, e.g.*, EX1001, claims 1 and 3; EX1002 ¶¶15-20.

The '755 patent has a total of 8 claims, including two independent claims—claims 1 and 3. Both claims 1 and 3 are directed to a system for reviewing an OS call issued by a downloadable. Each independent claim recites closely related subject matter, including a processor and memory, an OS probe, a runtime environment monitor, a response engine, a downloadable engine, a request broker, a file system probe, and a network system probe. The systems use the probes to detect behavior of a downloadable, compare that behavior to security policies using the runtime environment monitor, and uses the response engine to block calls that violate the security policy while allowing them otherwise. EX1002, ¶16.

For example, claim 3 reads as follows:

> 3. A system for reviewing an operating system call issued by a downloadable, comprising:
>
> at least one processor for accessing elements stored in at least one memory associated with the at least one processor and for executing instructions associated with the elements, the elements including:
>
> a downloadable engine for intercepting a request message being issued by a downloadable to an operating system, wherein the request message includes an extension call;
>
> a request broker for receiving a notification message from the downloadable engine regarding the extension call;
>
> a file system probe and a network system probe each being associated with an operating system function for receiving the request message from the downloadable engine, intercepting an operating system call being issued by the downloadable to an operating system and associated with the operating system function and providing an event message regarding the operating system call;
>
> a runtime environment monitor for receiving the notification message and the event message and comparing the extension call and the operating system call against a predetermined security policy before allowing the operating system to process the extension call and the operating system call; and
>
> a response engine for receiving a violation message from

> the runtime environment monitor when one of the extension call and the operating system call violate one or more rules of the predetermined security policy and blocking extension calls and operating system calls that are forbidden according to the predetermined security policy, and for allowing extension calls and operating system calls that are permitted according to the predetermined security policy.

Claim 1 recites the same elements as claim 3, and adds certain additional elements relating to an OS probe; interactions between the probe, the runtime environment monitor, and the response engine; and an event router. These additional elements are discussed in the grounds below.

The challenged dependent claims of the '755 patent—2 and 4-8—each depend from claim 1 or 3. Claims 2 and 4 depend from claims 1 and 3, respectively, and further recite that the downloadable is one of a Java component, an ActiveX control, executable code, or interpretable code. Claims 5-8 depend from claim 3, and further actions that are taken by the response engine—in claim 5 the response engine stores comparison results in an event log, in claim 6 the response engine informs the user, in claim 7 the response engine stores information in a database, and in claim 8 the response engine discards the downloadables when it violates the security policy. EX1002, ¶17.

To understand the claims of the '755 patent, one must look into its tangled chain of priority documents. The specification of the '755 patent lacks support for

its claims; in particular, nowhere does the specification of the '755 patent itself disclose an OS probe, a runtime environment monitor, a response engine, or various other elements of the '755 patent claims. EX1002, ¶18. The earliest priority document that mentions such elements is U.S. Patent No. 6,167,520 ("the '520 patent," EX1008), which discusses a system with OS probes, a runtime environment monitor, and a response engine. *See, e.g.,* EX1008 at 2:1-23.

The '755 patent itself is now expired. The application that led to the '520 patent was filed on January 29, 1997, making it the earliest-filed non-provisional application to which the '755 patent claims priority. The '755 patent received no patent term adjustment, and accordingly expired on January 29, 2017.

As far as priority is concerned, the '755 patent is not entitled to any priority date prior to January, 29, 1997. The '755 patent also claims priority to U.S. Provisional Application No. 60/030,639 ("the '639 provisional," EX1010), filed November 8, 1996. The '639 provisional, however, also fails to provide support for the claims of the '755 patent. EX1002, ¶19. For example, nowhere does the '639 provisional disclose an OS probe, a response engine, or various other elements recited in independent claims 1 and 3 of the '755 patent. *Id.*

Additionally, at least two of the patents in the priority chain of the '755 patent fail to make a specific reference to the '639 provisional as required by 35 U.S.C. § 120. That is, U.S. Patent Nos. 7,613,926 and 7,058,822 both lack a

specific reference to the ’639 provisional. These patents are necessary links in the priority chain. Notably, the application that led to U.S. Patent No. 7,613,926 was the only application pending in the chain from June 6, 2006 to May 26, 2009; the application that led to U.S. Patent No. 7,058,822 was the only application pending in the chain from October 12, 2004 to March 7, 2006; and the two applications were co-pending alone in the chain from March 7, 2006 to June 6, 2006. *See* EX1001 at [59]; EX1004 at 0051-52, 1463-64. Because of this break in the priority chain, the ’755 patent cannot rightfully claim priority benefit to the ’639 provisional. *See Encyclopaedia Britannica, Inc. v. Alpine Elecs. of Am.*, 609 F.3d 1345 (Fed. Cir. 2010); *see also Medtronic CoreValve, LLC v. Edwards Lifesciences Corp*., 741 F. 3d 1359 (Fed. Cir. 2014).

Accordingly, the priority date of the ’755 patent is no earlier than January 29, 1997, the filing date of the ’520 patent.

B. The Prosecution History and Related Cases

Several references used in this Petition—Jaeger, TBAV, and Arnold—were disclosed among several Information Disclosure Sheets, which collectively disclosed over 1000 different references. *See* EX1004 at 1518, 1527, 1667. None of these references, however, were ever used by the examiner in a rejection nor were they ever cited or mentioned by the examiner in any other context. The CORBA reference was not disclosed to the examiner during prosecution.

Though the prosecution of the application that led to the '755 patent is relatively unenlightening, there has been activity in the prosecution of applications to which the '755 patent claims priority that is relevant to this IPR.

In particular, U.S. Patent No. 6,480,962 ("the '962 patent," EX1013) is one of several patents that issued from parent applications of the '755 patent, with claims substantially similar to the '755 patent claims, that had all claims subsequently canceled in reexamination. While the '755 patent was pending, the '962 patent was subject to an *inter partes* reexamination (Control No. 95/001,836) challenging claims 1-55 of the '962 patent.[2] The examiner rejected each claim on multiple art grounds, including grounds of anticipation and obviousness over Jaeger similar to those presented below. On February 29, 2016, after the issuance of the '755 patent, the Board issued a decision affirming the examiner's rejection

[2] During prosecution of the '755 patent, the examiner rejected all claims for nonstatutory double patenting, because the claims are not patentably distinct from claims 29-32 and 50-51 of the '962 patent. EX1004 at 1488-89. Patent Owner obviated the rejection with a terminal disclaimer. *Id*. at 1609.

of all[3] claims over Jaeger alone or in combination with other references. *See* Ex1016 at 2, 3, 9, 11-13. A reexamination certificate issued on January 12, 2017, canceling all claims.

Certain claims of the '962 patent that the Board found invalid over the Jaeger reference are substantially similar to claims of the '755 patent challenged here. In the reexamination decision, the Board held that Jaeger anticipated claims 29, 32 and 50-51, and rendered obvious claims 30 and 31 of the '962 patent (EX1016 at 3-6, 9-11, 12). Claims 29-32 depend from claim 22 which reads:

> 22. A system for determining whether a Downloadable, which is received by a Downloadable engine, is suspicious, comprising:
> means for monitoring substantially in parallel a plurality of subsystems of the operating system during runtime for an event caused from a request made by a Downloadable;
> means for interrupting processing of the request;
> means for comparing information pertaining to the Downloadable against a predetermined security policy; and
> means for performing a predetermined responsive action based on the comparison.

[3] Claims 1, 5, 6, 12, 15, 21, 33, 37, 38, 45, 52, and 55 were not included in the Board's decision, because the Federal Circuit had already affirmed their invalidity in *Finjan, Inc. v. Symantec Corp.*, No. 2013-1682 (Fed. Cir. 2014) (rule 36).

The Board construed the "means" recited in this claim in a manner corresponding to the elements recited in the '755 patent claims; for example, the Board construed the recited "means for monitoring" as including "class extensions, probes, and equivalent functioning software between an application and an operating system," and concluded that Jaeger disclosed this element. EX1016 at 4. Claims 29-32 add additional elements corresponding to claims 5-8 of the '755 patent. Claims 50 and 51 of the '962 patent are also substantially similar to claims 1-4 of the '755 patent.

Claims 1 and 3 of the '755 patent recite systems with substantially similar elements to the unpatentable claims of the '962 patent, including probes, a runtime environment monitor, and a response engine. Claims 1-8 of the '755 patent add some additional details, but as the Examiner correctly found (*see* EX1004 at 1488-89), none of these claims is patentably distinct from the unpatentable claims (*e.g.*, claims 50 and 51) of the '962 patent. As shown below in Grounds 1-3, the Board was correct in finding Jaeger to anticipate each of the above-recited elements of the '962 patent, and each of the elements added in the challenged claims is disclosed by Jaeger or obvious in view of Jaeger and the other prior art references described herein. The Board was correct in finding Jaeger to anticipate each of the above-recited elements of the '962 patent. As shown in the independent analysis below, Jaeger also renders unpatentable numerous claims of the '755 patent, as the reference teaches or suggests nearly all aspects of the challenged claims.

C. The Scope and Content of the Prior Art

As explained in detail in the corresponding Declaration of Azer Bestavros, Ph.D. (EX1002) and as addressed in further detail below (Section VI), a person of ordinary skill in the art at the relevant time would have considered the challenged claims obvious.

As described by Dr. Bestavros, in the early- to mid-1990s, the internet was becoming increasingly popular. At the same time, it became increasingly apparent that large computer networks such as the internet posed a danger. Files distributed over such networks could contain malicious or otherwise dangerous code. This led to a proliferation of new software intended to combat this danger. *See, e.g.*, EX1012 at Table 2 (comparing an antivirus product to "four other shareware and commercial virus detection products"). Accordingly, by 1996, the market for antivirus protection was a crowded one, and the subject of a great deal of continuing research and development. EX1002, ¶40.

One strategy for virus protection that was common at the time was to enforce a system of access rights according to a security policy. EX1002, ¶41. Untrusted programs from the internet were potentially dangerous, so programs were written to monitor their behavior. This monitoring could include determining whether a program attempted an operation that exceeded its access rights, and then taking action to prevent such potentially malicious software from doing harm.

The claims of the '755 patent are directed to systems that monitor downloadables by reviewing their access requests and enforcing security policies to allow or deny such requests. But as demonstrated in the prior art references discussed below, systems to perform such methods were already known to individuals in the field as of 1996. EX1002, ¶¶42-43.

i. Jaeger

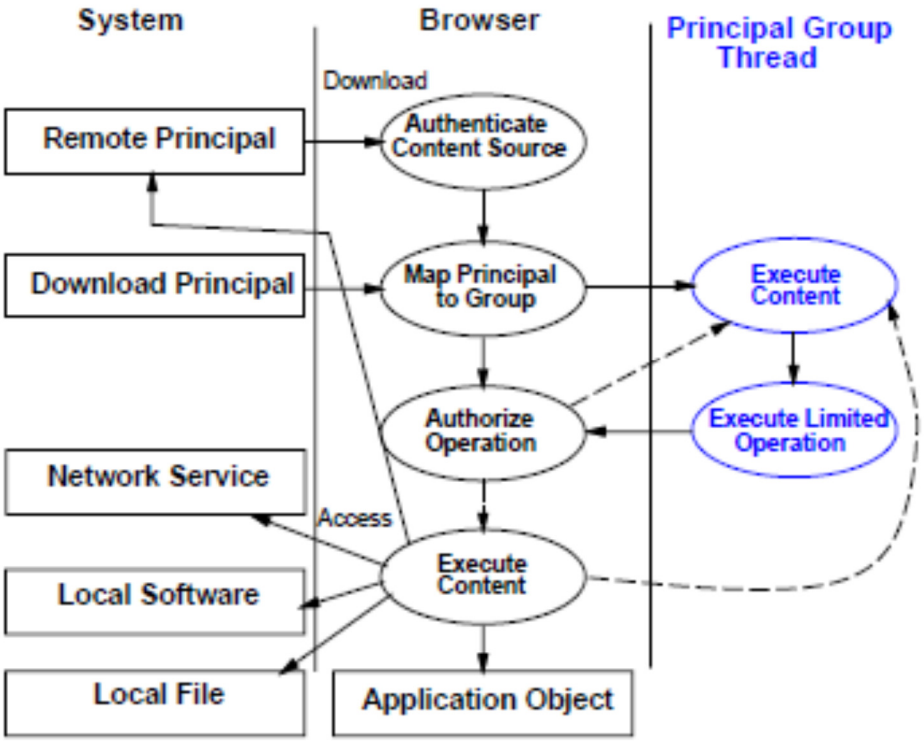
"Building Systems that Flexibly Control Downloadable Executable Content," by Jaeger *et al.* ("Jaeger," EX1005) was published in the Proceedings of the Sixth USENIX UNIX Security Symposium in July 1996. *See* EX1005 at 0001. Jaeger was distributed to attendees of the symposium in July 1996, as well as being made available online. *See* EX1005; see also EX1002, ¶¶44-48.[4] Accordingly, Jaeger is prior art under 35 U.S.C. § 102(a) because it was published prior to the January 29, 1997 effective filing date of the '755 patent.

Jaeger discloses an architecture for "flexible control" of downloaded executable content. EX1005 at 0002. The architecture provides that "downloaded executable content is properly controlled," thereby preventing a scenario in which

[4] To the extent Patent Owner would object to this evidence, all of these documents, including the cover to EX1005, qualify as ancient documents for purposes of the hearsay exception in Fed. R. Evid. 803(16). *See also* EX1002 at ¶¶44-48.

"a malicious remote principal may obtain unauthorized access to the downloading principal's resources." *Id*. In Jaeger's system, "operations are implemented by a call to authorize which checks the content access rights prior to calling the actual operation." *Id*. at 0016. "Since these commands are only available in the browser and are protected by the authorization operation, only authorized accesses to system objects are possible." *Id*. at 0016. Jaeger describes using "application specific interpreters [that] control access to application objects and determine the access rights of content within their domain." *Id*. at Fig. 3. By controlling access and determining access rights, Jaeger's system enables the protection of system resources and the detection of potentially malicious code. EX1002, ¶49.

The '755 patent recites numerous calls and messages between various system elements. However, Fig. 5 of Jaeger illustrates the same type of calls and messages for access control using similar system elements, as shown below:



In the above figure, a downloadable ("Download," "Content") is executed ("Execute Content") and executes a limited operation requiring authorization for access to an OS function ("Network Service," "Local File," etc.). A message corresponding to a call to the appropriate OS subsystem is intercepted by software in the "Browser" (probe). The message is forwarded to a runtime environment monitor ("Authorize Operation") to compare against rules of a security policy. The result is sent to a response engine ("Execute Content") that executes the requested OS call if it is authorized by the security policy.

As discussed in the Grounds below, Jaeger discloses nearly every element of the independent claims 1 and 3. For example, Jaeger discloses a system comprising a processor and memory, and which executes software with elements including

probes for reviewing OS calls of a downloadable, a downloadable engine, a runtime environment monitor, and a response engine.

ii. CORBA

"The Common Object Request Broker: Architecture and Specification" ("CORBA," EX1011) was published as a joint publication of the Object Management Group and X/Open in 1993. CORBA states on its face that it was published on December 29, 1993 as revision 1.2, OMG Document Number 93.xx.yy.[5] The Object Management Group website maintains a copy of CORBA, identified as document number 1993/93-12-43, for download in PDF or PostScript formats at http://www.omg.org/spec/CORBA/1.2/. From 1993 on, CORBA was widely distributed, including over the internet. EX1002, ¶50. For example, U.S. Patent No. 5,857,102 to McChesney *et al.* (EX1015), filed on March 14, 1995, states that "The Common Object Request Broker: Architecture and Specification, Rev. 1.2, OMG TC Document Number 93-12-43 [is] available by anonymous ftp to omg.org." EX1015 at 4:27-31. Accordingly, CORBA is prior art under 35 U.S.C. § 102(a) and § 102(b) because it was published more than a year prior to the January 29, 1997 effective filing date of the '755 patent.

[5] To the extent Patent Owner would object to this evidence, all of these documents qualify as ancient documents for purposes of the hearsay exception in Fed. R. Evid. 803(16).

CORBA describes an architecture for distributed computing that uses an Object Request Broker ("ORB") to provide interoperability between applications on different machines. EX1011 at 15. The ORB receives a request from a client and forwards the request to an object implementation. *Id*. at 30, Fig. 2. The ORB thus provides an interface for requesting execution of a code object that is "completely independent of where the object is located, what programming language it is implemented in, or any other aspect which is not reflected in the object's interface." *Id*. at 30. In other words, the ORB acts as a middleman between a client and an object requested for execution by the client. EX1002, ¶51.

As discussed in Ground 1 below, to the extent Jaeger does not explicitly disclose the request broker recited in claims 1 and 3, the Common Object Request Broker Architecture discloses this element.

iii. TBAV

The user manual for the ThunderBYTE antivirus scanner ("TBAV," EX1006) was published in 1995. The ThunderBYTE antivirus scanner was a popular antivirus program in the mid-1990s, and TBAV was distributed as a printed publication along with the ThunderBYTE antivirus scanner software. *See, e.g.*, EX1017 ("Qualified organizations, editors, reviewers or PC users groups who prefer to receive a disk-based evaluation version of our software complete with a printed manual, please e-mail us or contact one of our local distributors," archived

Dec. 4, 1996); *see also* EX1018 (showing page offering an evaluation version of the software along with the manual was visited nearly 1,000 times per day in April of 1996); EX1002, ¶52.[6]

Distribution of TBAV to the public is also demonstrated by U.S. Patent No. 5,696,822 ("the '822 patent," EX1012), filed September 28, 1995. The '822 patent lists the TBAV manual on its face and refers to the ThunderBYTE scanner in a list of "shareware and commercial virus detection products." EX1012 at 12:20-50, Table 2 col. 5; *see also* EX1002, ¶53.

Accordingly, TBAV is prior art under 35 U.S.C. § 102(a) and § 102(b) because it was published in 1995, more than a year prior to the January 29, 1997 effective filing date of the '755 patent.

TBAV describes a software program that protects computer systems against both known and unknown viruses. The TBAV software includes several utilities for virus protection, including the TbLog utility. *See* EX1006 at 1-5, 139. The TbLog utility generates log files in response to various alert messages from TBAV's virus detection utilities, including recording when a virus is detected.

[6] To the extent Patent Owner would object to this evidence, all of these documents qualify as ancient documents for purposes of the hearsay exception in Fed. R. Evid. 803(16).

EX1006 at 139. The logs generated include information such as a time stamp, the machine where the event occurred, and a message describing what occurred and what files were involved. *Id*. The TbLog utility is described as "primarily for network users," and allows a supervisor to "keep track of what's going on." *Id*.; EX1002, ¶55.

As discussed in Ground 2 below, to the extent that Jaeger does not explicitly disclose storing of comparison results related to malware in a log file, TBAV teaches this.

iv. Arnold

U.S. Patent No. 5,440,723 ("Arnold," EX1007) was filed January 19, 1993 and issued August 8, 1995. Accordingly, Arnold is prior art under 35 U.S.C. § 102(b) as it was patented more than 1 year before the January 29, 1997 effective filing date of the '755 patent.

Arnold discloses a system for detecting "anomalous behavior that may indicate the presence of an undesirable software entity such as a computer virus, worm, or Trojan Horse." EX1007 at Abstract. When such an entity is detected, Arnold's system extracts an identifying signature and adds the signature to a database so it may be used by a scanner to prevent subsequent infections of the system and other computers connected to the system in a network. *Id*.; *see also* EX1002, ¶57.

As discussed in Ground 3, to extent that Jaeger does not explicitly disclose them, Arnold teaches the elements of several dependent claims of the '755 patent, including responsive actions such as informing the user of dangerous program activity, storing malware-related information in a database, and deleting suspicious Downloadables.

D. The Level of Skill in the Art

As Dr. Bestavros explains, a person of ordinary skill in the relevant field prior to January 29, 1997 would include someone who had, through education or practical experience, the equivalent of a bachelor's degree in computer science or a related field and at least an additional three to four years of work experience in the field of computer security. EX1002, ¶¶31-34. Such a person would have been aware of and would have been working with computer security trends from the early- to mid-1990s, including trends toward access rights control of downloaded executable code using security policies. *Id.*; *see also* §I.C; *Custom Accessories, Inc. v. Jeffrey-Allan Indus. Inc.*, 807 F.2d 955, 962 (Fed. Cir. 1986) ("The person of ordinary skill is a hypothetical person who is presumed to be aware of all the pertinent prior art.").

II. GROUNDS FOR STANDING

Petitioner certifies that, under 37 C.F.R. § 42.104(a), the '755 patent is available for inter partes review, and Petitioner is not barred or estopped from

requesting inter partes review of the '755 patent on the grounds identified.

III. MANDATORY NOTICES UNDER 37 C.F.R. § 42.8

A. Real Party in Interest (§ 42.8(b)(1))

Blue Coat Systems LLC is the real party in interest. Blue Coat's parent company, Symantec Corp., is also a real party in interest.

B. Related Matters (§ 42.8(b)(2))

The '755 patent has been asserted in *Finjan, Inc. v. Blue Coat Systems, LLC*, No. 5:15-cv-03295-BLF (N.D. Cal.) ("*Finjan v. Blue Coat II*"), in an amended complaint filed on March 1, 2016.

Blue Coat is aware of the following other district court litigation in which the '755 patent has been asserted: *Finjan, Inc. v. ESET, LLC*, *et al.*. No. 3:16-cv-03731-JD (N.D. Cal.).

Blue Coat is not aware of any other patent office matters involving the '755 patent.

Patent Owner has followed a prosecution strategy of seeking a large number of patents with a large number of very similar claims. It has also pursued a litigation strategy of asserting overlapping but nonidentical groups of those claims against many different targets all at different times. Thus, proceedings involving Patent Owner's patents—both in district courts and at the Patent Office—have proliferated. Accordingly, Blue Coat has only listed matters directly involving the

'755 patent and not matters only involving related patents.

C. Lead and Back-Up Counsel (§ 42.8(b)(3))

Lead Counsel: Michael T. Rosato (Reg. No. 52,182)

Back-Up Counsel: Andrew S. Brown (Reg. No. 74,177)

D. Service Information (§ 42.8(b)(4))

Blue Coat consents to electronic service.

Email: mrosato@wsgr.com; asbrown@wsgr.com

Post: WILSON SONSINI GOODRICH & ROSATI

701 Fifth Avenue Suite 5100

Seattle, WA 98104-7036

Tel.: 206-883-2529, Fax: 206-883-2699

IV. STATEMENT OF THE PRECISE RELIEF REQUESTED FOR EACH CLAIM CHALLENGED

Blue Coat requests review and cancellation of claims 1-8 of the '755 patent under 35 U.S.C. § 311 and AIA § 6. The grounds for relief are as follows:

Ground	Claims	Description
1	1-4	Obvious under § 103(a) over Jaeger and CORBA
2	5	Obvious under § 103(a) over Jaeger, CORBA and TBAV
3	6-8	Obvious under § 103(a) over Jaeger, CORBA, and Arnold

V. CLAIM CONSTRUCTION

In an *inter partes* review of an expired patent, claim terms are given their ordinary and accustomed meaning as would be understood by one of ordinary skill in the art because the expired claims are not subject to amendment. *See Panel Claw, Inc. v. Sunpower Corp.*, IPR2014-00386, Paper 7 at 7; *see also In re Rambus, Inc.*, 694 F.3d 42, 46 (Fed. Cir. 2012), *Phillips v. AWH Corp.*, 415 F.3d 1303, 1312-13 (Fed. Cir. 2005) (*en banc*). The Board routinely institutes *inter partes* review of expired patents using this standard. *See, e.g.*, *Intel Corp. v. Fuzzysharp Techs. Inc.*, IPR2014-00001, Paper 7 at 9; *Harmonix Music Sys. Inc., v. Princeton Dig. Image Corp.*, IPR2014-00155, Paper 11 at 6-7. Because the '755 patent has expired, claim terms are to be given their ordinary and accustomed meaning, consistent with how they would be understood by one of ordinary skill in the art, for purposes of this review. A few terms that warrant discussion are identified and discussed below.

A. "downloadable"

The term "downloadable," as used throughout the claims of the '755 patent, includes "an executable application program, which is downloaded from a source computer and run on a destination computer." This construction corresponds to the definition of the term "downloadable" in the U.S. Pat. No. 6,092,194 ("the '194 patent," EX1009), from which the '755 patent claims priority. EX1010 at 1:44-55;

see also EX1002 ¶36.

B. "extension call"

The meaning of the term "extension call," as recited in independent claims 1 and 3, was objected to by the examiner as unclear during prosecution. EX1004 at 1485. Patent Owner did not provide a construction, but did point to portions of the '962 patent as allegedly providing requisite support. EX1004 at 1601-04. These sections included Figs. 3 and 4 of the '962 patent, as well as parts of the specification. For example, the '962 patent states

> The security system 135a includes Java™ class extensions 304, wherein each extension 304 manages a respective one of the Java™ classes 302. When a new applet requests the service of a Java class 302, the corresponding Java™ class extension 304 interrupts the request and generates a message to notify the request broker 306 of the Downloadable's request.

EX1013 at 4:10-17. The '962 patent further states that the classes managed by the extensions are "conventional Java™ classes 302. Each of the Java™ classes 302 performs a particular service such as loading applets, managing the network, managing file access, etc." EX1013 at 3:52-63.

Accordingly, for the purposes of this Petition, an extension call includes at least "a request corresponding to a class of program objects." *See* EX1002, ¶38.

C. "downloadable engine"

Independent claims 1 and 3 both recite a downloadable engine. The '755

patent provides no support for such a structure. EX1002, ¶39. The '520 patent describes a "browser 245 [that] includes a Downloadable engine 250 for managing and executing received Downloadables." EX1008 at 3:29-30. The '520 patent gives the Java virtual machine and the ActiveX platform as examples of downloadable engines. *Id*. at 3:42-46, 5:17-19. Accordingly, for the purposes of this Petition, the term "downloadable engine" includes at least "software for managing and executing downloadables." *See* EX1002, ¶39.

VI. DETAILED EXPLANATION OF GROUNDS FOR UNPATENTABILITY

A. [Ground 1] Claims 1-4 are Obvious under 35 U.S.C. § 103(a) over Jaeger in view of CORBA

As described further below, claims 1-4 of the '755 patent are obvious over Jaeger in view of CORBA. EX1002, ¶¶58-138. *See* §I.C.

Jaeger discloses an architecture for "flexible control" of downloaded executable content. EX1005 at 0002. Jaeger describes analyzing downloadable content for suspicious or malicious aspects, stating the purpose of this architecture as ensuring that "downloaded executable content [i.e., a 'downloadable'] is properly controlled," because without such control, "a malicious remote principal may obtain unauthorized access to the downloading principal's resources." *Id*. In Jaeger's architecture, "operations are implemented by a call to authorize which checks the content access rights prior to calling the actual operation." *Id*. at 0016.

"Since these commands are only available in the browser and are protected by the authorization operation, only authorized accesses to system objects are possible." *Id*. In combination with the browser, Jaeger's system uses "application specific interpreters [that] control access to application objects and determine the access rights of content within their domain." *Id*. at Fig. 3. *See* EX1002, ¶49.

Jaeger discloses or renders obvious nearly every element of the independent claims 1 and 3. For example, Jaeger discloses a system comprising a processor and memory, and which executes software with elements including probes for reviewing OS calls of a downloadable, a downloadable engine, a runtime environment monitor, and a response engine. Though Jaeger does not expressly disclose a request broker, including a request broker would have been obvious in view of CORBA.

CORBA discloses request brokers, referred to as Object Request Brokers (ORBs) to "provide[] interoperability between applications on different machines in heterogeneous distributed environments and seamlessly interconnect[] multiple object systems." EX1011 at 15. It was a known use of request brokers such as those disclosed in CORBA to facilitate communications between system components in distributed architectures, such as that disclosed in Jaeger. EX1002, ¶¶51, 90.

The below discussion provides an element-by-element discussion of the

aspects of the claims with respect to the cited prior art.

i. Independent claims 1 and 3

The architecture described by Jaeger, in view of the request brokers of CORBA, would have rendered independent claims 1 and 3 of the '755 patent obvious.

Claims 1 and 3 are similar, with claim 1 comprising each element of claim 3, and adding a few additional claim elements. Each claim has the same preamble: "A system for reviewing an operating system call issued by a downloadable, comprising." To the extent the preambles are limiting, Jaeger discloses each of these elements, as shown in the following claim chart:

'755 Patent	**Jaeger and CORBA**
[1.pre, 3.pre] A system for reviewing an operating system call issued by a downloadable, comprising:	Jaeger discloses a system for flexible control of downloadables: "In this paper, we describe an architecture that flexibly controls the access rights of downloaded content." EX1005 at 0002. The system reviews access requests (*e.g.*, "system call") from downloadables: "Obviously, the access requests made directly by content must be controlled." *Id.* at 0005. Calls to invoke procedures and access objects are reviewed: *Id.* at 0015. Control is provided to objects handled by an operating

	system: "In general, access to the following types of objects needs to be controlled … Application Objects … File System … Applications … Processes … Environment … Network Services … URLs." *Id*. at 0005. *See also id*. at Title, 0003, 0004, 0005, 0006-07, 0016 ("the exec call"); EX1002, ¶¶60, 105-06.

Jaeger discloses a system for reviewing an OS call issued by a downloadable. EX1005 at 0002. "Downloaded Executable Content" is a downloadable. The system will "(1) authenticate the source of the content; (2) determine the least privilege access rights for the content given its source; and (3) enforce those access rights throughout the execution of the content." *Id*. at 0004.

Jaeger describes the use of a "conventional protection model," in which "principals (e.g., users, groups, services, etc.) execute processes that perform operations (e.g., read, write, etc.) on objects (e.g., files, devices, etc.)." EX1005 at 0003; EX1002, ¶60, 106. The system ensures compliance with the protection model by reviewing OS calls. *Id*.

In particular, Jaeger lists several types of OS objects for which "access … needs to be controlled." EX1005 at 0005. These objects include including applications, file system objects, processes, network services, and the environment of each process. *Id*. To access each of these object types, a call would be made to

an OS. *Id.*; EX1002, ¶60, 106. For example, Jaeger's system uses "I/O commands … to control access to system objects" and includes an "operating system [that] has an unmodified trusted computing base." EX1005 at 0006-07. An I/O command to access system objects is an OS call. EX1002, ¶60, 106. Accordingly, Jaeger discloses a system for reviewing an OS call issued by a downloadable.

Jaeger in view of CORBA discloses **elements [1.1] and [3.1]:**

'755 Patent	**Jaeger and CORBA**
[1.1, 3.1] at least one processor for accessing elements stored in at least one memory associated with the at least one processor and for executing instructions associated with the elements, the elements including:	Jaeger discloses its system as computer-based, thereby including hardware and software components. The architecture stores and executes downloadable content: "Application performance can be improved because the program (i.e., executable content) can be downloaded to the location of the data (e.g., for a query) or the location of the user (e.g., for an interface-driven task)." *Id.* at 0002. *Id.* at 0006 (discussing "…computing base"). Processes can read, write, and execute: "A process can read and write objects in its memory [and] can perform read, write, and execute operations." *Id.* at 0005. *See also id.* at 0005, 0010, 0019, Figs. 2, 5, 8 (showing pseudocode for execution on a processor) EX1002, ¶¶61-63, 107-08.

Claim elements [1.1] and [3.1] are identical, each reciting a processor for

accessing and executing elements stored in an associated memory. Jaeger's system includes these elements.

Jaeger's system includes a processor capable of accessing and executing instructions stored in memory. Jaeger discloses "an architecture that flexibly controls the access rights of downloaded content" (EX1005 at 0002) that employs an "operating system [with] an unmodified trusted computing base." EX1005 at 0006; *see also id*. at Fig. 2 ("Network Computing Environments"). As described by Dr. Bestavros, a person of ordinary skill in the art would understand that the "computing base" would include a processor and associated memory, for executing instructions. EX1002, ¶¶61, 107.

Jaeger confirms this interpretation, disclosing that access to several types of programming objects needs to be controlled, including "Application Objects," a "File System," and "Applications," each of which relates to "a process." EX 1005 at 0005; EX1002, ¶¶62, 107. This includes a "process [that] can read and write objects in its memory." *Id*. This "process," running on a processor, would access and execute commands stored in the "memory" described by Jaeger. EX1002, ¶63, 108.

Additionally, the use of processors and memory storing software elements for execution would have been obvious. Such elements form the basic building blocks of computer science, and a person of ordinary skill would have had a

reasonable expectation of success in using them. EX1002, ¶63, 108. Indeed, Jaeger states that Jaeger's "model is influenced most strongly by the access control models of Hydra," which is a "kernel of a multiprocessor system." EX1005 at 0010, 0019.

Accordingly, Jaeger's system includes the limitations as recited in claims 1 and 3.

Jaeger in view of CORBA discloses **element [1.2]:**

'755 Patent	**Jaeger and CORBA**
[1.2] an operating system probe associated with an operating system function for intercepting an operating system call being issued by a downloadable to an operating system and associated with the operating system function;	Jaeger discloses a system with OS probes, which it refers to as interpreters: "Trusted, application-independent interpreters (which we will refer to as browsers from here on) control access to system objects by the application-specific interpreters and downloaded content. An application-specific interpreter controls access to application objects and specifies the access rights of downloaded content within its limited domain." EX1005 at 0007. The probes authorize OS operations by intercepting OS calls of downloadables: "[T]he browser authenticates the content and sends the content to be run in the appropriate application-specific interpreter. … Any controlled operation run by the content is authorized by the appropriate interpreter and if authorization is granted then

	the operation is executed. Operations are executed in the interpreter trusted with the operation." *Id*. at 0008. *See also id*. at 0005, Figs. 3, 5; EX1002, ¶¶64-71.

Claim element [1.2] does not have a corresponding element in claim 3, although claim element [3.4], discussed below, is similar. Element [1.2] recites an OS probe that receives an OS call from a downloadable. Jaeger's system discloses this element.

As discussed in more detail below, Jaeger's interpreters are substantially similar to the probes described in the '520 patent, which the '755 patent relies on to support its claims. The '520 patent describes "operating system probes" as respectively receiving from the downloadable "instructions sent to the file system … the network management system … the process system … [and] the memory system." EX1008 at 4:10-23; *see also id*. at Fig. 3 (showing the OS probes situated between the OS and the remainder of the security system, similar to Jaeger's interpreters). Jaeger's interpreters perform substantially the same function in substantially the same way. EX1002, ¶¶65-67, 70.

Jaeger discloses interpreters that are OS probes as recited in claim 1. With regard to the probes, Jaeger discloses a system including "browsers" that act as "Trusted, application-independent interpreters" to "control access to system objects" by "application-specific interpreters" that correspond to specific

applications and functions called by "downloaded content." EX1005 at 0007; *see also id*. at Fig. 3. Jaeger's interpreters act as probes that intercept calls from downloaded content to an OS. EX1002, ¶65. For example, Jaeger describes that the interpreters review intercepted OS calls to authorize them: "[a]ny controlled operation run by the content is authorized by the appropriate interpreter." EX1005 at 0008. To review a call, the probe intercepts it and confirms that it conforms to a security policy. *See id*. at Fig. 5; EX1002, ¶67.

These interpreters intercept messages corresponding to an OS call from a downloadable. The messages include access requests to subsystems including "Application Objects," "File System," "Processes," "Environment," or "Network Services." EX1005 at 0005; EX1002, ¶66. Jaeger teaches that each of these subsystems must be monitored. *Id*. The subsystems for which the intercepted calls request access include OS functions (such as "File System," "Processes," and "Network Services," for example). EX1002, ¶66.

Furthermore, Jaeger's probes intercept calls "being issued" by a downloadable as the downloadable is being executed. As Jaeger describes,

> Content is executed as shown in Figure 5. … Any controlled operation run by the content is authorized by the appropriate interpreter and if authorization is granted then the operation is executed. Operations are executed in the interpreter trusted with the operation.

EX1005 at 0008. As Jaeger's interpreters must authorize requests to the OS by a downloadable before the request is executed, they act as probes that intercept an OS call being issued by a downloadable to an OS and associated with the OS function. EX1002, ¶67.

The protocol for content execution in Jaeger's system is shown in Fig. 5:

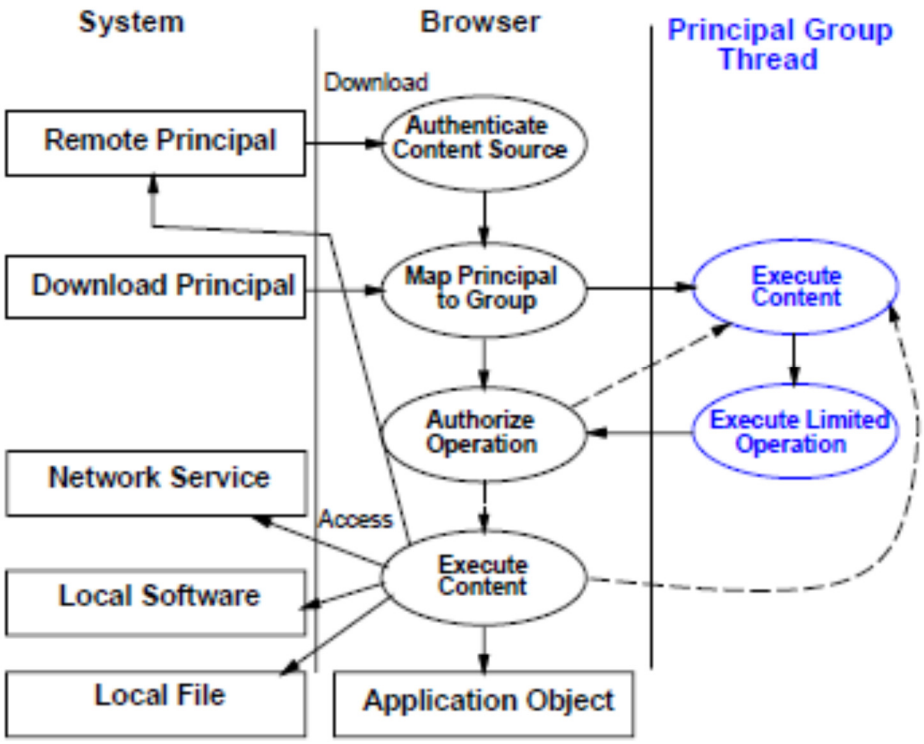


Figure 5: **Content Execution Protocol**: (1) Remote principal downloads content to browser; (2) Browser assigns content to principal group in application-specific interpreter (not shown); (3) content executes a limited operation which cause authorization in the browser; (4) if authorized the browser access system objects on behalf of content.

As discussed above (*see* §I.C.i.), this figure shows that when content is executed,

calls are intercepted by a probe (e.g., the browser, an application-independent interpreter), as indicated by the arrows leading from "Execute Content" to "Authorize Operation."

When a downloadable requests execution of an instruction by the OS, this would be a "call being issued by a downloadable to an operating system and associated with the operating system function." By monitoring the OS's subsystems for such requests, and intercepting those requests, Jaeger's interpreters act as OS probes intercepting the calls. EX1002, ¶68-69.

Accordingly, Jaeger in view of CORBA discloses claim element [1.2]. EX1002, ¶71.

Jaeger in view of CORBA discloses **element [1.3]:**

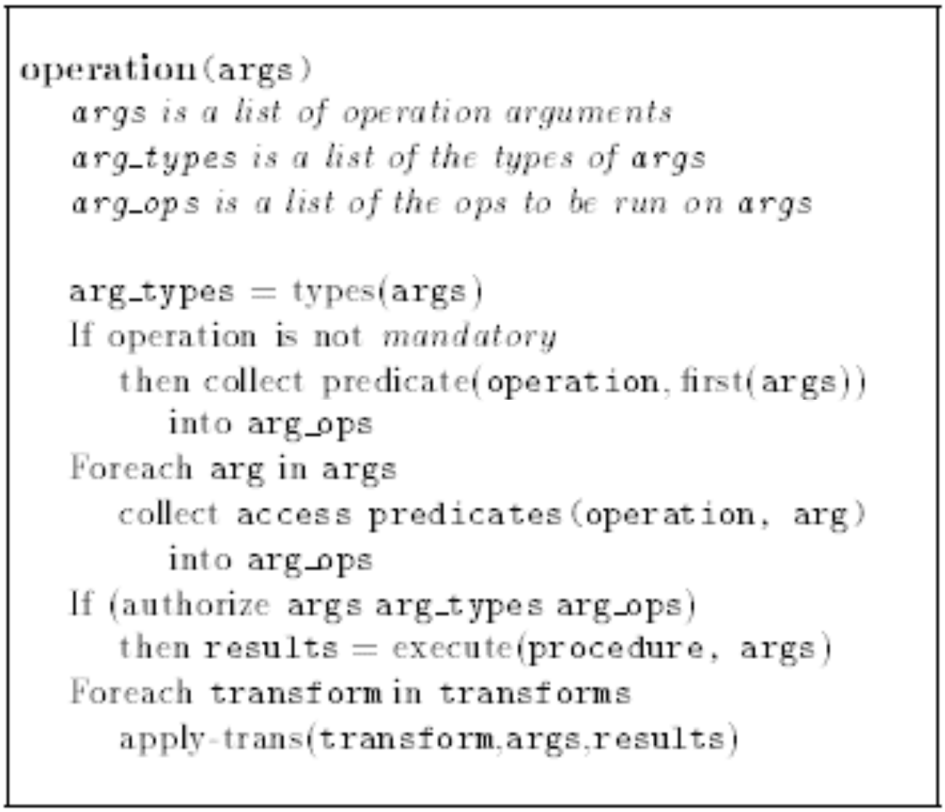
'755 Patent	**Jaeger and CORBA**
[1.3] a runtime environment monitor for comparing the operating system call against a predetermined security policy including multiple security rules to	Jaeger's access rights define a security policy: "The access rights of a remote principals content in interpreter t active at state s are the intersection of the rights of interpreter t and the rights of the principal group to which the remote principal is assigned in interpreter t at state s." EX1005 at 0012. The access rights determine what OS calls can be executed: "Content is executed as shown in Figure 5. … This content interpreter has access rights consistent with the identity of the remote principal. Any controlled operation run by the content

determine if execution of the operating system call violates one or more of the multiple security rules before allowing the operating system to process the operating system call and for forwarding a message to a response engine when the comparison by the runtime environment monitor indicates a violation of one or more of the multiple security rules;	is authorized by the appropriate interpreter and if authorization is granted then the operation is executed. Operations are executed in the interpreter trusted with the operation." *Id.* at 0008.
	
	The rules are predetermined: "[W]e want delegation to be implicit relative to some application rather than require the downloading principal to specify access rights at runtime." *Id.* at 0014.
	
	Fig. 9 shows pseudocode for a runtime environment monitor comparing calls to multiple security rules, and sends a message to the response engine:
	
	`authorize(args arg_types arg_ops)`
	`  For arg in args, arg_type in arg_types,`
	`  ops in arg_ops`
	`  Find object group for arg and arg_type`
	`  For op in ops`
	`  If domain rights grant op`
	`  AND no exception precludes op on arg`
	`  then continue`
	`  else return FALSE`
	`  return TRUE`
	
	See also id. at Figs. 5, 8, 9; EX1002, ¶¶72-76.

Claim element [1.3] does not have a corresponding element in claim 3,

although claim element [3.6], discussed below, is similar. Element [1.3] recites a runtime environment monitor that compares the call against a security policy comprising multiple security rules and sends a message to a response engine when there is a security rule violation. Jaeger's system discloses this element.

Jaeger discloses a runtime environment monitor (*e.g.*, as shown in the pseudocode of Fig. 9) that compares the OS call against a predetermined security policy including multiple security rules. The security policies of Jaeger are embodied in "access rights" for a downloadable, which are determined from the rights of the interpreter assigned to the downloadable and the group rights of the user ("rights of the principal group to which the remote principal is assigned"). EX1005 at 0012; EX1002, ¶72. These rights are specified in advance to speed up access rights determination at runtime. EX1005 at 0014; EX1002, ¶72.



```
operation(args)
   args is a list of operation arguments
   arg_types is a list of the types of args
   arg_ops is a list of the ops to be run on args

   arg_types = types(args)
   If operation is not mandatory
      then collect predicate(operation, first(args))
         into arg_ops
   Foreach arg in args
      collect access predicates(operation, arg)
         into arg_ops
   If (authorize args arg_types arg_ops)
      then results = execute(procedure, args)
   Foreach transform in transforms
      apply-trans(transform,args,results)
```

Jaeger provides pseudocode in Figs. 8 and 9 illustrating how a requested operation can be authorized and subsequently executed subject to that authorization. Fig. 8 (above) illustrates a process for handling an operation, and that process invokes the procedure in Fig. 9 (below) to determine whether the operations required by the request are authorized. EX1002, ¶73; *see also* §I.C.i. above (describing the "Authorize Content" step of Fig. 5.)

```
authorize(args arg_types arg_ops)
   For arg in args, arg_type in arg_types,
      ops in arg_ops
      Find object group for arg and arg_type
      For op in ops
         If domain rights grant op
            AND no exception precludes op on arg
            then continue
            else return FALSE
   return TRUE
```

As described by Dr. Bestavros, the "operation" procedure in Fig. 8 processes a request received from a downloadable by determining types ("arg_types") of the arguments ("args") and the predicate operations ("ops") required for the request, then invokes the "authorize" procedure of Fig. 9 using this information. EX1002, ¶74. The "authorize" procedure then determines whether the request violates any security rules, and returns its conclusion to the "operation" procedure, which executes the request if authorized, and blocks it otherwise. *Id.* Accordingly, as demonstrated by the "authorize" procedure, Jaeger comprises a runtime environment monitor that compares OS calls to multiple security rules. EX1002, ¶¶72-74.

Jaeger's system performs the comparison before allowing the OS to process the OS call. As shown in Fig. 8 of Jaeger, the "operation" function does not execute the procedure requested by the call until after evaluating whether it is

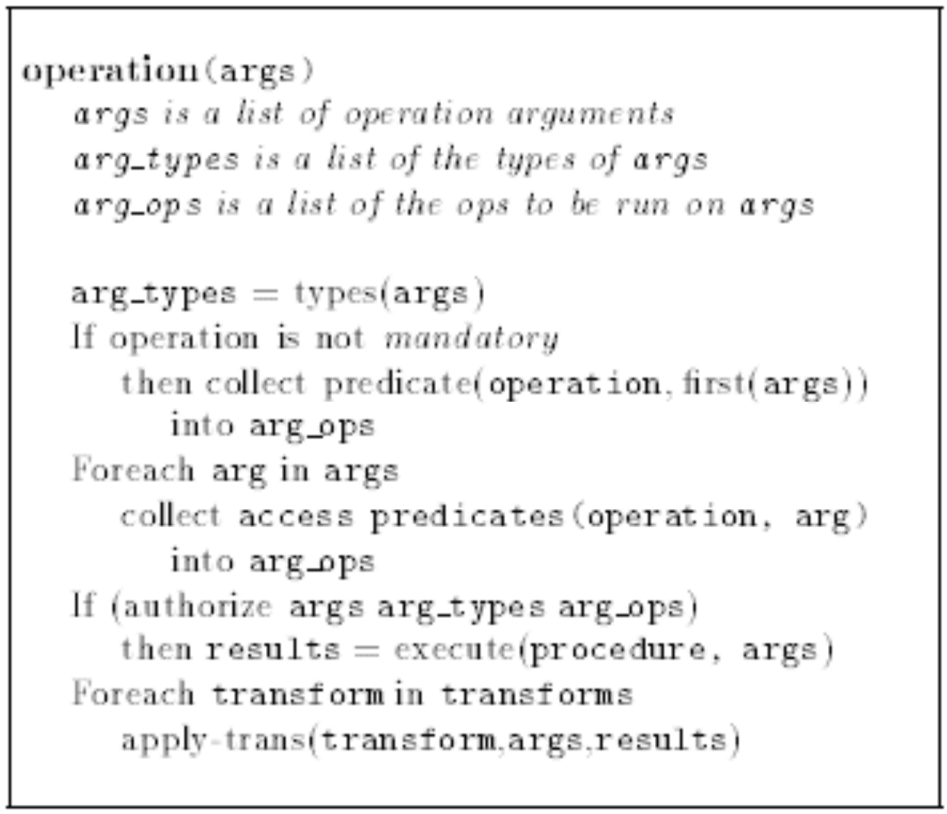
authorized. Indeed, the procedure is executed only if the comparison determines that it is authorized. Furthermore, it would also be obvious to a person of ordinary skill in the art that authorization must come before, not after execution, lest the system risk executing a malicious instruction. EX1002, ¶74.

Jaeger's runtime environment monitor forwards a message to a response engine when the comparison by the runtime environment monitor indicates a violation. As explained by Dr. Bestavros, the "authorize" function sends a message "FALSE" as its return value whenever it determines that a security rule has been violated, indicating that the operation is not authorized. EX1002, ¶75. Moreover, although not required by the claims of the '755 patent, it would have been obvious to add additional information to Jaeger's return message. For example, the message could include more information regarding what operations were unauthorized, or what rules were violated. *See* EX1002, ¶76. As explained by Dr. Bestavros, there would be various reasons to do this, including debugging, allowing users to modify access rights, and identifying attacks by malicious code. *Id*.; *see also* EX1014 at Abstract (describing logging of rule violations). Indeed, Jaeger's system already produces such information, so it would be straightforward to send it to the response engine as part of a message. EX1002, ¶76.

Accordingly, Jaeger in view of CORBA discloses claim element [1.3].

Jaeger in view of CORBA discloses **element [1.4]:**

’755 Patent	Jaeger and CORBA
[1.4] a response engine for compiling each rule violation indicated in the messages forwarded by the runtime environment monitor, for blocking execution of operating system calls that are forbidden according to the security policy when execution of the operating system calls would result in a violation of a predetermined combination of multiple security	The response engine (*e.g.*, Jaeger’s Fig. 8) executes authorized requests: “Once authorized, the browser can execute the procedure. EX1005 at 0015. Authorization is required for a request to be executed by the response engine: “Operations on application objects must be authorized to ensure that the principal is permitted to modify the object. Access to file system objects should be limited to prevent unauthorized access to private objects.” *Id*. at 0005. Fig. 8 shows pseudocode for a response engine that compiles a violation from the runtime environment monitor, then executes the requested procedure if and only it is authorized:

rules of the predetermined security policy and for allowing execution of operating system calls that are permitted according to the security policy;	 *See also id.* at 0015, Figs. 8, 9; EX1014 at Abstract; EX1002, ¶¶77-81.

```
operation(args)
   args is a list of operation arguments
   arg_types is a list of the types of args
   arg_ops is a list of the ops to be run on args

   arg_types = types(args)
   If operation is not mandatory
      then collect predicate(operation, first(args))
         into arg_ops
   Foreach arg in args
      collect access predicates(operation, arg)
         into arg_ops
   If (authorize args arg_types arg_ops)
      then results = execute(procedure, args)
   Foreach transform in transforms
      apply-trans(transform,args,results)
```

Claim element [1.4] does not have a corresponding element in claim 3, although claim element [3.6], discussed below, is similar. Element [1.4] recites a response engine that compiles any rule violation indicated in the message from the runtime environment monitor and compares the rule violation to the security policy, then allows permitted calls to be executed. Jaeger's system discloses this element.

The '520 patent describes a "response engine" as software that responds to a security violation compiled from the runtime environment monitor:

> If the OS request does not violate a security rule 330, then the response engine 318 in step 730 instructs the operating system 260 via the recognizing probe 310-316 to resume operation of the OS request.

EX1008 at 6:41-51. When there is a rule violation, the response engine can respond by "stopping execution of a suspicious Downloadable." EX1008 at 6:6-13.

Jaeger teaches a response engine for compiling each rule violation indicated in the messages forwarded by the runtime environment monitor. This function is performed by the "operation" procedure, for example. *See* EX1005 at Fig. 8; EX1002, ¶78; *see also* §I.C.i. above (describing the "Execute Content" step of Fig. 5). As explained by Dr. Bestavros, Jaeger's example pseudocode shows that any violation of the security rules is received as a message from the "authorize" function (the runtime environment monitor) as a FALSE return value. EX1002, ¶78. For each rule violation detected, the return value of FALSE is recorded and used to evaluate the IF statement in the line "If (authorize args arg_types arg_ops)." When the IF statement evaluates a FALSE value, it skips the THEN line that follows, thus avoiding execution of the requested operation. *Id.* As claim 1 does not recite a plurality of messages or violations, nor any particular content of the messages (e.g., describing the violation), evaluating this "IF" function for each message constitutes "compiling each rule violation indicated in the messages forwarded." EX1002, ¶78.

Nonetheless, it would also have been obvious to include additional information in each message, characterizing one or more violations. EX1002, ¶79. For example, it would have been obvious to identify which operations were found to violate rules of the security policy for purposes such as debugging, rights modification, or characterizing malicious code. *Id.* Moreover, such a modification would have been straightforward, as Jaeger's system already generates the information in the "authorize" procedure. *Id*. Compiling such violation information upon receipt would likewise be obvious, as failing to do so would defeat the purpose in sending it. *Id.; see also* EX1014 at Abstract ("compiled program code is allowed to execute and type enforcement violations are logged").

Jaeger's system blocks execution of OS calls that are forbidden according to the security policy when execution of the OS calls would result in a violation of a predetermined combination of multiple security rules of the predetermined security policy. Jaeger's "operation" function blocks execution of any operation that would violate a security policy because the operation is only executed if the "authorize" function returns the value TRUE. EX1005 at Figs. 8, 9; EX1002, ¶80. As discussed above in claim element [1.3], the "authorize" function returns TRUE if the operation comports with the security policy, and FALSE if it violates the security policy.

Furthermore, Jaeger's security policy is a predetermined combination of

security rules. Jaeger describes that the "access rights available to the content are an intersection" of several different sets of rights. EX1005 at 0007. Such an intersection is a predetermined combination of multiple security rules. EX1002, ¶80. This combination is used to allow or block execution by Jaeger's "operation" procedure. EX1005 at Fig. 8; EX1002, ¶80-81.

Accordingly, Jaeger in view of CORBA discloses a response engine as described in claim element [1.4].

Jaeger in view of CORBA discloses **elements [1.5] and [3.2]:**

'755 Patent	**Jaeger and CORBA**
[1.5, 3.2] a downloadable engine for intercepting a request message being issued by a downloadable to an operating system, wherein the request message includes an extension call;	Jaeger discloses controlling access requests from downloadables: "Obviously, the access requests made directly by content must be controlled." EX1005 at 0005. "Any controlled operation run by the content is authorized by the appropriate interpreter and if authorization is granted then the operation is executed." *Id*. at 0008. Jaeger suggests implementing class-based controls: "If the software should be further restricted or if we want to be sure that the software can be executed successfully, then we suggest the creation of classes for software to represent this information. Software classes contain strongly-typed constructors for executing the software. Also, access control predicates can be assigned to the arguments in the call." *Id*. at 0016.

	See also id. at 0002-05 (Java-enabled Netscape and Java appletviewer), 0014, Fig. 6 ("class operations"); EX1002, ¶¶82-87, 109-15.

Claim elements [1.5] and [3.2] are identical, each reciting "a downloadable engine for intercepting a request message being issued by a downloadable to an operating system, wherein the request message includes an extension call." Jaeger's system discloses these elements.

Jaeger discloses such a downloadable engine. Jaeger states that "[o]bviously, the access requests made directly by content must be controlled." EX1005 at 0005.By controlling execution of downloaded content, Jaeger's system manages and executes downloadables. EX1002, ¶82-83, 109-111. This corresponds to the meaning of "downloadable engine" in the '755 patent, as discussed in §V.C above. The downloadable engine intercepts request messages from downloadables to handle access requests. *Id*; *see also* §I.C.i. above (describing Fig. 5, which includes content execution in an application-specific interpreter).

Access request messages from downloadables would include extension calls. Jaeger suggests including components to handle requests corresponding to classes of program objects. *See* EX1005 at 0016; EX1002, ¶84, 112. Such requests would be extension calls. *See* §V.B, above.

Furthermore, it would have been obvious to employ a downloadable engine,

such as the Java virtual machine of Java-enabled Netscape, as disclosed by Jaeger (*see* EX1005 at 0002-03, 0004-05), for the execution of software such as applets. *See* EX1002, ¶¶85-86, 113-14; see also EX1022 at 5:17-29. To obtain authorization, Java extensions would generate request messages including an extension calls. EX1002, ¶¶85, 113. The application of class-based access rights, as suggested by Jaeger, to allow the interpreters of Jaeger to process Java extension calls, would have been a routine matter for a person of ordinary skill in 1996. *Id*. at 87, 115.

Accordingly, Jaeger in view of CORBA discloses a downloadable engine as described in elements [1.5] and [3.2].

Jaeger in view of CORBA discloses **elements [1.6] and [3.3]:**

'755 Patent	**Jaeger and CORBA**
[1.6, 3.3] a request broker for receiving a notification message from the downloadable engine regarding the	Jaeger discloses class-based rights that would include extension calls:"[O]perations are implemented by a call to authorize which checks the content access rights prior to calling the actual operation. Since these commands are only available in the browser and are protected by the authorization operation, only authorized accesses to svstem obiects are not possible." EX1005 at 0016. "If the software should be further restricted or if we want to be sure that the software can be executed successfully, then we suggest the creation of classes for software to represent this

extension call;	information. Software classes contain strongly-typed constructors for executing the software. Also, access control predicates can be assigned to the arguments in the call." *Id*. CORBA discloses a request broker: "FIG. 2 on page 30 shows a request being sent by a client to an object implementation. The Client is the entity that wishes to perform an operation on the object and the Object Implementation is the code and data that actually implements the object. The ORB is responsible for all of the mechanisms required to find the object implementation for the request, to prepare the object implementation to receive the request, and to communicate the data making up the request." EX1011 at 30. "[CORBA] is structured to allow integration of a wide variety of object systems." *Id*. at 29. "The Basic Object Adapter will start a program to provide the Object Implementation, in this example, a per-class server." *Id*. at 152. *See also id*. at 15, 20, Fig. 2; EX1005 at 0002, 0009, 0016, Fig. 5; EX1002, ¶¶88-91, 116-20.

Claim elements [1.6] and [3.3] are identical, each reciting "a request broker for receiving a notification message from the downloadable engine regarding the extension call." Jaeger in view of CORBA discloses these elements.

As discussed above, Jaeger sends messages from its downloadable engine to

a browser, as well as to application-specific interpreters, as appropriate. *See, e.g.*, EX1005 at Fig. 5. No function other than receiving and forwarding a message is attributed to the request broker of claims 1 and 3; accordingly, such a feature would have been obvious in view of Jaeger alone. EX1002, ¶¶88-89, 116-18; *see also* EX1005 at 0016 (discussing extension calls), Fig. 5.

Furthermore, it would have been obvious to include a request broker in Jaeger's architecture to receive and forward notification messages, as taught by CORBA. CORBA illustrates the structure of an object request broker ("ORB") in Fig. 2, which "shows a request being sent by a client to an object implementation." EX1011 at 30. CORBA describes that "ORB is responsible for all of the mechanisms required to find the object implementation for the request, to prepare the object implementation to receive the request, and to communicate the data making up the request." *Id*.

CORBA also describes that the architecture is "structured to allow integration of a wide variety of object systems." *Id*. at 29. For example, the request brokers of CORBA can receive and forward messages for objects in an object oriented code, including objects organized by class. *See, e.g.*, EX1011 at 152, EX1002, ¶¶90, 119. CORBA explicitly discloses a request broker capable of handling extension calls, Jaeger in view of CORBA discloses a request broker for receiving a notification message from the downloadable engine regarding the

extension call. EX1002, ¶¶90, 119.

It would have been obvious to use a request broker, as disclosed in CORBA, in the architecture of Jaeger. Jaeger discloses a distributed system architecture and suggests using "classes of objects" to control access rights. *E.g.*, EX1005 at 0002, 0009, 0016. Requests are forwarded from a downloadable to interpreters. *Id*. at 0007-08. CORBA discloses that request brokers can be used in to "provide[] interoperability between applications on different machines in heterogeneous distributed environments and seamlessly interconnect[] multiple object systems." EX1011 at 15. CORBA further explains that its "model is an example of a classical object model, where a client sends a message to an object." EX1011 at 20. This parallels the system of Jaeger, where content messages are sent to objects for execution. EX1002, ¶¶90-91, 119-20. Using the request broker of CORBA to pass notification messages regarding an extension call in Jaeger's system would be obvious, as it would represent a combination of prior art elements according to known methods to yield predictable results. *Id*.

Accordingly, Jaeger in view of CORBA discloses a request broker as described in elements [1.6] and [3.3].

Jaeger in view of CORBA discloses **elements [1.7] and [3.4]:**

'755 Patent	**Jaeger and CORBA**
[1.7, 3.4] a file system	*See* [1.2] above, describing why Jaeger's interpreters are

probe and a network system probe each being associated with an operating system function for receiving the request message from the downloadable engine and intercepting an operating system call being issued by the downloadable to an operating system and associated with the operating system function	OS probes. Jaeger discloses interpreters that provide file and network access for Java requests: "For example, content run using Java-enabled Netscape is prevented from performing any file system I/O and communicating with third parties …. The Java appletviewer permits some access rights to be granted to content." EX1005 at 0003. "By default, all these interpreters only grant content (remote content in the case of Java) the right to communicate to only ports at the IP address of the content. The addition of read and write access rights to files is possible in the appletviewer by specifying those rights in the ~./hotjava/properties file." *See also* EX1005 at 0005, 0007, 0008, Figs. 3, 5 (showing Jaeger's interpreter controlling network and file access); EX1002, ¶¶92-96, 121-25.

Claim elements [1.7] and [3.4] are identical, each describing file system and network system probes associated with an OS function that receive request messages from the downloadable engine and intercept OS calls being issued by the downloadable. Jaeger's system discloses these elements.

As discussed above with regard to claim element [1.2], Jaeger discloses OS probes as part of its interpreters, and that the probes intercept OS calls being issued

by the downloadable associated with OS functions. *See also* EX1002, ¶¶92-95, 121-24; §I.C.i. above (describing the "Browser" (probe) of Fig. 5 intercepting OS requests for "Local File" and "Network Services" operations).

Among the sources of OS calls to be intercepted by Jaeger's system would be an extension call from an object corresponding to a class (such as a Java object), where that call sought access to a file or network system. *See, e.g.*, EX1005 at 0003 (discussing "file system I/O and communicating with third parties" being allowed for "Java appletviewer."), 0005 (discussing granting to Java applets "the right to communicate to … IP addresses" and "read and write access rights to files"); EX1002, ¶¶96, 125. Such a call is handled by sending a message to the appropriate network or file system probe of Jaeger's browser, as shown in Fig. 5. EX1002, ¶¶96, 125. The probes of Jaeger that handle file and network access requests are the recited file and network probes.

Accordingly, Jaeger in view of CORBA discloses file and network system probes as described in elements [1.7] and [3.4].

Jaeger in view of CORBA discloses **elements [1.8] and [3.5]:**

'755 Patent	**Jaeger and CORBA**
[1.8] an event router for receiving the notification message	Jaeger discloses receiving multiple types of requests at a browser: "Lastly, the architecture provides support for enforcing the access rights specified above while executing

from the request broker regarding the extension call and an event message from one of the file system probe and the network system probe regarding the operating system call; [3.5] providing an event message regarding the operating system call;	content and any external software and network services used by the content. The browser: (1) authorizes content operations; (2) provides safe implementations of these operations; (3) enables controlled execution of external software and network services." EX1005 at 0015. "[The browser] authenticates the remote principal and finds the appropriate interpreter to execute the content. If the remote principal is authorized to execute content in that interpreter, then the content is executed." *Id*. at Fig. 4. *See also id*. at Fig. 5, EX1002, ¶¶97-98, 126.

Claim element [1.8] recites "an event router for receiving the notification message from the request broker regarding the extension call and an event message from one of the file system probe and the network system probe regarding the operating system call." Claim element [3.5] recites that the file and network system probes provide an event message regarding the OS call. Jaeger's system discloses each of these elements.

As discussed in claim element [1.2], the OS probes of Jaeger (which would include a file system probe and a network system probe) request authorization of

their respective OS calls. Such a request would be sent to the interpreter component for authorization, as shown in the "Authorize Operation" step of Fig. 5 and the pseudocode of Figs. 8 and 9. *See also* EX1005 at Fig. 3, EX1002, ¶98, 126.

The requests for authorization received by Jaeger's interpreters include the claimed notification messages and event messages for authorizing extension calls and OS calls. EX1002, ¶98, 126. As discussed above in element [1.2], event messages are be generated for OS calls, and these would include file system and network system calls handled by their respective probes. The part of Jaeger's code that receives each request would correspond to an event router, as recited in claim element [1.8]. *Id.* Furthermore, it would have been obvious to include an event router in Jaeger's system, as each message requesting authorization must be received by the authorization subroutines, as illustrated in Fig. 5. *See* EX1002, ¶98.

Jaeger in view of CORBA discloses **elements [1.9] and [3.6]:**

'755 Patent	**Jaeger and CORBA**
[1.9, 3.6] [a/the] runtime environment monitor for receiving the notification message and the event message [from the event router] and	*See* [1.3] above, describing the runtime environment monitor. Jaeger's security policy involves an intersection of sets of access rights: "The access rights of a remote principals content in interpreter t active at state s are the intersection of the rights of interpreter t and the rights of the principal group to which the remote principal is assigned in

comparing the extension call and the operating system call against a predetermined security policy before allowing the operating system to process the extension call and the operating system call; and	interpreter t at state s." EX1005 at 0012. The rights are used to authorize execution: "Any controlled operation run by the content is authorized by the appropriate interpreter and if authorization is granted then the operation is executed. Operations are executed in the interpreter trusted with the operation." *Id.* at 0008. *See also id.* at 0014, Figs. 5, 8, 9; EX1002, ¶¶99, 127-32.

Claim elements [1.9] and [3.6] both recite a runtime environment monitor for receiving the notification and event messages and comparing the respective calls against a security policy before allowing the OS to process them. Jaeger's system discloses these elements.

As discussed above with respect to claim element [1.3], Jaeger discloses a runtime environment monitor (embodied in the "authorize" function of Fig. 9) that receives messages related to calls such as extension and OS calls. *See also* §I.C.i. above (describing the "Authorize Content" step of Fig. 5). Furthermore, as discussed above, the runtime environment monitor of Jaeger compares each call to a security policy before allowing the OS to process it. *See* EX1002, ¶¶99, 127-32. Similar runtime environment monitor software would handle authorization for each

of the notification and event messages.

Moreover, as discussed in claim element [1.8], the extension and OS calls would be received from an event router, which itself would be the portion of Jaeger's interpreter that receives function calls and invokes the "authorize" procedure of the "operation" routine to authorize them. *See* EX1002, ¶¶97-99.

Accordingly, Jaeger in view of CORBA discloses a runtime environment monitor as described in elements [1.9] and [3.6].

Jaeger in view of CORBA discloses **elements [1.10] and [3.7]:**

'755 Patent	**Jaeger and CORBA**
[1.10, 3.7] [a/the] response engine for receiving a violation message from the runtime environment monitor when one of the extension call and the operating system call violate one or more rules of the predetermined security policy and blocking extension calls and operating system calls that are forbidden according to the predetermined security policy, and for allowing	*See* [1.4] above, describing the response engine shown in Fig. 8. Jaeger's response engine enables execution of requests conditioned on a message received from the authorize procedure: "Lastly, the architecture provides support for enforcing the access rights specified above while executing content and any external software and network services used by the content. The browser: (1) authorizes content operations; (2) provides safe implementations of these operations; (3) enables controlled execution of external software and network services." EX1005 at 0015.

extension calls and operating system calls that are permitted according to the predetermined security policy.	"**Any** controlled operation run by the content is authorized by the appropriate interpreter and if authorization is granted then the operation is executed." *Id*. at 0008 (emphasis added). *See also id* at Figs. 5, 8, 9; EX1002, ¶¶100-02, 133-36.

Claim elements [1.10] and [3.7] both recite a response engine for receiving a violation message from the runtime environment monitor when a call violates a security rule. The response engine blocks forbidden calls and allows permitted calls. Jaeger's system discloses these elements.

As discussed above with respect to claim element [1.4], Jaeger discloses a response engine that receives a message when a rule is violated by a call, such as an extension call or OS call. *See also* §I.C.i. above (describing the "Execute Content" step of Fig. 5). Furthermore, as discussed in element [1.4], the response engine determines whether or not the security policy is violated based on such messages. *See* EX1002, ¶¶100-02, 133-36.

Jaeger discloses a response engine, as illustrated in Fig. 8, that allows operations that are permitted and blocks operations that are forbidden according to a received message. This function is indicated by the code "If (authorize args arg_types arg_ops) then results = execute(procedure, args)." *See* EX1005 at Fig. 8,

EX1002, ¶¶100-01, 134-35. This code will execute the procedure requested by the call if and only if it is permitted, as determined by the return value of the authorize procedure. *Id*. In Jaeger's security policy, a call is forbidden if the runtime environment monitor indicates a rule violation, and permitted otherwise. *Id*. Moreover, the same rules would apply for extension calls and OS calls, as Jaeger states that "**Any** controlled operation run by the content is authorized by the appropriate interpreter and if authorization is granted then the operation is executed." *Id*. at 0008 (emphasis added), *see also id*. at Fig. 5 ("if authorized the browser access[es] system objects on behalf of content."); EX1002, ¶102, 136.

Accordingly, Jaeger in view of CORBA discloses a runtime environment monitor as described in elements [1.10] and [3.7].

Accordingly, independent claims 1 and 3 would have been obvious to a person of ordinary skill in view of Jaeger and CORBA. EX1002, ¶139.

ii. Claims 2 and 4

Claims 2 and 4 depend from claims 1 and 3 respectively, and further recite that the downloadable is one of a Java component, an ActiveX control, executable code, or interpretable code. Jaeger in view of CORBA discloses this element, in addition to the other elements discussed above with respect to claims 1 and 3.

As indicated by its title, Jaeger is directed to "Building Systems That Flexibly Control Downloaded Executable Content." Accordingly, Jaeger's system

includes a downloadable in the form of executable code. EX1002, ¶104, 138; see also EX1005 at Figs. 3, 5 (illustrating steps including "Download," "Authenticate Content Source," "Authorize Operation," and "Execute Content"). Similarly, each of the other recited types of downloadable would have been obvious, as each was a well-known type of Downloadable in 1996. EX1002, ¶104, 138; *see also* EX1005 at 0002-03.

Accordingly, claims 2 and 4 would have been obvious in view of Jaeger and CORBA. In addition to the discussion above, the following claim chart shows in further detail how Jaeger in view of CORBA renders claims 2 and 4 of the '755 patent obvious. EX1002, ¶139.

'755 Patent	**Jaeger and CORBA**
[2/4]. The system of claim [1/3], wherein the downloadable is one of a Java component, an ActiveX control, executable code, or interpretable code.	*See* Claims 1 and 3 above. "Building Systems That Flexibly Control Downloaded Executable Content." EX1005, Title. See EX1005 at Figs. 3,5, 0002-03 ("download and run executable content," "Java-enabled Netscape, Java's Appletviewer, and Tcl's safe interpreter"), 0014 ("Evaluation of arbitrary code"); EX1002, ¶¶103-04, 137-38.

B. [Ground 2] Claim 5 is Obvious under 35 U.S.C. § 103(a) over Jaeger in view of CORBA and further in view of TBAV

As described in further detail below, claim 5 of the '755 patent would have been obvious in view of Jaeger, CORBA, and TBAV. EX1002, ¶¶140-48.

Claim 5 depends from claim 3, and further recites that "the response engine stores results of the comparison in an event log." As discussed above with respect to Ground 1, Jaeger in view of CORBA renders claim 3 obvious.

Jaeger also discloses generating an event log. For example, Jaeger states that the contents of a file used to define mappings from object groups to system objects "should be based on analysis, *e.g.*, use in system that can log process actions of the interpreter t." EX1005 at 0013. Logging process actions of the interpreter would include storing results of the interpreter's comparison in a log. EX1002, ¶143.

To the extent that Jaeger does not explicitly disclose storing results of the comparison in an event log, it would have been obvious to do so. Indeed, storing such information in a log file was a well-established, routine practice when dealing with potentially malicious files. EX1002, ¶145. For example, TBAV discloses the generation of such log files.

TBAV is a user manual describing a software package for malware protection that includes event logging. EX1002, ¶145. TBAV describes a utility called "TbLog, which is designed primarily to create log files in response to various TBAV alert messages." EX1006 at 139. TBAV states that the TbLog

utility is "primarily for network users," and provides convenient logging of activity so that a "supervisor can easily keep track of what's going on." *Id*. TbLog is described as a "utility that writes a record into a log file whenever one of the resident TBAV utilities pops up with an alert message. It also records when a virus is detected." *Id*. TBAV also describes the contents of such a log entry:

> A TbLog record provides three pieces of information: The time stamp of when the event took place. The name of the machine on which the event occurred. An informative message about what happened and which files were involved. This information is very comprehensive and takes only one line.

Id.; EX1002, ¶146.

It would have been obvious to store the results of Jaeger's comparison in an event log for the purpose of identifying suspicious downloadables. Jaeger describes a purpose of the system as preventing a "malicious remote principal" from "obtain[ing] unauthorized access to the downloading principal's resources." EX1005 at 0002. If a downloadable attempted to access unauthorized resources, that would indicate that it was suspicious. EX1002, ¶144. Indeed, Jaeger recognizes that such behaviors are suspicious, stating that such a downloadable "could try to perform, either accidentally or maliciously, unauthorized actions." EX1005 at 0004.

Logging software, as disclosed in TBAV, would have been obvious to use

for this purpose. EX1002, ¶147. Furthermore, there would have been a reasonable expectation of success in making such a combination, as recording of data in log files has long been routine. *Id*.

Accordingly, claim 5 would have been obvious in view of Jaeger and TBAV. In addition to the discussion above, the following claim chart shows in further detail how Jaeger in view of CORBA and TBAV renders claim 5 of the '755 patent obvious. EX1002, ¶148.

'755 Patent	**Jaeger, CORBA, and TBAV**
5. The system of claim 3, wherein the response engine stores results of the comparison in an event log.	*See* Claim 3 above. Jaeger discloses logging of process actions in its interpreters: "The file … is used to define the mapping from object groups to system objects for interpreter *t*. The contents of this file should be based on analysis, *e.g.*, use in system that can log process actions of the interpreter *t*." EX1005 at 0013. TBAV discloses generation of log files: "TbLog, which is designed primarily to create log files in response to various TBAV alert messages" EX1006 at 139. The log files contain information about what events happened: "A TbLog record provides three pieces of information: The time stamp of when the event took place. The name of the machine on which the event occurred. An informative message about what happened and which files were involved. This information is very

	comprehensive and takes only one line." *Id*. The log files record when viruses are detected: "TbLog is a memory resident TBAV utility that writes a record into a log file whenever one of the resident TBAV utilities pops up with an alert message. It also records when a virus is detected." *Id*. *See also* EX1005 at Fig. 8; EX1002, ¶¶142-48.

C. [Ground 3] Claims 6-8 are Obvious under 35 U.S.C. § 103(a) over Jaeger in view of CORBA and further in view of Arnold

As described further below, claims 6-8 of the '755 patent would have been obvious to one of ordinary skill in the art in view of Jaeger, CORBA, and Arnold. EX1002, ¶¶149-65.

Claims 6, 7, and 8 each depend from claim 3. As discussed above with respect to Ground 1, Jaeger in view of CORBA renders claim 3 obvious. As discussed below, each of the recited elements of claims 6-8 (informing the user, storing relevant information in a database, and discarding a suspicious downloadable) would have been obvious in view of Arnold.

i. Claim 6

Claim 6 of the '755 patent depends from claim 3 and further recites that "the response engine informs the user when the security policy has been violated." To the extent that Jaeger does not explicitly disclose that the response engine informs the user when the security policy has been violated, such a feature would have

been obvious to include in Jaeger's system in view of Arnold. EX1002, ¶¶151 -55.

There are several reasons that a notifying a user of a security violation would be a desirable feature. For example, a security violation represents a danger, and informing a user of such danger would be obvious. *See, e.g.*, EX1005 at 0002, 0003, 0015; EX1002, ¶152. Furthermore, Jaeger's system blocks commands that violate the policy, so the user should be informed that a requested operation has failed, to explain the resulting change in program behavior. EX1002, ¶152. Finally, a security violation may occur when a program has been given the wrong domain rights; if alerted, the user could correct this and allow the program to execute. *Id*. Any or all of these reasons could motivate a person of ordinary skill to alert the user of a security violation.

Additionally, it was well known in the art of virus detection that the user should be alerted when potentially malicious content was detected. EX1002, ¶153. For example, Arnold discloses a system for detecting "anomalous behavior that may indicate the presence of an undesirable software entity such as a computer virus, worm, or Trojan Horse." EX1007 at Abstract. Arnold teaches that "a number of commercial computer virus scanners are successful in alerting users to the presence of viruses that are known apriori." *Id*. at 1:64-66. Arnold further discloses that "the use of the distress signal is appropriate to alert the user and/or an expert to the existence of a confusing, potentially dangerous situation." *Id*. at 20:62-64, *see*

also id. at Fig. 2; EX1006 at 75 (describing displaying a "virus alert window" if a scanner "finds a file to be suspicious"); EX1002, ¶153. A program being blocked due to a security violation would be an example of a "confusing, potentially dangerous situation." EX1002, ¶153.

As discussed above, a person of ordinary skill would be motivated to inform the user, as taught by Arnold, in response to a security violation in Jaeger's system, for various reasons. A person of ordinary skill would also have a reasonable expectation of success in doing so, as generating alert messages has long been routine practice. EX1002, ¶154. Accordingly, claim 6 would have been obvious in view of Jaeger, CORBA, and Arnold.

ii. Claim 7

Claim 7 of the '755 patent depends from claim 3 and further recites that "the response engine stores information on the Downloadable in a suspicious Downloadable database when it includes extension OS calls that are forbidden according to the predetermined security policy." To the extent that Jaeger does not explicitly disclose storing information on the Downloadable in a suspicious Downloadable database when it includes forbidden calls, such a feature would have been obvious to include in Jaeger's system. EX1002, ¶¶156-60.

Storing information in a database relating to potentially malicious content was well known prior to 1997. EX1002, ¶¶157-58. For example, Arnold discloses

storing information on the Downloadable in a suspicious Downloadable database. Arnold discloses a system in which, upon detection of a virus, an identifying signature is extracted and added to a signature database. EX1007 at Abstract; *see also id*. at Fig. 3, step E ("extract signature from code, add to scanner's database"); EX1002, ¶158. Extracting signatures of a detected virus helps detect copies, as well as variants of the virus. EX1007 at 9:13-41; *see also id*. at 5:28-68, 19:38-43, Fig. 2, step B.

A person of ordinary skill in the art would be motivated to store information on the Downloadable in a suspicious Downloadable database, as taught by Arnold, when Jaeger's system detected a security violation, because such a violation indicates that the content may be malicious. *See* EX1005 at 0002. By recording information about such malicious content to a database, such as the database of Arnold, would allow the system to more easily screen out future copies and variants of the content. Moreover, maintaining and updating databases was a common practice in computer science, and Arnold teaches that doing so helps protect a system from malicious or otherwise dangerous content. A person of ordinary skill would thus have a reasonable expectation of success in using the database of Arnold with the system of Jaeger in this way. EX1002. ¶159. Accordingly, claim 7 would have been obvious in view of Jaeger, CORBA, and Arnold.

iii. Claim 8

Claim 8 of the '755 patent depends from claim 3 and further recites that "the response engine discards the Downloadable when it includes extension calls or operating system calls that are forbidden according to the predetermined security policy." To the extent that Jaeger does not explicitly disclose discarding the Downloadable when it includes forbidden calls, such a feature would have been obvious to include in Jaeger's system. EX1002, ¶¶161-65.

When Jaeger's system detects a violation of a security policy, such as an unauthorized request, this indicates that downloadable making the request may be malicious. *See, e.g.*, EX1005 at 0002. Such a downloadable is dangerous, so discarding the downloadable upon detection (by deleting it, for example) is an obvious response. EX1002,¶162. Jaeger's system includes a file system that "can perform read, write, and execute operations" (EX1005 at 0005), which would be able to delete a file. EX1002, ¶162. Indeed, discarding a malicious downloadable would have been standard practice in 1996. *Id*.

For example, Arnold teaches that a malicious downloadable should be discarded when detected. Arnold describes that when malicious code is detected, a "cleanup" process is performed that "involves killing active worm processes (determined by tracing the process tree), and deleting worm executables and auxiliary files from storage media." EX1007 at 21:34-37; *see also id*. at Abstract

(describing detecting "undesirable software entities and taking remedial action if they are discovered"), 5:61-65 ("virus removed (killed) by traditional methods, such as … disinfection (removal of the virus from all of the software it has infected.)"). Accordingly, Arnold teaches discarding of malicious code upon detection. EX1002, ¶163.

A person of ordinary skill would be motivated to discard a Downloadable, as taught by Arnold, when Jaeger's system detected a security policy violation, because a violation indicates that the Downloadable is likely to be malicious, and should therefore be deleted. EX1002, ¶164. A person of ordinary skill would have a reasonable expectation of success, as Arnold teaches that discarding a potentially malicious Downloadable is one of the "traditional methods" for dealing with malicious software. EX1007 at 5:61-65; EX1002, ¶164.

Accordingly, claim 8 would have been obvious in view of Jaeger, CORBA, and Arnold.

In addition to the discussion above, the following claim chart shows in further detail how Jaeger in view of CORBA and Arnold renders claims 6-8 of the '755 patent obvious. EX1002, ¶155, 160, 165.

'755 Patent	**Jaeger, CORBA, and Arnold**
6. The system of claim 3, wherein the	*See* Claim 3 above.

response engine informs the user when the security policy has been violated.	Arnold teaches that anomalous behavior is suspicious. *See* EX1007 at Abstract. Arnold teaches that users should be alerted of viruses: "Currently, a number of commercial computer virus scanners are successful in alerting users to the presence of viruses that are known apriori." *Id.* at 1:64-66. Sending a distress signal is appropriate when a dangerous situation is identified: "As such, the use of the distress signal is appropriate to alert the user and/or an expert to the existence of a confusing, potentially dangerous situation." *Id.* at 20:62-64. *See also id.* at Fig. 2; EX1006 at 75; EX1002, ¶¶151-55.
7. The system of claim 3, wherein the response engine stores information on the Downloadable in a suspicious Downloadable database when it includes extension calls or operating system calls that are forbidden according	*See* Claim 3 above. Arnold teaches to extract a signature from a malicious executable and store it in a database: "[E]xtracting an identifying signature from the executable code portion and adding the signature to a signature database." EX1007 at Abstract. Storing information on a Downloadable such as a signature allows future copies to be identified: "As was noted above, the signature report(s) may be displayed on the display device 20 and/ or are written to a database for use by the virus scanner. As soon as the signature for the previously

to the predetermined security policy.	unknown virus has been extracted, the scanner of Step B uses the signature to search for all instances of the virus in the system." *Id*. at 19:38-43. *See also id*. at 5:28-68, 9:13-41, Figs. 2, 3; EX1005 at 0002; EX1002, ¶¶156-60.
8. The system of claim 3, wherein the response engine discards the Downloadable when it includes extension calls or operating system calls that are forbidden according to the predetermined security policy.	*See* Claim 3 above. Arnold teaches the deletion of malicious code: "Cleanup involves killing active worm processes (determined by tracing the process tree), and deleting worm executables and auxiliary files from storage media." EX1007 at 21:34-37. Remedial action should be taken when suspicious code is discovered: "[A]utomatic scanning for occurrences of known types of undesirable software entities and taking remedial action if they are discovered." *Id*. at Abstract. Removal of a virus is a traditional method of dealing with malicious code: "If the anomaly is found to be due to a known virus or some slight alteration of it, the method proceeds to Step B_1 where the user is alerted, and the virus removed (killed) by traditional methods, such as restoration from backup (either automatically or manually by the user) or disinfection (removal of the virus from all of the software it has infected.)" *Id*. at 5:59-65.

	See also id. at Fig. 2; EX1005 at 0002, 0005; EX1002, ¶¶161-65.

VII. CONCLUSION

For the reasons set forth above, the challenged claims of the '755 patent are unpatentable. Blue Coat therefore requests that an *inter partes* review of these claims be instituted and the claims be canceled.

Respectfully submitted,

Dated: March 1, 2017

/ Michael T. Rosato /
Michael T. Rosato, Lead Counsel
Reg. No. 52,182

VIII. CERTIFICATE OF COMPLIANCE

Pursuant to 37 C.F.R. §42.24(d), the undersigned certifies that this Petition complies with the type-volume limitation of 37 C.F.R. §42.24(a). The word count application of the word processing program used to prepare this Petition indicates that the Petition contains 13,942 words, excluding the parts of the brief exempted by 37 C.F.R. §42.24(a).

Respectfully submitted,

Dated: March 1, 2017

/ Michael T. Rosato /
Michael T. Rosato, Lead Counsel
Reg. No. 52,182

IX. PAYMENT OF FEES UNDER 37 C.F.R. §§ 42.15(A) AND 42.103

The required fees are submitted herewith. If any additional fees are due at any time during this proceeding, the Office is authorized to charge such fees to Deposit Account No. 23-2415.

X. APPENDIX – LIST OF EXHIBITS

Exhibit No.	Description
1001	U.S. Patent No. 9,219,755 to Shlomo Touboul
1002	Declaration of Dr. Azer Bestavros
1003	*Curriculum Vitae* of Dr. Azer Bestavros
1004	File History of U.S. Patent No. 9,219,755 (excerpts)
1005	Jaeger *et al.*, "Building Systems that Flexibly Control Downloadable Executable Content" (July 1996)
1006	ThunderBYTE Anti-Virus Utilities User Manual (1995)
1007	U.S. Pat. No. 5,440,723 to Arnold *et al.*
1008	U.S. Pat. No. 6,167,520 to Shlomo Touboul
1009	U.S. Pat. No. 6,092,194 to Shlomo Touboul
1010	U.S. Provisional Application No. 60/030,639
1011	"The Common Object Request Broker: Architecture and Specification," (December 1993)

1012	U.S. Pat. No. 5,696,822 to Carey Nachenberg
1013	U.S. Pat. No. 6,480,962 to Shlomo Touboul
1014	U.S. Pat. No. 5,867,647 to Haigh *et al.*
1015	U.S. Pat. No. 5,857,102 to McChesney *et al.*
1016	Decision in Reexam Control No. 95/001836
1017	ThunderBYTE Evaluation Request Page (1996)
1018	ThunderBYTE Server Statistics for April, 1996
1019	Proceedings of the Sixth Annual USENIX Security Symposium
1020	USENIX Proceedings Download Page for Jaeger (Nov. 14, 1996)
1021	USENIX Publication Order Form (Nov. 22, 1996)
1022	U.S. Pat. No. 5,754,830 to Butts *et al.*
1023	U.S. Pat. No. 5,819,093 to Davidson et al.

CERTIFICATE OF SERVICE

Pursuant to 37 C.F.R. §§ 42.6(e) and 42.105(a), this is to certify that I caused to be served a true and correct copy of the foregoing Petition for Inter Partes Review of U.S. Patent No. 9,219,755 (and accompanying Exhibits 1001-1023) by overnight courier (Federal Express or UPS), on this 1st day of March, 2017, on the Patent Owner at the correspondence address of the Patent Owner as follows:

FINJAN, INC.
2000 University Avenue
Suite 600
East Palo Alto, CA 94303

Dawn-Marie Bey
213 Bayly Court
Richmond, VA 23229

Respectfully submitted,

Dated: March 1, 2017

/ Michael T. Rosato /
Michael T. Rosato, Lead Counsel
Reg. No. 52,182