NOTE: This disposition is nonprecedential.

United States Court of Appeals for the Federal Circuit

PALO ALTO NETWORKS, INC.,
Appellant

v.

FINJAN, INC.,
Appellee

2017-2059

Appeal from the United States Patent and Trademark Office, Patent Trial and Appeal Board in Nos. IPR2015-02001, IPR2016-00157, IPR2016-00955, IPR2016-00956.

Decided: September 19, 2018

ORION ARMON, Cooley LLP, Broomfield, CO, argued for appellant.

PAUL J. ANDRE, Kramer Levin Naftalis & Frankel LLP, Menlo Park, CA, argued for appellee. Also represented by JAMES R. HANNAH.

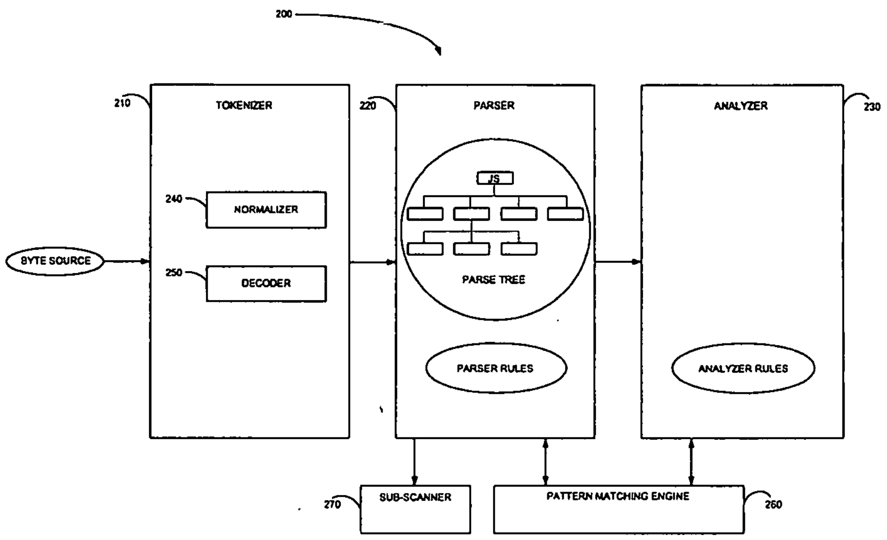
Before REYNA, SCHALL, and STOLL, *Circuit Judges.*

STOLL, *Circuit Judge.*

Appellant Palo Alto Networks, Inc. petitioned for two inter partes reviews of Appellee Finjan, Inc.'s U.S. Patent No. 8,225,408, alleging that certain claims were unpatentable as obvious. The Patent Trial and Appeal Board of the U.S. Patent and Trademark Office ("Board") found that there was insufficient evidence that Palo Alto Networks's proposed prior art combinations would have taught the "dynamically building" claim limitation. *Palo Alto Networks, Inc.*, No. IPR2015-02001, 2017 WL 1052502, at *4–10 (P.T.A.B. Mar. 17, 2017) ("*Board Decision*"). Therefore, the Board found that Palo Alto Networks failed to carry its burden of demonstrating, by a preponderance of the evidence, that any of the challenged claims would have been obvious. *Id.* Palo Alto Networks appeals. We affirm.

I

Finjan's '408 patent relates to methods and systems for detecting malware in data streamed from a network onto a computer. The patent relates to network security, including scanning code to determine whether there are potential viruses in the code. The patent describes a scanner system that preferably uses generic architecture, is language-independent, and is customized for a specific language by using a set of language-specific rules. The '408 patent explains that this adaptive rule-based scanner has three components (illustrated in Figure 2, below). Tokenizer 210 recognizes and identifies constructs (i.e., "tokens") within a byte source code. For example, code between { } or [ ] would become a token. Parser 220 controls the process of scanning incoming content, preferably by building a parse tree data structure that represents the incoming content. Finally, analyzer 230 checks for malware by searching for specific patterns of content that indicate malware.



’408 patent, Fig. 2.

Claims 1, 3–7, 9, 12–16, 18–23, 29, and 35 are at issue in this appeal, and independent claim 1 is illustrative:

> 1. A computer processor-based multi-lingual method for scanning incoming program code, comprising:
>
> receiving, by a computer, an incoming stream of program code;
>
> determining, by the computer, any specific one of a plurality of programming languages in which the incoming stream is written;
>
> instantiating, by the computer, a scanner for the specific programming language, in response to said determining, the scanner comprising parser rules and analyzer rules for the specific programming language, wherein the parser rules define certain patterns in terms of tokens, tokens being lexical constructs for the specific programming language, and wherein the analyzer rules identify

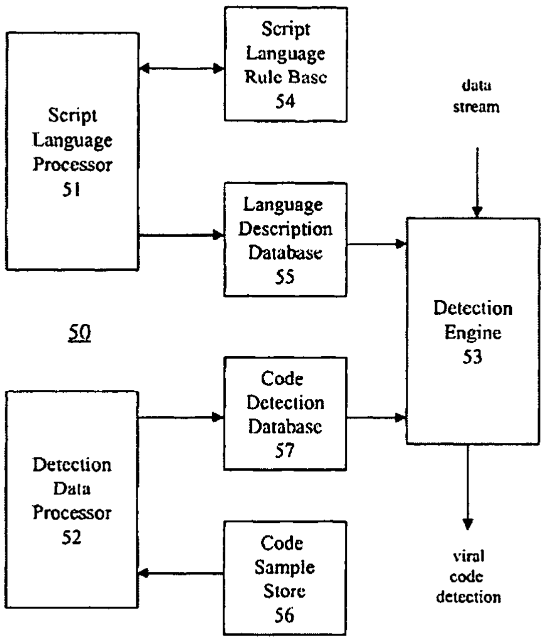
> certain combinations of tokens and patterns as being indicators of potential exploits, exploits being portions of program code that are malicious;
>
> identifying, by the computer, individual tokens within the incoming stream;
>
> *dynamically building*, by the computer while said receiving receives the incoming stream, a parse tree whose nodes represent tokens and patterns in accordance with the parser rules;
>
> dynamically detecting, by the computer while said dynamically building builds the parse tree, combinations of nodes in the parse tree which are indicators of potential exploits, based on the analyzer rules; and
>
> indicating, by the computer, the presence of potential exploits within the incoming stream, based on said dynamically detecting.

Id. claim 1 (emphasis added to highlight the disputed claim limitation). We focus on the claim limitation requiring "dynamically building" a parse tree, which is common to all the challenged claims. The Board construed "dynamically building" to mean: "requires that a time period for dynamically building overlap with a time period during which the incoming stream is being received." *Board Decision*, 2017 WL 1052502, at *3. This unopposed claim construction was proposed by Palo Alto Networks based on the plain claim language, which requires "dynamically building, by the computer while said receiving receives the incoming stream." '408 patent, claim 1.

II

Palo Alto Networks asserted that claims 1, 3–5, 9, 12–16, 18, 19, 22, 23, 29, and 35 of the '408 patent would have been obvious over U.S. Patent No. 7,636,945 ("Chandnani") and U.S. Patent No. 5,860,011 ("Kolawa") under 35 U.S.C. § 103. Palo Alto Networks also asserted that the same claims would have been obvious over Chandnani, Kolawa, and U.S. Patent No. 7,284,274 ("Walls").

Chandnani teaches a method of detecting malware in a data stream, including determining the programming language of the data stream and detecting viral code. Figure 2 from Chandnani (duplicated below) illustrates Chandnani's script language virus detection apparatus, including detection engine 53, one of the focal points of Chandnani's method:


Script Language Processor 51
Script Language Rule Base 54
data stream
Language Description Database 55
Detection Engine 53
50
Code Detection Database 57
Detection Data Processor 52
Code Sample Store 56
viral code detection

Chandnani, Fig. 2, col. 8 ll. 5–7. Detection engine 53 tokenizes the incoming data stream by breaking it into smaller pieces known as tokens. As part of that process, it receives the language check data from the language description module 55, as indicated in step 31 of Figure 6:

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.