UNITED STATES PATENT AND TRADEMARK OFFICE

_____

BEFORE THE PATENT TRIAL AND APPEAL BOARD

_____

BLUE COAT SYSTEMS LLC,
Petitioner,

v.

FINJAN, INC.,
Patent Owner.

_____

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

---

**PATENT OWNER'S PRELIMINARY RESPONSE**
**UNDER 37 C.F.R. § 42.107**

# TABLE OF CONTENTS

# TABLE OF AUTHORITIES

**Page(s)**

**Cases**



**Statutes**

**Other Authorities**

## I. INTRODUCTION

On March 1, 2017, Blue Coat Systems LLC ("Petitioner") submitted a Petition to institute an *inter partes* review ("IPR") challenging claims 1–8 of U.S. Patent No. 9,219,755 (the " '755 Patent") (Ex. 1001). Finjan, Inc. ("Patent Owner") requests that the Board not institute *inter partes* review because Petitioner has not demonstrated a reasonable likelihood that it would prevail in showing unpatentability of any of the challenged claims on the grounds asserted in its Petition, as required under 37 C.F.R. § 42.108(c).

The '755 Patent is generally directed to systems and methods for protecting client computers from malicious Downloadables, which are executable programs that are downloaded from a source computer and run on a destination computer. To protect client computers, the '755 Patent discloses monitoring two types of calls that may be issued by a Downloadable, operating system calls and extension calls. In order to monitor operating system calls, the '755 Patent discloses **operating system probes, such as a filesystem probe and a network system probe, each associated with an operating system function**, that intercept operating system calls issued by a downloadable to an operating system and provides an **event message** regarding the operating system call. In order to monitor extension calls, the '755 Patent discloses a downloadable engine that intercepts request messages issued by a downloadable, where the request message includes an **extension call**,

and a **request broker** that receives a **notification message** from the downloadable engine regarding the extension call.

A runtime environment monitor receives the notification message and the event message and compares 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. If an extension call or an operating system call violates a predetermined security policy, a response engine blocks the forbidden extension calls and operating system calls.

The references cited in the Petition do not disclose this approach to computer security. For example, the primary reference cited against independent claims 1 and 3, Jaeger *et al.*, *Building Systems that Flexibly Control Downloaded Executable Content* (Exhibit 1005), merely identifies I/O commands in a Browser in order "to control access to system objects" rather than using operating system probes—like the file system probe and the network system probe, **each being associated with an operating system function**, recited in independent claims 1 and 3—to intercept operating system calls issued by the downloadable to an operating system. Ex. 1005 at 6.

Although there are a variety of reasons why the '755 Patent is valid over Petitioner's asserted prior art references, this Preliminary Response focuses on only limited reasons why *inter partes* review should not be instituted. *See*

*Travelocity.com L.P. v. Cronos Techs., LLC*, Case CBM2014-00082, Decision Denying Request for Rehearing, Paper 12 at 10 (P.T.A.B. Oct. 16, 2014) ("[N]othing may be gleaned from the Patent Owner's challenge or failure to challenge the grounds of unpatentability for any particular reason."). Accordingly, while Patent Owner reserves its right to advance additional arguments in the event that trial is instituted on any ground, the deficiencies of the Petition noted herein are more than sufficient for the Board to find that Petitioner has not met its burden to demonstrate a reasonable likelihood that it would prevail in showing unpatentability of any of the challenged claims.

## II. THE '755 PATENT

### A. Overview of the '755 Patent

The '755 Patent discloses a "malicious mobile code runtime monitoring system and methods." '755 Patent at Title. To protect a network against hostile Downloadables, the systems and methods of the '755 Patent protect a client computer by using a number of operating system probes for monitoring the operating system for events caused by a file's behavior, such as a requests made by the Downloadable. U.S. Patent No. 6,480,962 (the " '962 Patent") (Ex. 1013) at 4:29–31; *id.* at 6:24–27; *id.* at FIG. 3 (reproduced below).[1]

[1] The '755 Patent claims priority to and incorporates by reference the '962 Patent.

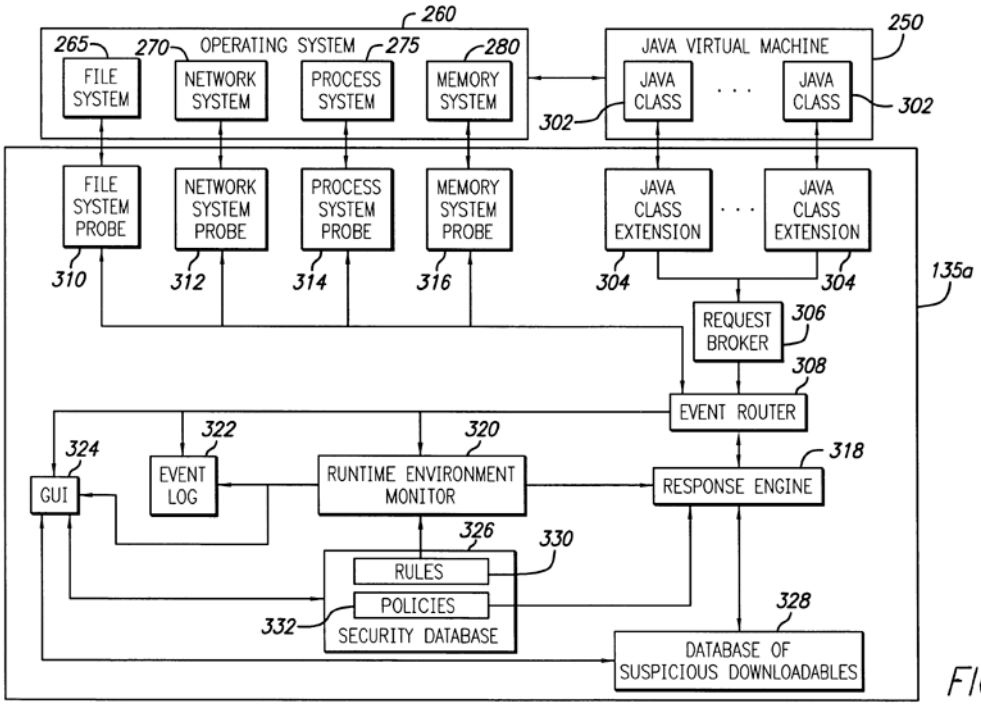



FIG. 3

For example, Figure 7 of the '962 Patent (reproduced below) illustrates a method for protecting a client from suspicious downloadables in which a series of operating system probes monitor the ***operating system*** for OS events, including "requests from Downloadables." '962 Patent at 6:24–27. When an operating system probe recognizes such an event, the request is interrupted and an event router forwards information about the event to a security engine for further processing. *Id.* at 6:27–35. The security engine includes a response engine that compares the OS request that caused the OS event alone or in combination with other information about the Downloadable with one or more security policies to determine how to respond to the event.

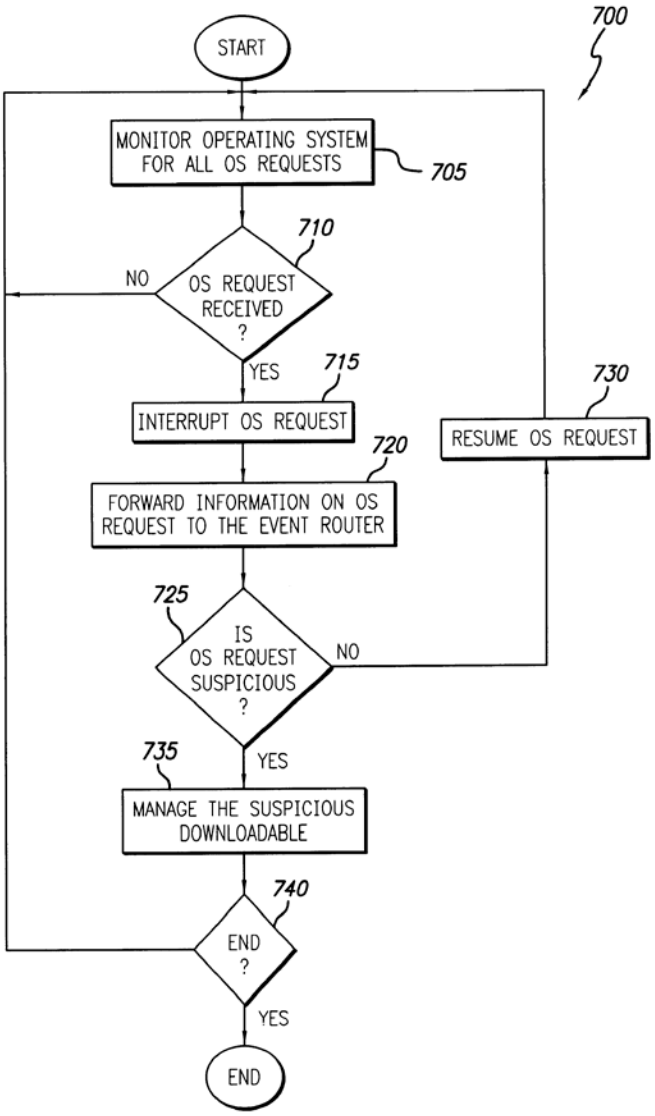



FIG. 7

The '962 Patent also discloses extensions, such as Java Class Extension 304, Messages Extension 404, DDE Extension 405, and DLL Extension 406. *See, e.g.*, '962 Patent, FIGS. 3 and 4. Java class extensions manage respective Java classes by interrupting requests to the Java class and generating a message to notify a request broker of the request (*see id.* at 4:10–18), while Messages Extension 404,

DDE Extension 405, and DLL Extension 406 handle message calls, DDE calls, and DLL calls, respectively (*id.* at 5:24–39).

In accordance with this general principle, the claims of the '755 Patent require, *inter alia*, (1) 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, (3) 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, (4) a request broker for receiving a notification message from the downloadable engine regarding the extension call, and (5) 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. *See, e.g.*, '755 Patent at claim 3.

**B. The Challenged Claims**

Petitioner challenges claims 1–8 of the '755 Patent, of which claims 1 and 3 are independent. The independent claims are reproduced below:

1. 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:

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;

a runtime environment monitor for comparing the operating system call against a predetermined security policy including multiple security rules to 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;

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 rules of the predetermined security policy and for allowing execution of operating system calls that are permitted according to the security policy;

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 and intercepting an operating system call being issued by the downloadable to an operating system and associated with the operating system function;

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;

the runtime environment monitor for receiving the notification message and the event message from the event router 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

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 extension calls and operating system calls that are permitted according to the predetermined security policy.

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.

## III. CLAIM CONSTRUCTION

In an *inter partes* review proceeding, "[a] claim in an unexpired patent that will not expire before a final written decision is issued shall be given its broadest reasonable construction in light of the specification of the patent in which it appears." 37 C.F.R. § 42.100(b); *see also* Office Patent Trial Practice Guide ("Trial Practice Guide"), 77 Fed. Reg. 48756 at 48766 (Aug. 14, 2012); *In re Cuozzo Speed Techs., LLC*, 793 F.3d 1268, 1278 (Fed. Cir. 2015) ("We conclude that Congress implicitly approved the broadest reasonable interpretation standard in enacting the AIA."). However, claims of expired patents are construed under the traditional *Phillips* standard. *In re: CSB-System Int'l, Inc.*, 832 F.3d 1335, 1341 (Fed. Cir. 2016) ("Even so, when an expired patent is subject to reexamination, the traditional *Phillips* construction standard attaches."), citing *In re Rambus Inc.*, 694 F.3d 42, 46 (Fed. Cir. 2012).

The '755 Patent has expired. The earliest nonprovisional patent application in its priority chain, U.S. Patent Application No. 08/790,097 (now U.S. Patent No.

6,167,520), was filed on January 29, 1997. The application that matured into the '755 Patent was not subject to any Patent Term Adjustment that would extend its term. Pursuant to 35 U.S.C. § 154(a)(2), the '755 Patent expired on January 29, 2017, and so the *Phillips* claim construction standard is proper.

Importantly for purposes of the instant proceeding, the interpretation claim terms arrived at during the reexamination of the related '962 Patent—interpretations based on the broadest reasonable interpretations ("BRI") of those terms—cannot be sustained under the *Phillips* standard. For example, whereas the Patent Trial and Appeal Board previously found that the "means for monitoring a plurality of subsystems of the operating system during runtime for an event caused from a request made by a downloadable" was properly construed as "include[ing] ***class extensions,*** probes, and equivalent-functioning software between an application and the operating system," and not requiring "monitoring of operating system calls to be performed at the operating system level," this construction is unsustainable under the *Phillips* standard. *See* Ex. 1016 at 4-5 (emphasis added).

**A. "downloadable" (all challenged claims)**

The proper construction of the term "downloadable" is "an executable application program, which is downloaded from a source computer and run on a destination computer." *See* Petition at 21–22.

### B. "extension call" (all challenged claims)

The term "extension call" should be accorded its plain and ordinary meaning. As Petitioner acknowledges, the '962 Patent discloses examples of extensions and how they operation with respect to original requests:

> The security system 135 a 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. The request broker 306 uses TCP/IP message passing protocol to forward the message to the event router 308.

'962 Patent at 4:10–18 (emphasis added). That is, an extension call is a call makes a request for the service of a Java class that is managed by an extension.

Petitioner's proposed construction, "a request corresponding to a class of program objects" is unrelated to the plain terminology of the claims as well as the section of the specification that Patent Owner specifically pointed to as supporting the term during prosecution. *See* Petition at 22; *see also* Ex. 1004 at 1601–04. It should, therefore, be rejected.

### C. "downloadable engine" (all challenged claims)

Petitioner states that the term "downloadable engine" means "software for managing and executing downloadables." *See* Petition at 22–23. Because this

construction does not affect Patent Owner's analysis, Patent Owner has applied Petitioner's construction of the term for the purposes of this response only.

### D. "operating system probe/file system probe/network system probe" (all challenged claims)

The term "operating system probe" means "software or hardware that monitors a **corresponding** subsystem of the operating system during runtime." In kind, the term "file system probe," being a type of operating system probe, is "software or hardware that monitors the file system subsystem of the operating system during runtime," and the term "network system probe," being a type of operating system probe, is "software or hardware that monitors the network system subsystem of the operating system during runtime."

Under the *Phillips* standard, "the person of ordinary skill in the art is deemed to read the claim term not only in the context of the particular claim in which the disputed term appears, but in the context of the entire patent, including the specification." *Phillips v. AWH Corp.*, 415 F.3d 1303, 1313 (Fed. Cir. 2005); *see also Medrad, Inc. v. MRI Devices Corp.*, 401 F.3d 1313, 1319 (Fed. Cir. 2005) ("We cannot look at the ordinary meaning of the term ... in a vacuum. Rather, we must look at the ordinary meaning in the context of the written description and the prosecution history.") (citation omitted); *V-Formation, Inc. v. Benetton Group SPA*, 401 F.3d 1307, 1310 (Fed. Cir. 2005) (finding that the intrinsic record "usually provides the technological and temporal context to enable the court to

ascertain the meaning of the claim to one of ordinary skill in the art at the time of the invention") (citation omitted).

The intrinsic record of the '755 Patent demonstrates that each operating system probe is dedicated to monitoring a particular subsystem of the operating system:

> The security system 135*a* further includes **operating system probes 310, 312, 314 and 316**. More particularly, a **file management system probe 310** recognizes applet instructions sent to the file system 265 of operating system 260, **a network system probe 312** recognizes applet instructions set to the network management system 270 of operating system 260, **a process system probe 314** recognizes applet instructions sent to the process system 275 of operating system 260, and **a memory management system probe 316** recognizes applet instructions sent to the memory system 280 of operating system 260. When any of the probes 310-316 recognizes an applet instruction, the recognizing probe 310-316 sends a message to inform the event router 308.

'962 Patent at 4:19–31 (emphasis added). This one-to-one correspondence between each operating system probe in the plurality of operating system probes and each operating system subsystem of the plurality of operating system subsystems is fully appreciable in Figure 3 of the '962 Patent (reproduced below):

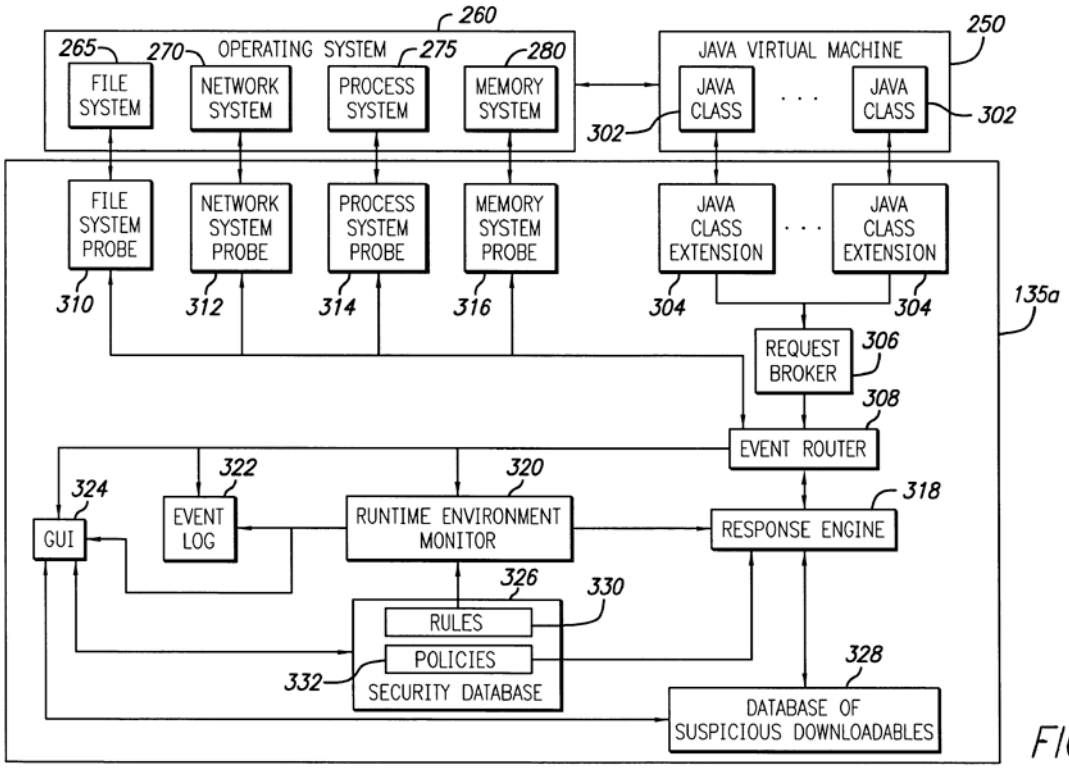



FIG. 3

Indeed, unlike the Jaeger's interpreters, the operating system probes disclosed in the '962 Patent do not even interact with the Downloadable Engine 250 that executes received Downloadables. '962 Patent at FIG. 3, *see also id.* at 3:33–40 (disclosing that "Downloadable engine 250 [is] for managing and executing received Downloadables 140"). Instead, these operating probes interact with the Downloadable only by observing operating system level events. *Id*. at FIG. 3; *id.* at 4:19–31 (explaining that the various operating system probes recognize instructions already sent to corresponding operating system subsystems).

This description is precisely consistent with the language used in the claims to define operating system probes, and file system and network system probes, which are particular examples of operating system probes:

- "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" (claim 1) (emphasis added);
- "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 [and] intercepting an operating system call being issued by the downloadable to an operating system and associated with the operating system function" (claims 1 and 3) (emphasis added).

Thus, because each operating system probe, such as the file system probe and the network system probe, is associated with an operating system function, it is associated with the corresponding subsystem of the operating system responsible for that function.

Notably, Petitioner's obviousness position relies on reading this claim term so broadly as to encompass undisclosed and uncontemplated scenarios. Such a construction would be improper under the *Phillips* standard as it would extend the metes and bounds of the claims outside of that which is supported by the written description. *See Ruckus Wireless, Inc. v. Innovative Wireless Sols., LLC*, 824 F.3d

999, 1004 (Fed. Cir. 2016) ("If, after applying all other available tools of claim construction, a claim is ambiguous, it should be construed to preserve its validity.") (citation omitted).

Patent Owner notes that its proposed construction in this case is consistent with the construction of the term "operating system probe" in prior district court litigation regarding the '962 Patent. *See* Ex. 2001, *Finjan, Inc., v. McAfee, Inc.*, No. 1:10-cv-00593-GMS, Dkt. 326, Order Construing the Terms of U.S. Patent Nos. 6,092,194 & 6,480,962 at 4 (D. Del. Feb. 29, 2012) ("The term 'operating system probe' is construed to mean 'interface for receiving and recognizing requests before allowing the operating system to execute the requests.'"). That is, the operating system probes are indeed "interface[s] for receiving and recognizing requests before allowing the operating system to execute the requests," which operate—in the context of the challenged claims—by being "associated with an operating system function" and "intercepting an operating system call being issued by a downloadable to an operating system and associated with the operating system function." *See* '755 Patent, claims 1 and 3.

For at least the foregoing reasons, the terms "operating system probe/file system probe/network system probe" are properly construed as "software or hardware that monitors a corresponding subsystem of the operating system during runtime."

## IV. SPECIFIC REASONS WHY THE CITED REFERENCES DO NOT INVALIDATE THE CLAIMS, AND WHY *INTER PARTES* REVIEW SHOULD NOT BE INSTITUTED

Petitioner's proposed Grounds rely on four references, Jaeger *et al.*, *Building Systems that Flexibly Control Downloadable Executable Content*," (Ex. 1005, "Jaeger"), *The Common Object Request Broker: Architecture and Specification* (Ex. 1011, "CORBA"), ThunderBYTE antivirus scanner user manual (Ex. 1006, "TBAV"), and Arnold, U.S. Patent No. 5,440,723 (Ex. 1007, "Arnold"). In particular, Petitioner's:

Ground 1 proposes that claims 1–4 of the '755 Patent are obvious under pre-AIA 35 U.S.C. § 103(a) over Jaeger in view of CORBA;

Ground 2 proposes that claim 5 is obvious over Jaeger in view of CORBA and further in view of TBAV; and

Ground 3 proposes that claims 6–8 are obvious over Jaeger in view of CORBA and further in view of Arnold.

### A. Claims 1 and 3 are Patentable Over Jaeger in view of CORBA

Jaeger discloses a system that "flexibly controls the access rights of downloaded content." Jaeger at Abstract. The system works by "(1) authenticating content sources; (2) determining content access rights based on its source and the application that it is implementing; and (3) enforcing these access

rights over a wide variety of objects and for the entire computation, even if external software is used." *Id.*

Importantly, the enforcing of access rights to objects—including system objects—occurs completely at the level of a "Trusted, Application-Independent Interpreter (Browser"). Accordingly, Jaeger does not disclose any operating system probes that monitory OS subsystems for events. Jaeger's procedure for enforcing access rights, which occurs within the "Browser" is shown in Figure 5:

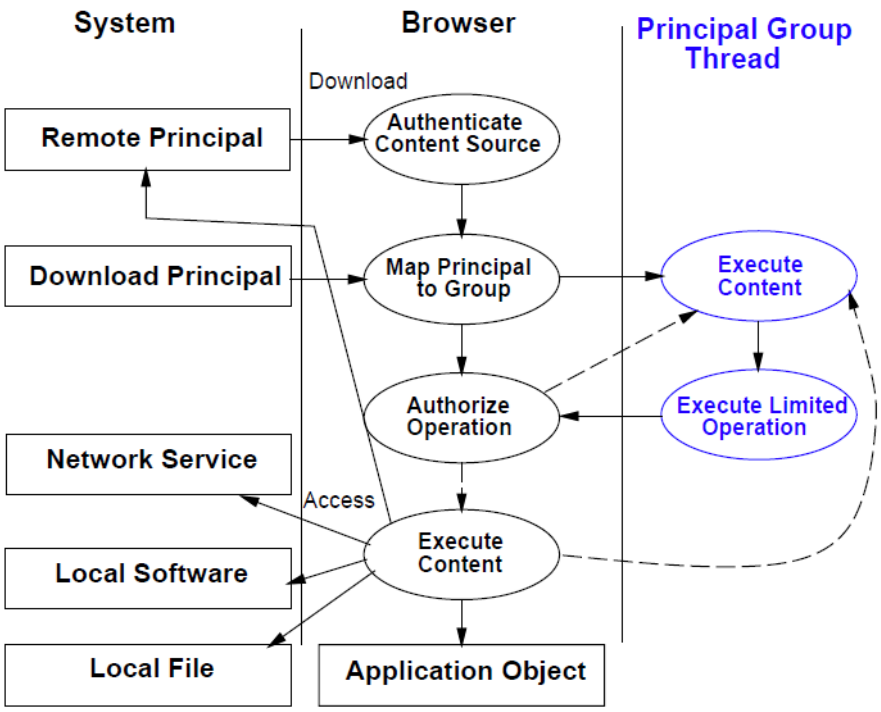



Figure 5: **Content Execution Protocol**: (1) Remote principal downloads content to browser; (2) Browser assigns content to principal group in application-specific interpreter (not shown); (3) content executes a limited operation which cause authorization in the browser; (4) if authorized the browser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

Jaeger at 0008; *see also id.* at 0015 ("Once authorized, the browser can execute the procedure. A procedure consists of two kinds of code: (1) calls to other procedures and (2) accesses to controlled objects."). Additionally, the Browser provides safe implementations of most operations, which "are implemented by a call to authorize which checks the content access rights prior to calling the actual operation." *Id*. at 0015–0016. Indeed, Petitioner recognizes that—rather than monitoring subsystems of an operating system for events—Jaeger's Browser merely intercepts attempts made by the browser itself. Petition at 13.

Jaeger also discloses an optional "application specific interpreter," which can run content if the content's "type refers to a known application-specific interpreter and the remote principal has the rights to execute content in that interpreter." Jaeger at 0007. Indeed, Jaeger discloses a "a hierarchy of interpreters where: (1) the browser is the master of the application-specific interpreters and (2) application-specific interpreters are masters of the content interpreters." *Id.* at 0010. However, these application specific interpreters and content interpreters do not control access to system objects. Such access is controlled only in the browser itself:

> Once authorized, the browser can execute the procedure. A procedure consists of two kinds of code: (1) calls to other procedures and (2) accesses to controlled objects. A browser executes calls to existing procedures in the content interpreter using the *slave* eval *args* call in

> Tcl. This ensures that all calls to other controlled procedures will be authorized using the principal group rights. Operations on controlled objects within the procedure are assumed to be authorized if access to the procedure is authorized. These commands must be run in the **browser because only the browser has access to the systems object**.

*Id*. at 0015 (emphasis added, italics in original).

In view of the foregoing, Petitioner has not met its burden to demonstrate that Jaeger in view of CORBA shows or suggests all elements of claims 1 and 3 of the '755 Patent, including at least:

- "an operating system probe associated with an operating system function" (claim 1);
- "a file system probe and a network system probe each being associated with an operating system function" (claims 1 and 3);
- "a request broker for receiving a notification message from the downloadable engine regarding the extension call" (claims 1 and 3); and
- "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 rules of the predetermined security policy and for allowing execution of operating system calls that are permitted according to the security policy" (claim 1).

Nor has Petitioner demonstrated that it would have been obvious for one of skill in the art at the time of the invention to modify Jaeger with CORBA to meet the claimed invention.

**1. Petitioner Fails to Identify in Jaeger "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" (claims 1 and 3).**

Petitioner fails to identify any element of Jaeger's system that corresponds to a "downloadable engine." Instead, Petitioner merely states that "Jaeger discloses such a downloadable engine" without positively identifying any element in Jaeger that allegedly meets its own proposed construction, "software for managing and executing downloadables." *See* Petition at 44 (vaguely referring to the downloadable engine as "Jaeger's system" that "manages and executes downloadables"); *see also id.* at 22–23 (proposing that the term downloadable engine means "software for managing and executing downloadables"). For example, Petitioner falsely contends that a "downloadable engine" is disclosed in Jaeger's Figure 5 that is separate and apart from the browser and application-specific interpreter. *See id.* at 46–47 ("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."). The Petition should, therefore, be rejected under 37 C.F.R. § 42.104(b)(4) for failing to "specify where

each element of the claim is found in the prior art patents or printed publications relied upon."

As should be appreciated with even a cursory review of Jaeger, its application-independent interpreter (browser) and optional application-specific interpreter are the software that "manages and executes downloadables." *See* Jaeger at 0002 ("The downloading principal uses an agent interpreter process running on his machine to execute the mobile agent (number 2 in the figure)."); *id*. at 0007 ("If the verification succeeds, the browser determines which interpreter to execute the content."); *id.* at 0008 ("Operations are executed in the interpreter trusted with the operation."). Petitioner plays coy with respect to this claim element because it relies on the same "software for managing and executing downloadables"—*i.e.*, Jaeger's interpreters—as allegedly disclosing the recited operating system probe, file system probe, and network system probe. *See* Petition at 30 ("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.").

Petitioner's argument that "it would have been obvious to employ a downloadable engine, such as the Java virtual machine of Java-enabled Netscape, as disclosed by Jaeger" is fundamentally wrong. *See* Petition at 44–45. The portions of Jaeger cited for this proposition demonstrate that the entire purpose of

the newly proposed architecture is to **replace** standard interpreters, like Java's applet viewer and Tcl's safe interpreter. *See* Jaeger at 0002–0003. This paradigm shift is appreciable by comparing Figures 2 and 3, which represent the "current architecture" and the new "architecture for flexible control of downloaded executable content," respectively:

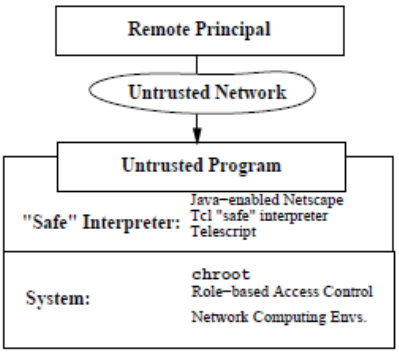



Figure 2: The current architecture for systems that utilize downloaded executable content

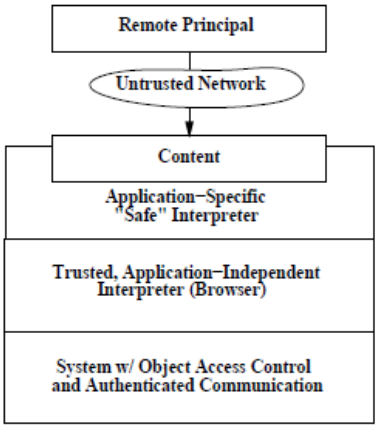



Figure 3: An architecture for flexible control of downloaded executable content: (a) system services control access outside the interpreter; (b) trusted, application-independent interpreters (*browsers*) control access to system objects by the less-trusted interpreters; (c) application-specific interpreters control access to application objects and determine the access rights of content within their domain; (d) downloaded content implements the actions of remote principals.

Jaeger at 0005, 0007. In fact, Jaeger's motivation for replacing these interpreters with the application-specific and application-independent interpreters is that "these [legacy] interpreters are not suitable for building an application where a downloading principal must grant rights to a specific remote principal." *Id.* at 0003. The sections cited by Petitioner only serve only to reinforce the fact that

Jaeger's interpreters cannot be the "operating system probe" that Petitioner relies upon them for.

This conflation of claimed "downloadable engine" and "operating system probe" (or file system and network system probes) demonstrates that Petitioner has not presented a *prima facie* case of obviousness and has not "specif[ied] where each element of the claim is found in the prior art patents or printed publications relied upon." For at least the foregoing reasons, the Board should decline to institute trial on independent claims 1 and 3 under Petitioner's proposed Ground 1. For at least the same reasons the Board should decline to institute trial on claims 2 and 4–8, which depend therefrom.

**2. Jaeger in View of CORBA Does not Disclose "an operating system probe associated with an operating system function" or "a file system probe and a network system probe each being associated with an operating system function" (claims 1 and 3).**

Petitioner has failed to meet its burden to demonstrate that Jaeger discloses an operating system probe associated with an operating system function" or "a file system probe and a network system probe each being associated with an operating system function," as recited in independent claims 1 and 3. As noted above, Petitioner conflates the terms "downloadable engine" and "operating system probe" because Jaeger does not discloses elements corresponding to both claim

elements. Indeed, Jaeger fails to disclose an operating system probe, a file system probe, or a network system probe as demonstrated in detail below.

*(a) Jaeger Does Not Disclose an Operating System Probe Because it Does Not Monitor the Operating System.*

As demonstrated above, the term "operating system probe" means "software or hardware that monitors a **corresponding** subsystem of the operating system during runtime." *See* § III.D, *supra*. For example, the recited file system probe monitors the file system subsystem, and the network system probe monitors the network system subsystem. *See* '962 Patent at FIG. 3; see also *id.* at 4:19–31.

Jaeger's interpreters—including the application-specific interpreters and the application-independent interpreter (or Browser), which Petitioner equates with the claimed operating system probe—do not work in this manner. *See* Petition at 29–33 (arguing that Jaeger's "interpreters," including "'browsers' that act as '[t]rusted, application-independent interpreters'" and "'application-specific interpreters,'" "correspond to specific applications and functions."). In fact, the '962 Patent notes problems associated with reliance upon interpreter level security measures, such as those used in Jaeger:

> Hackers have developed hostile Downloadables **designed to penetrate security holes in Downloadable interpreters**. In response, Sun Microsystems, Inc. has developed a method of restricting Downloadable access to resources (file system resources, operating system resources, etc.) on the destination computer, which effectively

> limits Downloadable functionality at the Java™ interpreter. Sun Microsystems, Inc. has also provided access control management for basing Downloadable-accessible resources on Downloadable type. However, the above approaches are difficult for the ordinary web surfer to manage, severely limit Java™ performance and functionality, and insufficiently protect the destination computer.

'962 Patent at 1:46–58 (emphasis added). As may be appreciated with respect to this disclosure, rather that implement security within the interpreter itself, the '962 Patent discloses **monitoring the operating system itself** using a number of operating system probes, where each probe monitors a corresponding subsystem. *See* § III.D; *see also* '962 Patent at 4:19–31. Jaeger, which teaches security monitoring **at the interpreter level**, is misplaced at least because rather than utilizing "operating system probes" it suffers the same flaws that exist in the prior art discussed in the '962 Patent.

*(b) Jaeger Does Not Disclose an Operating System Probe Because its Interpreters do Not Monitor Corresponding Subsystems of the Operating System.*

Additionally, the interpreters disclosed in Jaeger do not monitor **corresponding** subsystems of the operating system. Indeed, the Browser is not "associated with an operating system function" because it is agnostic to the type of operating system call in the request. In Jaeger's architecture, a single browser is responsible for monitoring every type of system access request from the browser:

> The browser provides safe implementations of open, socket, env, and exec operations. All these implementations are fairly straightforward except for exec which we describe in detail below. The other operations 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 system objects are possible.

Jaeger at 0015-0016 (emphasis added). This paradigm is illustrated in Figure 5, reproduced below:

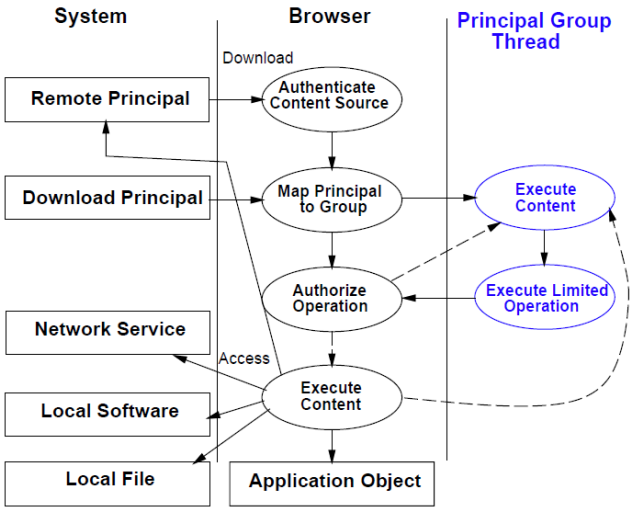



Figure 5: **Content Execution Protocol**: (1) Remote principal downloads content to browser; (2) Browser assigns content to principal group in application-specific interpreter (not shown); (3) content executes a limited operation which cause authorization in the browser; (4) if authorized the browser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

Jaeger at 0008; *see also id.* at 0007 ("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.").

As may be appreciated with respect to Jaeger's Figure 5, the browser controls **all** requests to access to what Petitioner states to be the plurality of subsystems of the operating system—namely, "Application Objects," "File System," "Processes," "Environment," or "Network Services." Petition at 31. This technique makes sense in the context of Jaeger's architecture because the Browser is not actually an operating system probe as it monitors requests at the application level rather than events at the system level, which then get reported to an event monitor. *See* '755 Patent, claims 1 and 3. Accordingly, Jaeger provides no operating system probe corresponding to each operating system subsystem.

*(c) Jaeger Does Not Disclose a File System Probe and a Network System Probe.*

For similar reasons, Jaeger does not disclose a "file system probe" and a "network system probe," as recited in independent claims 1 and 3. Indeed, Petitioner does not identify **any** teaching in Jaeger that corresponding to a network system probe or a file system probe. Rather, Petitioner only states that "[t]he probes of Jaeger than handle file and network access requests are the recited file and network probes." Petition at 50. However, the term "probe" does not appear in Jaeger, and Petitioner's position that there is some other "probe" situated in

Jaeger's Browser is inconsistent with Petitioner identification of the Browser *itself* as the operating system probe. *See id.* ("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."). There is no "network or file system probe" in Figure 5 of Jaeger.

Jaeger also does not disclose any "notification message from the downloadable engine regarding the extension call." As recited in independent claims 1 and 3, the file system probe and the network system probe are each "associated with an operating system function for receiving the request message from the downloadable engine regarding the extension call." Petitioner's conflation of the recited downloadable engine and the file and network system probes, highlights the misapplication of Jaeger to Patent Owner's claimed invention because, aside from failing to identify file and network system probes, Petitioner can identify no **request message** that is received at such a probe from the downloadable engine. *See* 37 C.F.R. § 42.104(b)(4). Rather than any alleged "network or file system probe" receiving such a request message (*see* Petition at 50), it is the browser itself—which is not disclosed as sending or receiving any request messages because such messages arise and are checked within the browser—authorizes any request for system resources. *See* Jaeger 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.").

Accordingly, the Board should decline to institute trial on independent claims 1 and 3 under Petitioner's proposed Ground 1. For at least the same reasons the Board should decline to institute trial on claims 2 and 4–8, which depend therefrom.

### 3. Jaeger in View of CORBA Does not Disclose "request broker for receiving a notification message from the downloadable engine regarding the extension call" (claims 1 and 3).

Petitioner has failed to meet its burden to demonstrate that Jaeger in view of CORBA discloses a "request broker for receiving a notification message from the downloadable engine regarding the extension call," as recited in independent claims 1 and 3. In particular, while Petitioner cites to CORBA as allegedly disclosing a "request broker," it both misunderstands the functionality of CORBA's request broker and fails to identify a request broker "for receiving a notification message from the downloadable engine regarding the extension call."

CORBA is a document that describes "The Common Object Request Broker: Architecture and Specification." CORBA at Title. This request broker is an interface that "provides interoperability between applications on different machines in distributed environments" (*see* CORBA at 3):

> As defined by the Object Management Group (OMG), the Object Request Broker (ORB) provides the mechanisms by which objects transparently make requests and receive responses. The ORB provides interoperability between applications on different machines in heterogeneous distributed environments and seamlessly interconnects multiple object systems

*Id*. at 15. That is, CORBA is designed to allow one object to request the services of another object even if the two objects are implemented differently. *See id.* at 24 ("Operations are (potentially) generic, meaning that a single operation can be uniformly requested on objects with different implementations, possibly resulting in observably different behavior. Genericity is achieved in this model via interface inheritance in IDL and the total decoupling of implementation from interface specification." (emphasis omitted)).

Aside its name, the CORBA request broker simply has nothing to do with the request broker described and claimed in the '786 Patent, which receives a notification message from the downloadable engine regarding the extension call. Neither Jaeger nor the invention of the '786 Patent operate in the type of distributed environment that requires "interoperability between applications on different machines" as disclosed in CORBA. CORBA at 15. Thus CORBA's request broker, which provides an interface between heterogeneous objects on different machines, is the improper tool to "receiv[e] a notification message from

the downloadable engine regarding the extension call." Indeed, CORBA is designed to "transparently make requests and receive responses," not to "receiv[e] a notification message from the downloadable engine regarding the extension call." CORBA at 15.

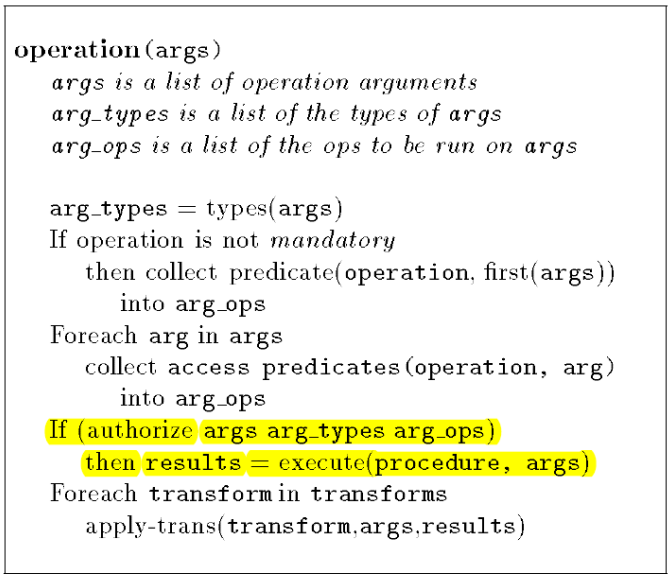
Moreover, Petitioner's statement that "CORBA explicitly discloses a request broker capable of handling extension calls" is unsupported by any evidence. Petition at 47 (citing Ex. 1002, ¶¶ 90, 119). In particular, although Petitioner cites to Dr. Bestavros' declaration as allegedly supporting this statement, Dr. Bestavros never makes that statement or any colorable approximation of that statement. Indeed, the CORBA reference does not mention the term "extension calls" at all. *See generally* CORBA (only mention the term "extension" in the context of a file name extension on page 48 and in its description on page 22 of an "extension of a type" as being "the set of values that satisfy the type at any particular time").

Accordingly, the Board should decline to institute trial on independent claims 1 and 3 under Petitioner's proposed Ground 1. For at least the same reasons the Board should decline to institute trial on claims 2 and 4–8, which depend therefrom.

**4. Jaeger in View of CORBA Does not Disclose "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 rules of the predetermined security policy and for allowing execution of operating system calls that are permitted according to the security policy" (claim 1).**

Petitioner has failed to meet its burden to demonstrate that Jaeger discloses 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 rules** of the predetermined security policy and for allowing execution of operating system calls that are permitted according to the security policy," as recited in independent claim 1. '755 Patent, claim 1 (emphasis added).

As Petitioner appreciates, Jaeger's "authorize" function returns a FALSE return value for each violation of the security rules. *See* Petition at 41. When the authorize procedure returns FALSE, the operation procedure its "IF" statement fails, and it does not execute the subject procedure:



```
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)
```

Figure 8: Generated operation with authorization and transformation calls

Jaeger at 0015. Notably, there is no **else** statement in the operation procedure that instructs the computer to take some other action upon receiving a FALSE return value from the authorize procedure. *See* Petition at 41 ("When the IF statement evaluates a FALSE value, it skips the THEN line that follows, thus avoiding execution of the requested operation."). As no affirmative action is taken upon receiving a FALSE return value, there can be no "compiling [of] each rule violation indicated in the messages forwarded by the runtime environment monitor." Petitioner's argument that "claim 1 does not recite a plurality of messages or violations" (Petition at 41) is facially false. *See* '755 Patent, claim 1 (reciting **multiple** messages being forwarded by the runtime environment monitor: "a response engine for compiling each rule violation indicated in the **messages forwarded by the runtime environment monitor**") (emphasis added).

The Board should ignore Dr. Bestavros' unsupported arguments that such a modification to Jaeger would have been obvious because "[a]ffidavits expressing an opinion of an expert must disclose the underlying facts or data upon which the opinion is based." *See* Trial Practice Guide, 77 Fed. Reg. 48756 at 48763 (citations omitted); *see also* 37 C.F.R. § 42.65(a) ("Expert testimony that does not disclose the underlying facts or data on which the opinion is based is entitled to little or no weight."). Rather than provide a supported opinion, Dr. Bestravos' Declaration "[m]erely repeat[s] an argument from the Petition in the declaration" and therefore "does not give that argument enhanced probative value." *Kinetic Techs., Inc. v. Skyworks Sols., Inc.*, Case IPR2014-00529, Decision Denying Institution of *Inter Partes* Review, Paper 8 at 15 (P.T.A.B. Sept. 23, 2014).

Additionally, Petitioner's argument that Jaeger's disclosure that "'access rights available to the content are an intersection' of several different sets of rights" corresponds to the claimed feature of "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 rules** of the predetermined security policy" is meritless. In context, Jaeger describes that the access rights for a particular piece of content are at the intersection of "(1) the rights the downloading principal grants to the remote principal to execute the application; (2) the rights that the downloading principal

grants to the application developer for the application; and (3) the rights that the application grants to the remote principal." Jaeger at 0007-0008. That is, this intersection is how the access rights for the content are determined. Jaeger is completely silent as to blocking execution of operating system calls when a "**predetermined combination of multiple security rules**" is violated.

Accordingly, the Board should decline to institute trial on independent claims 1 and 3 under Petitioner's proposed Ground 1. For at least the same reasons the Board should decline to institute trial on claims 2 and 4–8, which depend therefrom.

### 5. The Petition Provides Inadequate Motivation to Combine the Jaeger and CORBA.

Petitioner's Ground 1 should also be denied because the Petition does not "identify a reason that would have prompted a person of ordinary skill in the relevant field to combine the ***elements in the way the claimed new invention does***." *See KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 418 (2007) (emphasis added). In particular, Petitioner fails to explain why a person of ordinary skill in the art would be motivated to modify Jaeger's browser-based technique for controlling downloaded executable content with CORBA's "Common Object Request Broker," which is not applicable to downloaded executable content.

As described above, Jaeger's system applies to "distributed systems architectures, such as mobile agent and replicated process architectures, [which]

enable processes to download and run executable content." Jaeger at 0002. This architecture is shown in Jaeger's Figure 1, which shows a Downloading Principal downloading executable code from a Remote Principal:

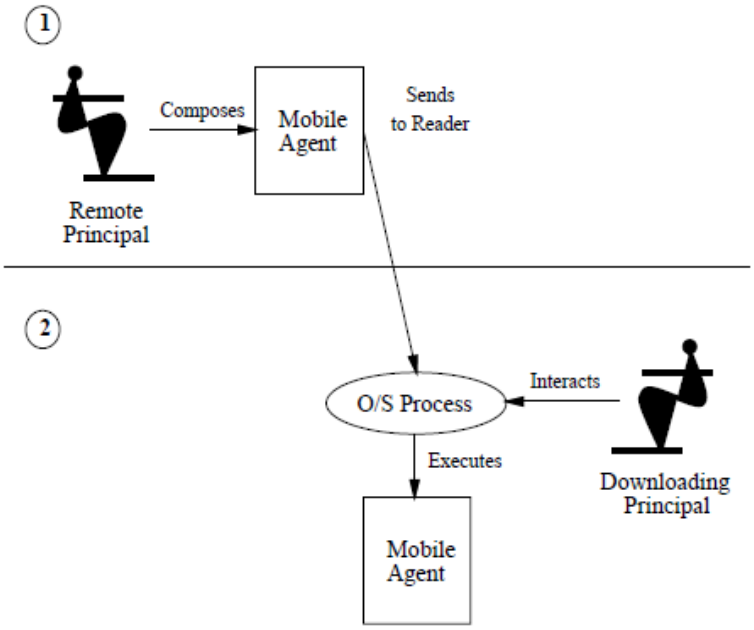



Figure 1: Mobile agent architecture

Jaeger at 0003.

In stark contrast, CORBA relates to an entirely different paradigm, one that allows for transparent communications between objects. *See* CORBA at 15 ("As defined by the Object Management Group (OMG), the Object Request Broker (ORB) provides the mechanisms by which **objects transparently make requests and receive responses**.") (emphasis added). That is, CORBA describes an interface for an application on one machine (e.g., a client) to request services of an

application on another machine (e.g., a host). *See id.* ("The ORB provides interoperability **between applications** on different machines in **heterogeneous distributed environments** and seamlessly interconnects **multiple object systems**.") (emphasis added). CORBA simply is not necessary in the environment disclosed in Jaeger, and so a POSITA would not have combined the two systems.

Rather than requiring "interoperability between applications of different machines in heterogeneous distributed environments," Jaeger's system—once the executable content is downloaded—runs entirely on the client, and access requests are either allowed or disallowed within the same environment that runs the content. *See* Jaeger at 0007 ("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.") (emphasis omitted); *see also id.* at FIG. 3, reproduced below:

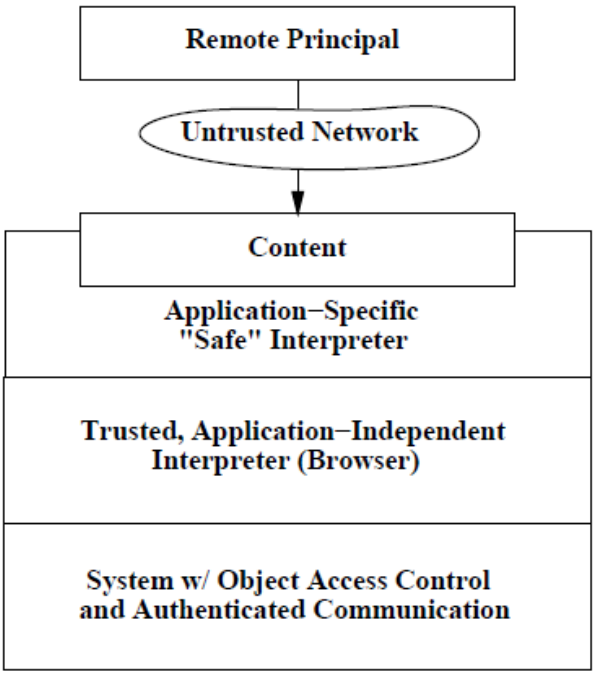



Figure 3: An architecture for flexible control of downloaded executable content: (a) system services control access outside the interpreter; (b) trusted, application-independent interpreters (*browsers*) control access to system objects by the less-trusted interpreters; (c) application-specific interpreters control access to application objects and determine the access rights of content within their domain; (d) downloaded content implements the actions of remote principals.

Jaeger at 0007.

Petitioner provides no explanation as to why a system CORBA, which is used to "provide[] interoperability between applications on different machines in heterogeneous distributed environments and seamlessly interconnect[] multiple object systems" would be used in Jaeger's client-based, homogeneous interpreter environment. Indeed, none exists.

Because the motivation proposed by Petitioner is insufficient to provide a reasonably likelihood of success on the merits, the Board should decline to institute trial on Petitioner's proposed Ground 1.

### B. Claim 6 is Patentable Over Proposed Ground 2

Claim 6 depends from independent claim 3, which has been demonstrated to be patentable over the combination of Jaeger and CORBA. *See* § IV.A, *supra*. Petitioner does not assert that TBAV cures any of the deficiencies noted above. Accordingly, the Board should decline to institute trial on independent claims 6 under Petitioner's proposed Ground 2.

### C. Claims 7–9 are Patentable Over Proposed Ground 3

Claims 7–9 depends from independent claim 3, which has been demonstrated to be patentable over the combination of Jaeger and CORBA. *See* § IV.A, *supra*. Petitioner does not assert that Arnold cures any of the deficiencies noted above. Accordingly, the Board should decline to institute trial on independent claims 6 under Petitioner's proposed Ground 3.

## V. CONCLUSION

Petitioner has not established a reasonable likelihood that it will prevail in establishing that claims 1–8 of the '755 Patent are invalid. Finjan accordingly requests that the Board deny institution of *inter partes* review of the '755 Patent.

Respectfully submitted,

Dated: June 15, 2017

/James Hannah/

James Hannah (Reg. No. 56,369)
Kramer Levin Naftalis & Frankel LLP
990 Marsh Road
Menlo Park, CA 94025
Tel: 650.752.1700 Fax: 212.715.8000

Jeffrey H. Price (Reg. No. 69,141)
Kramer Levin Naftalis & Frankel LLP
1177 Avenue of the Americas
New York, NY 10036
Tel: 212.715.7502 Fax: 212.715.8302

(Case No. IPR2017-00997) *Attorneys for Patent Owner*

BLUE COAT SYSTEMS LLC
v.
FINJAN, INC.
Case IPR2017-00997
U.S. Patent No. 9,219,755

**PATENT OWNER'S EXHIBIT LIST**

| | |
|---|---|
| **Exhibit-2001** | *Finjan, Inc., v. McAfee, Inc.*, No. 1:10-cv-00593-GMS, Dkt. 326, Order Construing the Terms of U.S. Patent Nos. 6,092,194 & 6,480,962 (D. Del. Feb. 29, 2012) |

**CERTIFICATE OF SERVICE**

The undersigned certifies, in accordance with 37 C.F.R. § 42.6(e), that service was made on the Petitioner as detailed below.

| | |
|---|---|
| *Date of service* | June 15, 2017 |
| *Manner of service* | Electronic Mail (mrosato@wsgr.com; asbrown@wsgr.com) |
| *Documents served* | PATENT OWNER'S PRELIMINARY RESPONSE |
| *Persons Served* | Michael T. Rosato<br>Andrew S. Brown<br>WILSON SONSINI GOODRICH & ROSATI |

/James Hannah/
James Hannah
Registration No. 56,369
Counsel for Patent Owner