Trials@uspto.gov
571-272-7822

Paper 14
Entered: January 23, 2017

UNITED STATES PATENT AND TRADEMARK OFFICE

---

BEFORE THE PATENT TRIAL AND APPEAL BOARD

---

BLUE COAT SYSTEMS, INC.,
Petitioner,

v.

FINJAN, INC.,
Patent Owner.

---

Case IPR2016-01441
Patent 8,225,408 B2

---

Before JAMES B. ARPIN, PATRICK M. BOUCHER, and ZHENYU YANG, *Administrative Patent Judges*.

BOUCHER, *Administrative Patent Judge*.

# DECISION
Denying Institution of *Inter Partes* Review
*37 C.F.R. § 42.108*

On July 15, 2016, Blue Coat Systems, Inc. ("Petitioner") filed a Petition (Paper 2, "Pet.") pursuant to 35 U.S.C. §§ 311–319 to institute an *inter partes* review of claims 2, 8, 11, 24–28, and 30–33 of U.S. Patent No.

8,225,408 (Ex. 1001, "the '408 patent"). Finjan, Inc. ("Patent Owner") filed a Preliminary Response (Paper 6, "Prelim. Resp.") on November 18, 2016. Pursuant to our authorization, Petitioner filed a Reply (Paper 12, "Reply"), and Patent Owner filed a Sur-reply (Paper 11, "Sur-reply"), limited to addressing certain issues identified below.

Based on the particular circumstances of this case, we exercise our discretion under 35 U.S.C. §§ 314(a) and 325(d) and do not institute an *inter partes* review of the challenged claims.

## I. BACKGROUND

### *A. The '408 Patent*

The '408 patent relates to network security, including scanning content that includes "mobile code" to produce a diagnostic analysis of potential exploits, such as viruses, within the code. Ex. 1001, col. 1, ll. 19–20, col. 1, ll. 59–64. Figure 2 of the '408 patent is reproduced below.

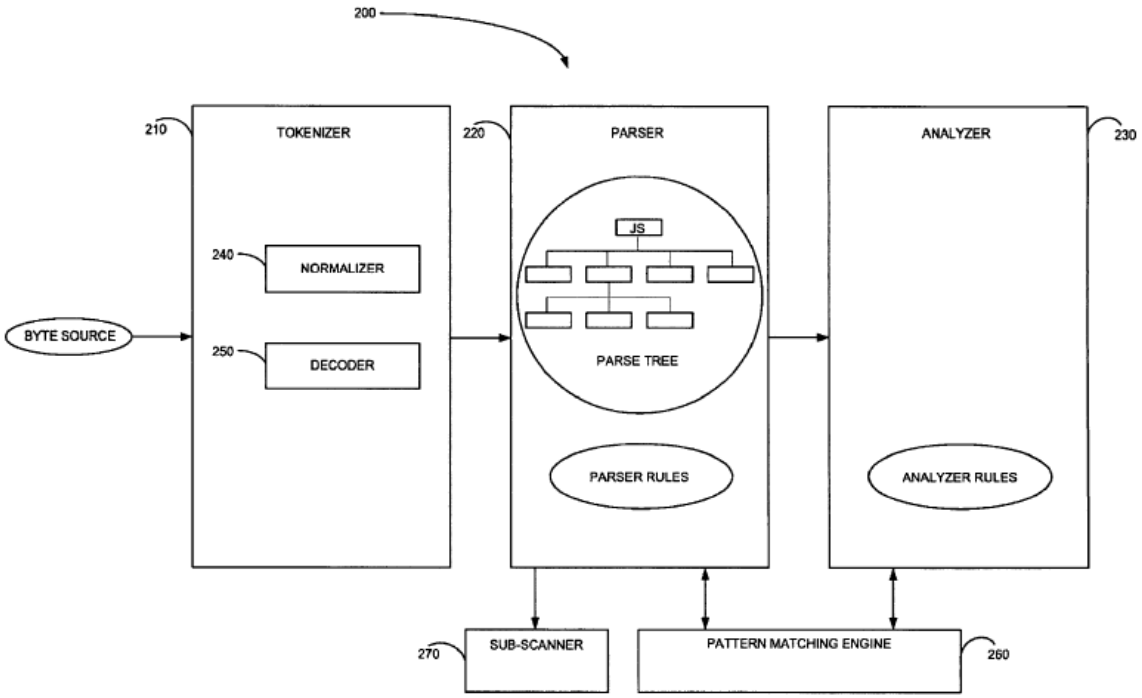


FIG. 2

Figure 2 provides a simplified block diagram of an adaptive rule-based content scanner system. *Id.* at col. 6, ll. 14–17.

The ’408 patent explains that the adaptive rule-based scanner of Figure 2 “is preferably designed as a generic architecture that is language-independent, and is customized for a specific language through use of a set of language-specific rules.” *Id.* at col. 6, ll. 17–20. In addition, “security violations, referred to as exploits, are described using a generic syntax, which is also language-independent.” *Id.* at col. 6, ll. 28–30. Adaptive rule-based scanner 200 includes three main components: (1) tokenizer 210, which recognizes and identifies constructs (i.e., “tokens”) within a byte source code; (2) parser 220, which controls the process of scanning incoming content, such as with a parse tree data structure that represents the incoming content; and (3) analyzer 230, which checks for exploits by searching for specific patterns of content that indicate an exploit. *Id.* at

col. 6, ll. 50–54, col. 8, ll. 18–27, col. 9, ll. 19–22. Sub-scanner 270 is another adaptive rule-based scanner used to scan a subsection of input being processed by scanner 200. *Id.* at col. 9, ll. 7–8. Pattern matching engine 260 performs pattern matching for both parser 220 and analyzer 230, such as by accepting an input list of regular-expression elements describing a pattern of interest and an input list of nodes from the parse tree to be matched against the pattern of interest, and outputting a Boolean flag indicating whether a pattern is matched. *Id.* at col. 9, ll. 44–58.

Using a "scanner factory," such adaptive rule-based scanners may be produced "on demand" for different types of content. *Id.* at col. 15, ll. 15–16. The scanner factory "instantiates" a scanner repository, which produces a single instance of multiple scanners, such as "a scanner for HTML content, a scanner for JavaScript content, and a scanner for URI content," each "able to initialize itself and populate itself with the requisite data." *Id.* at col. 15, ll. 34–41. When content is downloaded, a pool of thread objects is created and stores the scanner-factory instance as member data. *Id.* at col. 15, ll. 53–55. When a thread object has content to parse, it requests an appropriate scanner from its scanner-factory object; when the thread finishes scanning the content, it returns the scanner instance to its scanner factory, "to enable pooling the [adaptive rule-based] scanner for later re-use." *Id.* at col. 15, ll. 56–63.

### *B. Illustrative Claim*

All of the challenged claims are dependent claims. Independent claim 1, from which challenged claims 2 and 8 depend, is illustrative of the claims at issue and is reproduced below.

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*. at col. 19, l. 45–col. 20, l. 7.

### *C. References*

Petitioner relies on the following references. Pet. 3–9.

| | | | |
|---|---|---|---|
| Chandnani | US 2002/0073330 A1 | June 13, 2002 | Ex. 1007 |
| Kolawa | US 5,860,011 | Jan. 12, 1999 | Ex. 1008 |
| Huang | US 6,968,539 B1 | Nov. 22, 2005 | Ex. 1010 |
| Walls | US 7,284,274 B1 | Oct. 16, 2007 | Ex. 1011 |

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.