EXHIBIT 10

HIGHLY CONFIDENTIAL – ATTORNEY'S EYES ONLY

UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF CALIFORNIA

OAKLAND DIVISION

FINJAN LLC., a Delaware Limited Liability Company,

Plaintiff,

v.

QUALYS INC., a Delaware Corporation,

Defendant.

Case No. 4:18-cv-07229-YGR (TSH)

Hon. Yvonne Gonzalez Rogers

EXPERT REPORT OF MICHAEL GOODRICH, PH.D.

via a communications system or network."[4] Likewise, a POSITA would understand the term "receiver" to be a term of art. This understanding is supported, for example, by the *Microsoft Computer* Dictionary, as I cite above and in the footnotes. Thus, a POSITA would understand that in the context of distributed computing, a plain-and-ordinary meaning of "receiver" is a component that accepts data from another component via a communications system or network.

74. This understanding is further supported by the textbook, *Computer Networks*, by Tanenbaum,[5] which is a widely adopted textbook used in many undergraduate computer science curricula. Tanenbaum uses the term "receiver" without needing to provide to the reader further structural definition for the concepts he is discussing, confirming that the terms are sufficient to convey structure to a POSITA.

75. For example, Tanenbaum writes in the introductory chapter describing networking, "Point-to-point transmission with exactly one sender and exactly one ***receiver*** is sometimes called **unicasting.**" Tanenbaum at 17 [bold-italics added, bold as in the original] (*see* also Tanenbaum 4/e at 20). Tanenbaum also writes, "An allocation problem that occurs at every level is how to keep a fast sender from swamping a slow ***receiver*** with data." Tanenbaum at 34 [emphasis added] (*see* also Tanenbaum 3/e at 21). Further, Tanenbaum writes, "The essential aspect of a connection is that it acts like a tube: the sender pushes objects (bits) in at one end, and the ***receiver*** takes them out at the other end." Tanenbaum at 35 [emphasis added] (*see* also Tanenbaum 3/e at 23). In addition, Tanenbaum writes, "Another issue that arises in the data link layer (and most of the higher layers as well) is how to keep a fast transmitter from drowning a slow ***receiver*** in data." Tanenbaum at 43 (*see* also Tanenbaum 3/e at 30). Moreover, in this same introductory chapter, Tanenbaum writes, "The most practical approach [to connect office and laptop computers to the Internet] is to equip both the office and laptop computers with short-range radio transmitters and ***receivers*** to allow them to talk." Tanenbaum at 70 [emphasis added] (*see* also Tanenbaum 4/e at 58). Tanenbaum illustrates this idea, along with a concept known as "multipath fading" that can occur in such scenarios, in a figure, which I excerpt below:

---

[4] *See*, e.g., *Microsoft Computer Dictionary, 5/e*, 2002. ("**receive** *vb.* To accept data from an external communications system, such as a local area network (LAN) or a telephone line, and store the data as a file.") This definition is unchanged from the third edition (1997).

[5] Tanenbaum, *Computer Networks*, 5/e, Prentice Hall, 2011, 2003 (4/e), 1996 (3/e), 1989 (2/e), 1981 (1/e).

medium and executed by the computer, for receiving an incoming stream of program code." For example, the '408 Patent shows an exemplary architecture in Fig. 2:

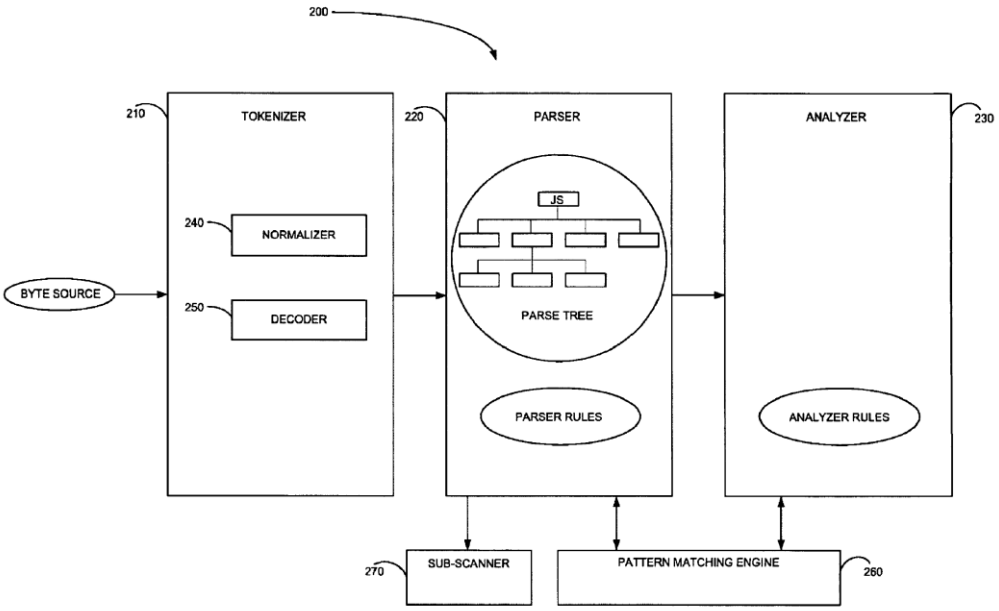

FIG. 2

79. A POSITA would understand that this figure shows a "byte source" being ***received*** by the tokenizer 210, which a POSITA would understand is disclosing a tokenizer as an embodiment of a "receiver," e.g., a component that "accept[s] data from an external communications system, such as a local area network (LAN)."[6] For example, the '408 Patent states, "The function of tokenizer 210 is to recognize and identify constructs, referred to as tokens, within a byte source, such as JavaScript Source code." '408 Patent at 6:51-54. The '408 Patent also states, "Preferably, tokenizer 210 reads bytes sequentially from a content source, and builds up the bytes until it identifies a complete token." '408 Patent at 6:60-62.

80. Further, the '408 also discloses a normalizer 240 as an embodiment that a POSITA would understand to be a "receiver." For instance, the '408 Patent states, "In accordance with a preferred embodiment of the present invention, normalizer 240 translates a raw input stream into a

[6] *See, e.g., See,* e.g., *Microsoft Computer Dictionary, 5/e,* 2002. ("**receive** *vb.* To accept data from an external communications system, such as a local area network (LAN) or a telephone line, and store the data as a file.") This definition is unchanged from the third edition (1997).

and identifying a scripting virus and reasonably identifiable polymorphs of the scripting virus by representing the scripting virus in a ***language independent form***." Li at 5:45-49 (emphasis added). Further, Li describes its solutions as "resulting in a very flexible virus signature." Li at 8:6-7.

101. A POSITA would recognize that the systems respectively described in Li and Zurko are incompatible, and that there would be no expectation of success in an alleged combination of their teachings. For example, a POSITA would understand that the approach of Li teaches away from the use of a hierarchical structure, such as a DOM tree, which is an essential component in the system of Zurko, given Zurko's reliance on comparing DOM trees for its functionality (*see, e.g.*, Zurko at Fig. 4, 0027, 0033, 0038, 0040). Li instead relies on a "flattened" linearized form. As noted above, the Parties agree that "parse tree" should be construed as "a hierarchical structure of interconnected nodes built from scanned content." The tokenized source code in Li is input to the threadizor which eliminates "noise" from the tokenized source code 204' based on a dictionary of key actions, it converts the tokens to a language-independent representation, and it "***flattens***" the function-calling representation of key actions into a ***linearized*** form, or executing thread. Li at 6:41-49. A POSITA would understand that this step does not produce a parse tree, due to the "flattening" and linearization actions,[8] and instead is incompatible with a hierarchical approach, as taught in Zurko. A POSITA would understand that the linear nature of this from is an essential feature in Li, as it allows for the patterns of key actions to be identified and matched. See, e.g., Li at 5:42-61, 6:13-61, 8:46-56, 9:12-27, 10:46-56, Figs. 7 and 11.

102. Further, the flattened linearized form used in Li discards as "noise" tokens that are not key actions; that is, Li discards tokens corresponding to punctuation, variables, and user-defined functions. *See, e.g.*, Li at Figs. 5A, 5B, 6, 7, 9A, 9B, 10, 11, and at 6:50-61, 7:52-59, 8:40-56, 9:4-11 ("leaving only a tokenized skeleton of the original scripting source code"), 10:41-45.

---

[8] *See, e.g., Microsoft Computer Dictionary, Fifth Edition* (2002) (FINJAN-QUALYS 770349-354). ("**linear structure** *n.* A structure in which items are organized according to strict rules of precedence. In a linear structure, two conditions apply: if X precedes Y and Y precedes Z, then X precedes Z; and if X precedes Y and X precedes Z, then either Y precedes Z or Z precedes Y.") Contrast: ("**hierarchy** *n.* A type of organization that, like a tree, branches into more specific units, each of which is "owned" by the higher-level unit immediately above. Hierarchies are characteristic of several aspects of computing because they provide organizational frameworks that can reflect logical links, or relationships, between separate records, files, or pieces of equipment. For example, hierarchies are used in organizing related files on a disk, related records in a database, and related (interconnected) devices on a network. In applications such as

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.