

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

TREND MICRO, INC.,
Petitioner,

v.

OPEN TEXT INC.,
Patent Owner.

IPR2023-00692
Patent 10,284,591 B2

Before KRISTINA M. KALAN, MICHELLE N. WORMMEESTER, and
AARON W. MOORE, *Administrative Patent Judges*.

KALAN, *Administrative Patent Judge*.

DECISION
Denying Institution of *Inter Partes* Review
35 U.S.C. § 314

I. INTRODUCTION

Trend Micro, Inc. (“Petitioner”) filed a Petition (Paper 2, “Pet.”) requesting *inter partes* review of claims 1–18 of U.S. Patent No. 10,284,591 B2 (Ex. 1001, “the ’591 patent”). Open Text Inc. (“Patent Owner”) filed a Preliminary Response (Paper 6, “Prelim. Resp.”).

To institute *inter partes* review, we must determine that the information presented in the Petition shows “there is a reasonable likelihood that the petitioner would prevail with respect to at least 1 of the claims challenged in the petition.” 35 U.S.C. § 314(a) (2018). For the reasons discussed below, after considering the parties’ submissions and the evidence of record, we determine that Petitioner has not shown a reasonable likelihood of prevailing with respect to at least one of the challenged claims. We accordingly do not institute an *inter partes* review of claims 1–18 of the ’591 patent.

A. *Related Proceedings*

The parties identify the following district court litigations for the ’591 patent:

- *Webroot, Inc. and Open Text, Inc. v. Trend Micro Inc.*, Case No. 6:22-cv-00239 (W.D. Tex.).
- *Webroot, Inc. and Open Text, Inc. v. Sophos LTD.*, Case No. 6:22-cv-00240 (W.D. Tex.).
- *Webroot, Inc. and Open Text, Inc. v. CrowdStrike, Inc. and CrowdStrike Holdings, Inc.*, Case No. 6:22-cv-00241 (W.D. Tex.).
- *Webroot, Inc. and Open Text, Inc. v. AO Kaspersky Lab*, Case No. 6:22-cv-00243 (W.D. Tex.).

IPR2023-00692
Patent 10,284,591 B2

Pet. 1–2; Paper 4, 1. Patent Owner adds that the ’591 patent is also at issue in the following PTAB proceeding: *CrowdStrike, Inc. v. Webroot Inc.*, IPR2022-01522 (PTAB Sept. 30, 2022). Paper 4, 1.

B. Real Party-in-Interest

Petitioner states that the “real parties-in-interest are Trend Micro, Inc., Trend Micro America, Inc. and Trend Micro Incorporated.” Pet. 1.

Patent Owner states that the real parties-in-interest are Open Text Inc. and Webroot LLC, and further identifies Carbonite, LLC, Open Text Corporation, and Open Text Holdings, Inc. as “affiliates.” Paper 4, 1.

C. The ’591 Patent

The ’591 patent describes “anti-exploit systems and methods,” where an “exploit is a piece of code, software, data, or a sequence of commands that takes advantage of a bug, glitch or vulnerability in order to cause unintended or unanticipated behavior to occur on computer software, hardware, or any other electronic component.” Ex. 1001, 1:12–16, 1:33–35.

Figure 3 of the '591 patent is reproduced below.

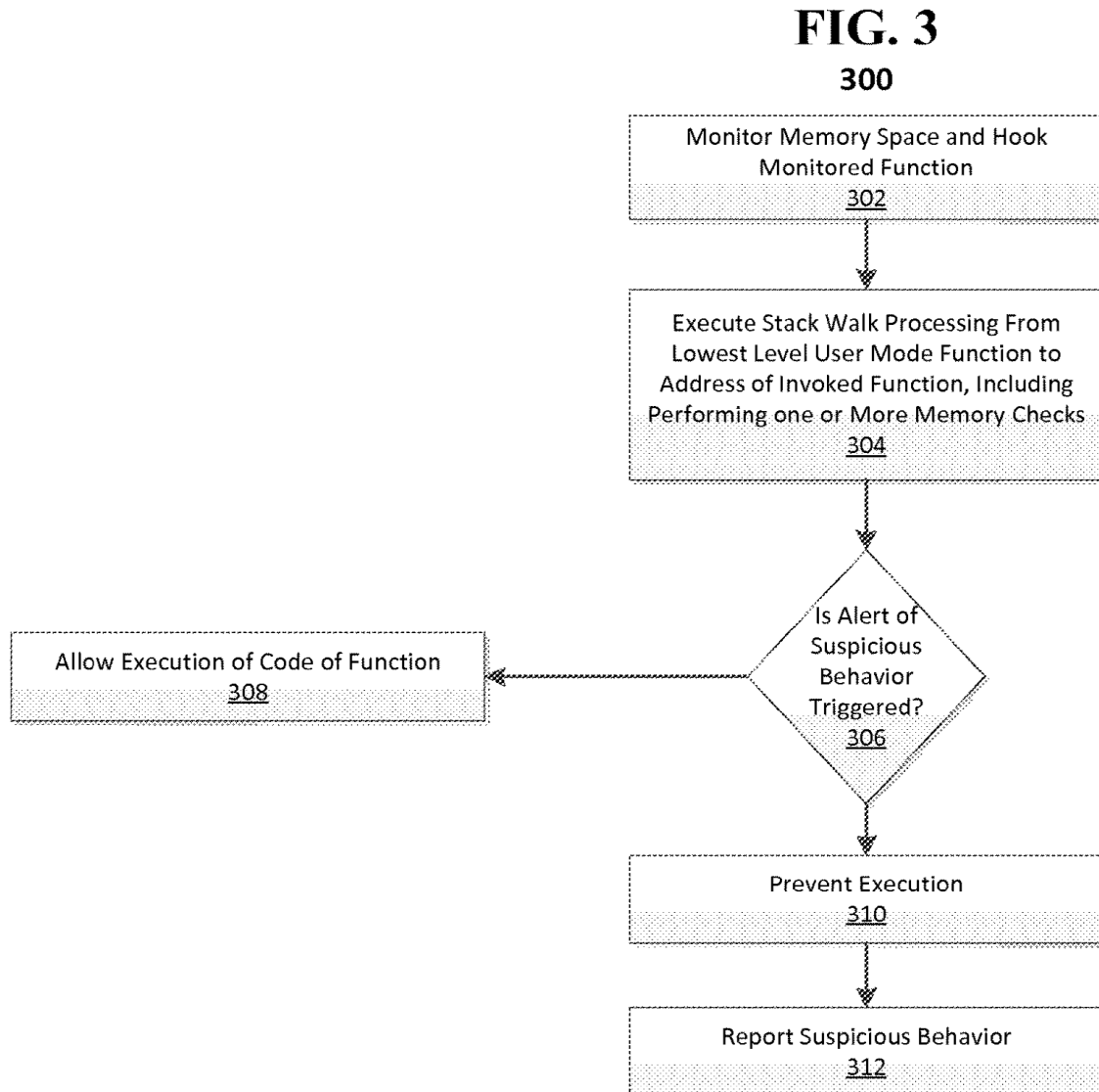


Figure 3 “illustrates an exemplary method 300 for implementation of anti-exploit systems and methods described herein.” *Id.* at 2:23–25. Method 300 begins at operation 302, “where a memory space of a process is monitored for execution of at least one of a plurality of functions,” which may include “any functions that attempt execution of another process or loading of a module into a process memory, implementable by/in other operation systems.” *Id.* at 5:10–15, 10:15–17. “The list of monitored functions may

change perpetually and can be updated (e.g., by an administrator or software update) at any time to improve the robustness of the functions monitored and improve security against software exploits.” *Id.* at 5:15–19.

The ’591 patent notes an example in which operation 302 “monitors functions associated with memory management in the memory space by a user mode function (e.g., a call initiated from a user mode).” *Id.* at 10:18–21. The ’591 patent additionally notes an example of “applying, upon detecting a function call of one of the plurality of functions in the monitored memory space, a function hooking technique to hook a lowest level user mode function of a called process (e.g., API).” *Id.* at 10:21–26. Monitoring operation 302 “may further comprise detecting an attempt by unidentifiable code to allocate a new executable page of memory and monitoring the allocated new executable page of memory.” *Id.* at 10:26–29.

When one of the monitored functions is invoked, stack walk processing is performed at operation 304 for an associated stack frame. *Id.* at 10:30–35. This is “a process that walks a stack frame for a hooked function[] from a lowest level user mode function to an address of the invoked function.” *Id.* at 10:35–39. Operation 304 also includes performing memory checks during the stack walk processing, such as identifying an originating address associated with the invoked function. *Id.* at 10:39–44.

Decision operation 306 determines whether a suspicious-behavior alert is triggered based on execution of the memory checks. *Id.* at 10:45–48.

This includes the following

- Is code execution attempted from non-executable memory;
- Is a base pointer of a stack frame identified as being invalid;
- Is invalid stack return address identified for a stack frame;

- Is attempted execution of a return-oriented programming technique detected;
- Is the base pointer of a stack frame detected as being outside a current thread stack; and
- Is a return address detected as being inside a virtual memory area.

Id. at 8:18–33. If no suspicious-behavior alert is triggered, code associated with the invoked function is allowed to execute at operation 308. *Id.* at 10:54–58. Otherwise, if an alert is triggered, code execution associated with the invoked function is prevented at operation 310, and suspicious behavior may be reported at operation 312. *Id.* at 10:58–65.

D. Illustrative Claim

Claims 1, 9, and 17 are independent. Claim 1 is directed to a computer-implemented method, claim 9 is a corresponding system claim, and claim 17 is directed to a “computer-readable storage device containing instructions, that when executed on at least one processor, caus[e] the processor to execute a process” corresponding to the method of claim 1. Claim 1 is reproduced below; the bracketed annotations include Petitioner’s identifiers for the preamble and limitations of claim 1 (Pet. 14–26).

- [1.a] A computer-implemented method comprising:
 - [1.b] monitoring a memory space of a process for execution of at least one monitored function of a plurality of functions, wherein monitoring the memory space comprises loading a component for evaluating the at least one monitored function in the memory space;
 - [1.c] invoking one of the plurality of functions as a result of receiving a call from an application programming instance;
 - [1.d] executing stack walk processing upon the invocation of one of the plurality of functions in the monitored memory space; and
 - [1.e] performing, during the executing of the stack walk processing before an address of an originating caller function is

reached, a memory check for a plurality of stack entries identified during the stack walk processing to detect suspicious behavior, wherein an alert of suspicious behavior is triggered when the performing of the memory check detects at least one of the following:

- code execution is attempted from non-executable memory,
- a base pointer is identified as being invalid,
- an invalid stack return address is identified,
- attempted execution of a return-oriented programming technique is detected,
- the base pointer is detected as being outside a current thread stack, and
- a return address is detected as being inside a virtual memory area,

[1.f] wherein when an alert of suspicious behavior is triggered, preventing execution of a payload for the invoked function from operating.

Ex. 1001, 13:29–61.

E. Asserted Ground of Unpatentability

Petitioner asserts that all claims would have been unpatentable on the following, sole ground:

Claims Challenged	35 U.S.C. § ¹	References/Basis
1–18	103	Dalcher, ² Pappas ³

¹ The Leahy-Smith America Invents Act, Pub. L. No. 112-29, 125 Stat. 284 (2011) (“AIA”), amended 35 U.S.C. § 103. Because the ’591 patent has an effective filing date after the effective date of the applicable AIA amendments, we refer to the AIA version of § 103.

² U.S. Patent Publication No. 2013/0275981 A1 to Dalcher (Ex. 1004).

³ Vasilis Pappas et al., *Transparent ROP Exploit Mitigation Using Indirect Branch Tracing*, 22nd USENIX Security Symposium (August 14–16, 2013) (Ex. 1005).

Pet. 4. In support of its unpatentability arguments, Petitioner relies on the Declaration of Dr. Seth James Nielson, Ph.D. Ex. 1002. Patent Owner relies on the Declaration of Dr. Nenad Medvidovic. Ex. 2001.

II. ANALYSIS

A. Claim Construction

We apply the claim construction standard articulated in *Phillips v. AWH Corp.*, 415 F.3d 1303 (Fed. Cir. 2005). 37 C.F.R. § 42.100(b) (2023). Under *Phillips*, claim terms are afforded “their ordinary and customary meaning.” *Phillips*, 415 F.3d at 1312. The “ordinary and customary meaning of a claim term is the meaning that the term would have to a person of ordinary skill in the art in question at the time of the invention.” *Id.* at 1313. Only terms that are in controversy need to be construed, and then only to the extent necessary to resolve the controversy. *Realtime Data, LLC v. Iancu*, 912 F.3d 1368, 1375 (Fed. Cir. 2019) (“The Board is required to construe ‘only those terms . . . that are in controversy, and only to the extent necessary to resolve the controversy.’” (quoting *Vivid Techs., Inc. v. Am. Sci. & Eng’g, Inc.*, 200 F.3d 795, 803 (Fed. Cir. 1999))).

Petitioner submits: “In parallel litigation, Webroot argues all claim terms should be given their plain and ordinary meanings. For purposes of this proceeding only, Petitioner proposes the Board adopt Webroot’s proposal.” Pet. 9–10. Patent Owner and its expert “apply the plain and ordinary meaning of all claim terms consistent with the district court’s *Markman* order.” Prelim. Resp. 28 (citing Ex. 2001 ¶ 52).

On this record, we find that we need not expressly construe any claim terms to reach a decision on institution in this proceeding. *See Realtime Data*, 912 F.3d at 1375.

B. Level of Ordinary Skill in the Art

Petitioner proposes that the person of ordinary skill in the art (POSITA or PHOSITA) would have had a “bachelor’s degree in computer science, or a related field, or an equivalent technical degree or equivalent work experience, and an additional one to two years of education or experience in the field of computer security topics such as malware/exploit detection[; m]ore education can supplement practical experience and vice versa.” Pet. 9 (citing Ex. 1002 ¶ 150). Patent Owner proposes that the POSITA would have had a “bachelor’s degree in an accredited program of electrical engineering, computer engineering, computer science, or a similar discipline, and 2–3 years of practical work or research experience in the fields of electrical engineering, computer science, networking, communications, and/or device and network security” and more “advanced degrees and/or training in a related discipline can compensate for shorter work experience;” Patent Owner notes that “minor distinctions in Petitioner’s and Patent Owner’s definitions would not alter the analysis.” Prelim. Resp. 28 (citing Ex. 2001 ¶¶ 53–57).

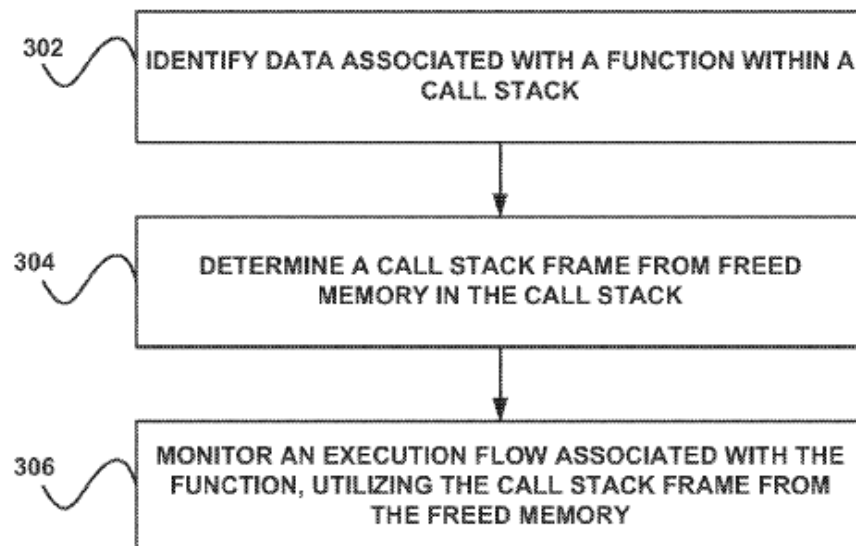
As Patent Owner does not dispute Petitioner’s characterization of the level of skill in the art, and because we find it generally consistent with the disclosures of the ’591 patent and the prior art, we adopt it for purposes of this analysis. The level of ordinary skill in the art is also reflected by the prior art of record. *See Okajima v. Bourdeau*, 261 F.3d 1350, 1355 (Fed. Cir. 2001).

C. Asserted Prior Art

1. Dalcher (Ex. 1004)

Dalcher is titled “System, Method, and Computer Program Product for Monitoring an Execution Flow of a Function.” Ex. 1004, code (54). Dalcher “relates to monitoring system activity, and more particularly to behavioral monitoring and analysis.” *Id.* ¶ 1.

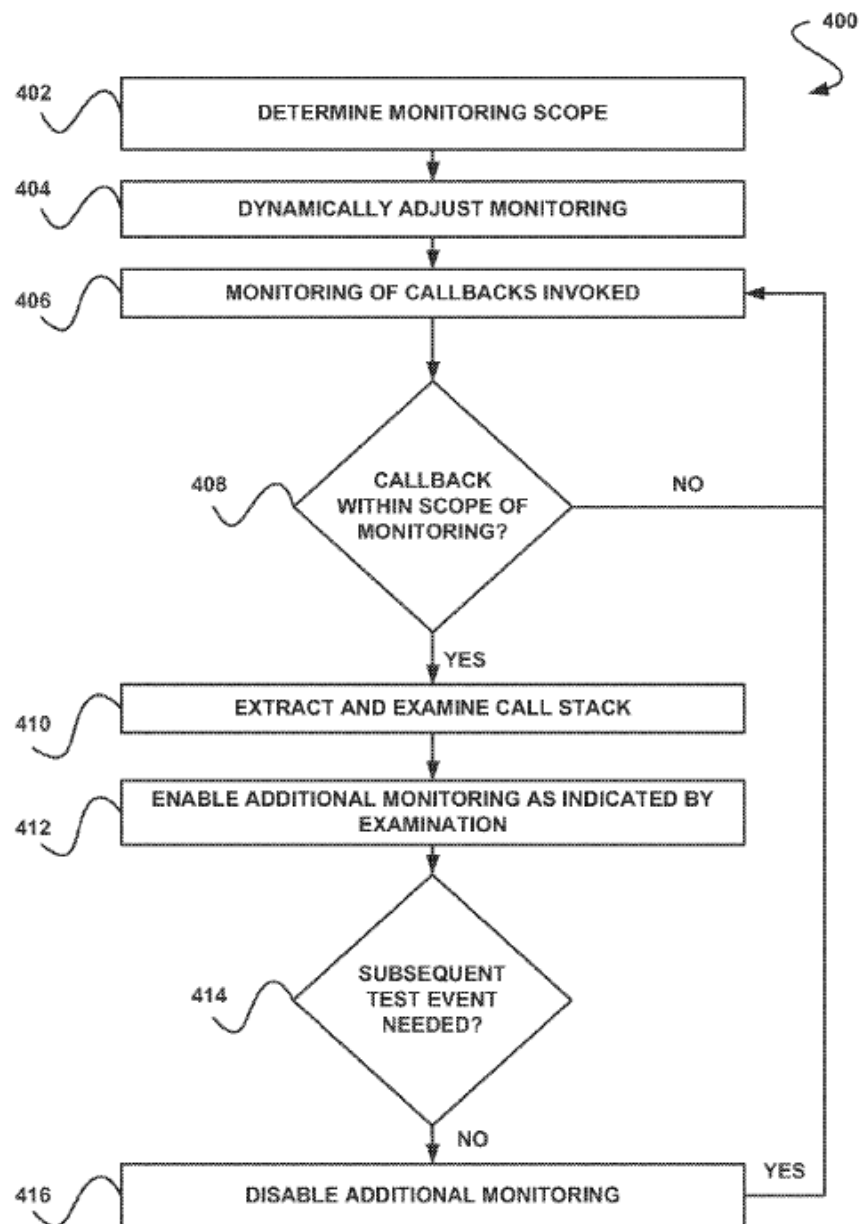
Dalcher summarizes the invention as “monitoring an execution flow of a function” via three features. *Id.* ¶ 4. First, “data associated with a function is identified within a call stack.” *Id.* Second, “a call stack frame is determined from freed memory in the call stack.” *Id.* Third, “an execution flow of the function is monitored, utilizing the call stack frame from the freed memory.” *Id.* Dalcher’s Figure 3 is reproduced below.



Dalcher’s Figure 3 illustrates a flowchart that “shows a method for monitoring an execution flow of a function.” *Id.* ¶ 7. In “operation 302, data associated with a function is identified within a call stack.” *Id.* ¶ 17. In “operation 304, a call stack frame is determined from freed memory in the call stack,” where “the call stack frame may include memory in the stack

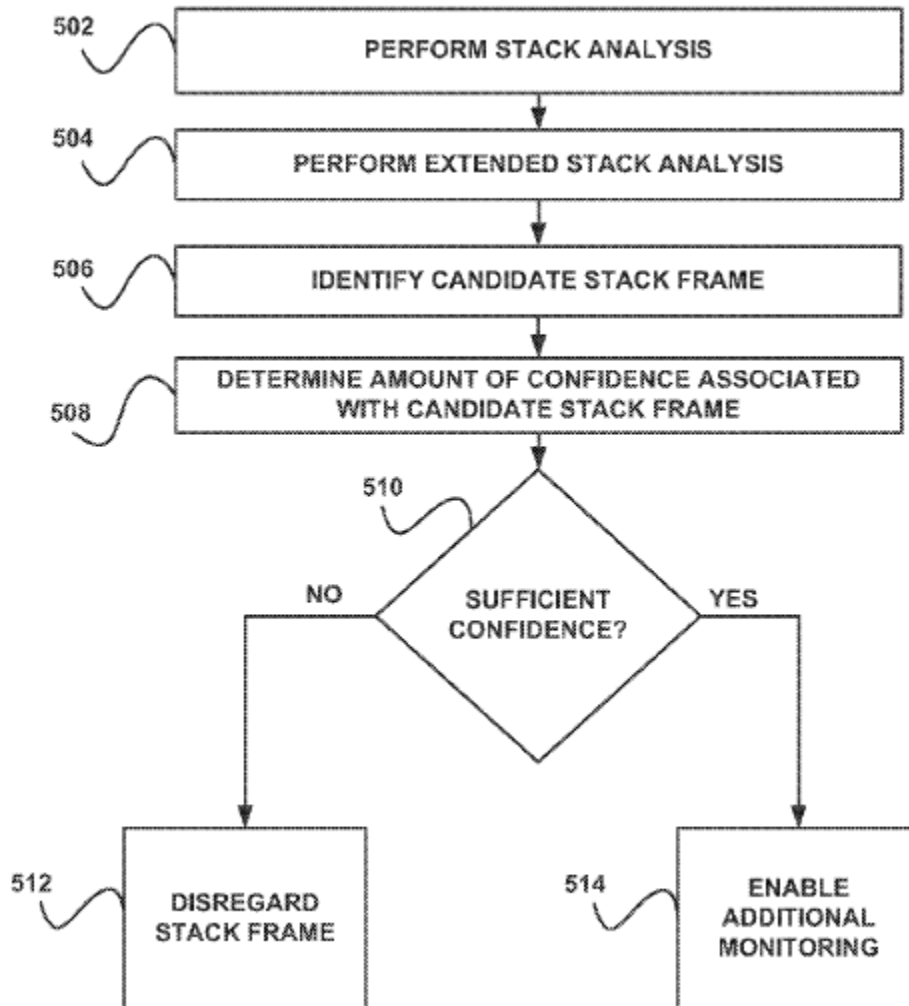
that previously held an active stack entry” and “may include memory outside (e.g., below, etc.) a stack pointer associated with the call stack.” *Id.* ¶ 21. In “operation 306, an execution flow of the function is monitored, utilizing the call stack frame from the freed memory,” and “a monitoring point (e.g., a hook, etc.) may be placed on the call stack frame” and “may identify code involved in the handling of the test event.” *Id.* ¶ 23.

Dalcher’s Figure 4 is reproduced below.



Dalcher's Figure 4 "shows a method for performing execution flow monitoring using enhanced call stack analysis." *Id.* ¶ 8. In "operation 402, a scope of monitoring to be performed is determined" in which "a test event may be generated, and the monitoring may be limited to the handling of the test event." *Id.* ¶ 28. In "operation 404, the monitoring "may be adjusted according to the determination of the scope of the monitoring." *Id.* ¶ 29. In "operation 406, monitoring of callbacks is invoked . . . for callbacks that are identified as a result of the dynamic adjustments." *Id.* ¶ 30. In "decision 408, it is determined whether a callback is within a scope of the monitoring," if decision 408 is negative, "then monitoring of callbacks is again invoked in operation 406." *Id.* ¶ 32. If decision 408 is affirmative ("within the scope"), then operation 410 is performed, in which "a call stack for the operation that triggered the monitoring callback is extracted and examined." *Id.* ¶ 33. In "operation 412, additional monitoring is enabled as indicated by the examination of the call stack," which "may include placing one or more additional monitoring points (e.g., hooks, probes, etc.) in the call stack or the freed memory in the call stack." *Id.* In "decision 414, it is determined whether a subsequent test event is needed;" if decision 414 is affirmative, "then monitoring of callbacks is again invoked in operation 406;" if decision 414 is negative, then operation 416 is performed and thereby "the additional monitoring enabled in operation 412 is disabled." *Id.* ¶ 34.

Dalcher's Figure 5 is reproduced below.



Dalcher's Figure 5 "shows a method 500 for performing extended call stack analysis." *Id.* ¶ 35. In "operation 502, stack analysis of a call stack is performed" which may include "identifying one or more call frames of the call stack by performing a stack walk" such as "walking back in the stack when stack frame pointers are not present," or "extracting a call return address from an identified call frame" and "identifying an origin of code containing the return address." *Id.* ¶ 36. In "operation 504, extended call stack analysis is performed," which may include "identifying a current top of the call stack," because "the call stack may grow from top to bottom as

functions are called, and may shrink from bottom to top as functions return,” or “moving through freed stack memory . . . to identify one or more preserved call stack frames in the freed stack memory.” *Id.* ¶ 37. In “operation 506, a candidate stack frame is identified as a result of the extended call analysis.” *Id.* ¶ 39. In “operation 508, an amount of confidence associated with the candidate stack frame is determined.” *Id.* A “suspected call return address may be extracted from the candidate call stack frame,” “it may be determined whether the suspected call return address points to code,” and “the code pointed to by the suspected call return address may be . . . examined for a preceding call that may explain the invocation of the candidate stack frame.” *Id.* The “level of confidence may be associated with a determination of whether the candidate call stack frame is genuine,” if it is “determined that a suspected call return address points to code, and the code includes a call to the candidate stack frame, a high level of confidence may be associated with the candidate stack frame.” *Id.* ¶ 41. In “decision 510, it is determined whether a sufficient amount of confidence is associated with the candidate stack frame.” *Id.* ¶ 42. If decision 510 is negative (“sufficient amount . . . is not associated”), “then in operation 512 the candidate stack frame is disregarded.” *Id.* ¶ 43. If decision 510 is affirmative (“sufficient amount . . . is associated”), “then in operation 514 additional monitoring is placed in order to validate the candidate stack frame.” *Id.*

2. *Pappas (Ex. 1005)*

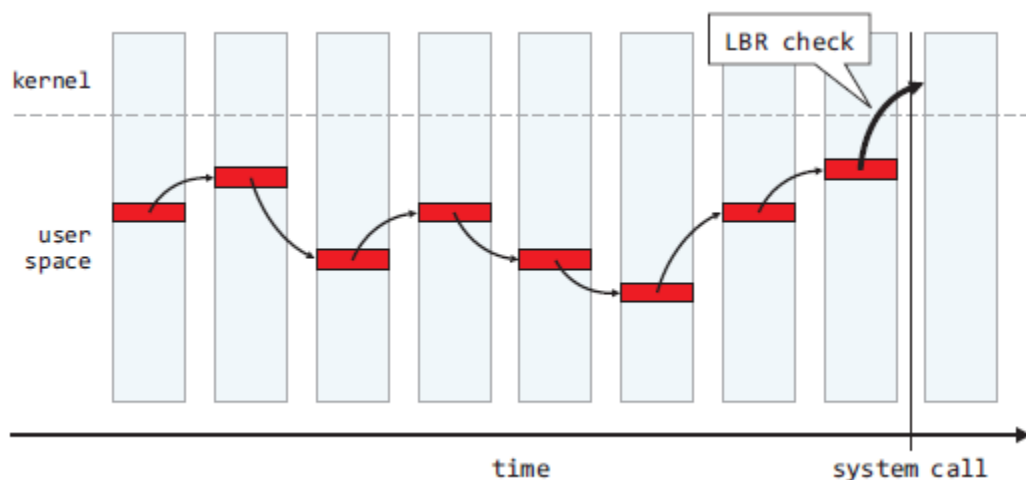
Pappas is a paper titled “Transparent ROP [Return-Oriented Programming] Exploit Mitigation using Indirect Branch Tracing.” Ex. 1005, 447 (title). Pappas relates to a “runtime ROP exploit prevention technique

for the protection of third-party applications” based on “the detection of abnormal control transfers that take place during ROP code execution.”

Id. at 447:1. Pappas teaches protecting the call stack by monitoring branching, control flow transfers, and other functions in the call stack using a hook. *Id.* at 448–450, 454

Pappas’s “approach is based on monitoring the executed indirect branches at critical points during the lifetime of a process, and identifying abnormal control flow transfers that are inherently exhibited during the execution of ROP code,” is “built around Last Branch Recording (LBR), a recent feature of Intel processors,” and relies “mainly on hardware for instruction-level monitoring,” which “allows for minimal runtime overhead and completely transparent operation, without requiring any modifications to the protected applications.” *Id.* at 448:1. Pappas terms its implementation “*kBouncer*,” and states that it “can identify and block a ROP exploit before its code accomplishes any critical operation.” *Id.* at 448:1–2.

Pappas’s Figures 1 and 3 are reproduced below.



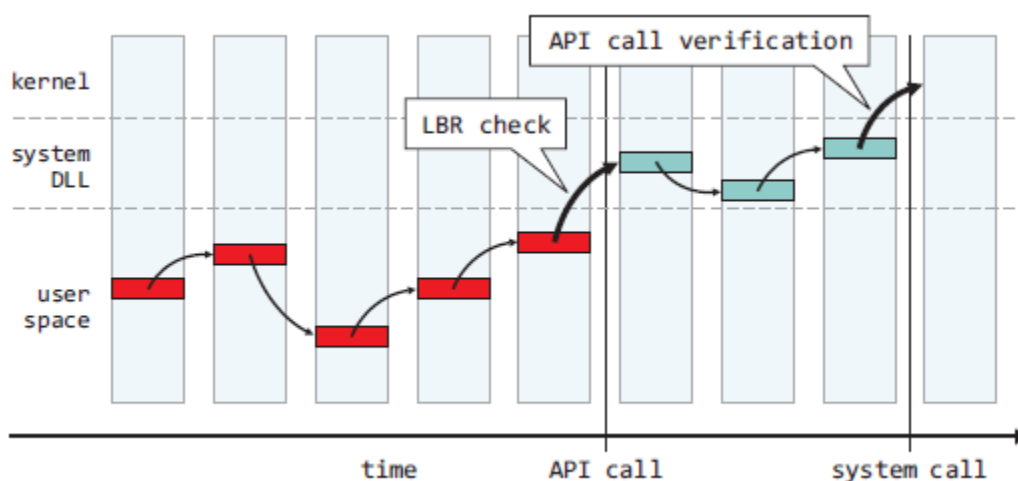


Figure 1 (top) illustrates “a basic scheme for ROP code detection,” and Figure 3 (bottom) illustrates “the detection scheme of *kBouncer*.” *Id.* at 450:1, 451:1. Figure 1 is captioned: “Whenever control is transferred from user to kernel space (vertical line), the system inspects the most recent indirect branches to decide whether the system call was invoked by ROP code or by the actual program.” *Id.* at 450:1. Figure 3 is captioned: “Overview of the detection scheme of *kBouncer*” and the caption elaborates that before “the invocation of protected Windows API functions, the system inspects the LBR stack to identify whether the execution path that led to the call was part of ROP code, and writes a checkpoint,” and to “account for ROP code that would bypass the check by jumping over *kBouncer*’s function hook, the system then verifies the entry point of the API function at the time of the corresponding system call invocation.” *Id.* at 451:1. The relationship between Figure 1 and Figure 3 is termed “Using Last Branch Recording for ROP Prevention.” *Id.* at 449:2 (section heading 2.2).

D. Analysis

In an *inter partes* review, “the petitioner has the burden from the onset to show with particularity why the patent it challenges is unpatentable.”

Harmonic Inc. v. Avid Tech., Inc., 815 F.3d 1356, 1363 (Fed. Cir. 2016) (citing 35 U.S.C. § 312(a)(3) (requiring *inter partes* review petitions to identify “with particularity . . . the evidence that supports the grounds for the challenge to each claim”)); *see also* 37 C.F.R. § 42.104(b) (requiring a petition for *inter partes* review to identify how the challenged claim is to be construed and where each element of the claim is found in the prior art patents or printed publications relied upon). Petitioner cannot satisfy its burden of proving obviousness by employing “mere conclusory statements,” but “must instead articulate specific reasoning, based on evidence of record, to support the legal conclusion of obviousness.” *In re Magnum Oil Tools Int’l, Ltd.*, 829 F.3d 1364, 1380 (Fed. Cir. 2016).

A patent claim is unpatentable under 35 U.S.C. § 103 if the differences between the claimed subject matter and the prior art are such that the subject matter, as a whole, would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. *KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 406 (2007). The question of obviousness is resolved based on underlying factual determinations including: (1) the scope and content of the prior art; (2) any differences between the claimed subject matter and the prior art; (3) the level of ordinary skill in the art; and (4) when presented, objective evidence of nonobviousness, i.e., secondary considerations.⁴ *Graham v. John Deere Co.*, 383 U.S. 1, 17–18 (1966). Subsumed within the *Graham* factors are the requirements that all claim limitations be found in the prior art references or

⁴ Patent Owner does not present any objective evidence of nonobviousness as to any of the challenged claims. *See also* Pet. 10–11 (Petitioner “is unaware of any secondary considerations relevant to the ’591 patent”).

the knowledge available to a skilled artisan and that such an artisan would have had a reasonable expectation of success in combining the prior art references to achieve the claimed invention. *Pfizer, Inc. v. Apotex, Inc.*, 480 F.3d 1348, 1361 (Fed. Cir. 2007). Additionally, the obviousness inquiry typically requires an analysis of “whether there was an apparent reason to combine the known elements in the fashion claimed by the patent at issue.” *KSR*, 550 U.S. at 418 (citing *In re Kahn*, 441 F.3d 977, 988 (Fed. Cir. 2006) (requiring “articulated reasoning with some rational underpinning to support the legal conclusion of obviousness”)).

Petitioner argues that claims 1–18 would have been obvious over the combination of Dalcher and Pappas. Pet. 10–64. Patent Owner opposes. Prelim. Resp. 36–66.

1. Petitioner’s Arguments

Beginning with its asserted motivation to combine Dalcher and Pappas, Petitioner argues that one of ordinary skill in the art would have been motivated to combine the references “since both references relate to detecting malware using the same or similar processes of call stack analysis and return-oriented programming,” and would have implemented them together “to address the common exploitation technique of return oriented programming by analyzing the relationship between stack frames (calls)” with “predictably successful results.” Pet. 12–13 (citing Ex. 1002 ¶¶ 106–109, 178–179, 183–186). “A PHOSITA would have looked to the teachings of Pappas to modify the teachings in Dalcher in order to enhance the tracking of calls, jumps, and returns and to reduce false positives in compromise detection rates.” *Id.* at 13 (citing Ex. 1002 ¶¶ 185–189).

More particularly, with respect to claim 1, Petitioner argues as follows.

[1.a] A computer-implemented method comprising:

Petitioner argues that Dalcher discloses a memory monitoring process that operates on a computer system. *Id.* at 14 (citing Ex. 1004 ¶¶ 12–16; Ex. 1002 ¶¶ 194–198).

[1.b] monitoring a memory space of a process for execution of at least one monitored function of a plurality of functions, wherein monitoring the memory space comprises loading a component for evaluating the at least one monitored function in the memory space;

Petitioner argues that Dalcher discloses monitoring a call stack containing one or more functions to be executed. *Id.* at 14–16 (citing Ex. 1004 ¶¶ 17–23, 30–33; Ex. 1002 ¶¶ 203–212). Petitioner also argues that Pappas discloses this limitation using its “kBouncer” module. *Id.* at 16–17 (citing Ex. 1005, 455, Fig. 8; Ex. 1002 ¶¶ 214–221). “Therefore,” argues Petitioner, “Dalcher and Pappas each disclose this limitation.” *Id.* at 17 (citing Ex. 1002 ¶¶ 202–223).

[1.c] invoking one of the plurality of functions as a result of receiving a call from an application programming instance;

Petitioner argues that “Dalcher teaches monitoring these OS-supplied functions to ensure proper handling,” which one of ordinary skill in the art would understand to “invoke that function or another API function.” *Id.* at 17 (citing Ex. 1002 ¶¶ 227–228; Ex. 1004 ¶¶ 26–32).

“To the extent that a call from an API is not explicitly disclosed in Dalcher,” Petitioner argues, “Pappas teaches this limitation.” *Id.* at 18 (citing Ex. 1002 ¶¶ 232–238; Ex. 1005, 450–451, 455, Fig. 8).

“Therefore, Dalcher in view of Pappas discloses this limitation.” *Id.* (citing Ex. 1002 ¶¶ 225–238).

[1.d] executing stack walk processing upon the invocation of one of the plurality of functions in the monitored memory space; and

Petitioner argues that “Dalcher teaches a stack walk as stack analysis.” *Id.* at 19 (citing Ex. 1004 ¶¶ 34–36 (“For example, stack frames may be discovered by walking back in the stack when stack frame pointers are not present.”), Fig. 5; Ex. 1002 ¶¶ 248–253, 260–264). Petitioner also relies on Dalcher’s disclosure that further analysis, i.e. a stack walk, may be performed if “monitoring indicates that [] other software was involved in this handling.” *Id.* at 20 (citing Ex. 1004 ¶¶ 26–34, Ex. 1002 ¶¶ 250–253).

“To the extent that Dalcher is considered to not execute stack walk processing upon invocation of a function,” Petitioner argues, “Pappas discloses Windows API function as an originally monitored function.” *Id.* (citing Ex. 1002 ¶¶ 219–221, 233–237; Ex. 1005, 450, 455, Fig. 8).

Petitioner concludes: “Therefore, Dalcher discloses this limitation.” *Id.* at 21 (citing Ex. 1002 ¶¶ 241–266).

[1.e] performing, during the executing of the stack walk processing before an address of an originating caller function is reached, a memory check for a plurality of stack entries identified during the stack walk processing to detect suspicious behavior, wherein an alert of suspicious behavior is triggered when the performing of the memory check detects at least one of the following:

code execution is attempted from non-executable memory,
a base pointer is identified as being invalid,
an invalid stack return address is identified,
attempted execution of a return-oriented programming technique is detected,

*the base pointer is detected as being outside a current thread stack, and
a return address is detected as being inside a virtual memory area,*

Petitioner argues that Dalcher discloses “executing a stack walk that walks back in the stack from a function call (e.g., a current top of the stack)” and this “tracing direction from recent calls to less recent calls and on to an initially called function is an essential aspect of walking back in a call stack, which is in a direction opposite to the call order (e.g., from subsequently-called function back to originally monitored/hooks (and called) function).” *Id.* at 21–22 (citing Ex. 1004 ¶¶ 36–37; Ex. 1002 ¶¶ 62–85, 91–95, 270–276). “Second, Dalcher also discloses detection,” Petitioner argues, “of at least two elements of the list of alternatives.” *Id.* at 23 (citing Ex. 1004 ¶ 24; Ex. 1002 ¶¶ 289–290, 292).

Petitioner also argues that “Pappas teaches monitoring of a memory stack (including a stack walk and memory checks) that is consistent with Dalcher and also teaches at least one of the alternatives elements of this limitation.” *Id.* at 24 (citing Ex. 1005, 451; Ex. 1002 ¶¶ 161–162, 165, 172). Additionally, Petitioner argues that “Dalcher and Pappas disclose ‘wherein an alert of suspicious behavior is triggered when the...memory check detects’ the above-discussed alternative elements.” *Id.*

“Therefore,” argues Petitioner, “Dalcher discloses elements listed in this limitation and Pappas discloses compatible and consistent memory stack monitoring, any of which are sufficient to

render the limitation obvious.” *Id.* at 25 (citing Ex. 1002 ¶¶ 268–299).

[1.f] wherein when an alert of suspicious behavior is triggered, preventing execution of a payload for the invoked function from operating.

Petitioner argues that Dalcher discloses “monitoring to determine whether a compromise (suspicious behavior) occurred and a level of confidence in the determination, and if the level of confidence that a compromise has occurred is above a certain threshold, a flag is generated indicating the compromise (corresponding to ‘alert of suspicious behavior is triggered’).” *Id.* at 25 (citing Ex. 1004 ¶¶ 24–26; Ex. 1002 ¶¶ 303–304).

Petitioner also argues that “Pappas discloses terminating execution of code or frames” that have been “identified by memory analysis to be compromised/suspicious,” and a “PHOSITA would have looked to Pappas for further compatible details on preventing execution of quarantined software or functions when a compromise (suspicious behavior) is detected.” *Id.* at 25–26 (citing Ex. 1002 ¶¶ 305–307).

“Therefore, Dalcher and Pappas disclose this limitation.” *Id.* at 26 (citing Ex. 1002 ¶¶ 302–308).

Petitioner also presents arguments directed to claims 2–18 based on the combination of Dalcher and Pappas. Pet. 26–64. Petitioner’s arguments directed to independent claim 9 (*id.* at 38–51) and independent claim 17 (*id.* at 56–60) mirror or refer back in substantial part to Petitioner’s arguments for claim 1.

2. Patent Owner’s Arguments

Patent Owner presents three categories of arguments against the combination of Dalcher and Pappas: (A) “Petitioner does not adequately

explain the combination it asserts as its sole ground”; (B) “Petitioner does not show that a POSITA would have been motivated to combine Dalcher with Pappas and, in fact, a POSITA would not have been so motivated”; and (C) “Petitioner has failed to show that the combination of Dalcher and Pappas discloses or renders obvious all limitations of any challenged claim.” Prelim. Resp. 37–38.

a) “Petitioner Does Not Adequately Articulate a Combination of Dalcher’s and Pappas’s Teachings”

Regarding Petitioner’s asserted failure to articulate the combination, Patent Owner argues that “the petition reads like a choose-your-own adventure book.” Prelim. Resp. 38 (“Dalcher discloses.... Pappas also discloses.... Therefore, Dalcher and Pappas each disclose this limitation.”). That is, according to Patent Owner, “Petitioner repeatedly recites what Dalcher and Pappas purportedly teach *individually* but fails to assert ‘what the **combined teachings** of those references would have suggested to those of ordinary skill in the art,’” but “neither the general motivation-to-combine section nor the cursory mentions of a combination in certain limitations meets Petitioner’s burden to explain what the combined teachings of Dalcher and Pappas would have suggested to a POSITA.” Prelim. Resp. 38, 40.

First, Patent Owner argues that Petitioner’s general motivation-to-combine discussion “makes only conclusory statements that a POSITA would have combined the references without ever identifying or explaining a particular combination of prior art teachings,” and “none of Petitioner’s statements about what might have motivated a POSITA to combine the teachings actually identifies which of Dalcher’s teachings a POSITA would

have modified, how they would have been modified, and by which teachings of Pappas.” *Id.* at 41.

Second, Patent Owner argues that “Petitioner’s limitation-specific analysis of the independent claims merely discusses Dalcher’s and Pappas’s teachings individually and still does not provide an ‘account of the contemplated workings *of the combination*.’” *Id.* at 42. Patent Owner presents a table that assertedly “includes, for each limitation in claim 1, the entirety of Petitioner’s discussion of the purported combination.” *Id.*

Claim 1 Limitation	Petitioner’s Asserted Reference
1.a	“Therefore, Dalcher discloses the preamble.” (Petition 14.)

1.b	“Therefore, Dalcher and Pappas each disclose this limitation.” (Petition 17.)
1.c	“Therefore, Dalcher in view of Pappas discloses this limitation.” (Petition 18.)
1.d	“Therefore, Dalcher discloses this limitation.” (Petition 21 (while also explaining how Pappas individually discloses this limitation).)
1.e	“Therefore, Dalcher discloses elements listed in this limitation and Pappas discloses compatible and consistent memory stack monitoring, any of which are sufficient to render the limitation obvious.” (Petition 25 (quoting a standard for anticipation by the prior art).)
1.f	“A PHOSITA would have looked to Pappas for further compatible details on preventing execution of quarantined software or functions when a compromise (suspicious behavior) is detected. Therefore, Dalcher and Pappas disclose this limitation.” (Petition 26.)
Conclusion	“[A] PHOSITA ... would have found claim 1 of the ’591 patent to be obvious in view of the combination of Dalcher and Pappas
	because these prior art publications disclose each and every limitation recited in the claim.” (Petition 26.)

Prelim. Resp. 42–44. Moreover, argues Patent Owner, “instituting an IPR on such a petition would be prejudicial to Patent Owner” because Patent

Owner is limited in its ability to address “some unspecified combination of disclosures from the petition,” and Petitioner could “develop arguments not initially presented in the petition.” *Id.* at 45.

b) “*There Is No Motivation to Combine the Teachings of Dalcher and Pappas*”

Patent Owner argues that there is no motivation to combine the teachings of Dalcher and Pappas. *Id.* at 46.

First, Patent Owner argues that Petitioner fails to show why a POSITA would have been motivated to combine Dalcher with Pappas to practice the claimed invention. *Id.* at 46–47 (citing *ActiveVideo Networks, Inc. v. Verizon Comms., Inc.*, 694 F.3d 1312 (Fed. Cir. 2012)). Patent Owner argues that “Petitioner’s alleged motivation to combine relies on generalized statements about Dalcher’s and Pappas’s fields of endeavor, allegedly similar features, or the purported results that could be achieved through some unspecified combination of their teachings,” which is insufficient to meet Petitioner’s burden. *Id.* at 47.

Second, Patent Owner argues that Petitioner fails to establish a reasonable likelihood of success. *Id.* at 48. “Petitioner alleges that a POSITA would have modified Dalcher in view of Pappas ‘in order to [1] enhance the tracking of calls, jumps, and returns and [2] to reduce false positives,’” but fails to explain why one of ordinary skill in the art (1) “would expect that modifying Dalcher in view of Pappas would enhance the tracking of calls, jumps, and returns” and Petitioner’s expert does not elaborate; or (2) “would expect that modifying Dalcher in view of Pappas would result in a lower false positive rate,” because Petitioner fails to explain how Pappas’s threshold of eight chained gadgets could be applied to

Dalcher, or result in fewer false positives than Dalcher's own method. *Id.* at 48–49.

Third, Patent Owner argues that one of ordinary skill in the art would not have been motivated to combine the teachings of Dalcher and Pappas. *Id.* at 51. First, Patent Owner argues that there is a “significant difference between Dalcher’s *software*-based analysis and Pappas’s ‘[reliance] mainly on *hardware* for instruction-level monitoring.’” *Id.* Second, Patent Owner argues that “Dalcher is directed to monitoring a call stack in process memory, whereas Pappas is directed to monitoring LBR registers in the CPU,” that is, “Dalcher and Pappas are directed to monitoring different things.” *Id.* at 52. Finally, Patent Owner argues that, while “Dalcher and Pappas both arguably analyze a stack, this is as far as their similarities go,” because the two references have “entirely distinct methods and goals, and the mere fact that both mention stack analysis does not mean that a POSITA would have been motivated to apply any teaching from one to the other.” *Id.* at 53–54.

c) *“Petitioner Fails to Show That the Combination of Dalcher and Pappas Renders Obvious Any Independent Claim”*

Patent Owner argues that Petitioner fails to show that the combination of Dalcher and Pappas renders obvious any independent claim. *Id.* at 55–65. First, Patent Owner argues that “Petitioner fails to show that the combination of Dalcher and Pappas renders obvious performing a memory check before an address of an originating caller function is reached (1.e, 9.f, 17.e).” *Id.* at 56. Second, Patent Owner argues that “Petitioner fails to show that the combination of Dalcher and Pappas renders obvious triggering an alert when the memory check detects one of the enumerated elements (1.e, 9.f, 17.e).”

Id. at 60. Third, Patent Owner argues that “Petitioner fails to show that the combination of Dalcher and Pappas renders obvious preventing execution of a payload (1.f, 9.g, 17.f).” *Id.* at 63.

3. *Analysis*

Based on our review of the record, we find that Petitioner does not adequately explain the combination it asserts as its sole ground, and, also, that Petitioner does not show that a POSITA would have been motivated to combine Dalcher with Pappas.

Regarding the explanation of the proposed combination, we agree that the Board tends to reject a “pick-and-choose” approach to the presentation of evidence, which is “challenging to follow and makes it difficult to piece together in order to understand Petitioner’s proposed combination or modification.” *Lyft, Inc. v. Quartz Auto Techs., LLC*, IPR2020-01450, Paper 7 at 17–20 (PTAB Mar. 4, 2021) (denying institution where petitioner “does not articulate adequately how it proposes to modify one of the references or to combine the two references’ teachings”). In view of the directive that a “clear, evidence-supported account of the contemplated workings of the combination is a prerequisite to adequately explaining and supporting a conclusion that a relevant skilled artisan would have been motivated to make the combination and reasonably expect success in doing so,” we find that Petitioner’s case is lacking a clear roadmap of which reference it relies on to disclose which limitation. *Personal Web Techs., LLC v. Apple, Inc.*, 848 F.3d 987, 994 (Fed. Cir. 2017). The Petition is also unclear about how, with respect to any given limitation, the proposed combination discloses that limitation. *ActiveVideo*, 694 F.3d at 1327 (affirming JMOL of no invalidity where the defendant’s expert “failed to

explain how specific references could be combined, which combination(s) of elements in specific references would yield a predictable result, or how any specific combination would operate or read on the asserted claims”). As shown in Patent Owner’s table (Prelim. Resp. 42–44), Petitioner does not provide a clear explanation of its combinations with respect to any limitation of claim 1, and it would be difficult for the Board or Patent Owner to analyze or respond to Petitioner’s case.

Regarding whether Petitioner demonstrates that a POSITA would have been motivated to combine Dalcher and Pappas, we find the Petition insufficient on this count as well. Petitioner does not specify in its limitation-by-limitation analysis of claim 1 how the two references would be combined, and Petitioner also does not specify, for limitations in which both references are cited, why one of ordinary skill in the art would have been motivated to combine them (as in, e.g., limitations 1.b, 1.c, 1.d, 1.e, 1.f). Regarding Petitioner’s general motivation to combine allegations (Pet. 12–13), we disagree that Petitioner’s arguments that “both references relate to” the same field, “both provide stack analysis,” both “provide solutions to the same problem,” and both use a “common exploitation technique” are adequate. These statements merely allege that the two references are in the same field of endeavor, but the same field of endeavor analysis is merely the jumping-off point in reaching the determination of whether a claimed invention is obvious; these statements alone fall short of providing a motivation to combine. *See K-TEC, Inc. v. Vita-Mix Corp.*, 696 F.3d 1364, 1375 (Fed. Cir. 2012) (to qualify as prior art in an obviousness analysis, references must be analogous art—either from the same field of endeavor, or reasonably pertinent to the problem with which the inventor is involved);

Microsoft Corp. v. Enfish, LLC, 662 F. App’x 981 (Fed. Cir. 2016) (unpublished) (insufficient motivation to combine where the only reason given was “that references were directed to the same art or same techniques”).

Regarding Petitioner’s remaining argument about modifying the teachings of Dalcher with Pappas to “enhance the tracking of calls, jumps, and returns and to reduce false positives in compromise detection rates” (Pet. 13), we do not find this substantive or supported enough to carry Petitioner’s burden on motivation to combine. Petitioner and its expert do not provide sufficient reasoning as to why or how one of ordinary skill in the art would need to enhance the tracking of calls, jumps, and returns in Dalcher’s system. *Id.*; Ex. 1002 ¶¶ 185–189. Petitioner and its expert also do not explain how applying Pappas’s alleged teachings about lowering false positives rates would result in fewer false positives in Dalcher’s method, or why one of ordinary skill in the art would have been motivated to apply Pappas’s teachings when, purportedly, “doing so would serve no purpose in Dalcher’s method and would not, in fact, reduce false positives.” Prelim. Resp. 50 (citing Ex. 2001 ¶¶ 78–81); Pet. 13.

In sum, we find that Petitioner does not adequately provide an explanation of which reference is meant to map to each of the limitations of claim 1, or an explanation of how Dalcher and Pappas might be combined. Moreover, there is uncertainty about Petitioner’s proffered motivation to combine Dalcher and Pappas, particularly, how Pappas’s tracking of calls, jumps, and returns and its false positive detection technique might be beneficial in a system like that of Dalcher. We have also reviewed Petitioner’s arguments for claims 2–18, but find that those arguments do not

cure the shortcomings identified. For these reasons, we find Petitioner fails to establish on this record a reasonable likelihood that Petitioner can prove the challenged claims unpatentable in view of Dalcher and Pappas.

III. CONCLUSION

For the reasons set forth above, we determine that Petitioner has not demonstrated a reasonable likelihood of prevailing with respect to at least one challenged claim of the '591 patent, and, thus, we do not institute *inter partes* review.⁵

IV. ORDER

For the reasons given, it is ORDERED that no *inter partes* review is instituted in this proceeding.

⁵ Because we deny the Petition on the merits, we do not reach Patent Owner's request that we exercise our discretion to deny the Petition based on the state of the related litigation. *See* Prelim. Resp. 3–20.

IPR2023-00692
Patent 10,284,591 B2

FOR PETITIONER:

Robert M. Hansen
Scott B. Amankwatia
THE MARBURY LAW GROUP, PLLC
rhansen@marburylaw.com
samankwatia@marburylaw.com

FOR PATENT OWNER:

Brian Eutermoser
Russell E. Blythe
Mikaela Stone
KING & SPALDING LLP
beutermoser@kslaw.com
rblythe@kslaw.com
mikaela.stone@kslaw.com