IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

_____

BEFORE THE PATENT TRIAL AND APPEAL BOARD

_____

NETSKOPE, INC.,

Petitioner,

v.

FORTINET, INC.,

Patent Owner.

_____

PTAB Case No. IPR2026-00515
U.S. Patent No. 12,126,695

_____

PETITION FOR *INTER PARTES* REVIEW OF U.S. PATENT NO. 12,126,695

TABLE OF CONTENTS

EXHIBIT LIST v

I. INTRODUCTION 1

II. MANDATORY NOTICES 2

A. Real party in interest 2

B. Related matters 2

C. Counsel and service information 2

D. Fees 3

III. IPR REQUIREMENTS 3

A. Standing 3

B. Challenged claims and relief requested 3

1. Prior art 4

2. Grounds 4

3. No Discretionary Denial or 325(d) Factors Are Present 5

IV. TECHNOLOGY DESCRIPTION 6

A. Threat Intelligence 6

B. Cloud Computing 7

C. Cloud Deployment Configuration Settings 8

D. Infrastructure as Code (IaC) 9

V. The ’695 Patent 10

A. Alleged invention 10

B. Prosecution history 12

C. Priority 12

VI. LEVEL OF ORDINARY SKILL IN THE ART 12

VII. CLAIM CONSTRUCTION 12

VIII. GROUNDS 13

A. GROUND 1: CHERNIN RENDERS CLAIMS 1-3, 5, 8-11, 13, 16-19 OBVIOUS 13

1. Overview of Chernin 13

2. Claims 1, 9, and 17 15

1[pre]/9[pre]/17[pre] 15

3. Claims 2/10/18 26

2[pre]/ 10[pre]/18[pre] 26

2[a]/10[a]/18[a] 26

4. Claims 3/11/19 27

3[pre]/11[pre]/19[pre] 27

3[a]/11[a]/19[a] 27

5. Claims 5/13 28

5[pre]/13[pre] 28

5[a]/13[a] 28

5[b]/13[b] 29

6. Claims 8/16 30

8[pre]/ 16[pre] 30

8[a]/16[a] 30

B. GROUND 2: THE COMBINATION OF CHERNIN AND JEUK RENDERS CLAIMS 1-3, 5, 8-11, 13, 16-19 OBVIOUS.....31

1. Overview of Jeuk 31

2. Jeuk's Role in Parent's Prosecution.....33

3. Motivation to Combine Chernin and Jeuk. 35

4. Claims 1, 9, and 17.....38

1[pre]/9[pre]/17[pre] 38

1[a]/9[a]/17[a] 40

1[b]/9[b]/17[b] 41

1[c]/9[c]/17[c] 41

1[d]/9[d]/17[d] 41

5. Claims 2/3/5/10/11/13/18/19 42

6. Claims 8/16 42

8[pre]/16[pre] 42

8[a]/16[a].....43

C. GROUND 3: THE COMBINATION OF HUDIS AND SHANKS RENDERS CLAIMS 1-3, 5, 8-11, 13, 16-19 OBVIOUS. 43

1. Overview of Hudis 43

2. Overview of Shanks 47

3. Motivation to Combine Hudis and Shanks 52

4. Claims 1, 9, and 17.....55

1[pre]/9[pre]/17[pre] 55

1[a]/9[a]/17[a] 62

1[b]/9[b]/17[b] 64

1[c]/9[c]/17[c] 67

1[d]/9[d]/17[d] 70

5. Claims 2/10/18 72

2[pre]/10[pre]/18[pre] 72

2[a]/10[a]/18[a] 72

6. Claims 3/11/19 73

3[pre]/11[pre]/19[pre] 73

3[a]/11[a]/19[a] 73

7. Claims 5/13 74

5[pre]/13[pre] 74

5[a]/13[a] 74

5[b]/13[b] 76

8. Claims 8/16 77

8[pre]/16[pre] 77

8[a]/16[a] 77

IX. CONCLUSION 78

EXHIBIT LIST

Exhibit No.	Description
1001	United States Patent No. 12,126,695 ("the '695 Patent")
1002	Prosecution History of U.S. Patent No. 12,126,695 ("the '695 Prosecution History")
1003	Declaration of Dr. Michael Franz
1004	Curriculum Vitae of Dr. Michael Franz
1005	Prosecution History of U.S. Patent No. 11,785,104 ("the '104 Prosecution History")
1006	US 2016/0366174A1 ("Chernin")
1007	US 2019/0123973A1 ("Jeuk")
1008	US 2017/0207980A1 ("Hudis")
1009	US 2018/0063085A1 ("Shanks")
1010	Cloud Computing: Benefits and Risks of Moving Federal IT into the Cloud, July 01, 2010
1011	Cloud Computing Adoption Continues Accelerating In the Enterprise, November 22, 2014
1012	Fortinet and Palo Alto Networks Co-Found the Industrie's First Cyber Defense Consortium, May 30, 2014
1013	The Mitre Corporation, STIX Language_ Version 1.0 (Official), 2012
1014	The Mitre Corporation, TAXII Services Specification, Version 1.0 (draft) by Mark Davidson and Charles Schmidt, Nov. 16, 2012
1015	NIST Special Publication 800-150, Guide to Cyber Threat Information Sharing, 2016

1016	Ex. D to 2026-01-22 Fortinet's Disclosure of Infringement Contentions
1017	OASIS Advances Automated Cyber Threat Intelligence Sharing with STIX, TAXII, CybOX, July 16, 2015

I. INTRODUCTION

Issued in 2024, the ’695 patent is directed to “learning from other cloud deployments to combat security threats.” ’695, cl. 1. But sharing threat intelligence information across cloud environments was standard practice long before the ’695 patent’s 2017 priority claim. Back in 2012, MITRE developed the Structured Threat Information eXpression (STIX) and Trusted Automated eXchange of Intelligence Information (TAXII) standards for machine-readable threat sharing. Additionally, in 2014, companies like MacAfee, Palo Alto Networks, Symantec, and Fortinet formed the Cyber Threat Alliance to enable the kind of cross-organization defensive learning the ’695 patent now claims to have invented.

Unsurprisingly, there is ample prior art about sharing threat intelligence from before the ’695 patent. This petition rests on three such prior art references and confirms that the challenged claims should have never issued. For starters, Chernin—assigned to Soltra, a later contributor to STIX/TAXII—alone renders obvious every claim (Ground 1). But even if it does not, combining it with Jeuk independently renders the claims obvious (Ground 2). Indeed, during prosecution of the ’695 patent’s parent application, the examiner found that Jeuk disclosed the same structural framework recited in the challenged claims—identifying cloud deployments for cross-customer learning, receiving information from those deployments, and identifying configurations to adopt. The applicant overcame Jeuk

only by adding limitations that are absent from the challenged claims. Finally, the combination of Hudis and Shanks provides yet another basis to find the challenged claims obvious (Ground 3).

No other factors weigh against institution: the patent is only two years old, no trial date has been set in the parallel district court proceeding, and none of the primary references was a basis for rejecting the claims. In fact, the challenged claims breezed through prosecution without a single office action. Accordingly, Netskope, Inc. ("Petitioner") respectfully requests *inter partes* review of claims 1-3, 5, 8-11, 13, and 16-19 of the '695 patent and cancellation of the challenged claims.

II. MANDATORY NOTICES

A. Real party in interest

Netskope, Inc. is the sole real party-in-interest.

B. Related matters

Patentee originally asserted the '695 Patent against Petitioner in the United States District Court for the Northern District of California, Case No. NDCA-4-25-cv-02360-HSG. That claim has now been severed into Case No. 4:26-cv-08886-HSG.

C. Counsel and service information

Lead Counsel: Thomas N. Millikan (Reg. No. 72,316).

Back-up counsel: Joseph P. Reid (Reg. No. 72,317) and Andrew Klein (*pro hac vice* forthcoming).

These attorneys can be reached by mail at Sheppard, Mullin, Richter & Hampton LLP, 12275 El Camino Real, Suite 100, San Diego, California, 92130, by phone at (858) 720-8900, and by fax at (858) 509-3691.

Petitioner consents to electronic service. All services and communications to the attorneys listed above may be sent to:

LegalTm-NetskopeFortinetII-IPRs@sheppard.com.

A Power of Attorney for Petitioner is being filed concurrently.

D. Fees

Petitioner paid the required fees. The Office is authorized to charge any fee deficiency, or credit any overpayment, to Deposit Account No. 50-4561.

III. IPR REQUIREMENTS

A. Standing

Petitioner certifies the '695 patent is available for IPR and that Petitioner is not barred or estopped from challenging the claims.

B. Challenged claims and relief requested

Petitioner respectfully requests cancellation of claims 1-3, 5, 8-11, 13, and 16-19 under §103.

1. Prior art

Petitioner relies on the references listed in the Table of Exhibits, including:

- U.S. Patent Application Publication No. 2016/0366174 A1 ("Chernin")
- U.S. Patent Application Publication No. 2019/0123973 A1 ("Jeuk"),
- U.S. Patent Application Publication No. 2017/0207980 A1 ("Hudis")

  U.S. Patent Application Publication No. 2018/0063085 A1 ("Shanks").

All references are prior art to the '695 patent, the earliest claimed priority date of which is November 27, 2017:[1] Chernin published December 15, 2016; Jeuk was filed October 24, 2017; Hudis published July 20, 2017; and Shanks was filed August 23, 2016.

Petitioner also relies on the declaration of Dr. Michael Franz (Ex. 1003), a Distinguished Professor of Computer Science at the University of California, Irvine, specializing in software systems, computer security, and trustworthy computing.

2. Grounds

Petitioner requests *inter partes* review and cancelation of the challenged claims on the following grounds:

[1] Petitioner does not concede that the '695 patent is entitled to this or any other claimed priority date, and reserves the right to challenge any such priority date.

Ground	Basis	Reference(s)	Challenged Claims
1	§103	Chernin	1-3, 5, 8-11, 13, and 16-19
2	§103	Chernin and Jeuk	1-3, 5, 8-11, 13, and 16-19
3	§103	Hudis and Shanks	1-3, 5, 8-11, 13, and 16-19

3. No Discretionary Denial or 325(d) Factors Are Present

No discretionary denial or 325(d) factors weigh against review. Specifically:

- The '695 Patent issued only 2 years ago (October 22, 2024) and likely has a priority date no earlier than October 18, 2021 due to new matter in a continuation-in-part application.
- No schedule is set, including no trial date, in the parallel district court proceeding.
- Petitioner has not taken inconsistent positions on claim construction—claim construction proceedings have not begun in the parallel district court proceeding.
- Netskope is a U.S.-based cloud security company.
- None of Chernin, Hudis, or Shanks was before the Examiner during prosecution. Additionally, while Jeuk is cited on the face of the patent, it was not a basis for rejection and it was not presented in combination with Chernin, which discloses the security aspect of the challenged claims. Thus, the combination presents a new and distinct invalidity challenge.

IV. TECHNOLOGY DESCRIPTION

A. Threat Intelligence

Before 2017, cyber threats were an important problem and significant resources were devoted to mitigating them. Cloud deployments were particularly attractive targets, since a vulnerability in a shared resource might make everyone using that resource vulnerable. Franz, ¶¶48-49.

Defensive approaches using shared threat intelligence were well-known before 2017. For example, in 2014, companies in cyber defense (Fortinet and Palo Alto Networks) formed the Cyber Threat Alliance, an organization with the goal of "deep collaboration on threat intelligence and sharing of prevention measures." Ex. 1012. Avoiding configuration settings that had led to exploitation, and adopting settings that had thwarted exploit attempts, would naturally help improve the defensive posture of multiple participants in a threat intelligence sharing network. Franz, ¶49.

Around this same period, MITRE released the first specification drafts (in 2012) for two complementary threat-sharing standards: STIX, a machine-readable language for describing cyber threat data, and TAXII, a REST/HTTPS-based protocol for transporting that data. Exs. 1014, 4-15; Ex. 1013. By 2015, OASIS was leading further development of both standards with participation from Boeing,

Cisco, Dell, Fortinet, IBM, Palo Alto Networks, Soltra (Chernin's assignee), the Department of Homeland Security, and NIST, among others. Ex. 1017; Franz, ¶50.

B. Cloud Computing

In general terms, cloud computing describes the on-demand delivery of computing resources and services—servers, storage, databases, networking, and software—delivered over a network. Franz, ¶36.

NIST's 2010 publication confirms that the underlying concepts of cloud computing were not new but had matured into a well-established industry by that time:

> [T]he ideas behind cloud computing reach back through decades. In the early 1960s, researchers proposed the idea of computing as a utility, similar to other services such as gas or electricity. . . By the 2000s, the term cloud computing was being used to describe computing services delivered over a network, and, in 2010, a substantial and growing number of vendors are developing cloud computing offerings for government, industry, and the general public.

Ex. 1010, pg. 2; Franz, ¶37. Cloud computing adoption continued to accelerate; by 2014, an IDG study found that "69% of enterprises have either applications or infrastructure running in the cloud...." Ex. 1011; Franz, ¶38.

By 2017, "cloud computing" was well established—even to lay persons—and had differentiated into a range of service models:

- Infrastructure as a Service (IaaS): virtual servers, storage, and networking on demand, eliminating the need for physical data centers. Examples include Amazon AWS, Google Cloud, and Microsoft Azure.
- Platform as a Service (PaaS): managed runtimes and application platforms enabling developers to build, test, and deploy applications without managing underlying services or infrastructure. Examples include Google AppEngine and Azure App Services.
- Software as a Service (SaaS): complete software applications hosted and delivered over the web, such as Microsoft's Office 365.

These models can be deployed on public clouds (e.g., AWS, Azure), on private clouds in which all resources are dedicated to a single tenant, or hybrid clouds combining the two. By 2017, a POSITA would have known the major cloud service models, providers, and the distinction between public, private, and hybrid clouds. Franz, ¶¶39-41.

C. Cloud Deployment Configuration Settings

Cloud offerings (including those available in 2017) are typically designed to be highly flexible and have many user-configurable parameters. A set of parameters customizing a cloud deployment is often called a configuration setting, stored in a configuration file. Franz, ¶42.

Two approaches to cloud configuration are imperative (step-by-step commands executed in sequence) and declarative (the desired end state is described statically, and the platform determines how to reach it). By the time of the alleged invention, declarative configuration was dominant. HashiCorp's TerraForm (cited in the '695 Patent at 60:36-39) was a widely known provider-agnostic tool for managing cloud deployment configurations in 2017. Franz, ¶¶43-44.

D. Infrastructure as Code (IaC)

In the declarative approach, a developer specifies the desired end result and automated tooling does the rest. The declarative specification is essentially a program in a cloud configuration language, enabling developers to treat infrastructure as software. This led to the term "Infrastructure as Code" or "IaC," originating in IaaS to specify provisioning of virtual servers, networks, and storage. TerraForm had its initial open-source release in July 2014. By the time of the alleged invention, IaC had expanded beyond IaaS to encompass declarative configuration of PaaS and SaaS deployments as well. Franz, ¶¶45-46.

A POSITA would have understood that a declarative configuration setting is a program in a special language. To adjust a deployed cloud, a developer changes the declarative description, and the tools automatically determine the migration

steps. The underlying infrastructure thereby becomes almost as malleable as software. Franz, ¶47.

V. The '695 Patent

A. Alleged invention

The '695 patent is a continuation-in-part of an earlier chain of provisional and non-provisional applications. Much of the specification addresses matters outside the scope of the challenged claims. Columns 69-84 and Figures 5-12 are most relevant to the challenged claims. Franz, ¶30.

At a high level, the challenged claims recite a method for one cloud deployment to combat security threats by learning from another cloud deployment. For instance, a cloud deployment can receive configuration settings used by other deployments to combat a security threat and identify ones that are worth adopting. The disclosed examples of adoptable settings include IP-address blacklists and network-domain blacklists. Franz, ¶31.

Figure 10 (below) depicts a process flow related to this process. Franz, ¶32.

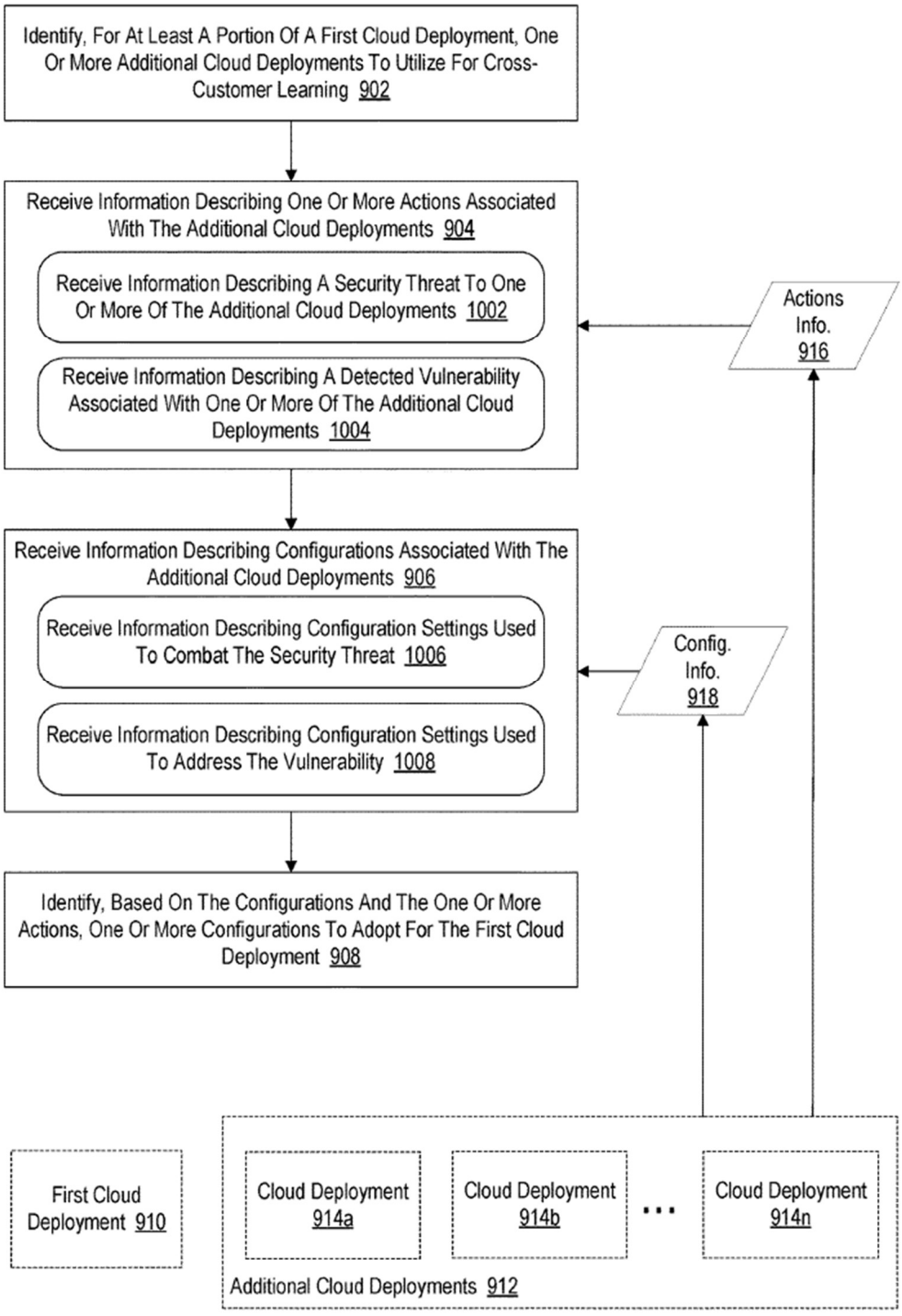

Identify, For At Least A Portion Of A First Cloud Deployment, One Or More Additional Cloud Deployments To Utilize For Cross-Customer Learning 902
Receive Information Describing One Or More Actions Associated With The Additional Cloud Deployments 904
Receive Information Describing A Security Threat To One Or More Of The Additional Cloud Deployments 1002
Receive Information Describing A Detected Vulnerability Associated With One Or More Of The Additional Cloud Deployments 1004
Actions Info. 916
Receive Information Describing Configurations Associated With The Additional Cloud Deployments 906
Receive Information Describing Configuration Settings Used To Combat The Security Threat 1006
Receive Information Describing Configuration Settings Used To Address The Vulnerability 1008
Config. Info. 918
Identify, Based On The Configurations And The One Or More Actions, One Or More Configurations To Adopt For The First Cloud Deployment 908
First Cloud Deployment 910
Cloud Deployment 914a
Cloud Deployment 914b
Cloud Deployment 914n
Additional Cloud Deployments 912

Fig. 10

B. Prosecution history

The ’695 patent breezed through prosecution without an office action, requiring only a terminal disclaimer over US Patent No. 11,785,104. (Ex. 1002); Franz, ¶33.

C. Priority

The ’695 patent has an extensive priority chain. Petitioner submits the ’695 patent is not entitled to the benefit of its priority chain because the claims recite subject matter absent from U.S. Application No. 17/504,311 and all applications to which it claims priority. But because Petitioner’s grounds rest on prior art predating the earliest purported priority claim, Petitioner does not address these deficiencies here but reserves the right to do so. Franz, ¶¶27-29.

VI. LEVEL OF ORDINARY SKILL IN THE ART

A POSITA in 2017 would have had at least a bachelor’s degree in computer science, computer engineering, electrical engineering, or a related field, and at least two years of work or research experience in cloud and distributed computing with exposure to security issues. Work experience can substitute for formal education and vice versa. Franz, ¶25.

VII. CLAIM CONSTRUCTION

For this proceeding, Petitioner does not believe claim construction is needed.

VIII. GROUNDS

Claims 1-3, 5, 8-11, 13, and 16-19 of the '695 patent are obvious over Chernin (Ground 1), Chernin and Jeuk (Ground 2), and Hudis and Shanks (Ground 3).

A. GROUND 1: CHERNIN RENDERS CLAIMS 1-3, 5, 8-11, 13, 16-19 OBVIOUS

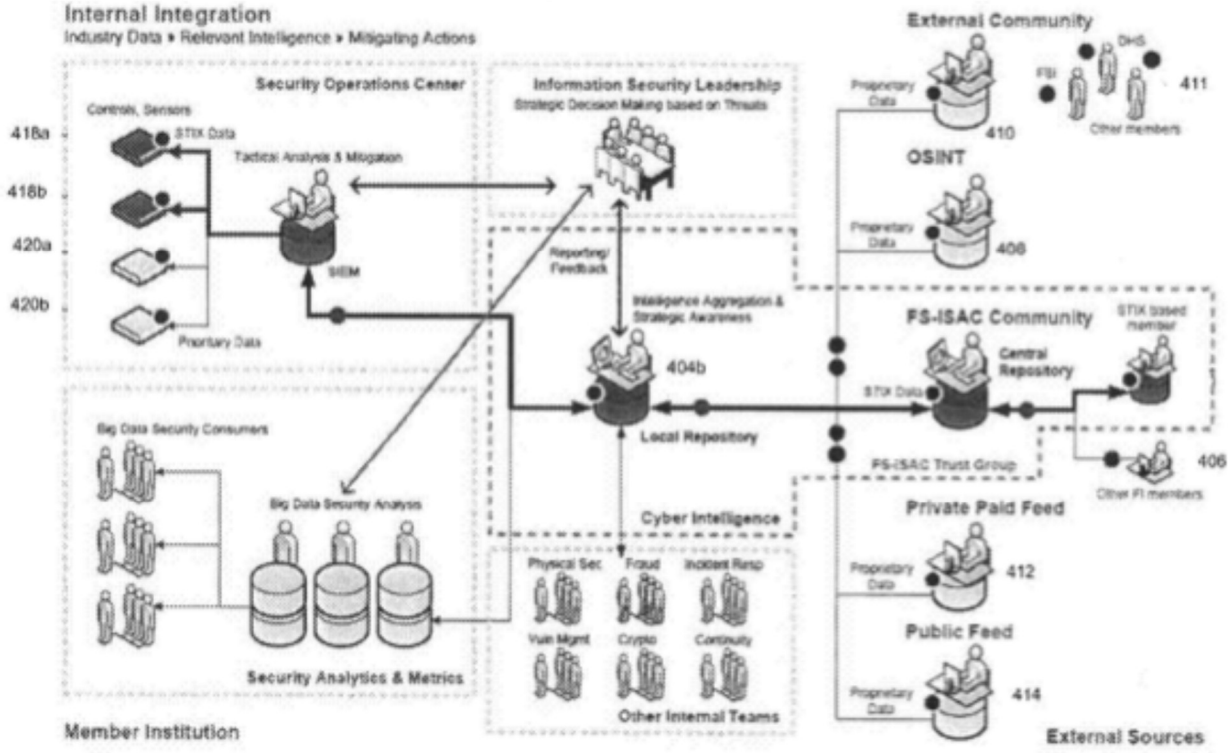
1. Overview of Chernin

Chernin is directed to sharing cyber-threat information among entities in a standardized, secure format. Chernin, Abstract. As Chernin explains, distributing this information helps prevent attackers from reusing successful techniques, because potential targets learn both the methods used and the defenses available to counter them. *Id.* Standardizing the information format further allows participating entities' systems to automatically analyze it and deploy defensive measures, *id.*, and Chernin gives each entity control over which threat information it discloses and to whom. Chernin, [0009]; Franz, ¶56.

For example, Chernin teaches that a detected malicious IP address can be distributed to other users, who can then block it from their networks. Chernin, [0006]. A detected malware filename can be shared so other firms can warn employees not to download it. *Id.* Thus, even when an attack succeeds, sharing information about it assists in detecting and preventing future attempts. *Id.*; Franz, ¶57.

Under Chernin's cross-customer learning model, each participating entity operates its own distributed server with a local threat-information repository. Chernin, [0037]-[0038]. Chernin contemplates two sharing architectures for this purpose: a "hub and spoke" model, where a central repository ingests, aggregates, processes, and redistributes threat data, Chernin, [0034], and a "peer to peer" model, where each entity chooses which peers to learn from directly. Chernin, [0059]-[0060]. In some implementations a directory of threat-information "feeds" helps operators locate feeds suitable for cross-customer learning. Chernin, [0063]; Franz, ¶58.

Chernin discloses threat information in STIX format transmitted via TAXII. Chernin, [0068], [0091]. STIX encodes observable properties such as file information (filename, hash, size), registry key values, services started or changed, HTTP requests, network traffic on specific IP addresses, and communication on certain ports. Chernin, [0071]. STIX also encodes "courses of action (COAs)," representing specific measures to address cyberthreats—either corrective or preventative. Chernin, [0082]. A COA may include an intrusion prevention system rule or automatic patch or remediation. *Id.* Below is an exemplary architecture for sharing threat intelligence. Franz, ¶59.



Chernin, Fig. 4.

2. Claims 1, 9, and 17

1[pre]/9[pre]/17[pre]

Chernin teaches that an entity with a cloud deployment can learn from another entity with a cloud deployment by sharing information about security threats. Abstract ("Computerized systems and methods for sharing identified cyber-threat information in a standardized and secure format. The sharing of cyber-threat information assists in preventing malicious actors from replicating successful cyber-attacks by informing potential targets of the methods employed by the malicious actors, and the defensive measures that those targets should to implement to prevent those methods from succeeding."); *see also id.*, [0002].

Chernin thus recites both a "method" (as in 1[pre]) and a "system" (as in 9[pre]) for sharing threat information. Moreover, Chernin teaches that these systems and methods "can also predict future threats (based on recognized patterns of behavior) and preemptively prescribe and apply appropriate preventative and defensive measures before threats have even been carried out anywhere within the system." Chernin, [0033]; Franz, ¶¶60-61.

For example, Chernin recognizes the growing use of computer systems because of "the recent proliferation of mobile devices, ***cloud computing***, and the so-called 'Internet of things.'" Chernin, [0003]. It recognizes that entities place their data "in the cloud" and face resulting security risks. *Id.*, [0005]. Chernin discloses that sharing cyber-threat information "can act as an 'immune system' for participating entities" by enabling them to be "vaccinated" against threat vectors detected by other members. *Id*., [0006]. Chernin further teaches systems and methods for sharing threat information that:

> are fully automated, and can not only automatically identify, filter, and respond appropriately to threat information received by other entities, but can also predict future threats (based on recognized patterns of behavior) and preemptively prescribe and apply appropriate preventative and defensive measures before threats have even been carried out anywhere within the system.

Id., [0033]; Franz, ¶62.

Figure 1 shows a "hub and spoke" architecture with a central repository 102 hosted on a centralized server and member entities 104a-104d, e.g., "individuals, firms, government agencies, and associations," which are hosted on distributed server(s) and connected to the central repository via a network such as the Internet. *Id*., [0036], [0038]; Franz, ¶63.

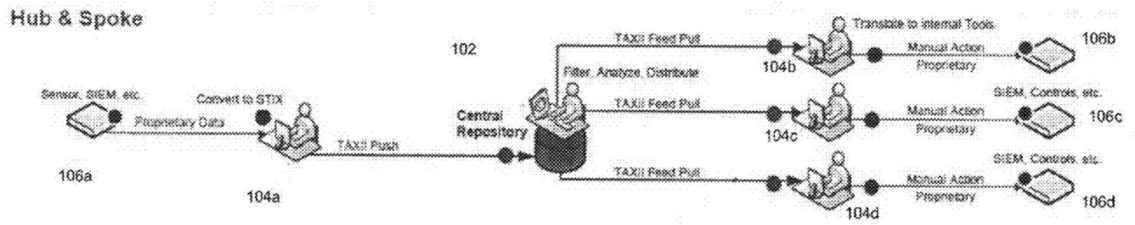


FIG. 1

Id., FIG. 1.

Given these disclosures, including paragraphs [0003], [0005], [0006], [0037] and [0038], a POSITA would have readily found that Chernin's "member entities such as individuals, firms, government agencies and associations, each of which are hosted on one or more distributed servers" are "cloud deployments," since each one is hosted on at least one distributed server. Franz, ¶64.

Chernin teaches that local entities receive threat information from sensors/controls 106a-106d which "can monitor, identify, and locate known and unknown cyberattacks and events in real time," e.g., can detect "intrusions,

anomalies, and vulnerabilities." Chernin, [0038]. Chernin teaches that when these sensors and controls observe an event or property, they "generate one or more items of threat information describing that event or property in a machine-readable customized language used by that particular local entity." *Id*., [0039]. The local entities then "translate the items of threat information from the machine-readable, customized language into a common language used by the system 100 to describe items of threat information" and transmit the items to centralized repository 102. *Id*., [0040]. The centralized repository receives, stores, and analyzes the items of threat information, and the items of threat information are distributed directly ("hub-and-spoke") or indirectly ("peer-to-peer") among the local entities 104a-104d for use in protecting against threats in a manner such as elaborated with reference to other elements of claim 1. *Id*., [0046]-[0048]. This corresponds to "learning from other cloud deployments to combat security threats." Franz, ¶65.

Chernin discloses performing the relevant operations as part of a "method" as in 1[pre], and a "system" as in 9[pre]. *See, e.g.*, Chernin, Abstract; Franz, ¶66.

Chernin discloses that the central server comprises "at least one multi-core processor" hosting a central repository that synchronizes with each of the distributed servers by exchanging threat information. Chernin, [0023]. A POSITA would have recognized that in order for a processor to perform the listed tasks of "hosting," "synchronizing," and "exchanging," it would be running a program with

instructions. Given this, a POSITA would have readily recognized that Chernin teaches a "computer program product." Franz, ¶67.

1[a]/9[a]/17[a]

In Chernin, "entities such as individuals, firms, government agencies and associations, each of which are hosted on one or more distributed servers" correspond to "cloud deployments". One entity corresponds to a "first cloud deployment" while others correspond to "additional cloud deployments." Chernin discloses several mechanisms by which entities can choose who they exchange information with, including the ability to control "which ***other entities they share them with***." Chernin, [0009]; Franz, ¶¶70-71.

Threat data can be associated with a quality metric that can be based on a confidence metric. Chernin, [0084]-[0085]. The confidence metric can be based on factors including an identity of the entity who was responsible for creating the threat data. Chernin explains:

> [0086] For example, there may be a high confidence metric and quality metric that the information contained in a construct is accurate and valid if the entity that first observed a threat has a long historical record of accurately characterizing cyberattacks, […].
>
> [0087] In contrast, if the entity that identified the cyberthreat has just joined the information-sharing network and has no history of correctly identifying cyberthreats […] the confidence and quality metrics for that STIX language construct will be relatively low.

> [0088] ***In certain embodiments of the invention, member entities may choose to respond to constructs with low confidence/quality scores […] by filtering out the low-quality threat information altogether so that it is ignored***.

In other words, Chernin teaches identifying additional cloud deployments to learn from by providing a mechanism to ignore cloud deployments that provide low-quality threat information and using information from the entities whose data survives the filter for learning. Franz, ¶¶72-73.

Additionally, Chernin discloses that entities can form "communities" with "access control groups" defining which entities and/or repositories are authorized to access certain categories of threat information. Chernin, [0054], [0057]. Communities can be public, private, or secret. *Id.*, [0057]. This enables custom information sharing—for example, sharing sensitive information with some repositories while excluding others. Chernin, [0055]; *see also* FIG. 2; Franz, ¶¶74-75.

These groups allow an administrator to customize information sharing between entities. Chernin, [0055]. For example, in Figure 2, access control group 212 includes "local repositories 204a, 204b, and 204c and central repository 202, while excluding local entity 210." *Id.* In this example, if a piece of threat information would be damaging if revealed to the general public, it would be shared with repositories 202, 204a, and 204b, but not local entity 210. *Id.* As

another example, "information coded as 'RED' [may] not to be shared with any parties outside of the specific exchange, meeting, or conversation in which it was initially disclosed." Chernin, [0058]; Franz, ¶76.

Additionally, Chernin discloses that member entities can subscribe to each other's threat feeds. In peer-to-peer, local repositories may each publish a feed of threat information, and an operator may browse, search, or receive suggestions for feeds to subscribe to (e.g., feeds suggested by the directory based on the similarity of the two), causing those feeds to automatically push or pull threat information. Chernin, [0063]. A POSITA would have found it obvious that in the hub-and-spoke architecture, such a feed directory would be part of the centralized server, since all local entities connect to it directly. This setup would allow the feeds to be accessed and managed from a central location and be simpler. Franz, ¶77.

A POSITA also would have readily understood or found obvious that Chernin's cross-customer learning would not necessarily be relevant to an entire cloud deployment because much threat information is tied to a particular hardware platform or operating system. For example, a deployment running Ubuntu Linux servers gains little from vulnerability data specific to Windows Server, so learning of that kind would only apply where the platforms genuinely overlap, given how little vulnerabilities tend to correlate across different operating-system versions. Franz, ¶78.

1[b]/9[b]/17[b]

As discussed above for the preamble, Chernin teaches that local entities generate threat information from their sensors/controls, translate it into STIX, and transmit it to the centralized repository, which in turn distributes threat information to the other local entities. Chernin, [0038]-[0040], [0046]-[0048]. For example, Chernin discusses "SIEM sensors ... observ[ing] an event or property." *Id.*, [0039]. The local entity takes that observable event and generates an item of threat information in its local language. A POSITA would have understood that this is the event mentioned in [0013] and [0039] and that it is translated into, for example, the "Observable" construct in STIX mentioned in [0071]. Franz, ¶80-82.

1[c]/9[c]/17[c]

The '695 patent provides examples of "configuration settings":

> The configuration settings used to combat the security threat may be embodied, for example, as configuration settings that caused the additional cloud deployments 912 to blacklist certain IP addresses or network domains, as configuration settings that caused the additional cloud deployments 912 to close vulnerabilities that allowed an attack to succeed, or some other configuration that would be useful in detecting/preventing/mitigating a security threat.

'695 patent, 81:43-51.

Chernin discloses that entities store and distribute threat information to one another and/or to the centralized repository using a "common language,"

specifically the STIX language. Chernin, [0034], [0039], [0068]. The local entities translate their threat information into STIX and transmit it to the centralized repository, which in turn distributes items of threat information to the local entities. *Id.*, [0039]-[0040], [0046]. The STIX language includes "Observable," "Incident" and "CourseOfAction (COA)" constructs, among others. As described in the Chernin overview, Observables can include information about files, registry keys, services, HTTP requests, network traffic on specific IP addresses, and communication on certain ports. Chernin, [0071]. Incident includes "information discovered or decided during the investigation into how to respond to the Incident," "a log of actions taken during and in response to the Incident," and the responsive COA taken. *Id.*, [0074]. COA includes "its relevant stage in cyberthreat management (e.g., remedy of an ExploitTarget or response to an Incident) … the structured representation of the COA (such as an IPS rule or automated patch or remediation)…." *Id.*, [0082]; Franz, ¶¶84-85.

A POSITA would have understood that a COA's structured representation—for instance, an "IPS rule or automated patch or remediation" describes configuration settings used to combat the security threat, since an Intrusion Prevention System rule is a configuration directive telling the IPS appliance to inspect traffic and block or drop packets matching known malicious patterns. Franz, ¶86.

Similarly, applying an "automated patch or remediation" changes a system's software or settings to close a known vulnerability, which is likewise a configuration change. Chernin provides an example: "a detected malicious IP address can be distributed to other users of the information-sharing system, who can then block that IP address from gaining access to their networks." Chernin, [0006]. Blocking a malicious IP address necessarily means adding it to a firewall blacklist or access-control list—itself a configuration setting. When the local entities send threat information to the central repository (*see* 1[b]/9[b]/17[b]) using STIX, the central repository and/or another local entity "receiv[es] information describing configuration settings used to combat the security threat," just as recited in 1[c]/9[c]/17[c]. Franz, ¶87.

Additionally, Chernin discloses that the COA can include more than one configuration setting. Chernin, [0082] ("CourseCfAction (COA)…is a representation of ***one or more specific measures*** to be taken to address cyberthreats."); Franz, ¶88.

1[d]/9[d]/17[d]

Chernin teaches that the local entities send threat information to the central repository or to one another using STIX (*see* 1[c]/9[c]/17[c]). The central repository or other local entity (each of which is associated with a local repository) thus "receiv[es] information describing configuration settings used to combat the

security threat," particularly the COA's representation such as an "IPS rule or automated patch or remediation." Chernin, [0082]. Indeed, the entire purpose of the COA in the STIX data is to represent "one or more specific measures to be taken to address cyberthreats." *Id*.; Franz, ¶91.

Chernin further teaches that the central repository identifies, based on the threat information it receives (in the hub-and-spoke architecture), one or more configurations to adopt for the local repositories: "the analysis performed by central repository 102 may allow the central repository 102 to determine…a particular action that could be taken to prevent, mitigate, or remediate the effects of at least one of the multiple observed events or properties, the entity responsible for at least one of the multiple observed events or properties." Chernin, [0044]. The central repository then distributes items of threat information to the local entities via "pull" or "push," and in some embodiments "is capable of determining unique items of the subsets of threat information to be distributed to each of the local entities." *Id*., [0047]-[0048]; Franz, ¶92.

Chernin teaches a wide variety of actions that can be taken to prevent, mitigate, or remediate the effects of cyberattacks, including blocking a malicious IP address, sharing detected malware filenames, and distributing COAs. Chernin, [0006], [0082]; Franz, ¶93.

Working from the STIX data it receives from an attacked entity, the central repository can thus identify configurations—such as an IP block or an IPS rule—that other entities should adopt. The COA construct's structured representation identifies configurations to adopt, since the purpose of the COA is to represent "one or more specific measures to be taken to address cyberthreats." Chernin, [0082]. Chernin further teaches that the central repository's analysis results, including "a particular action that could be taken to prevent, mitigate, or remediate the effects of at least one of the multiple observed events or properties," are added back to the threat items that are shared. Chernin, [0016], [0044]; *see also* Chernin, Claim 16. Where the identified remediation is, say, a firewall rule or patch, the repository can thus annotate the existing threat item accordingly and push it out to the other local entities through the same STIX/TAXII pipeline, letting each recipient identify and implement configurations to reduce its own exposure. Franz, ¶94.

3. Claims 2/10/18

2[pre]/ 10[pre]/18[pre]

Claim 2 depends from claim 1; claim 10 depends from claim 9; and claim 18 depends from claim 17. Chernin discloses or renders obvious claims 1/9/17.

2[a]/10[a]/18[a]

As discussed for 1[c]/9[c]/17[c], Chernin teaches that "a detected malicious IP address can be distributed to other users of the information-sharing system, who can then block that IP address from gaining access to their networks" [0006] and that the "base' construct of the STIX language" used to communicate threats can include "network traffic on specific IP addresses" [0071]; Franz, ¶98.

Thus, a POSITA would have found it obvious that the STIX data reaching the central repository, or the threat information exchanged among the entities (which correspond to cloud deployments as discussed for 1[pre]/9[pre]/17[pre]), would obviously identify a known-malicious IP address for blacklisting—satisfying the claim's requirement of "configuration settings that caused the one or more of the additional cloud deployments to blacklist one or more internet protocol (IP) addresses," as recited in 2[a]/10[a]/18[a]. Franz, ¶99.

4. Claims 3/11/19

3[pre]/11[pre]/19[pre]

Claim 3 depends from claim 1; claim 11 depends from claim 9; and claim 19 depends from claim 17. Chernin discloses or renders obvious claims 1/9/17.

3[a]/11[a]/19[a]

Chernin discloses or renders obvious this claim for reasons similar to claim 2. The base "Observable" construct includes "an HTTP request being sent (or received)." Chernin, [0071]. A POSITA would have found it obvious, for example,

that HTTP requests carry URLs that may reveal malicious domain names identified by the attacked local entity (which corresponds to a cloud deployment as discussed for 1[pre]/9[pre]/17[pre]), and because Chernin's STIX-based threat information is designed to let local entities "prevent, mitigate, or remediate the effects of the observable event or property," a POSITA would have found it obvious that this includes blocking a previously identified malicious domain based on its name (among other things). Franz, ¶¶101-102.

5. Claims 5/13

5[pre]/13[pre]

Claim 5 depends from claim 1; and claim 13 depends from claim 9. Chernin discloses or renders obvious claims 1/9.

5[a]/13[a]

The '695 patent describes "actions" as including messages, requests, or other data communications exchanged between cloud deployments and external actors ('695 patent, 79:41-46) and user interactions such as which resources are accessed by particular users or business units (*id.*, 79:51-59); Franz, ¶¶106, 109.

A POSITA would have understood that communication activity between an entity and an external third party could indicate a security attack (e.g., outward communication with a command-and-control node or inward exploitation of a vulnerability), and that configuring firewalls to block such communication would

be a natural response. Chernin's "Incident" construct expressly includes "a log of actions taken during and in response to the Incident." Chernin, [0074]. This "log of actions" corresponds to "information describing one or more actions associated with the additional cloud deployments" as recited in 5[a]/13[a], and it is part of the STIX data shared among the local entities and the central repository. Moreover, Chernin teaches that threat information may be "manually generated" at local entities (Chernin, [0039]), which a POSITA would understand to constitute a user interaction with a cloud deployment. Franz, ¶107, 110-11.

As discussed above, Chernin's STIX Observable construct includes information about files, services, HTTP requests, network traffic, and communication on certain ports—all of which describe actions associated with cloud deployments. Chernin, [0071]; Franz, ¶108.

5[b]/13[b]

As discussed above for claims 1/9/17, Chernin discloses or renders obvious "identifying…one or more configurations to adopt for the first cloud deployment." And as explained for 5[a]/13[a], Chernin discloses or renders obvious that the threat information describes actions associated with the additional cloud deployments via the "Observable" construct in STIX language. Chernin discloses or renders obvious that configurations to adopt for a given local entity are based on actions of another local entity. STIX includes several nested constructions based

on the Observables, such as "Indicator," which conveys information about patterns of Observables representing behaviors of interest; "Incident," which is a discrete instance "in which an Indicator affects an organization, along with information discovered or decided during the investigation into how to respond to the incident," including "related Observables;" and "CourseOfAction (COA)," which represents "one or more specific measures to be taken to address cyberthreats" and is based on Observable and Indicator constructs. Chernin, [0073]-[0083]; *see* 1[c]/9[c]/17[c]; Franz, ¶113-114.

A POSITA would have recognized that based on the STIX data—including information about actions—that Chernin's central repository receives from an attacked entity, the central repository can use the Incident, Observables, COA, and other constructs to identify configurations that another local entity should adopt. Moreover, a POSITA would have recognized that an entity receiving such threat information would identify and implement configurations to reduce the threat. Franz, ¶115.

6. Claims 8/16

8[pre]/ 16[pre]

Claim 8 depends from claim 1; and claim 16 depends from claim 9. Chernin teaches claims 1/9.

8[a]/16[a]

In its infringement contentions (Ex. 1016), Fortinet alleges:

> Netskope Cloud Exchange may receive information from third party cloud deployments that can include ***information concerning the name of the file associated with an identified threat***, which is insubstantially different from information concerning the deployment processes employed in the cloud deployment as the file name (as incorporated in the "comment" field) can establish the component (e.g., program) within the cloud deployment associated with the identified threat.

Infringement Contentions at 43. Chernin teaches the same thing that Fortinet accuses of infringement. Because STIX's "Observable" construct captures "information about a file (such as the filename, hash, or size)," Chernin, [0070]-[0071], filenames are necessarily among the data the central repository obtains from attacked entities, as well as what entities exchange directly with one another. Franz, ¶118-22.

B. GROUND 2: THE COMBINATION OF CHERNIN AND JEUK RENDERS CLAIMS 1-3, 5, 8-11, 13, 16-19 OBVIOUS

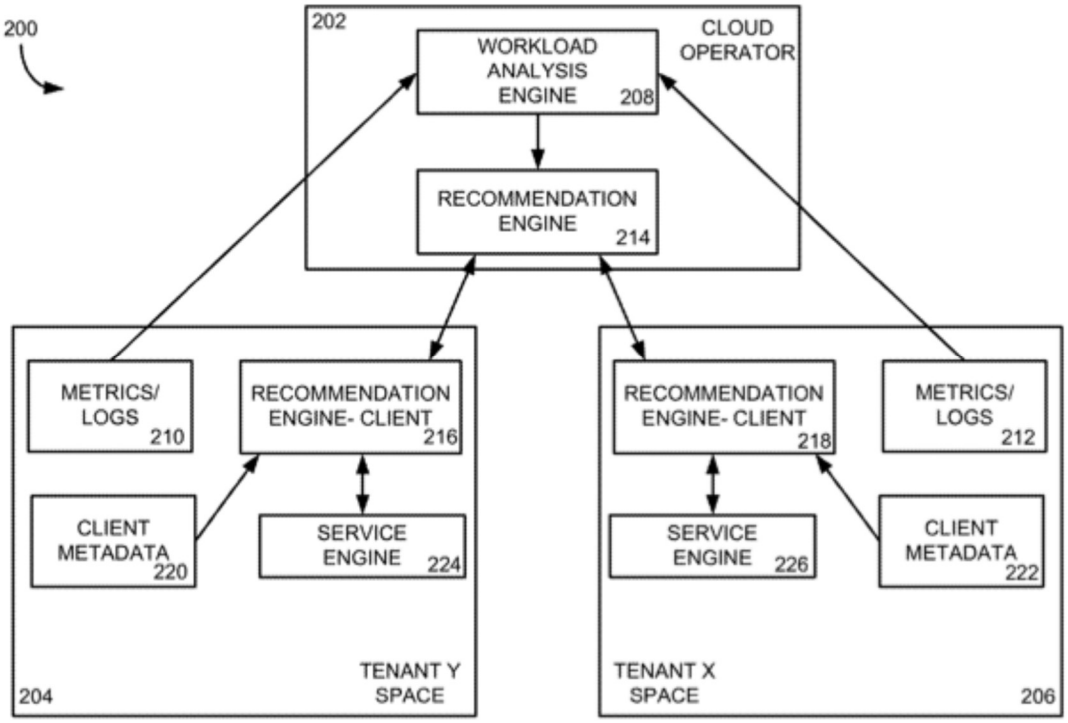
1. Overview of Jeuk

Jeuk's disclosure centers on collecting and correlating workload performance data across multiple customers' separate "tenant spaces," then applying cross-customer learning to that correlated data to generate deployment-improvement recommendations for a cloud operator or customer. Jeuk, Abstract, [0017]; Franz, ¶124.

Specifically, Jeuk discloses a "workload analysis engine" builds a first performance pattern from metrics and logs gathered from one tenant space, and a second performance pattern from a corresponding data set gathered from a second tenant space. Jeuk, [0014], [0027]. Applying machine-learning techniques to this performance data, the engine can then detect patterns across the different tenant deployments, Jeuk, [0017]—effectively spotting a characteristic or pattern suggesting that a similar application is running across multiple tenants. *Id.*; Franz, ¶125.

A recommendation engine, working in conjunction with the workload analysis engine then correlates the received metrics and log data and uses it to formulate deployment recommendations, Jeuk, [0018], [0037], analyzing performance characteristics across the various tenant applications to arrive at a "best fit" or optimized deployment of the application. Jeuk, [0018]; Franz, ¶126.

Once generated, that optimized deployment can be delivered to a cloud environment administrator and/or one or more tenants, Jeuk, [0019], as a suggested improvement to the deployed application sent to a client recommendation engine within a tenant space. *Id.* The client recommendation engine then decides whether the recommended optimized deployment should actually be executed within the tenant space. *Id.*; Franz, ¶127.



Jeuk, Fig. 2.

2. Jeuk's Role in Parent's Prosecution

During prosecution of the '695 patent's parent application (Serial No. 17/671,199), the examiner relied on Jeuk to challenge the novelty of the claims.

The examiner rejected claims 1-2, 9-10, and 17-18 under 35 U.S.C. § 102(a)(1) as anticipated by Jeuk alone, and rejected claims 3-6, 8, 11-14, 16, and 19-20 under 35 U.S.C. § 103 as obvious over Jeuk with secondary references. Ex. 1005, 318-332. The examiner found that Jeuk met several core limitations of the parent application's independent claims. Specifically:

Limitation (Parent App Claim 1)	Examiner's Basis in Jeuk
Preamble: "A method of learning from	Jeuk generally

similar cloud deployments"	
1[a]: "identifying, for at least a portion of a first cloud deployment, one or more additional cloud deployments to utilize for cross-customer learning"	Jeuk Abstract – workload engine detects performance patterns across different tenant deployments
1[b]: "receiving information describing configurations associated with the additional cloud deployments"	Jeuk [0036] – workload analysis engine correlates identified performance pattern with a particular configuration of cloud resources
1[c]: "identifying, based on the configurations, one or more configurations to adopt for the first cloud deployment"	Jeuk [0018] – recommendation engine determines a "best fit" or optimized deployment
Claim 8: "receiving information describing one or more deployment processes associated with the additional cloud deployments"	Jeuk Abstract – applications provide performance logs and metrics to inter-tenant workload engine

The applicant overcame the rejections by amending the claims to incorporate subject matter from dependent claims directed to resolving user-privilege issues. *Id.*, 297-308. The examiner then allowed the claims. *Id.*, 264-266.

Several limitations the examiner found Jeuk disclosed mirror limitations in the Challenged Claims, such as:

Limitation Found in Jeuk (Parent App)	**Corresponding '695 Patent Claim Limitation**
Preamble: "A method of learning from similar cloud deployments"	1[pre]: "A method of learning from other cloud deployments to combat security threats"
1[a]: "identifying, for at least a portion of a first cloud deployment, one or more additional cloud deployments to utilize for cross-customer learning"	1[a]: "identifying, for at least a portion of a first cloud deployment, one or more additional cloud deployments to utilize for cross-customer learning"

	(identical)
1[c]: "identifying, based on the configurations, one or more configurations to adopt for the first cloud deployment"	1[d]: "identifying, based on the information describing configuration settings used to combat the security threat, one or more configurations to adopt for the first cloud deployment" (security-specific)
Claim 8: "receiving information describing one or more deployment processes associated with the additional cloud deployments"	Claim 8: "receiving information describing one or more deployment processes associated with the additional cloud deployments" (identical)

The core structural framework of the challenged claims—identifying additional cloud deployments for cross-customer learning, receiving information from those deployments, and identifying configurations to adopt—is the same as what the examiner found Juek disclosed during prosecution of the parent. All the challenged claims do is add a security-threat overlay to Jeuk's cross-tenant learning architecture. But adapting Jeuk's cross-customer learning architecture to the security context, like the one in Chernin, would have been obvious. Franz, ¶¶128-135.

3. Motivation to Combine Chernin and Jeuk.

A POSITA would have been motivated to combine Jeuk and Chernin with a reasonable expectation of success.

To the extent Chernin does not utilize "cloud deployments" (*see* Ground 2, elements 1[pre]/9[pre]/17[pre]), a POSITA implementing Chernin's information-

sharing scheme would have appreciated the value of running it within cloud deployments of the kind Jeuk describes, given that cloud deployments were already assuming a growing role in enterprise computing, infrastructure, and security well before November 2017 (*see supra* §IV.B). That growing importance would have motivated a POSITA to deploy Chernin's scheme in the cloud, particularly since Chernin itself acknowledges the heightened risk of placing organizational data "in the cloud" ("An institution that has placed its data in a repository 'in the cloud' now faces the prospect of having its information stolen and compromised by a hacker who gains access to that single repository." Chernin, [0005].). Franz, ¶¶136-137.

Additionally, a POSITA implementing Chernin's threat-information-sharing scheme would have recognized that Jeuk's pattern correlation and recommendation approach could be advantageously adapted to optimize defenses for Chernin's cloud deployments. A POSITA would have been motivated to incorporate Jeuk's correlation and recommendation techniques into Chernin for cyber defense. This combination would have had a reasonable expectation of success, since both references share a similar architecture in which a central authority collects information from cloud deployments, processes it, and communicates back to at least a subset of the deployments. Franz, ¶138.

Chernin's system collects, translates, and disseminates threat information, whereas Jeuk's collects performance data, computes deployment improvements, and communicates recommendations; a POSITA reviewing both would have recognized the value in applying Jeuk's "recommending calculated improvements" concept to the threat information Chernin already gathers. Jeuk itself recognizes that its invention can provide recommendations for "other applications and workloads." Jeuk, [0063]; Franz, ¶139.

This modification would provide additional benefits to Chernin while changing nothing substantive about how Chernin's threat-sharing functions operate. Adding Jeuk's correlation and recommendation features to Chernin's central repository would have been straightforward since the functionality needs to be added in a single location. The central repository already ingests, stores, and processes STIX-formatted threat data, and Jeuk's engine would operate on that same data to identify cross-tenant patterns and generate recommendations. The existing TAXII distribution infrastructure could deliver recommendations back to client deployments—no new communication protocol or network architecture would be required. Franz, ¶140.

Indeed, Chernin already discloses or suggests a similar "correlation and recommendation" approach. The central repository can analyze received threat information—including "***comparing the received items of threat information to***

previously stored items of threat information," "***determining patterns***," and "making future predictions"—***and*** can determine "a particular action that could be taken to prevent, mitigate, or remediate ***the*** effects of at least one ***of*** the multiple observed events or properties." Chernin, [0043]-[0044]. Such analysis results can be added to the shared information items. Chernin, [0016], Claim 16; Franz, ¶¶141-42.

The combination of Chernin and Jeuk would therefore have been a straightforward undertaking for a POSITA, adding new capability to Chernin while leaving its existing functionality intact. Franz, ¶143.

4. Claims 1, 9, and 17

1[pre]/9[pre]/17[pre]

For the reasons explained in Ground 1, Chernin discloses or renders obvious limitations 1[pre]/9[pre]/17[pre].

But to the extent Chernin does not specifically disclose sharing information between "cloud deployments," a POSITA would have found it obvious to deploy Chernin in such deployments in view of Jeuk.

As discussed above, the examiner found that Jeuk disclosed the concept of learning from cloud deployments during examination of the '695 patent's parent. Jeuk's architecture is similar to Chernin's "hub-and-spoke" model in which tenants on a cloud learn from another to improve computing performance. Jeuk discloses a

centralized cloud computing environment that may include private, public, and/or hybrid clouds providing various services to clients. Jeuk, [0022]-[0024]; FIG. 1; Franz, ¶¶145-147.

Jeuk teaches that cloud environment 102 may include a cloud operator environment with a workload analysis engine 208 that "analyze[s] the performance information received from the tenants 204, 206 of the cloud environment to identify performance patterns of the executing applications and/or correlate performance patterns across tenant configurations," "correlate[s] each performance pattern generated for each tenant to a particular tenant configuration or deployment," and provides the correlated pattern information to a recommendation engine 214. Jeuk, [0026], [0028], [0036], [0037]. The recommendation engine then "provide[s] a recommendation to one or more tenants…for improving the execution or deployment of workloads on the environment." *Id.*, [0037]. This parallels Chernin: in both references a centralized entity collects and analyzes information from individual participants and redistributes actionable output back to them—workload improvements in Jeuk's case, threat countermeasures in Chernin's. Franz, ¶148.

As discussed above, a POSITA would have been motivated to combine Chernin and Jeuk, applying Chernin's teachings to cloud deployments. It would have been obvious to deploy Chernin's "entities such as individuals, firms,

government agencies and associations, each of which are hosted on one or more distributed servers" in the cloud, as Jeuk teaches. Franz, ¶149.

1[a]/9[a]/17[a]

As discussed above, Chernin discloses or renders obvious this limitation. If Chernin does not, the combination of Chernin and Jeuk does.

Because Jeuk's workload analysis engine "may identify a performance characteristic or pattern that may indicate the execution of a similar application by different cloud environment tenants," Jeuk, [0017], it is identifying which tenant deployments run comparable workloads—in other words, identifying additional cloud deployments for cross-customer learning. Moreover, Jeuk teaches that the engine "correlate[s] each performance pattern generated for each tenant to a particular tenant configuration or deployment," Jeuk, [0036], associating the identified deployments with their configurations so the recommendation engine can issue targeted recommendations. Franz, ¶153.

As discussed above, the examiner already found that Jeuk disclosed the concept of learning from cloud deployments during examination of the '695 patent's parent. For example, the examiner found that Jeuk teaches that a workload engine "may ***identify a performance characteristic or pattern that may indicate the execution of a similar application by different cloud environment tenants"*** and "a similar application is identified as executed or deployed by different tenants

on the cloud environment….” Jeuk, Abstract, [0017]; *see also id*., [0031]; Franz, ¶154.

As discussed above, a POSITA would have been motivated to combine Jeuk and Chernin. In the combination, Jeuk’s engine would be deployed in Chernin’s central repository and would identify cloud deployments for cross-customer learning using the same analysis disclosed in Jeuk. *E.g.*, Jeuk at Abstract, [0017] Thus, the system would correlate deployments using, for example, similar applications, which would improve information sharing in the event of an attack. Franz, ¶155.

1[b]/9[b]/17[b]

Chernin discloses or renders obvious this claim limitation when combined with Jeuk for the same reasons discussed in Ground 1. Franz, ¶157.

1[c]/9[c]/17[c]

Chernin discloses or renders obvious this claim limitation when combined with Jeuk for the same reasons discussed in Ground 1. Franz, ¶158.

1[d]/9[d]/17[d]

As discussed above, Chernin discloses or renders obvious this limitation. But to the extent Chernin does not specifically disclose “identifying… configurations to adopt for the first cloud deployment,” a POSITA would have found it obvious to modify Chernin in view of Jeuk to do so.

As discussed in Jeuk's summary, Jeuk discloses a recommendation engine that correlates metrics and log information to generate deployment recommendations. Jeuk, [0018], [0037]. The recommendation engine analyzes the performance characteristics across multiple tenant applications and determines a "best fit" or optimized deployment of the application. Jeuk, [0018]. Then, the recommendation engine provides the optimized deployment to a cloud environment administrator and/or one or more of the tenants. Jeuk, [0019]. During prosecution of the parent patent, the examiner found Jeuk discloses "identifying, based on the configurations, one or more configurations to adopt for the first cloud deployment." Franz, ¶¶160-62.

A POSITA would have been motivated to incorporate Jeuk's correlation and recommendation engine into Chernin's system for the reasons discussed in the motivation to combine section. Franz, ¶163.

5. Claims 2/3/5/10/11/13/18/19

Chernin discloses or renders obvious these claims when combined with Jeuk for the same reasons discussed in Ground 1. Franz, ¶¶165, 167, 169-170.

6. Claims 8/16

8[pre]/16[pre]

Claim 8 depends from claim 1; and claim 16 depends from claim 9. The Chernin-Jeuk combination teaches claims 1/9.

8[a]/16[a]

As discussed in Ground 1, Chernin discloses or renders obvious this claim limitation. If Chernin does not, the combination of Chernin and Jeuk does.

During prosecution of the parent patent, the examiner found that Jeuk disclosed this limitation, as described above. Specifically, the examiner reasoned that Jeuk disclosed that "applications deployed and executed on the cloud environment may provide performance logs and / or metrics to an inter - tenant workload engine of the cloud environment." Franz, ¶174.

Beyond that, Jeuk teaches that its recommendation engine "may receive or obtain[] deployment information for the workloads or applications related to the identified performance patterns" and that "[s]uch deployment information may be any type of information related to the deployment of the applications or workloads...." Jeuk, [0043]. Franz, ¶175.

C. GROUND 3: THE COMBINATION OF HUDIS AND SHANKS RENDERS CLAIMS 1-3, 5, 8-11, 13, 16-19 OBVIOUS.

1. Overview of Hudis

Hudis's recommendation system works by analyzing feature values used by cloud deployments and recommending a feature value for a target deployment's target feature based on how similar the target deployment is to those other deployments. Hudis, Abstract, [0019]. Hudis illustrates this with a firewall example, "if...features of the target deployment are most similar to features of

deployments that use a firewall with a certain name, then the recommendation system recommends that the target deployment use the firewall with that certain name." *Id.*, [0020]; Franz, ¶178.

Hudis provides exemplary display pages showing (1) how a user can select which recommendations it wants to receive and (2) a recommendation for a firewall based on which firewalls other, similar deployments are using.

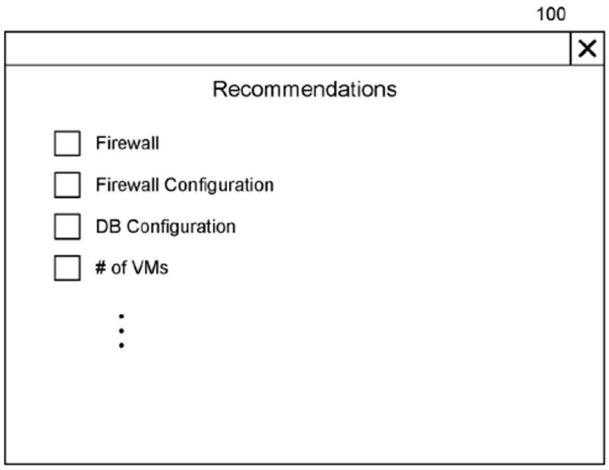


FIG. 1

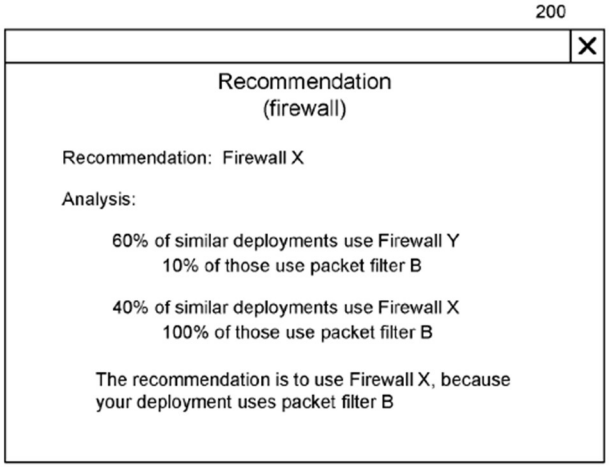


FIG. 2

Generating these recommendations requires the system to first collect feature values for each deployment. *Id*., [0019]. These features span data center resources (e.g., virtual machines and storage), appliances (e.g., firewalls and packet filters), configuration information and logs for those resources and appliances, usage patterns, network data, and indicators of compromise and threat intelligence. *Id*., [0019], [0040]. The system also collects—and a deployment may purchase—

threat intelligence that, for example, "identifies Internet Protocol ('IP') addresses whose access should be blocked by a firewall." *Id*., [0040]; Franz, ¶¶179-180.

Rather than generating a single model across all deployments, the system can generate separate models for clusters of similar deployments. Hudis, [0022]. The system clusters the deployments based on similarity between features of the deployments, for example using a k-means or k-nearest neighbor clustering algorithm. *Id*. The system then generates a model for each cluster using training data from the deployments within that cluster. *Id*. To generate a recommendation, the system first identifies the cluster to which the target deployment is most similar. *Id*., [0023]. The system determines similarity, for example, by computing cosine similarity between a representative feature vector for each cluster and the feature vector for the target deployment. *Id*. After identifying the most similar cluster, the system applies the model for that cluster to the target deployment's feature vector to generate the recommendation. *Id*.; Franz, ¶181.

Hudis further discloses that the recommendation system can monitor the features of deployments in real time and provide recommendations based on changes in those features. Hudis, [0024]. Hudis describes a scenario in which "several deployments are the subject of a certain type of attack" and "some administrators of deployments may quickly change the configuration of their firewalls in an attempt to thwart the attack." *Id*. The recommendation system can

use such changes as a trigger to generate recommendations for changing the configurations of firewalls of other deployments. *Id*. The trigger can be based on a recommendation criterion such as a percentage of deployments that changed their configurations, sophistication of the deployments, or reputation of deployment administrators. *Id*. The system then "generates a recommendation relating to the configuration of the firewall for other deployments and provides the recommendation for each deployment." *Id*.; Franz, ¶182.

Hudis also teaches that a deployment may authorize the automatic implementation of recommendations. *Id*., [0024]. In that case, "the recommendation system may direct a configuration manager of the deployment to change the configuration to automatically implement the recommended configuration without the explicit approval of an administrator of the deployment." *Id*. Hudis thus provides a mechanism by which firewall configuration settings that have proven effective at thwarting attacks on some deployments can be identified and automatically propagated to other similar deployments within the data center. *Id*., [0024]-[0025]. This is a straightforward application of automated configuration management, a technique that was already well established by 2017. Franz, ¶183.

2. Overview of Shanks

Shanks relates to automatically configuring firewalls to minimize errors caused when administrators manually configure their firewalls. Shanks, [0002].

Manual firewall configuration is error-prone and automated approaches to reduce human error were sought throughout the industry. Shanks's architecture is geared towards firewall appliance providers, which makes its practical applicability all the more evident. The assignee of Shanks is Cisco, a major player in network security that manufactures and sells firewalls. Shanks also discloses that cloud management system customers could "purchase a specific service" to have their firewalls configured automatically, which would be an attractive additional recurring revenue stream for a firewall manufacturer/seller. *Id*., [0068]; Franz, ¶184.

Shanks discloses "automatic firewall configuration based on aggregated cloud managed information" using cross-customer learning. Shanks, [0011]. A cloud management system (which can itself be cloud-based) communicates with multiple client computing environments that may belong to different customers. *Id.*, [0061], [0068]. Client computing environments can continuously transmit "security event data" to the cloud management system. *Id.*, [0060], [0063]. This architecture is depicted in Figures 4 and 5, below. Franz, ¶185.

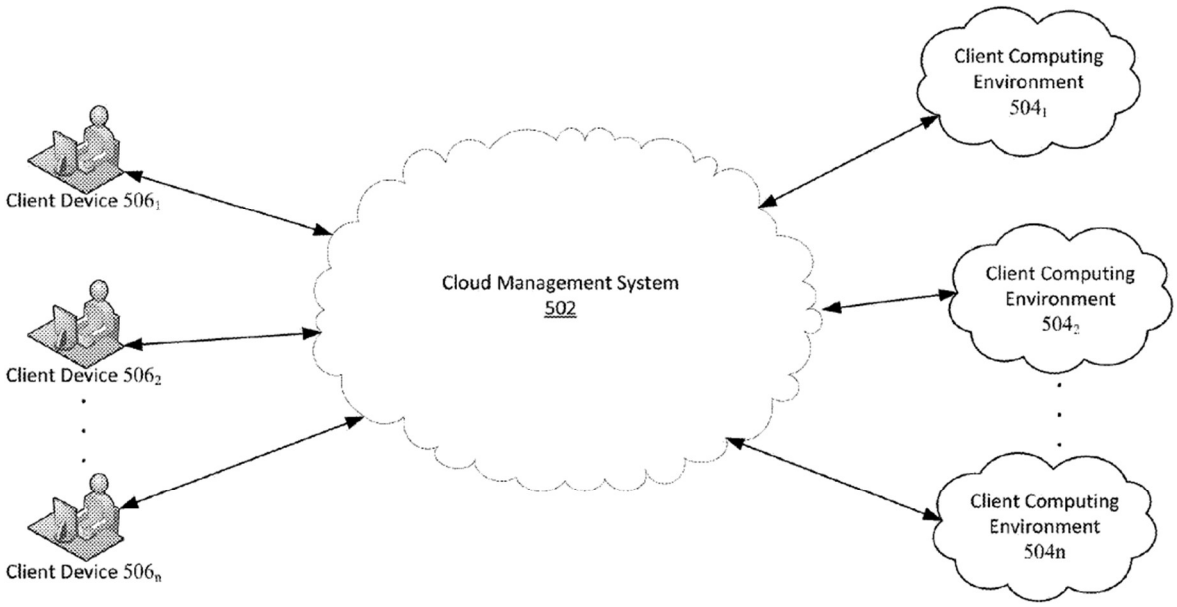
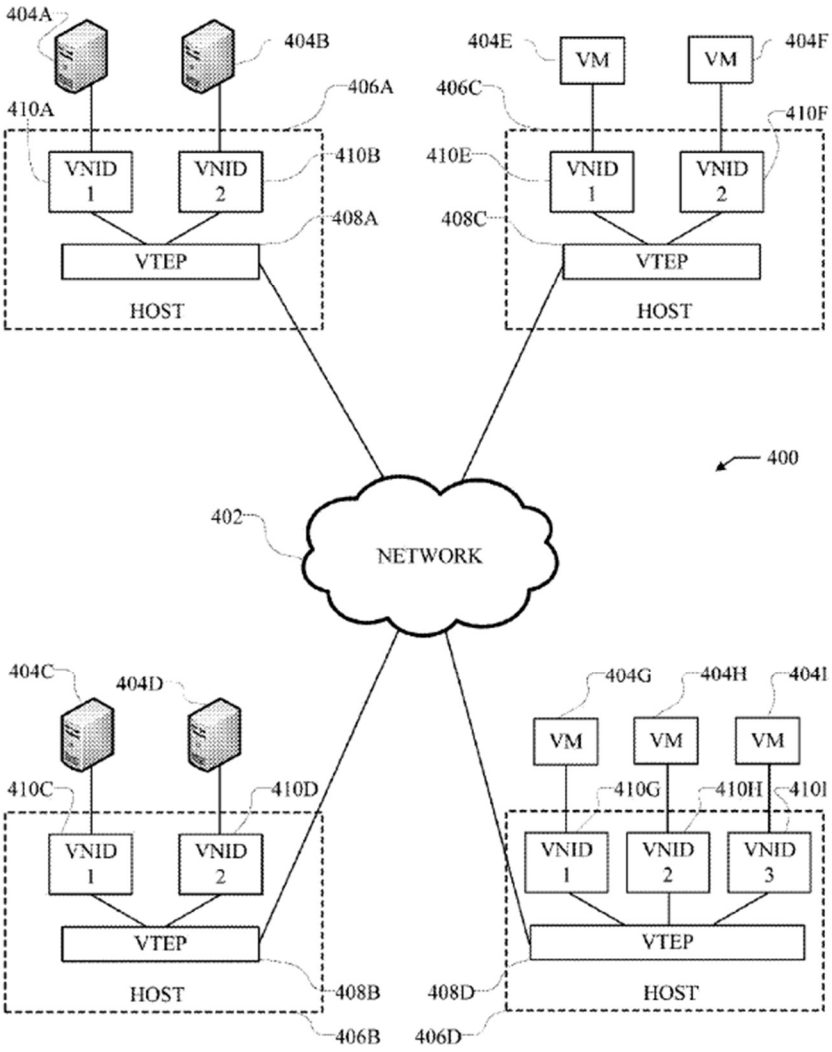
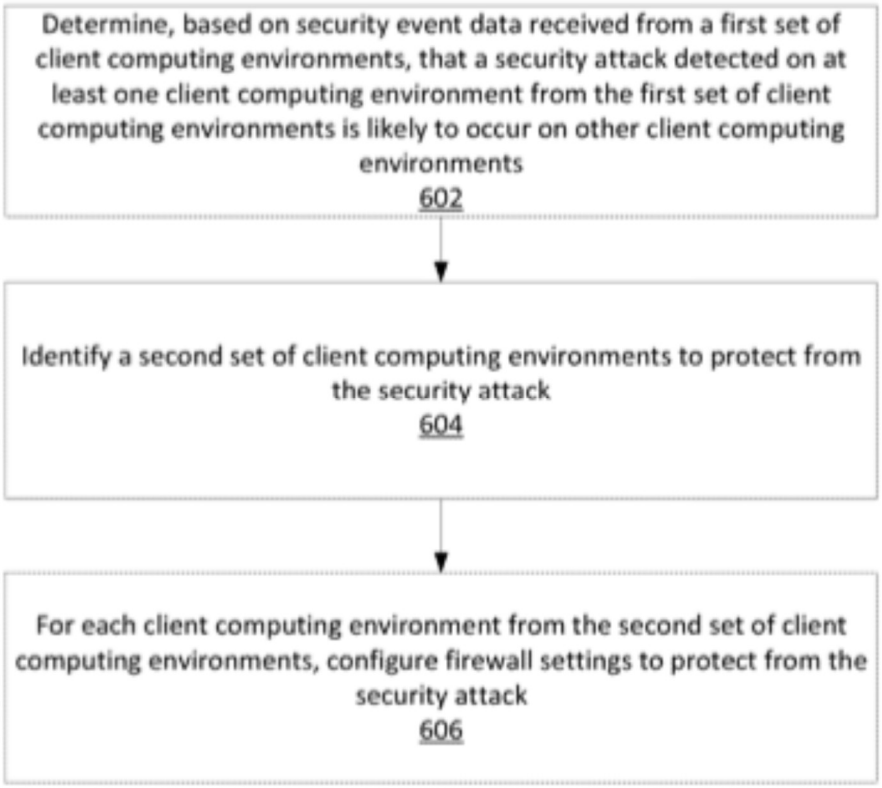

Client Device 506$_1$
Client Device 506$_2$
Client Device 506$_n$
Cloud Management System
502
Client Computing Environment 504$_1$
Client Computing Environment 504$_2$
Client Computing Environment 504n

FIG. 5

FIG. 4



Security event data encompasses any data describing detected security events on a client computing device, or data from which such events can be derived. Shanks, [0063]. A security event, in turn, is any observable deviation from normal system, device, environment, process, or workflow behavior that could signal a malicious breach. *Id.*; Franz, ¶¶185-186.



(Shanks, Fig. 6)

The cloud management system can analyze security event data received from client computing environments to identify security attacks and determine whether the attacks are likely to occur on other client computing environments. *Id.*, [0060], [0064]. If so, the cloud management system identifies a second set of client computing environments to protect from the attack. *Id.*, [0060], [0073]. For each of the client computing environments from this second set, it can then automatically configure firewall settings. *Id.*, [0068], [0074]. For example, it can apply firewall rules to block and/or filter data packets transmitted from a source associated with

the security attack or to block and/or filter data packets transmitted to a destination port associated with the security attack. *Id.*, [0069]. Franz, ¶187.

Shank's cloud management system can provide users with a management interface that allows setting configurations, such as security settings, firewall settings, whitelists, blacklists, etc., for their respective client computing environment. *Id.*, [0059], [0062]. Users (e.g., system administrators) communicate with the cloud management system to access the management interface and set/adjust configurations. *Id.*, [0062]. These received configurations can then be then sent to client computing environments for implementation. *Id.*, [0059], [0062]; Franz, ¶188.

3. Motivation to Combine Hudis and Shanks

A POSITA would have been motivated to combine Hudis and Shanks with a reasonable expectation of success.

Both Hudis and Shanks are cloud-based systems that aggregate data across cloud deployments to bolster the security of other deployments. Hudis aggregates deployment data and generates recommendations for configurations, including firewalls, and can use changes in firewall configurations during an attack as a trigger to generate recommendations for other deployments. Hudis, [0019], [0024]. Shanks, similarly, aggregates security event data across cloud deployments and uses that data to generate configuration settings to thwart attacks in another

deployment. Shanks, [0011]. The two references thus occupy the same field of endeavor and address a common problem. Franz, ¶190.

Moreover, a POSITA would have been motivated to combine these references because doing so would create a materially improved system leveraging the respective strengths of each. Franz, ¶191.

As discussed in the Hudis overview, Hudis teaches that its recommendation system can use changes in firewall configurations during an attack as a trigger to recommend new configurations for other deployments. Hudis, [0024]. In this scenario, though, the recommended setting is community-driven—derived from statistical analysis of configuration changes across deployments—which might not yield the optimal solution. Furthermore, this approach leaves open the risk that an administrator introduces errors when manually configuring the firewall, a problem Shanks specifically identifies and seeks to solve. Franz, ¶192.

A more reliable and streamlined alternative—exactly what Shanks provides—would have a cloud security management system either recommend or directly implement the firewall configuration itself. A system of that kind, particularly one operated by a firewall appliance vendor, is uniquely positioned to (1) identify the optimal configuration setting for a given attack, given its intimate knowledge of its own product's capabilities and known vulnerabilities, and (2) roll

that setting out quickly across its existing and new customer base alike. Franz, ¶193.

Hudis itself suggests this solution, teaching that "a provider of security appliances may request that its customers provide their feature values" and "can then, based on those feature values, make recommendations to its current customers for changes in feature values, new customers for initial feature values, and so on." Hudis, [0025], [0049]. A POSITA would have been motivated to combine the references' teachings so that similar cloud deployments are clustered (as Hudis teaches), information is gathered from cloud deployments (as both Hudis and Shanks teach), information is shared with the security appliance provider (as Hudis teaches), a targeted configuration setting is identified by the appliance provider (as Shanks teaches), and then automatically implemented (as both teach). Franz, ¶¶194-95.

As noted above, Shanks is assigned to Cisco Technology, Inc. Shanks itself teaches that customers can opt in for automatic firewall configuration (Shanks, [0068]), which is the type of managed security service a network security appliance provider such as Cisco would offer its customers. Appliance vendors have long sought to differentiate their products through automated, intelligence-driven configuration capability. A POSITA would have recognized that Shanks's automatic firewall configuration approach represents the kind of security-specific

enhancement that a firewall vendor would incorporate into a broader cross-deployment recommendation platform like the one Hudis discloses. Franz, ¶196.

The two systems are technically compatible, too, because both operate as centralized cloud management platforms collecting data from distributed deployments. Incorporating Shanks's automatic firewall learning into Hudis's general recommendation system represents an additive, predictable, and low-risk enhancement. This is a straightforward case of combining known, well-tested elements to yield predictable results—no new protocols, data formats, or fundamental architectural changes are required. The mapping below identifies overlapping and analogous disclosures across the references that further support the ease of combination. Franz, ¶197.

4. Claims 1, 9, and 17

1[pre]/9[pre]/17[pre]

Hudis teaches learning from other cloud deployments to combat security threats. For example, Hudis teaches that a "target deployment" learns from configuration settings (e.g., firewall configuration settings) used by other cloud deployments to thwart an attack. Franz, ¶199.

As discussed in the Hudis overview, Hudis discloses a recommendation system that recommends feature values for a target deployment based on similarity

to other deployments, including deployments in cloud-based data centers. Hudis, [0019], [0049]; Franz, ¶200.

Hudis "recommends a feature value for the target feature based on similarity between the features of the target deployment and the features of other deployments in the data center." Hudis, [0019]. Hudis gives an example of sharing a feature value in the context of a firewall. In this example, Hudis explains that "if features of the target deployment are most similar to features of deployments that use a firewall with a certain name, then the recommendation system recommends that the target deployment use the firewall with that certain name." *Id.*, [0020]; Franz, ¶201.

Figures 1 and 2 illustrate an interface for selecting features for recommendation (e.g., "firewall," "firewall configuration") and an example firewall recommendation based on similar deployments. Franz, ¶202.

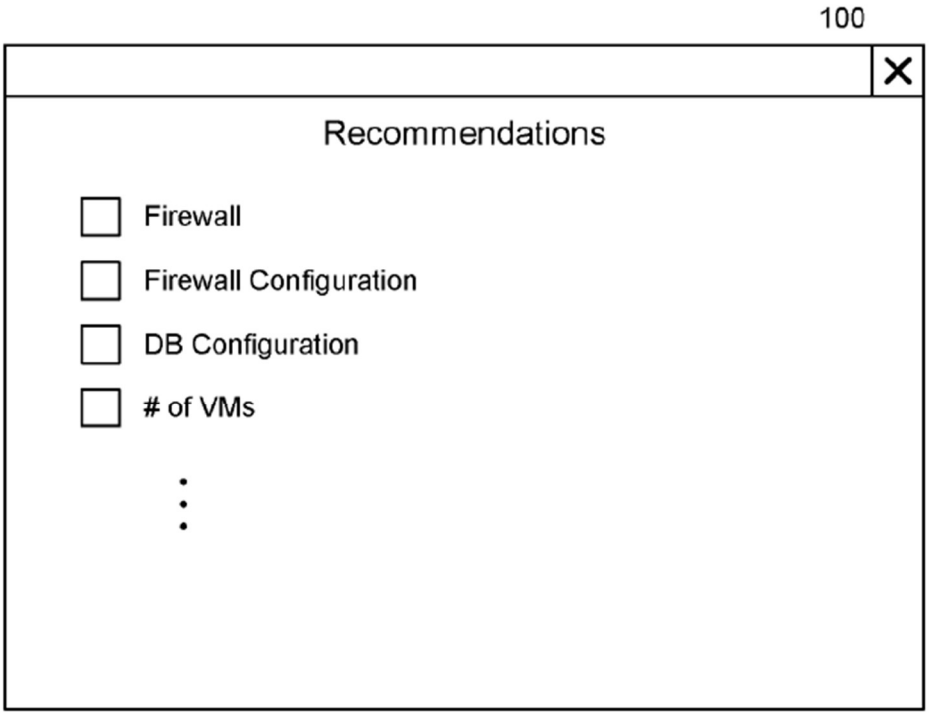

100
Recommendations
Firewall
Firewall Configuration
DB Configuration
# of VMs

FIG. 1

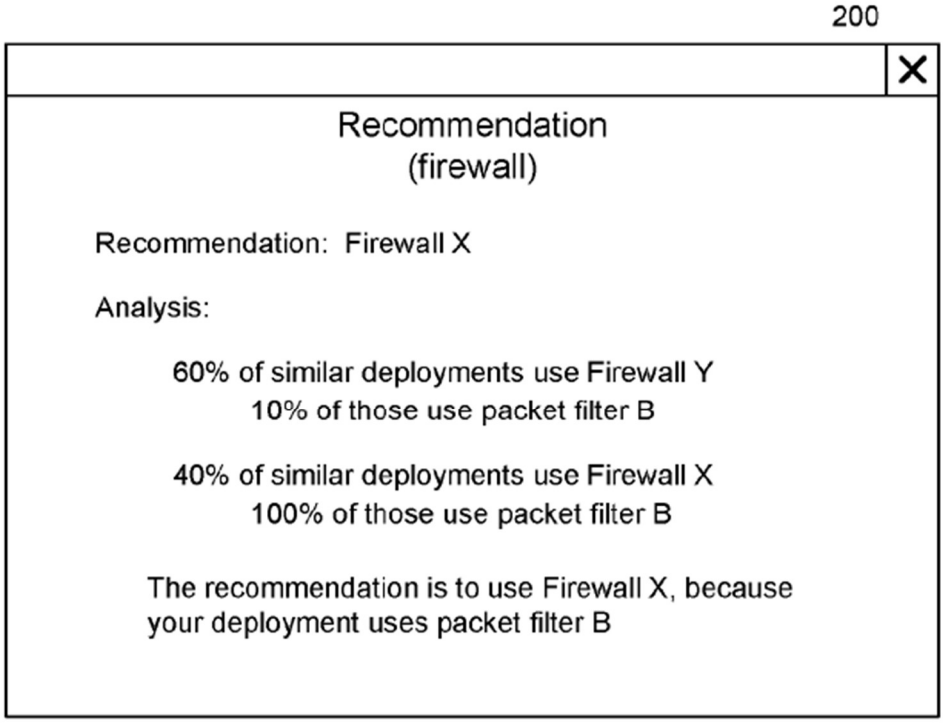


FIG. 2

As discussed in the Hudis overview, the recommendation system can monitor deployments in real time and, when several deployments change firewall configurations to thwart an attack, use those changes as a trigger to generate recommendations for other deployments, which can be automatically implemented:

> For example, if several deployments are the subject of a certain type of attack, some administrators of deployments may quickly change the configuration of their firewalls in an attempt to thwart the attack. The recommendation system may use such changes in the configurations as a trigger to launch the generating of recommendations for changing the configurations of firewalls of other deployments. … The recommendation system generates a recommendation relating to the configuration of the firewall for other deployments and

> provides the recommendation for each deployment….In some embodiments, a deployment may authorize the automatic implementing of recommendations**.**

Hudis, [0024]; [0025] ("As other deployments within the data center change their configurations to thwart an attack, the recommendation system may direct the target deployment to change its configuration accordingly."); Franz, ¶¶203-204.

Shanks is consistent with Hudis. Shanks teaches a cloud management system that receives security event data from CCEs and analyzes this aggregated information to determine if security attacks are likely to occur on other CCEs, so as to combat the threats. For example:

> In response to determining that the security attack detected on at least one client computing environment from the first set of client computing environments is likely to occur on other client computing environments, the cloud management device can identify a second set of client computing environments to protect from the security attack. For each client computing environment from the second set of client computing environments, the cloud management device can configure firewall settings to protect from the security attack.

Shanks, Abstract. As discussed in the Shanks overview, Shanks's cloud management system allows users to manage CCEs and set configurations including security settings, firewall settings, whitelists, and blacklists**.** Shanks, [0059]. Shanks further teaches with regards to FIG. 5, reproduced below, that "[c]loud management system 502 can be a cloud based computing system comprising one

or more computing device (e.g., cloud management devices) that allows users to remotely manage their client computing environments." Shanks, [0061]. Shanks further teaches that each CCE "can include multiple computing nodes, such as firewalls, servers, routers, access points, etc." *Id*., [0061]; Franz, ¶¶205-206.

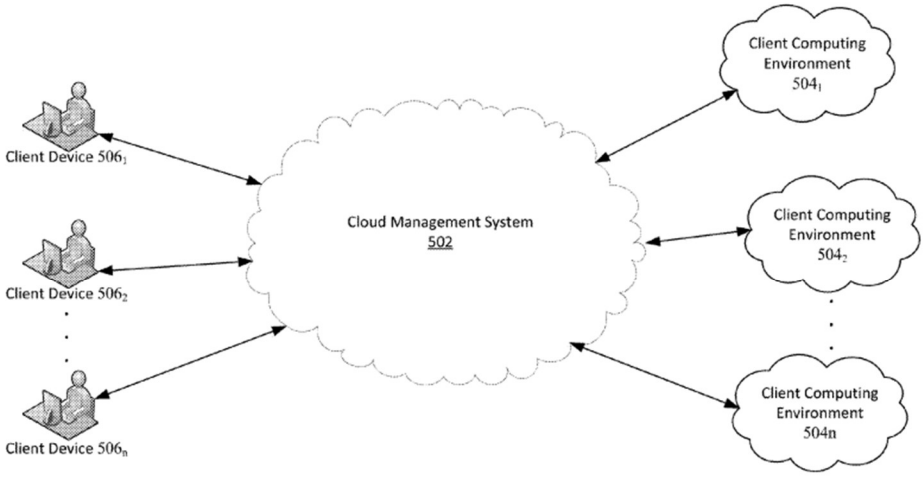


FIG. 5

Shanks's CCEs are "cloud deployments"—FIG. 5 uses cloud icons to illustrate the CCEs. *See* Shanks, [0021], [0061], FIG. 5; Franz, ¶¶205-207.

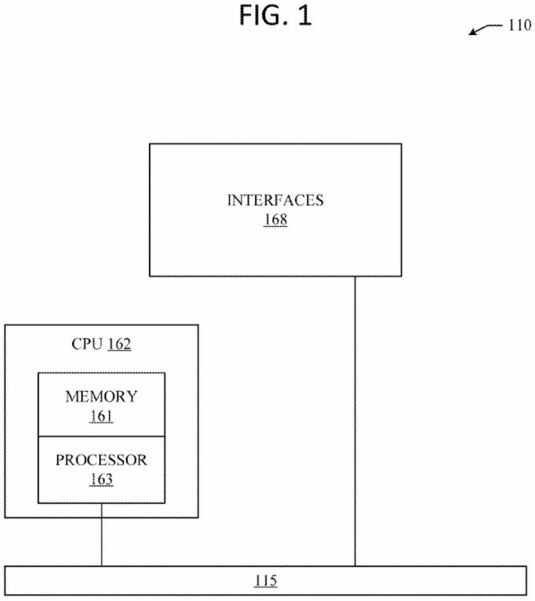
Shanks teaches that CCEs continuously transmit security event data to the cloud management system, which analyzes the data to identify security attacks, determines whether they are likely to occur on other CCEs, and automatically configures firewall settings to protect those CCEs. Shanks, [0060], [0063]; Franz, ¶¶207-208.

Shanks further discloses that "[a] security event can be any type of observable change to the normal behavior of a computing system, device,

environment, process, workflow, etc., that can indicate a malicious security breach." Shanks, [0063]. Shanks's disclosure of identifying security attacks/security events, by analyzing security event data that the client computing networks continuously transmit to the cloud management system, corresponds precisely to "learning from other cloud deployments to combat security threats" as recited in 1[pre]/9[pre]/17[pre]. Franz, ¶208.

Hudis teaches that the recommendation system executes on a computing system including a processor, memory, storage devices, and network interfaces, and may access computer-readable storage media. Hudis, [0033]-[0034], [0045], [0047]; claim 11. Shanks likewise teaches a system including a processor and memory storing instructions that cause the cloud management system to perform the disclosed steps. Shanks, [0023], [0078], claim 8; FIG. 1; Franz, ¶¶209-211.

FIG. 1



Shanks also discloses non-transitory computer-readable storage media including computer-executable instructions that perform the recited steps. Shanks, [0078], [0083]-[0084]; Franz, ¶¶211-212.

1[a]/9[a]/17[a]

As discussed in the Hudis overview, Hudis teaches a recommendation system that collects feature values from cloud deployments, generates models based on similarity, and applies those models to recommend feature values for target deployments. As part of the recommendation system, Hudis discloses identifying cloud deployments to utilize for cross-customer learning. Hudis, [0019]-[0021].

Hudis provides an example of cross-deployment learning:

> [I]f the features of the target deployment are most similar to features of deployments that use a firewall with a certain name, then the recommendation system recommends that the target deployment use the firewall with that certain name. The recommendation system thus performs collaborative filtering to base recommendations on deployments with similar features.

Hudis, [0020].

To identify deployments for cross-learning, the system can generate separate models for clusters of similar deployments using k-means or k-nearest neighbor clustering, identify the most similar cluster to a target deployment, and apply that cluster's model to generate the recommendation. Hudis, [0021]-[0023]. Additionally, Hudis discloses that the system can identify the cluster "to which the target deployment is most similar" based on cosine similarity between representative feature vectors, and then apply the model for that cluster to generate the recommendation. *Id.*, [0023]; Franz, ¶¶215-219.

As discussed in the Hudis overview, Hudis teaches applying this cross-deployment learning in the context of thwarting security attacks: the recommendation system uses changes in firewall configurations as a trigger to generate and provide recommendations for other deployments. *Id.*, [0024]-[0025]; Franz, ¶220.

Shanks's cloud management system is consistent with Hudis'. As discussed in the Shanks overview, Shanks teaches that the cloud management system

receives security event data from CCEs, analyzes it to identify security attacks, identifies a second set of CCEs to protect, and configures firewall settings for those CCEs. Shanks, [0060], [0064], [0071], [0073]; Franz, ¶¶221-22.

Shanks further teaches that CCEs continuously transmit security event data—including "any type of data describing security events detected in client computing devices"—to the cloud management system, which analyzes the data to identify attacks likely to occur on other CCEs. Shanks, [0063]-[0064]. Shanks teaches that CCEs ("cloud deployments") continuously transmit, to cloud management system, security data describing security events that can indicate a malicious security breach. Franz, ¶¶223-26.

Shanks teaches that cloud management system identifies a second set of CCEs to protect—for example, those that have opted in to automatic firewall configurations—by analyzing vulnerability scores and determining whether they exceed a threshold. Shanks, [0064], [0073]. Shanks discloses that "customers (e.g., businesses, schools, government institutions, etc.) that utilize cloud management system 502 to manage their client computing environment may select to have their firewalls automatically configured to protect from likely security threats." Shanks, [0068]. This opt-in mechanism corresponds to "cross-customer learning." Franz, ¶227.

1[b]/9[b]/17[b]

The ’695 patent provides the following examples of information describing a security threat to a cloud deployment:

> The information describing a security threat to one or more of the additional cloud deployments 912 can include, for example, information describing where an attack originated from, information describing the components within the cloud deployment that were targeted by an attack, information describing data access patterns that were associated with an attack, or some other information that would be useful in detecting subsequent security threats.

’695 patent, 81:31-38; Franz, ¶230.

As discussed in the Hudis overview, Hudis teaches that when deployments are subject to an attack, the system monitors configuration changes and generates recommendations. Hudis, [0024]-[0025]. Hudis also discloses collecting “various indicators of compromise regarding attacks and threat intelligence from the deployments.” *Id.*, [0040]; Franz, ¶231.

While Hudis does not expressly disclose receiving information “describing” a security threat, it is suggested by Hudis. Hudis discloses generating recommendations with configuration settings to thwart an attack. Hudis, [0024]-[0025]. Hudis also discloses collecting “various indicators of compromise regarding attacks and threat intelligence from the deployments” and IP addresses that should be blocked. *Id.*, [0040] (“In block 907, the component collects various indicators of compromise regarding attacks and threat intelligence from the

deployment and then completes. The component may, for example, analyze usage patterns to determine whether an attack (e.g., denial of service) was attempted. A deployment may purchase threat intelligence that, for example, identifies Internet Protocol ("IP") addresses whose access should be blocked by a firewall."). Thus, Hudis suggests collecting information describing an attack (e.g., indicators of compromise, threat intelligence) from cloud deployments. Indeed, such information would have been important to generating a recommendation for thwarting the attack (e.g., one would need to know the IP address to block before being able to recommend blocking it). Franz, ¶232.

Even if Hudis does not expressly teach this element, it would have been obvious in light of Shanks.

Shanks teaches that CCEs continuously transmit security event data to the cloud management system. Shanks, [0063]. Shanks teaches the received security event data can include "any type of data describing security events detected in client computing devices and/or data that can be used to derive security events that have occurred on a client computing device" and "[a] security event can be any type of observable change to the normal behavior of a computing system, device, environment, process, workflow, etc., that can indicate a malicious security breach." *Id*. Shanks further discloses applying firewall rules to block "a source associated with the attack" or "destination port associated with the security attack,"

which naturally requires receiving information describing the attack from the deployment that experienced it. Shanks, [0069]. This data correspond to "information describing a security threat to one or more of the additional cloud deployments" as in 1[b]/9[b]/17[b]. Franz, ¶¶233-37.

A POSITA would have been motivated to combine the teachings of Hudis and Shanks for the reasons discussed above. In the combination, Shanks' teachings about security event data would be part of the information that Hudis' recommendation system collects from each cloud deployment. Indeed, Hudis already discloses collecting indicators of compromise and threat intelligence, like blacklisted IP addresses. A POSITA would have been motivated to collect that information in real time during an attack because it would be most relevant at that time. Indeed, threat information sharing was so widespread before the '695 patent that the STIX and TAXII standards were developed. Accordingly, it would have been obvious that the recommendation system—and, ultimately, any cloud management system, for example one operated by an appliance provider, and target deployment—would receive information about the attack (e.g., source IP address, targeted firewall, attack pattern) to increase the chances of thwarting it for others. Franz, ¶238.

1[c]/9[c]/17[c]

The ’695 patent provides examples of configuration settings, including configuration settings that caused cloud deployments to blacklist IP addresses or network domains or to close vulnerabilities. ’695 patent, 81:43-51; Franz, ¶241.

Hudis teaches that deployments have “various parameters or configuration settings” that are part of the deployment’s configuration, including parameters for security sub-systems such as “a firewall appliance, an anti-virus appliance, an encryption appliance, and so on.” Hudis, [0002]; Franz, ¶242.

Hudis discloses administrators changing firewall configurations to thwart an attack. Hudis, [0024]. Hudis teaches that the configuration settings used to thwart the attack can be shared with other deployments. *Id.*, (“The recommendation system generates a recommendation relating to the configuration of the firewall for other deployments and provides the recommendation for each deployment.”); [0025] (“As other deployments within the data center change their configurations to thwart an attack, ***the recommendation system may direct the target deployment to change its configuration accordingly***.”); Franz, ¶243.

Hudis further discloses gathering indicators of compromise and threat intelligence—such as IP addresses warranting blocking—from cloud deployments and sharing that information with others. Hudis, [0040] (“A deployment may purchase threat intelligence that, for example, identifies Internet Protocol (“IP”) addresses whose access should be blocked by a firewall.”); Franz, ¶244.

Taken together, these disclosures confirm that both the system and the target cloud deployments receive information describing the configuration settings used to thwart an attack, and it would similarly have been obvious to route that information to a cloud security management system, such as one operated by an appliance provider, as discussed earlier. Franz, ¶245.

Shanks likewise teaches that its cloud management system can receive configurations—including security settings, firewall settings, whitelists, and blacklists—through a management interface and communicate with computing nodes to implement them. Shanks, [0059], [0062]. A POSITA would have understood that entering a new configuration through a management interface is functionally how the cloud management system "receives" that configuration, meaning the "management interface" of Shanks's cloud management system both transmits and receives configurations, such as firewall settings, for each CCE as a normal part of its operation. Franz, ¶246.

Shanks likewise teaches that cloud management device 502 communicates with CCEs to implement any "configurations, such as security settings, firewall settings, whitelists, blacklists, etc." *See* Shanks, [0062]. Shanks further teaches "[t]he cloud management system can communicate with computing nodes in client computing environments to implement any received configurations." Shanks, [0059]. A POSITA would readily have understood this to mean that the cloud

management device receives configurations from customers reacting to a security attack and stores them so that other customers can use the same configurations to combat that attack. *See* Shanks, [0068]-[0069]; Franz, ¶247.

1[d]/9[d]/17[d]

As discussed in the Hudis overview, when several deployments are subject to a certain type of attack, the recommendation system can use changes in their firewall configurations as a trigger to generate recommendations for other deployments. Hudis, [0024]. The system then "generates a recommendation relating to the configuration of the firewall for other deployments and provides the recommendation for each deployment." *Id.* If elected, the system can automatically implement the recommended configuration. *Id.*, [0025]. Even more broadly, Hudis teaches that the recommendation system "applies the model to the features of the target deployment to identify a target feature value for the target feature" and "provides the identified target feature value as a recommendation for the target feature for the target deployment." Hudis, [0004]; Franz, ¶250.

Also, Hudis teaches that the system can "direct a configuration manager of the deployment to change the configuration to automatically implement the recommended configuration without the explicit approval of an administrator of the deployment." Hudis, [0024]. This automatic implementation confirms that the system not only identifies configurations to adopt but can also cause them to be

adopted. It would have been obvious to a POSITA that, in order to recommend and automatically implement a configuration in a target deployment, the system would have needed to identify a configuration setting to adopt from the configuration settings received from the various cloud deployments. Additionally, where the deployment is not configured for automatic configuration, an administrator would obviously have considered and adopted one or more recommendations to avoid being left open to the attack. Thus, Hudis discloses "identifying...one or more configurations to adopt for the first cloud deployment." Franz, ¶¶250.

Shanks teaches that the cloud management system "automatically configure[s] firewall setting on client computing environments 506 to protect from the security attack," including applying firewall rules to block sources or destination ports associated with the attack. Shanks, [0068]-[0069]; *see also id*., [0060]. By applying firewall rules at CCEs to prevent the security attack, Shanks teaches that the cloud management system identifies configuration settings (firewall rules) to combat the security threat. Franz, ¶252.

A POSITA would have been motivated to combine Hudis and Shanks. In the combination of the two references, this step of identifying could be performed by the recommendation system, cloud [security] management system operated by the appliance provider, or target deployment, depending on the configuration. Franz, ¶253.

5. Claims 2/10/18

2[pre]/10[pre]/18[pre]

Claim 2 depends from claim 1; claim 10 depends from claim 9; and claim 18 depends from claim 17. The Hudis-Shanks combination teaches claims 1/9/17 as explained above.

2[a]/10[a]/18[a]

Shanks teaches that the cloud management system's management interface "allows users to set configurations, such as security settings, firewall settings, whitelists, blacklists, etc., for their respective client computing environment." Shanks, [0062]. Shanks further teaches that "in response to determining that a detected security attack is likely to occur on other client computing environments, cloud management system 502 can automatically configure firewall setting on client computing environments 506 to protect from the security attack." Shanks, [0068]. Shanks provides various examples of "firewall rules to prevent the security attack" that the cloud management system can apply; for example, blocking or filtering packets from a suspicious source or destination port. Shanks, [0069]; *see also id.*, [0074]; Franz, ¶257.

Additionally, Hudis teaches that deployments can "purchase threat intelligence that, for example, identifies Internet Protocol ("IP") addresses whose access should be blocked by a firewall." Hudis, [0040]; Franz, ¶258.

Given these disclosures—spanning blacklists, firewall rules, and IP addresses flagged for blocking—a POSITA would have recognized that the configuration setting at issue identifies an IP address to add to a blacklist, IP blacklisting being among the most basic and widely used defensive measures in network security for decades. Franz, ¶259.

6. Claims 3/11/19

3[pre]/11[pre]/19[pre]

Claim 3 depends from claim 1; claim 11 depends from claim 9; and claim 19 depends from claim 17. The Hudis-Shanks combination teaches claims 1/9/17 as explained above.

3[a]/11[a]/19[a]

Shanks's and Hudis's teachings regarding firewall rules are discussed with respect to 2[a]/11[a]/19[a]. *See, e.g.*, Shanks, [0069], [0074]; Franz, ¶263.

As already noted for elements 2[a]/11[a]/19[a], Hudis teaches gathering Indicators of Compromise from deployments and sharing them with others. Before the '695 patent, well-known indicators of compromise and threat intelligence included suspicious or malicious "network domains," as reflected, for example, in NIST's 2016 guide to threat information sharing, which identified "a suspicious Domain Name System (DNS) domain name" and "a Uniform Resource Locator

(URL) that references malicious content" as example indicators. Ex. 1015 at 10; Franz, ¶264.

It follows that a POSITA would have expected the threat intelligence shared with other deployments to include malicious network domains for blocking, since domain-based blocking had long been a routine defensive measure by 2017, and any network-security engineer would have anticipated a threat-sharing system communicating that kind of intelligence. Franz, ¶265.

7. Claims 5/13

5[pre]/13[pre]

Claim 5 depends from claim 1 and claim 13 depends from claim 9. The Hudis-Shanks combination teaches claims 1/9 as explained above.

5[a]/13[a]

The '695 patent describes examples of "actions" to include user interactions (such as which resources are accessed by particular users or business units) and data communications exchanged with external actors. '695 patent, 79:41-46, 79:51-59; Franz, ¶¶269, 272.

Hudis discloses receiving information describing actions associated with cloud deployments, teaching that its recommendation system "may monitor the features of the deployments in real time and provide recommendations based on changes in the features of the deployments." Hudis, [0024]. From such teachings, a

POSITA would have understood an administrator's act of changing a firewall configuration to be exactly such a user action, continuously monitored by the Hudis system in real time. Franz, ¶270.

Hudis further teaches collecting "usage patterns of the resources" (including "patterns of users accessing the deployment, patterns of applications accessing resources of the deployment") and "logs of the resources and the appliances" ("the logs may indicate when each user logged on to an application of a deployment"). Hudis, [0040]. These usage patterns and logs describe actions taken by users and applications within the cloud deployments—they constitute "information describing one or more actions associated with the additional cloud deployments" as recited in 5[a]/13[a]. Franz, ¶271.

Shanks teaches that "security event data" can include "any type of data describing security events detected in client computing devices and/or data that can be used to derive security events that have occurred on a client computing device" and that a "security event can be any type of observable change to the normal behavior of a computing system, device, environment, process, workflow, etc., that can indicate a malicious security breach." Hudis, [0063]. A POSITA would have understood or found obvious that such observable changes encompass communications with external actors. Franz, ¶273.

Sudden communication activity between a CCE and an external third party, for instance, would typically indicate either outbound communication from local malware to a command-and-control node or inbound exploitation of a software vulnerability by an outside party. Once such activity is confirmed as an attack, the natural response for protecting other CCEs would be to configure their firewalls to block further communication with that same external actor. Franz, ¶274.

This is exactly the type of information Shanks's cloud management device receives from the CCEs, which continuously transmit security event data that the system analyzes to identify attacks and prevent their recurrence elsewhere. *See* Shanks, [0060]; Franz, ¶275.

From such teachings, a POSITA would have understood that Shanks's "security event data" encompasses information describing "actions" associated with client computing devices. A POSITA would have recognized that such "actions" fall within Shanks's teachings of "information describing one or more actions associated with the additional cloud deployments." Put another way, a POSITA would have recognized that in the context of Shanks's cloud management system, the "actions" associated with a cloud deployment are a subset of the security event data the cloud management system is already designed to receive. Franz, ¶276.

5[b]/13[b]

As discussed for 1[d]/9[d]/17[d], the Hudis-Shanks combination teaches "identifying…one or more configurations to adopt for the first cloud deployment." And, as discussed for 5[a]/13[a], the Hudis-Shanks combination teaches that the received security event data describes one or more actions associated with the additional cloud deployments. Franz, ¶279.

Shanks teaches identifying configurations to adopt based on received security event data. Shanks uses that data to identify attacks, determine whether they are likely to occur on other CCEs, and automatically configure firewall settings. Shanks, [0060]. Because the security event data includes actions as per 5[a]/13[a], the Hudis-Shanks combination teaches 5[b]/13[b]. Additionally, as discussed for 5[a]/13[a], Hudis collects administrators changing firewall configurations. Naturally, the configuration to adopt in another cloud deployment would be based on that. Franz, ¶¶280-81.

8. Claims 8/16

8[pre]/16[pre]

Claim 8 depends from claim 1; and claim 16 depends from claim 9. The Hudis-Shanks combination teaches claims 1/9.

8[a]/16[a]

Fortinet's Infringement Contentions ("ICs," Ex 1016) allege that receiving information concerning the name of a file associated with an identified threat is

"insubstantially different from information concerning the deployment processes employed in the cloud deployment." ICs at 43; Franz, ¶284.

At the time of the alleged invention, file names, file extensions, and file content hashes were well-established indicators used to detect and mitigate security threats. A POSITA would have understood their use in identifying potential threats was conventional and had been in widespread use for decades. Franz, ¶285.

Although Shanks does not explicitly disclose using file names as part of "security event data," a POSITA would have well understood that file names, extensions, and hashes were valuable and usable information related to security threats and that their uses in detecting and mitigating such security threats were conventional and had been in widespread use for decades. Ex. 1015 at 10-11. To a POSITA, it would have been obvious to include file names, file extensions, and file content hashes among the set of "security event data" that one would use to detect and counteract a threat. Franz, ¶286.

IX. CONCLUSION

Accordingly, the Board should institute IPR and cancel the challenged claims.

Dated: September 30, 2026

Respectfully submitted,

SHEPPARD, MULLIN,
RICHTER & HAMPTON LLP

12275 El Camino Real, Ste 100
San Diego, California, 92130
Phone: (858) 720-8900
Fax: (858) 509-3691

/s/ *Thomas Millikan*
Thomas Millikan (Reg. No. 72,316)

Back-up Counsel
Joseph P. Reid (Reg. No. 72, 317)
Andrew Klein (*PHV* forthcoming)

Attorneys for Netskope, Inc.

CLAIM LISTING

Claim Element	Claim language
1[pre]	A method of learning from other cloud deployments to combat security threats, the method comprising:
1[a]	identifying, for at least a portion of a first cloud deployment, one or more additional cloud deployments to utilize for cross-customer learning;
1[b]	receiving information describing a security threat to one or more of the additional cloud deployments;
1[c]	receiving information describing configuration settings used to combat the security threat; and
1[d]	identifying, based on the information describing configuration settings used to combat the security threat, one or more configurations to adopt for the first cloud deployment.
2	The method of claim 1, wherein the configuration settings comprise configuration settings that caused the one or more of the additional cloud deployments to blacklist one or more internet protocol (IP) addresses.
3	The method of claim 1, wherein the configuration settings comprise configuration settings that caused the one or more of the additional cloud deployments to blacklist one or more network domains.
5	The method of claim 1 further comprising: receiving information describing one or more actions associated with the additional cloud deployments; and wherein identifying the one or more configurations to adopt for the first cloud deployment is further based on the one or more actions.
8	The method of claim 1 further comprising receiving information describing one or more deployment processes associated with the additional cloud deployments.
9[pre]	A system for learning from other cloud deployments to combat security threats, the system comprising at least one processor and memory storing computer program instructions that, when executed, cause the system to carry out the steps of:
9[a]	identifying, for at least a portion of a first cloud deployment, one or more additional cloud deployments to utilize for cross-

	customer learning;
9[b]	receiving information describing a security threat to one or more of the additional cloud deployments;
9[c]	receiving information describing configuration settings used to combat the security threat; and
9[d]	identifying, based on the information describing configuration settings used to combat the security threat, one or more configurations to adopt for the first cloud deployment.
10	The system of claim 9, wherein the configuration settings comprise configuration settings that caused the one or more of the additional cloud deployments to blacklist one or more internet protocol (IP) addresses.
11	The system of claim 9, wherein the configuration settings comprise configuration settings that caused the one or more of the additional cloud deployments to blacklist one or more network domains.
13	The system of claim 9, wherein the steps further comprise: receiving information describing one or more actions associated with the additional cloud deployments; and wherein identifying the one or more configurations to adopt for the first cloud deployment is further based on the one or more actions.
16	The system of claim 9 wherein the steps further comprise receiving information describing one or more deployment processes associated with the additional cloud deployments.
17[pre]	A computer program product for learning from other cloud deployments to combat security threats, the computer program product disposed on a non-transitory computer readable medium, the computer program product including computer program instructions that, when executed, carry out the steps of::
17[a]	identifying, for at least a portion of a first cloud deployment, one or more additional cloud deployments to utilize for cross-customer learning;
17[b]	receiving information describing a security threat to one or more of the additional cloud deployments;
17[c]	receiving information describing configuration settings used to combat the security threat; and
17[d]	identifying, based on the information describing configuration

	settings used to combat the security threat, one or more configurations to adopt for the first cloud deployment.
18	The computer program product of claim 17, wherein the configuration settings comprise configuration settings that caused the one or more of the additional cloud deployments to blacklist one or more internet protocol (IP) addresses.
19	The computer program product of claim 17, wherein the configuration settings comprise configuration settings that caused the one or more of the additional cloud deployments to blacklist one or more network domains.

CERTIFICATE OF WORD COUNT UNDER 37 CFR §42.24(D)

Under 37 C.F.R. §42.24(d), the undersigned certifies that the word count for this Petition for *Inter Partes* Review totals 13,475 words, excluding the parts exempted by 37 C.F.R. §42.24(a). The word count was made using the built-in word count function in the Microsoft® Word 365 software used to prepare this document.

Dated: September 30, 2026

Respectfully submitted,

12275 El Camino Real, Ste 100
San Diego, California, 92130
Phone: (858) 720-8900
Fax: (858) 509-3691

SHEPPARD, MULLIN,
RICHTER & HAMPTON LLP

/s/ *Thomas Millikan*
Thomas Millikan (Reg. No. 72,316)

Back-up Counsel
Joseph P. Reid, Reg. No. 72,317
Andrew Klein, PHV forthcoming
Jaime D. Choi, Reg. No. 59,198

Attorneys for Netskope, Inc.

CERTIFICATE OF SERVICE

The undersigned hereby certifies that true copies of this PETITION FOR *INTER PARTES* REVIEW and supporting materials (Exhibits 1001-1017 and Power of Attorney) have been served via USPS Express Mail and electronically on counsel for patent owner as follows:

Law Office of Dorian Cartwright
P.O. Box 6629
San Jose, CA 95150
UNITED STATES
info@cartwrightesq.com

In addition, the undersigned certifies that a true copy of the petition and supporting materials have been served on Patent Owner via email to its legal counsel:

Andrew Michael Holmes - drewholmes@quinnemanuel.com
Jordan Brock Kaericher - jordankaericher@quinnemanuel.com
Sean S. Pak - seanpak@quinnemanuel.com
Tigran Guledjian - tigranguledjian@quinnemanuel.com

Dated: September 30, 2026

Respectfully submitted,

12275 El Camino Real, Ste 100
San Diego, California, 92130
Phone: (858) 720-8900
Fax: (858) 509-3691

SHEPPARD, MULLIN,
RICHTER & HAMPTON LLP

/s/ Thomas Millikan
Thomas Millikan (Reg. No. 72,316)

Back-up Counsel
Joseph P. Reid, Reg. No. 72, 317
Andrew Klein, PHV forthcoming
Jaime D. Choi, Reg. No. 59,198