US 20250258816A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2025/0258816 A1**

Sharan et al. (43) **Pub. Date: Aug. 14, 2025**

(54) **PARALLEL AND DISTRIBUTED QUERY ENGINE FOLLOW-UP QUERY LANGUAGE**

(71) Applicants: **Dhiraj Sharan**, Brookings, SD (US); **Jeremy Fisher**, Brookings, SD (US); **Matt Eberhart**, Brookings, SD (US)

(72) Inventors: **Dhiraj Sharan**, Brookings, SD (US); **Jeremy Fisher**, Brookings, SD (US); **Matt Eberhart**, Brookings, SD (US)

(73) Assignee: **Query.AI, Inc.**, Atlanta, GA (US)

(21) Appl. No.: **19/195,355**

(22) Filed: **Apr. 30, 2025**

Related U.S. Application Data

(63) Continuation-in-part of application No. 18/222,998, filed on Jul. 17, 2023, now Pat. No. 12,111,830.

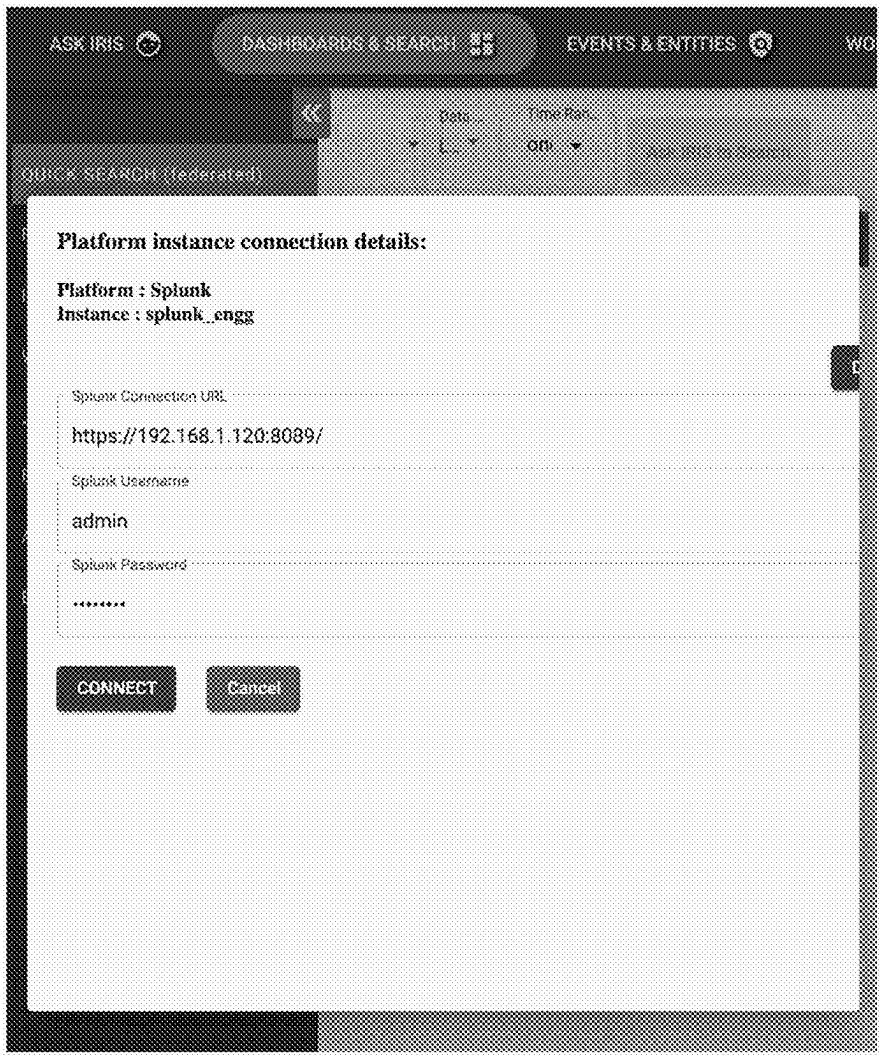
Publication Classification

(51) **Int. Cl.**
G06F 16/2453 (2019.01)
G06F 16/248 (2019.01)

(52) **U.S. Cl.**
CPC ***G06F 16/24532*** (2019.01); ***G06F 16/248*** (2019.01)

(57) **ABSTRACT**

A parallel and distributed query engine for federated searching is disclosed herein. As contemplated by the present disclosure, the system may provide a single application programming interface that allows a user to access and analyze multiple enterprise data storage locations remotely and simultaneously while presenting and reporting information from the multiple sources in a single, uniform display. Such a solution may allow a user to analyze and cross-reference data stored in multiple locations by using multiple queries in real time without requiring the actual data files to be displaced or combined. The system may further implement interactive artificial intelligence assistant, natural language processing, and workflow-based operations for improved user access and functionality.



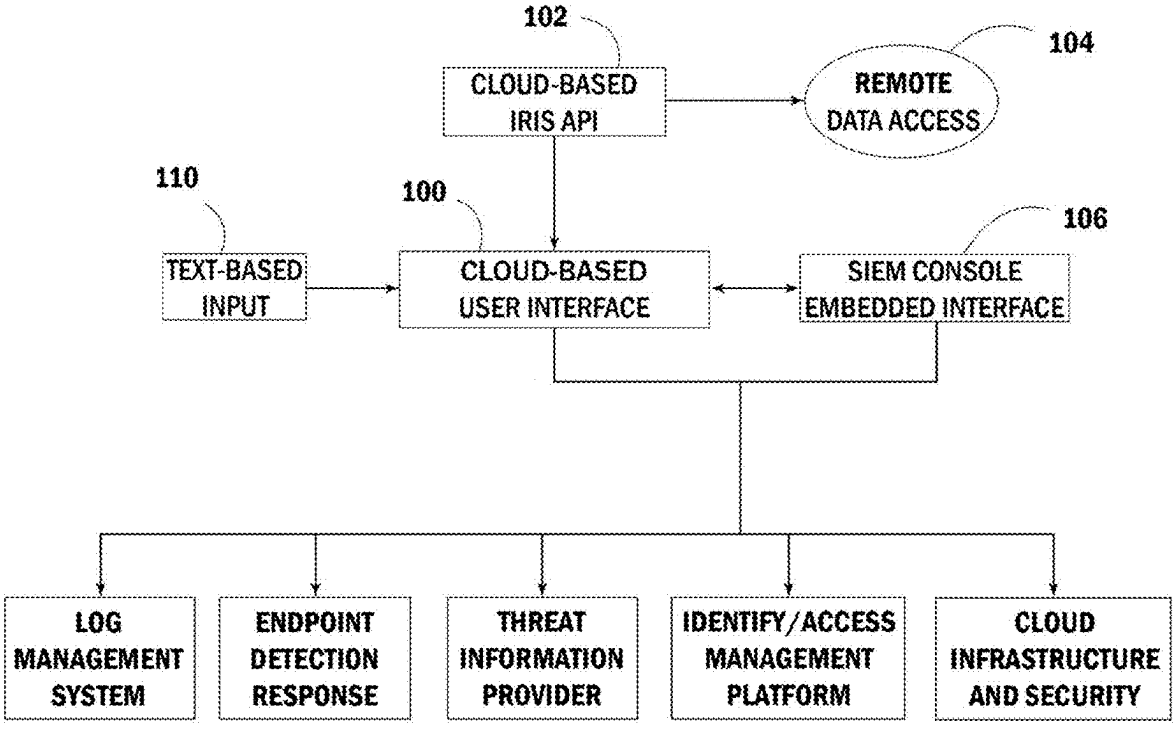

102
CLOUD-BASED IRIS API
104
REMOTE DATA ACCESS
110
TEXT-BASED INPUT
100
CLOUD-BASED USER INTERFACE
106
SIEM CONSOLE EMBEDDED INTERFACE
LOG MANAGEMENT SYSTEM
ENDPOINT DETECTION RESPONSE
THREAT INFORMATION PROVIDER
IDENTIFY/ACCESS MANAGEMENT PLATFORM
CLOUD INFRASTRUCTURE AND SECURITY
108

FIG. 1

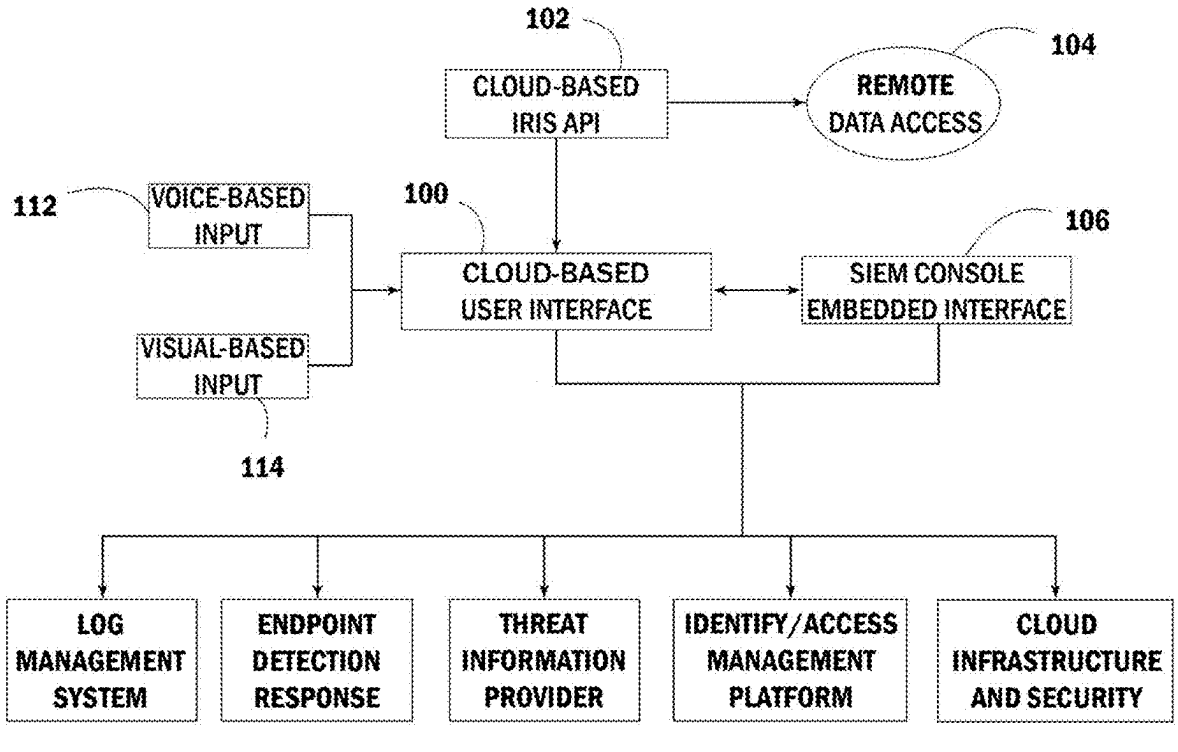

102
CLOUD-BASED IRIS API
104
REMOTE DATA ACCESS
112
VOICE-BASED INPUT
100
CLOUD-BASED USER INTERFACE
106
SIEM CONSOLE EMBEDDED INTERFACE
VISUAL-BASED INPUT
114
LOG MANAGEMENT SYSTEM
ENDPOINT DETECTION RESPONSE
THREAT INFORMATION PROVIDER
IDENTIFY/ACCESS MANAGEMENT PLATFORM
CLOUD INFRASTRUCTURE AND SECURITY
108

FIG. 2

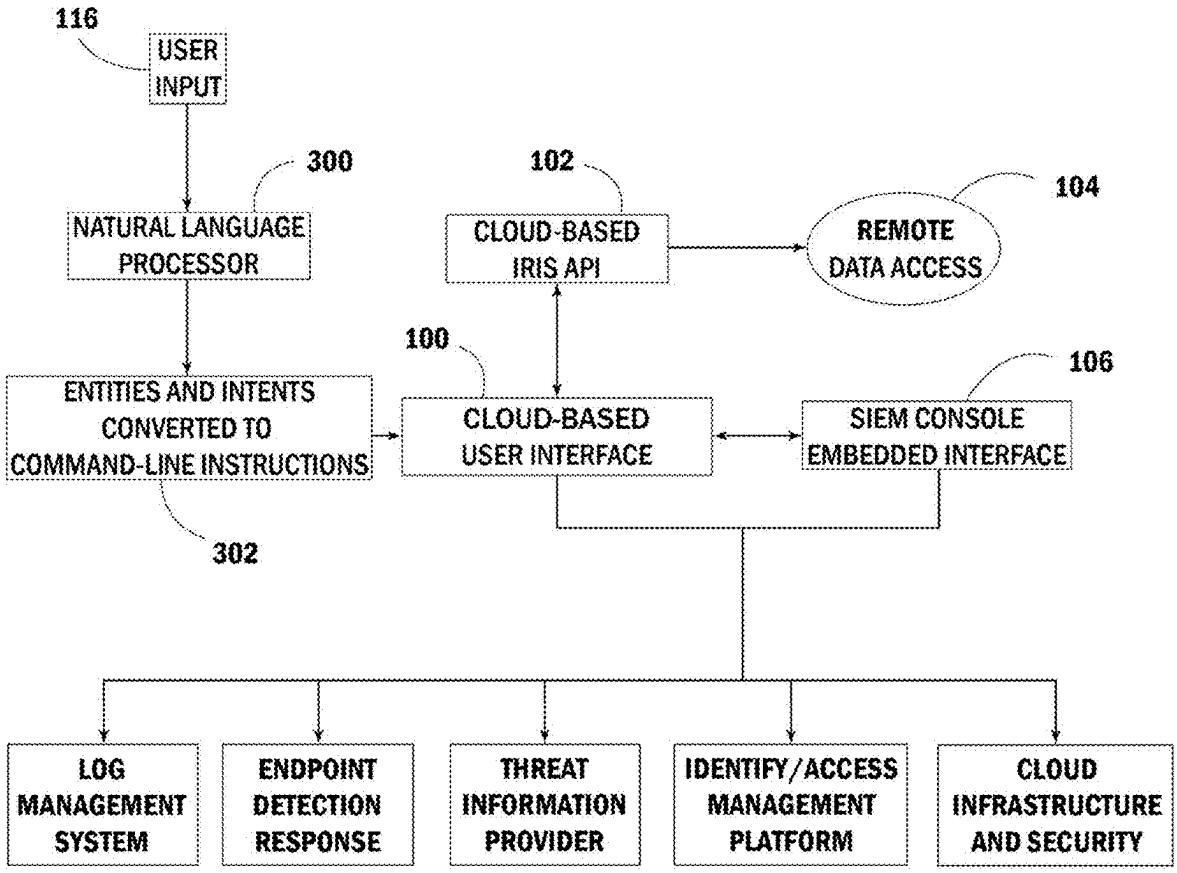

116
USER INPUT
300
NATURAL LANGUAGE PROCESSOR
102
CLOUD-BASED IRIS API
104
REMOTE DATA ACCESS
100
ENTITIES AND INTENTS CONVERTED TO COMMAND-LINE INSTRUCTIONS
CLOUD-BASED USER INTERFACE
106
SIEM CONSOLE EMBEDDED INTERFACE
302
LOG MANAGEMENT SYSTEM
ENDPOINT DETECTION RESPONSE
THREAT INFORMATION PROVIDER
IDENTIFY/ACCESS MANAGEMENT PLATFORM
CLOUD INFRASTRUCTURE AND SECURITY
108

FIG. 3

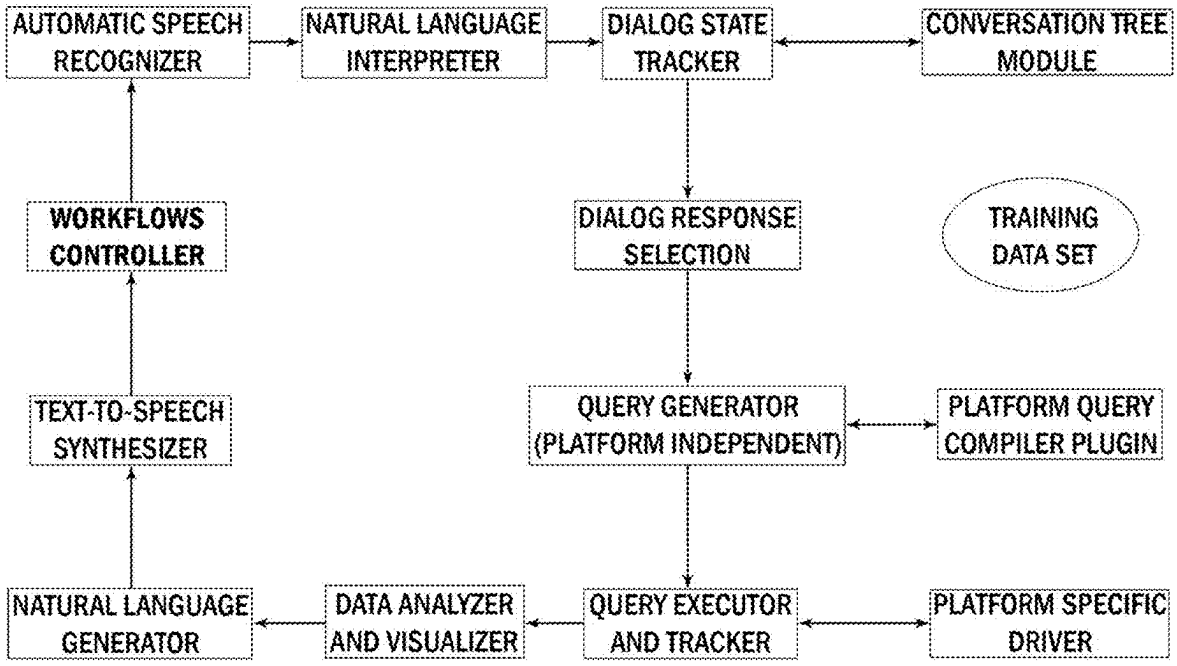

AUTOMATIC SPEECH RECOGNIZER
NATURAL LANGUAGE INTERPRETER
DIALOG STATE TRACKER
CONVERSATION TREE MODULE
WORKFLOWS CONTROLLER
DIALOG RESPONSE SELECTION
TRAINING DATA SET
TEXT-TO-SPEECH SYNTHESIZER
QUERY GENERATOR (PLATFORM INDEPENDENT)
PLATFORM QUERY COMPILER PLUGIN
NATURAL LANGUAGE GENERATOR
DATA ANALYZER AND VISUALIZER
QUERY EXECUTOR AND TRACKER
PLATFORM SPECIFIC DRIVER

FIG. 4

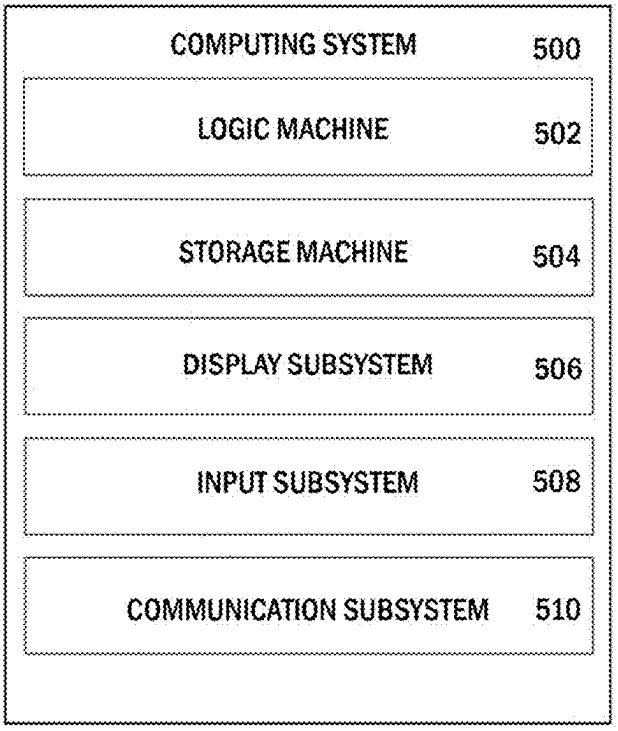

COMPUTING SYSTEM 500
LOGIC MACHINE 502
STORAGE MACHINE 504
DISPLAY SUBSYSTEM 506
INPUT SUBSYSTEM 508
COMMUNICATION SUBSYSTEM 510

FIG. 5

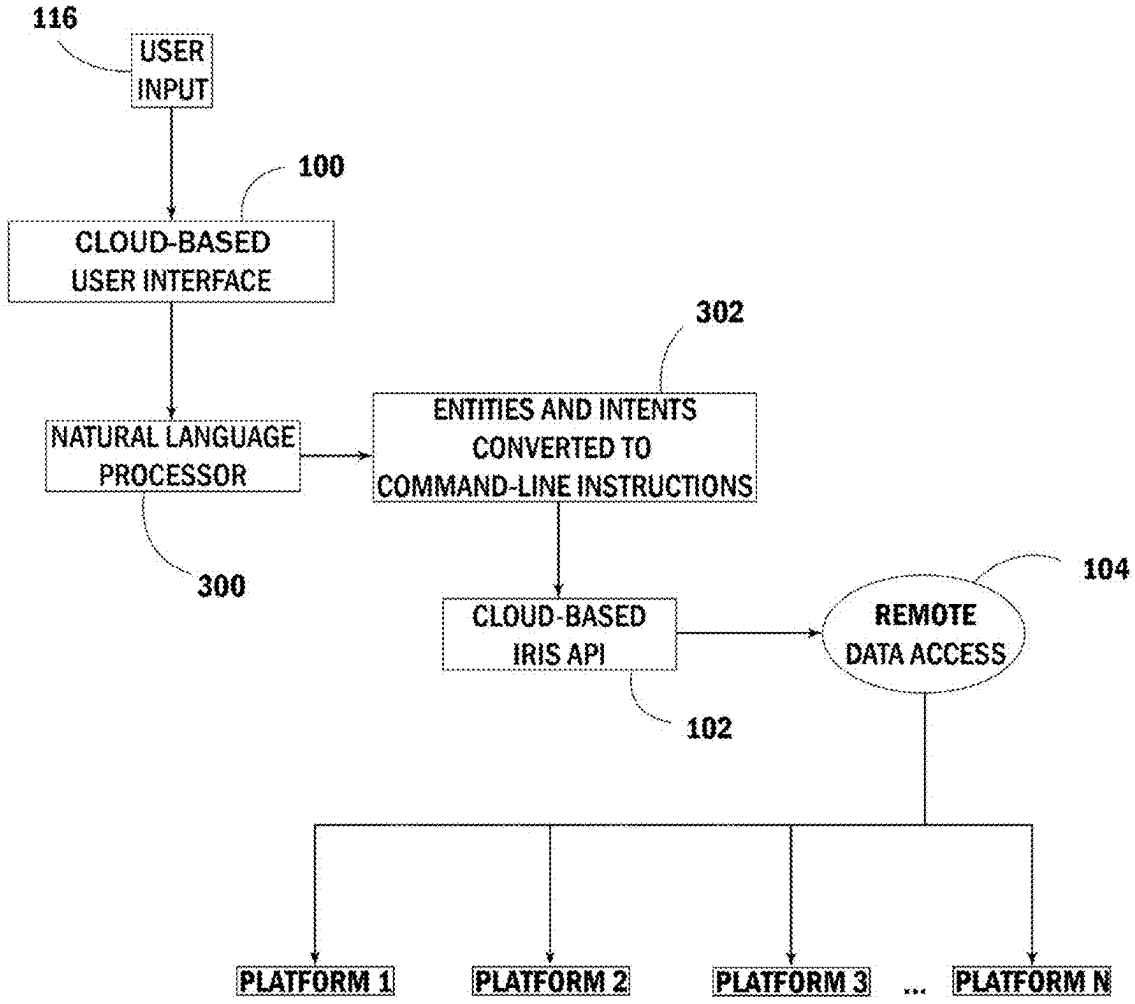

116
USER INPUT
100
CLOUD-BASED USER INTERFACE
302
NATURAL LANGUAGE PROCESSOR
ENTITIES AND INTENTS CONVERTED TO COMMAND-LINE INSTRUCTIONS
300
CLOUD-BASED IRIS API
104
REMOTE DATA ACCESS
102
PLATFORM 1
PLATFORM 2
PLATFORM 3
...
PLATFORM N

FIG. 6

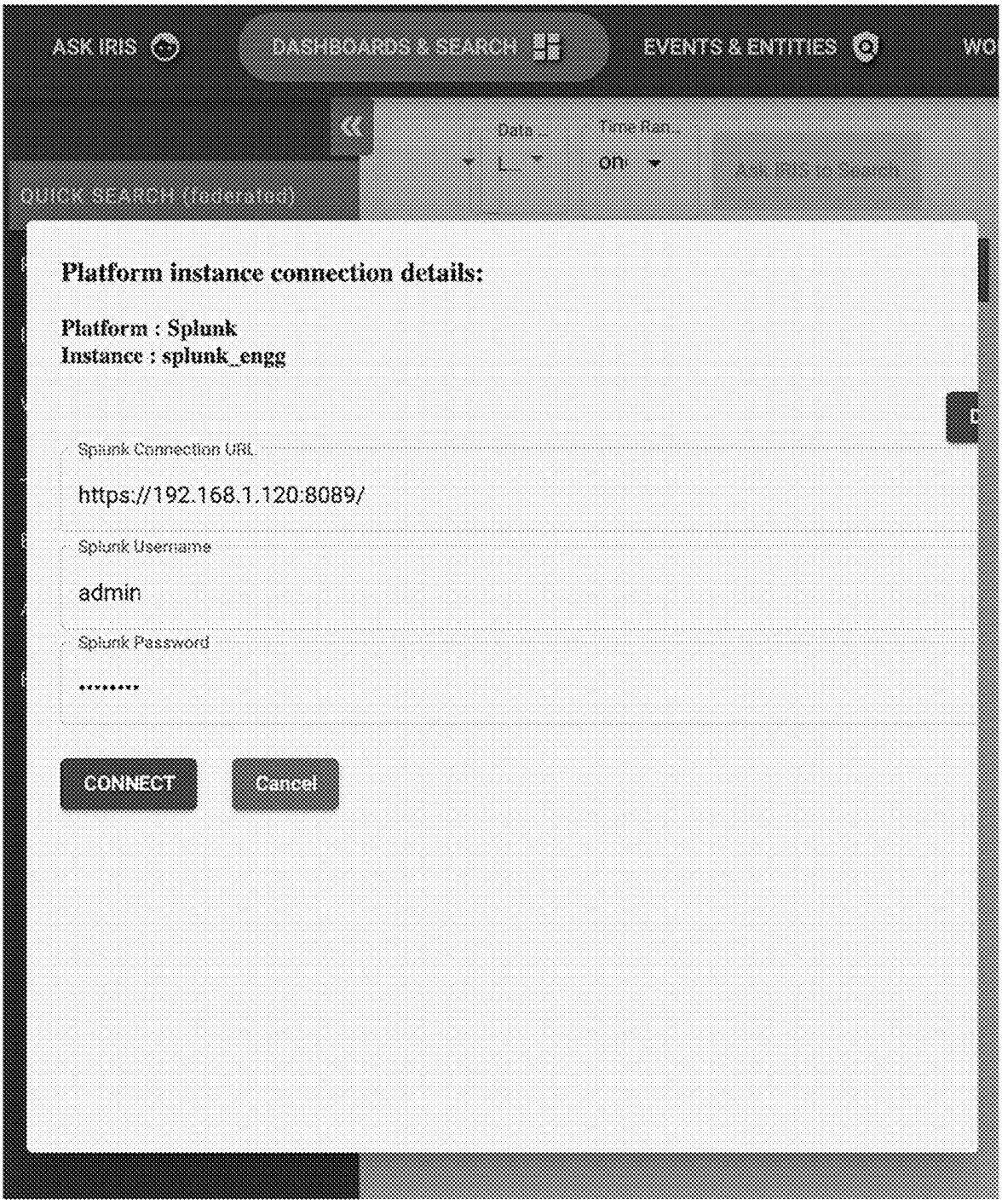

ASK IRIS
DASHBOARDS & SEARCH
EVENTS & ENTITIES
Platform instance connection details:
Platform : Splunk
Instance : splunk_engg
Splunk Connection URL
https://192.168.1.120:8089/
Splunk Username
admin
Splunk Password
••••••••
CONNECT
Cancel

FIG. 7

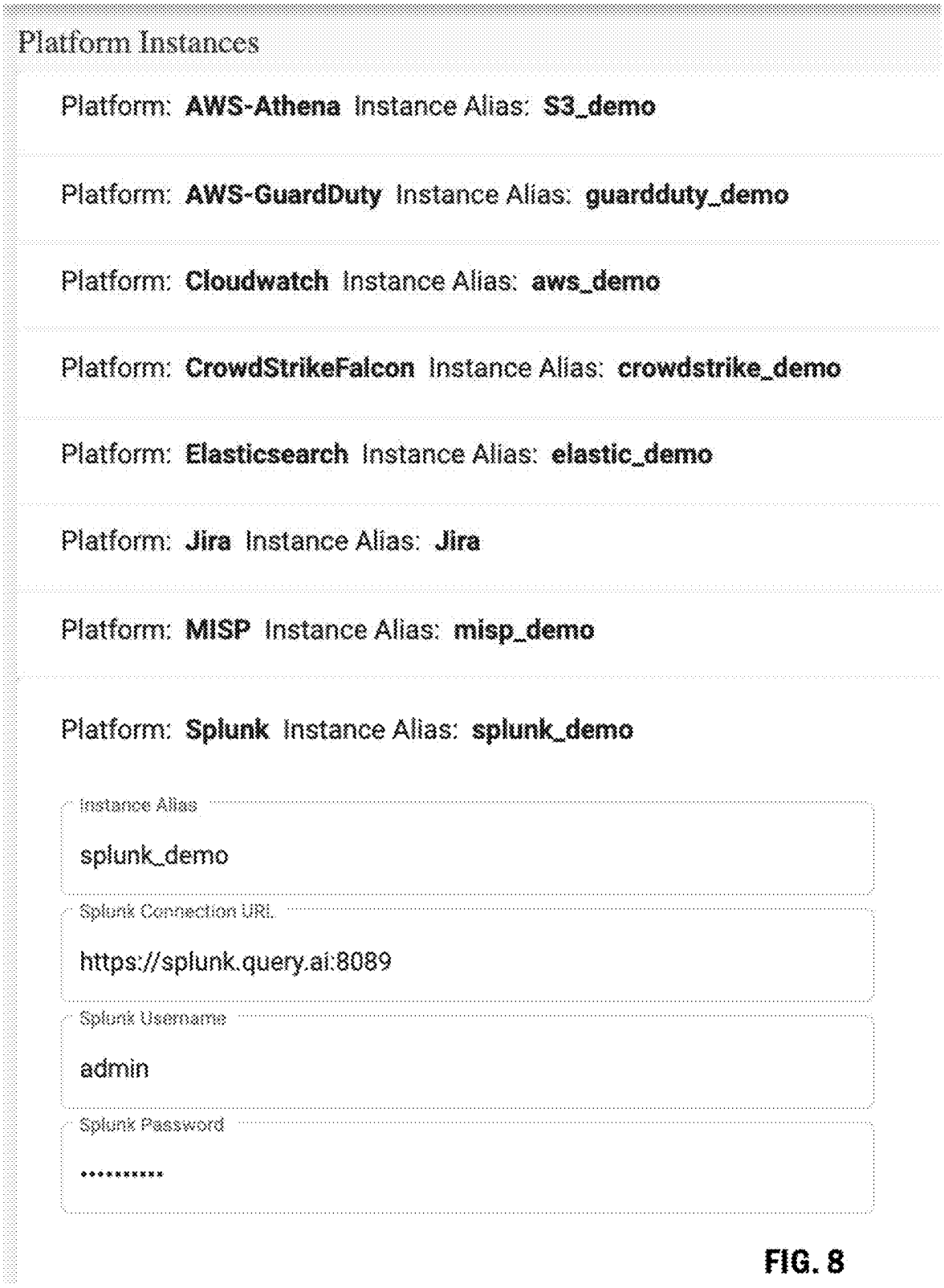

Platform Instances
Platform: AWS-Athena Instance Alias: S3_demo
Platform: AWS-GuardDuty Instance Alias: guardduty_demo
Platform: Cloudwatch Instance Alias: aws_demo
Platform: CrowdStrikeFalcon Instance Alias: crowdstrike_demo
Platform: Elasticsearch Instance Alias: elastic_demo
Platform: Jira Instance Alias: Jira
Platform: MISP Instance Alias: misp_demo
Platform: Splunk Instance Alias: splunk_demo
Instance Alias
splunk_demo
Splunk Connection URL
https://splunk.query.ai:8089
Splunk Username
admin
Splunk Password
••••••••••

FIG. 8

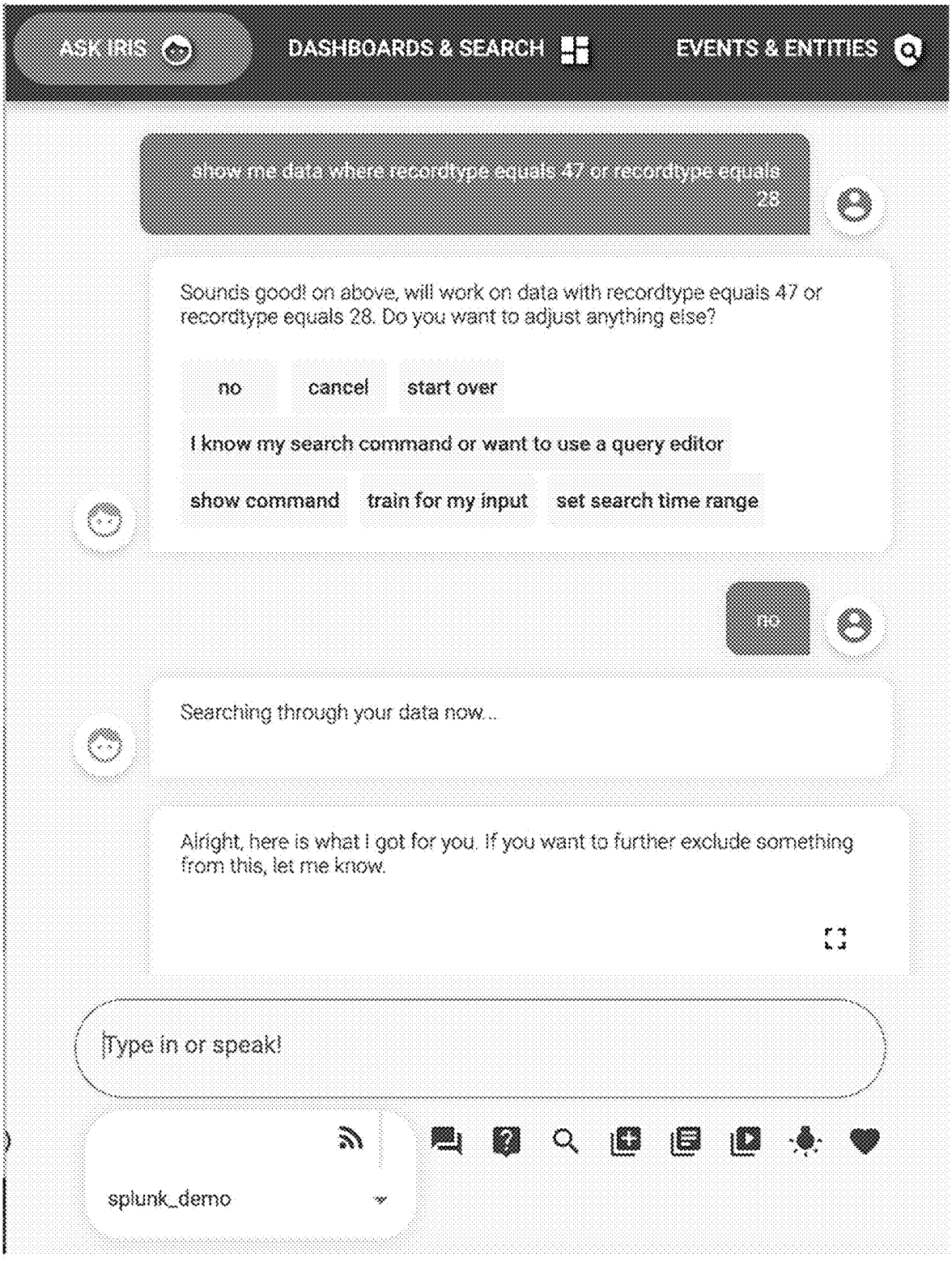

ASK IRIS
DASHBOARDS & SEARCH
EVENTS & ENTITIES
show me data where recordtype equals 47 or recordtype equals 28
Sounds good! on above, will work on data with recordtype equals 47 or recordtype equals 28. Do you want to adjust anything else?
no
cancel
start over
I know my search command or want to use a query editor
show command
train for my input
set search time range
no
Searching through your data now...
Airight, here is what I got for you. If you want to further exclude something from this, let me know.
Type in or speak!
splunk_demo

FIG. 9

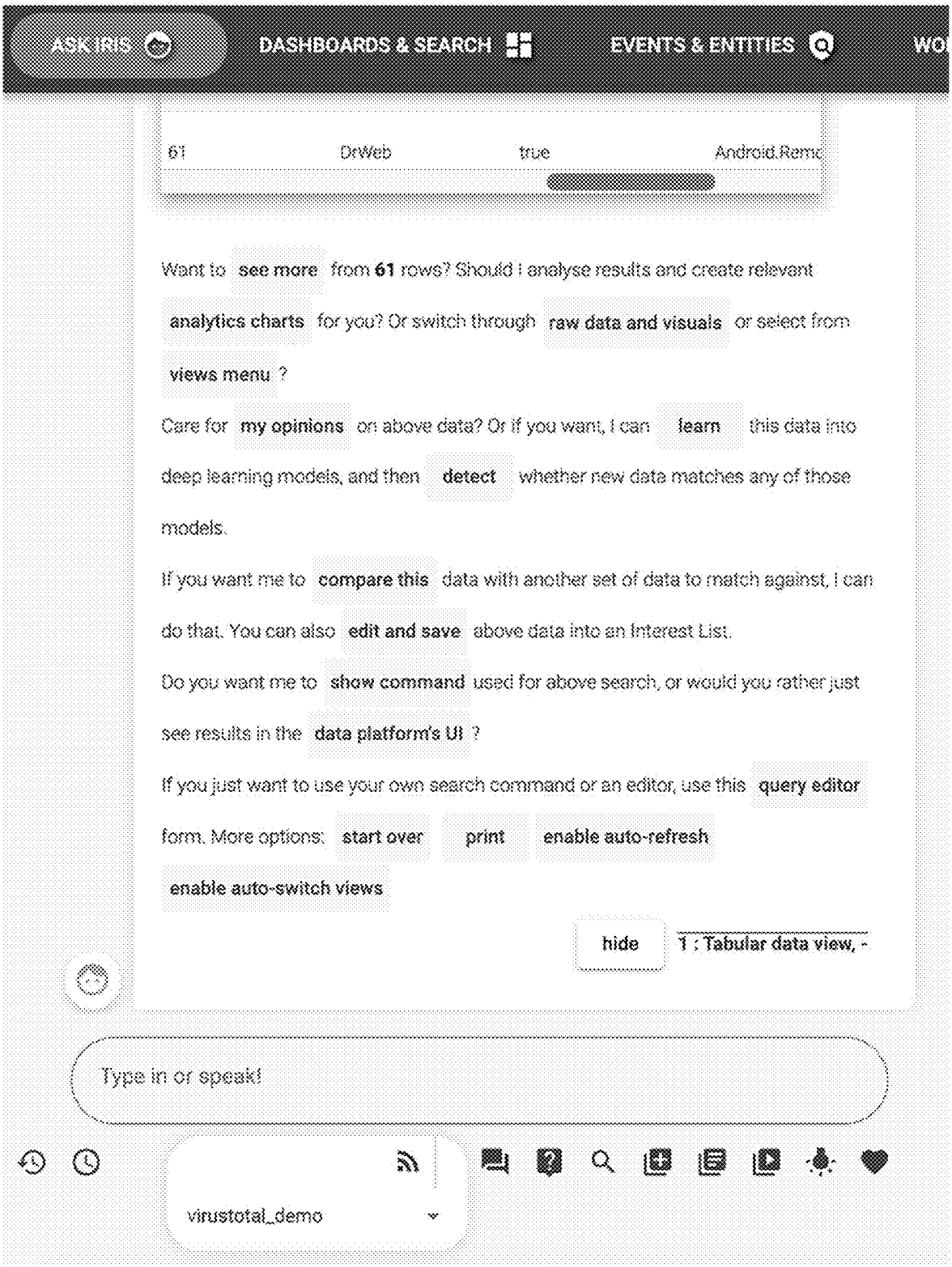

ASK IRIS
DASHBOARDS & SEARCH
EVENTS & ENTITIES
61 DrWeb true Android.Remo
Want to see more from 61 rows? Should I analyse results and create relevant analytics charts for you? Or switch through raw data and visuals or select from views menu ?
Care for my opinions on above data? Or if you want, I can learn this data into deep learning models, and then detect whether new data matches any of those models.
If you want me to compare this data with another set of data to match against, I can do that. You can also edit and save above data into an Interest List.
Do you want me to show command used for above search, or would you rather just see results in the data platform's UI ?
If you just want to use your own search command or an editor, use this query editor form. More options: start over print enable auto-refresh enable auto-switch views
hide 1 : Tabular data view, -
Type in or speak!
virustotal_demo

FIG. 10

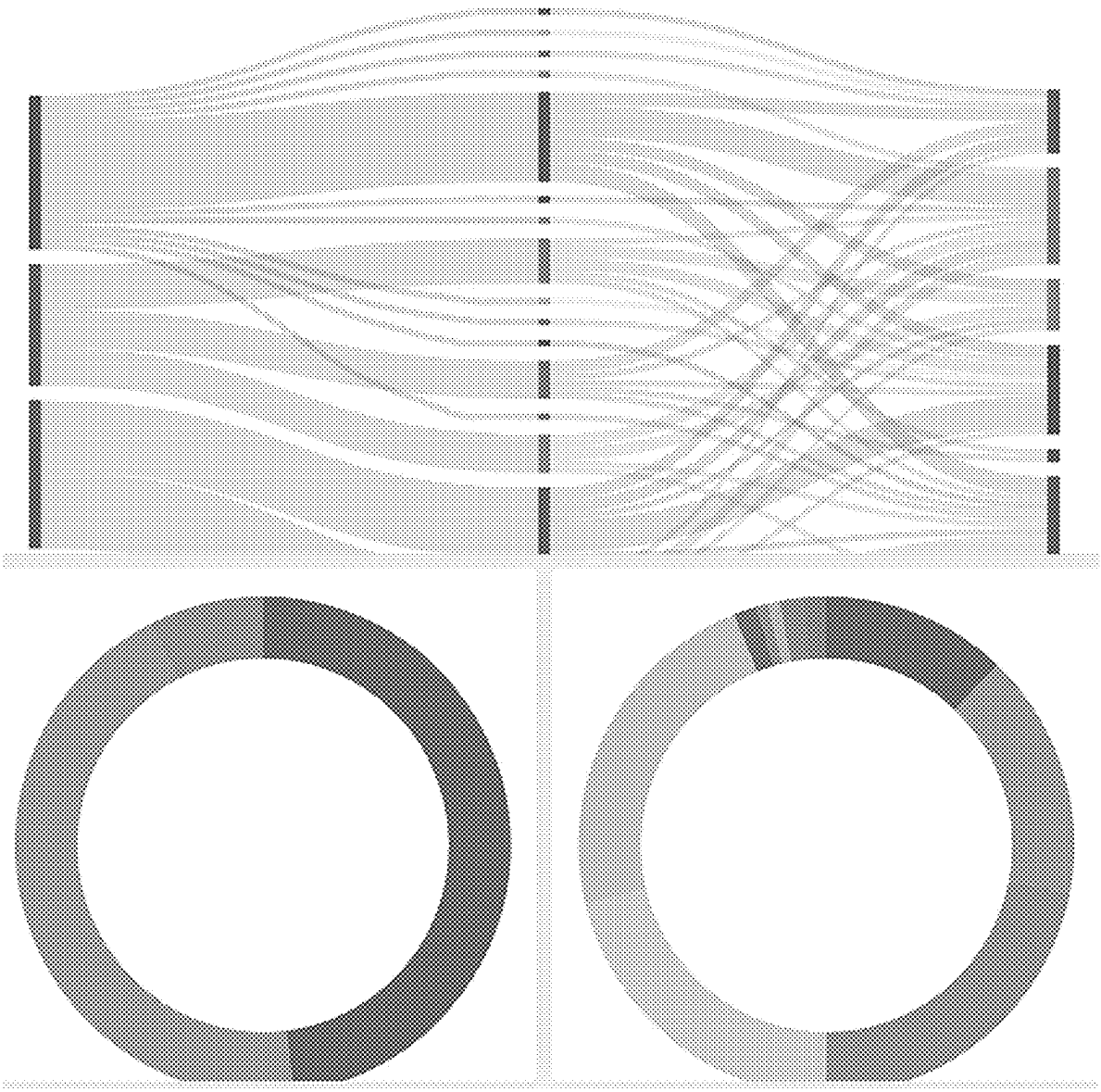


FIG. 11

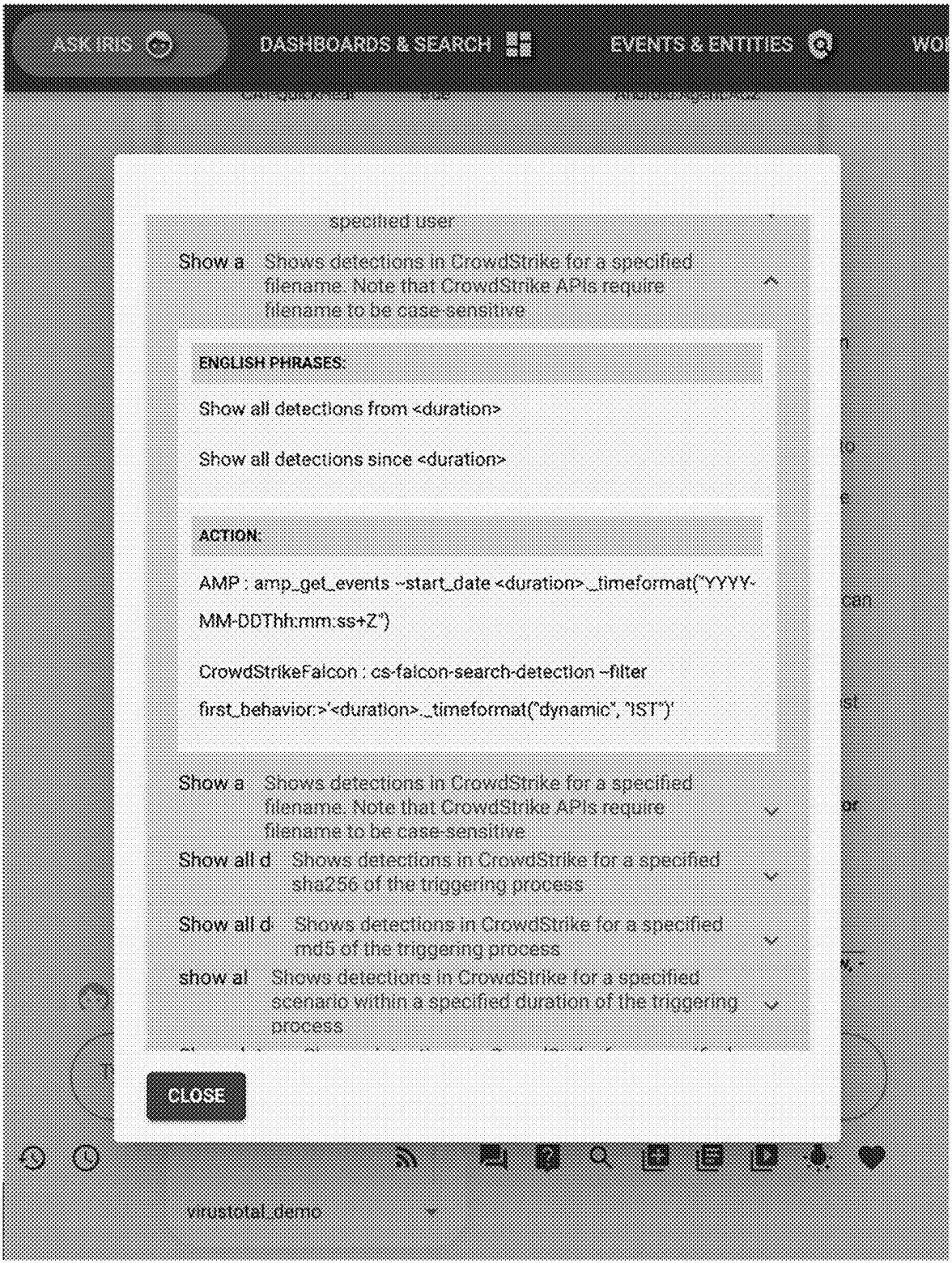

ASK IRIS
DASHBOARDS & SEARCH
EVENTS & ENTITIES
specified user
Show a Shows detections in CrowdStrike for a specified filename. Note that CrowdStrike APIs require filename to be case-sensitive
ENGLISH PHRASES:
Show all detections from <duration>
Show all detections since <duration>
ACTION:
AMP : amp_get_events --start_date <duration>._timeformat("YYYY-MM-DDThh:mm:ss+Z")
CrowdStrikeFalcon : cs-falcon-search-detection --filter first_behavior:>'<duration>._timeformat("dynamic", "IST")'
Show a Shows detections in CrowdStrike for a specified filename. Note that CrowdStrike APIs require filename to be case-sensitive
Show all d Shows detections in CrowdStrike for a specified sha256 of the triggering process
Show all d Shows detections in CrowdStrike for a specified md5 of the triggering process
show al Shows detections in CrowdStrike for a specified scenario within a specified duration of the triggering process
CLOSE
virustotal_demo

FIG. 12

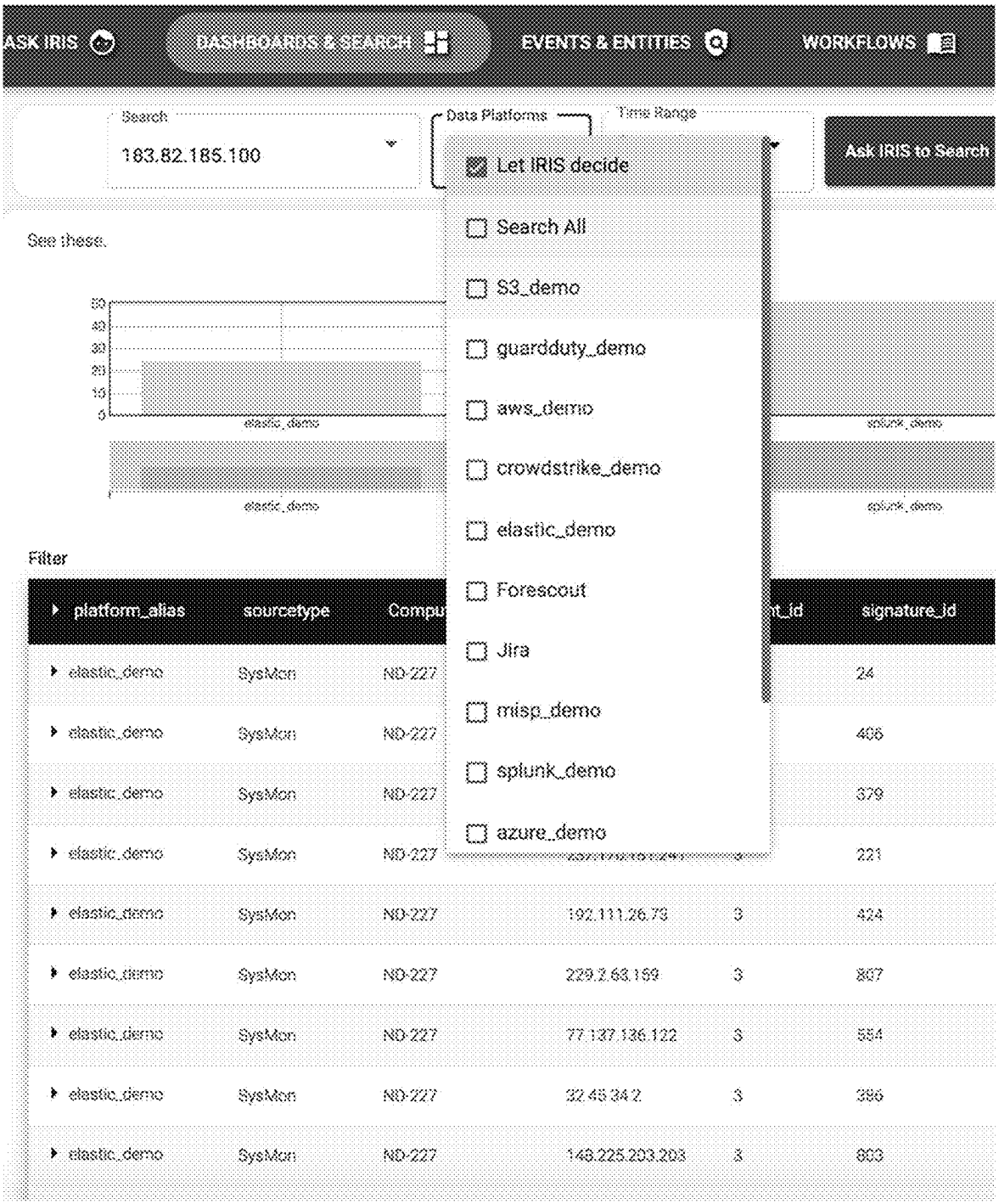

ASK IRIS
DASHBOARDS & SEARCH
EVENTS & ENTITIES
WORKFLOWS
Search
183.82.185.100
Data Platforms
Time Range
Ask IRIS to Search
Let IRIS decide
Search All
S3_demo
guardduty_demo
aws_demo
crowdstrike_demo
elastic_demo
Forescout
Jira
misp_demo
splunk_demo
azure_demo
See these.
Filter
platform_alias
sourcetype
signature_id
elastic_demo SysMon ND-227 24
elastic_demo SysMon ND-227 406
elastic_demo SysMon ND-227 379
elastic_demo SysMon ND-227 221
elastic_demo SysMon ND-227 192.111.26.73 3 424
elastic_demo SysMon ND-227 229.2.63.159 3 807
elastic_demo SysMon ND-227 77.137.136.122 3 554
elastic_demo SysMon ND-227 32.45.34.2 3 386
elastic_demo SysMon ND-227 148.225.203.203 3 803

FIG. 13

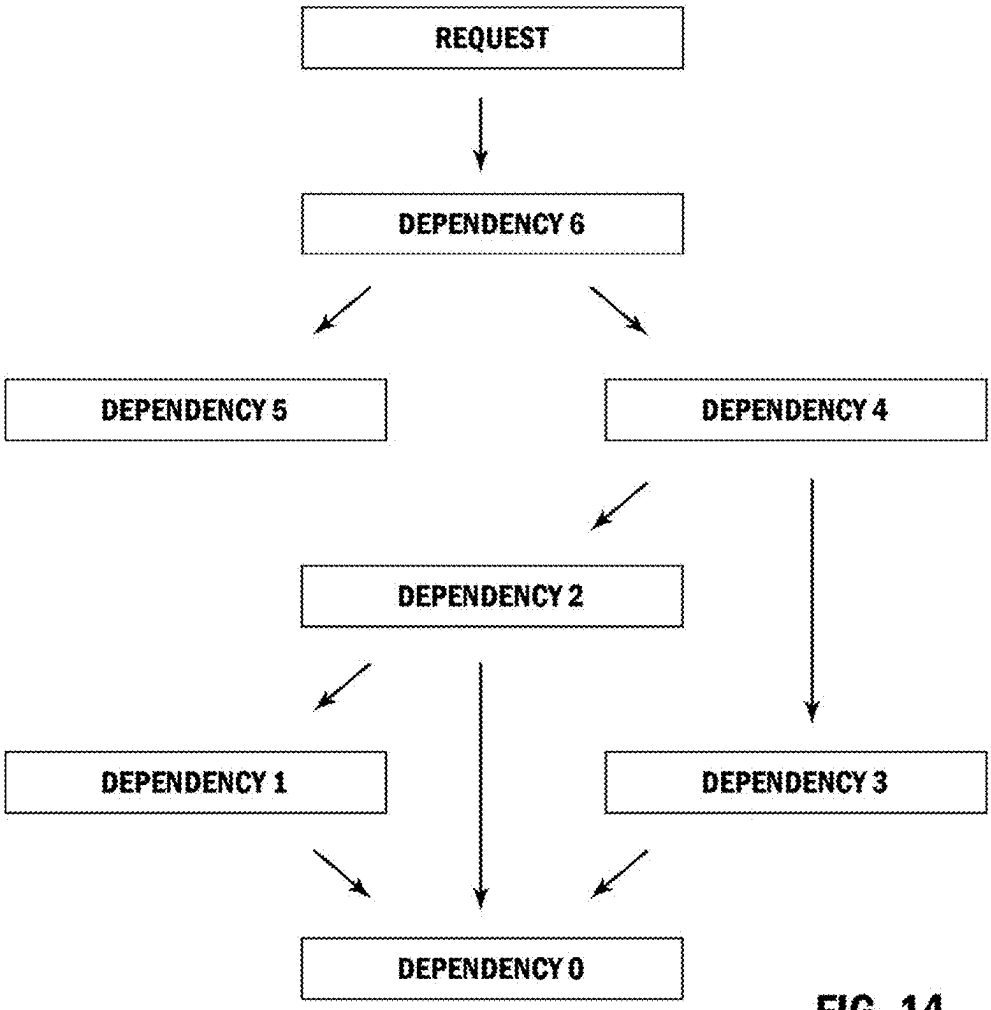

REQUEST
DEPENDENCY 6
DEPENDENCY 5
DEPENDENCY 4
DEPENDENCY 2
DEPENDENCY 1
DEPENDENCY 3
DEPENDENCY 0

FIG. 14

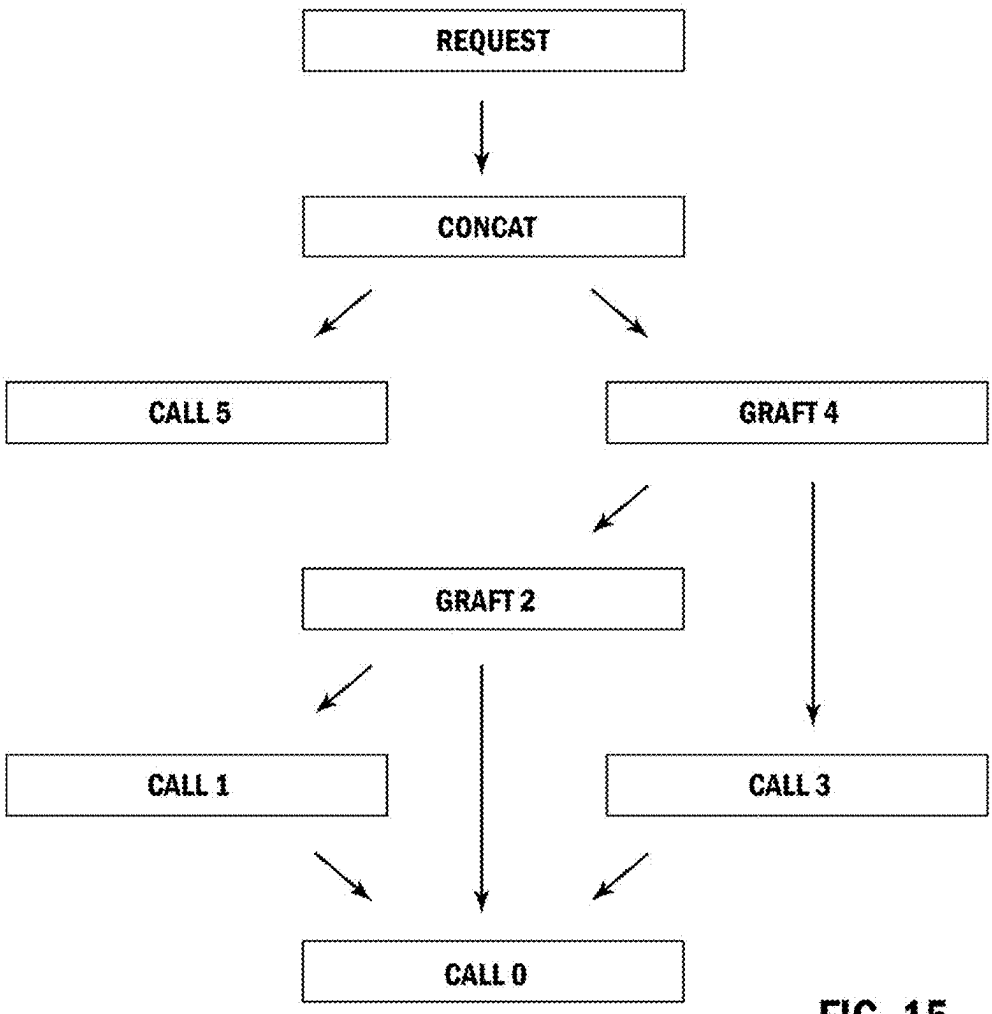


FIG. 15

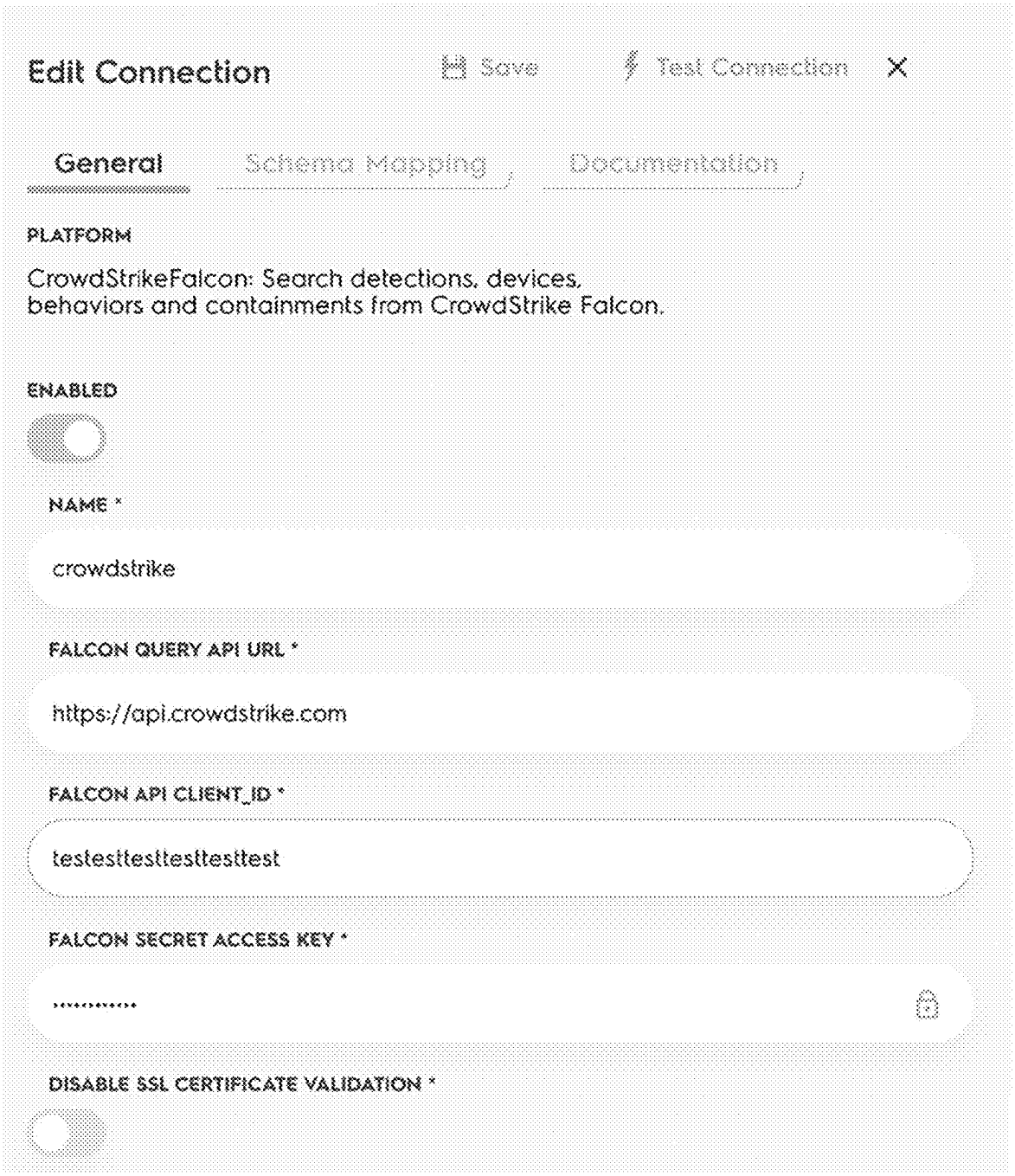

Edit Connection
Save
Test Connection
General
Schema Mapping
Documentation
PLATFORM
CrowdStrikeFalcon: Search detections, devices, behaviors and containments from CrowdStrike Falcon.
ENABLED
NAME *
crowdstrike
FALCON QUERY API URL *
https://api.crowdstrike.com
FALCON API CLIENT_ID *
testesttesttesttesttest
FALCON SECRET ACCESS KEY *
DISABLE SSL CERTIFICATE VALIDATION *

FIG. 16

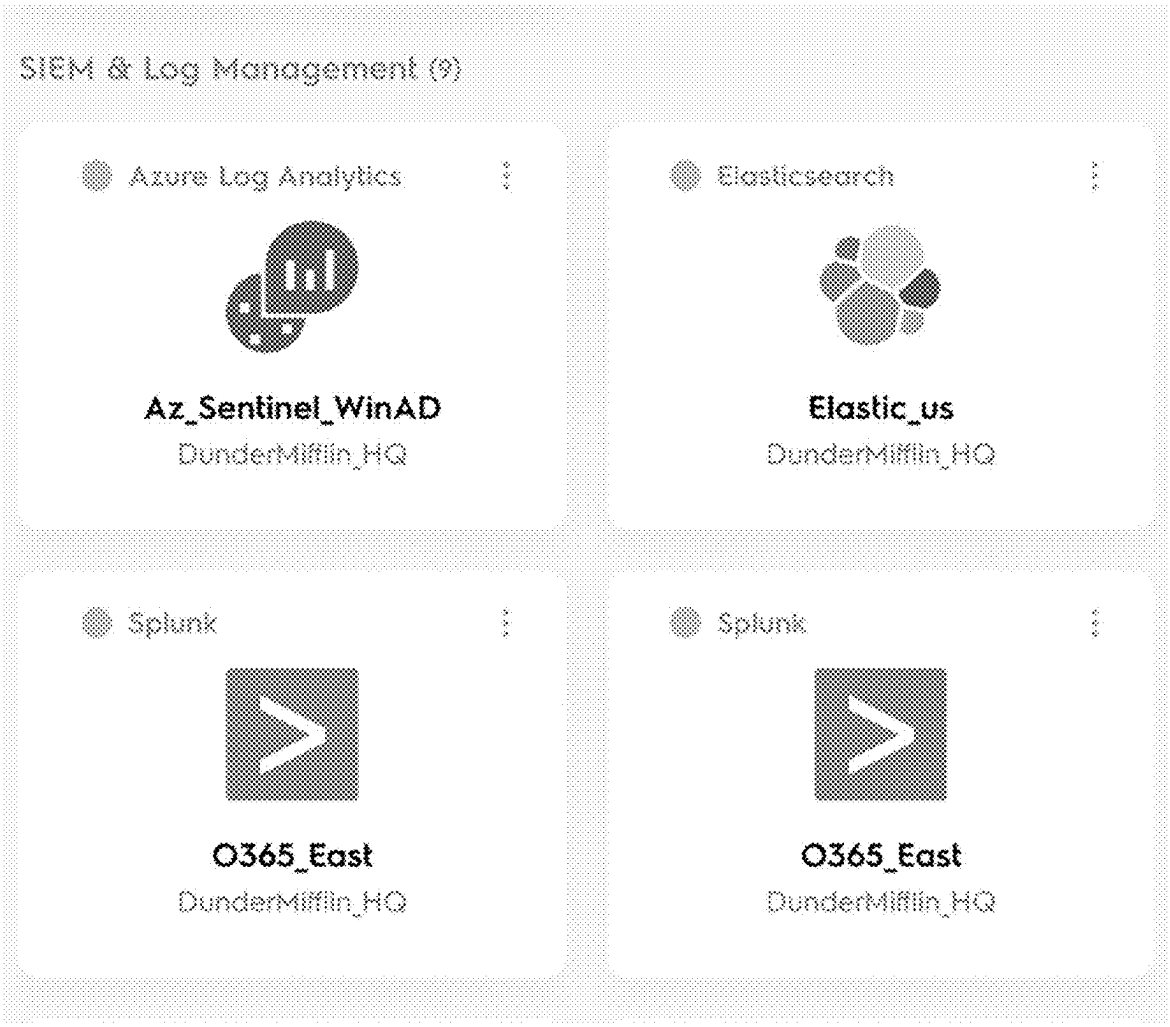

SIEM & Log Management (9)
Azure Log Analytics
Az_Sentinel_WinAD
DunderMifflin_HQ
Elasticsearch
Elastic_us
DunderMifflin_HQ
Splunk
O365_East
DunderMifflin_HQ
Splunk
O365_East
DunderMifflin_HQ

FIG. 17

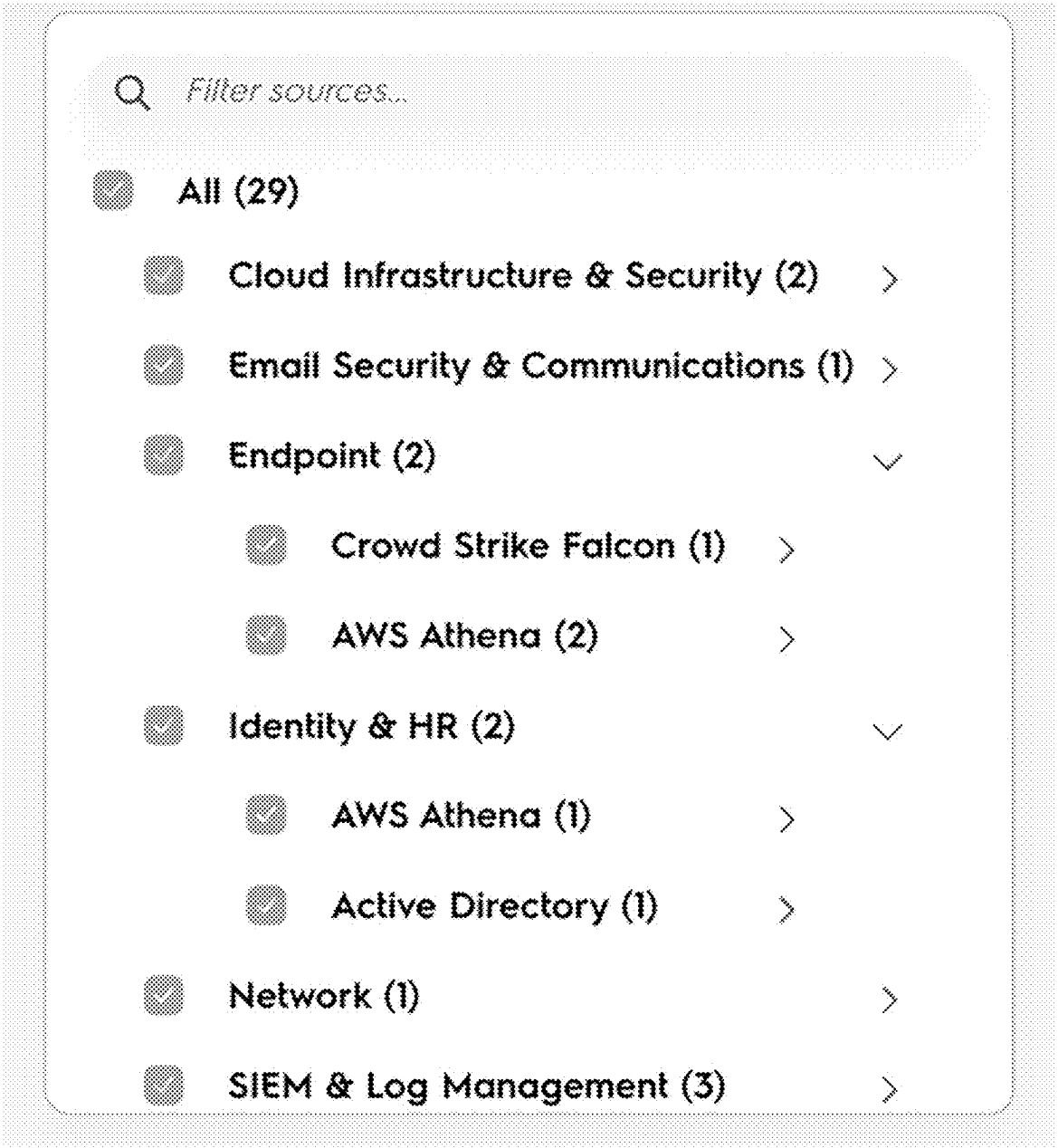

Filter sources...
All (29)
Cloud Infrastructure & Security (2)
Email Security & Communications (1)
Endpoint (2)
Crowd Strike Falcon (1)
AWS Athena (2)
Identity & HR (2)
AWS Athena (1)
Active Directory (1)
Network (1)
SIEM & Log Management (3)

FIG. 18

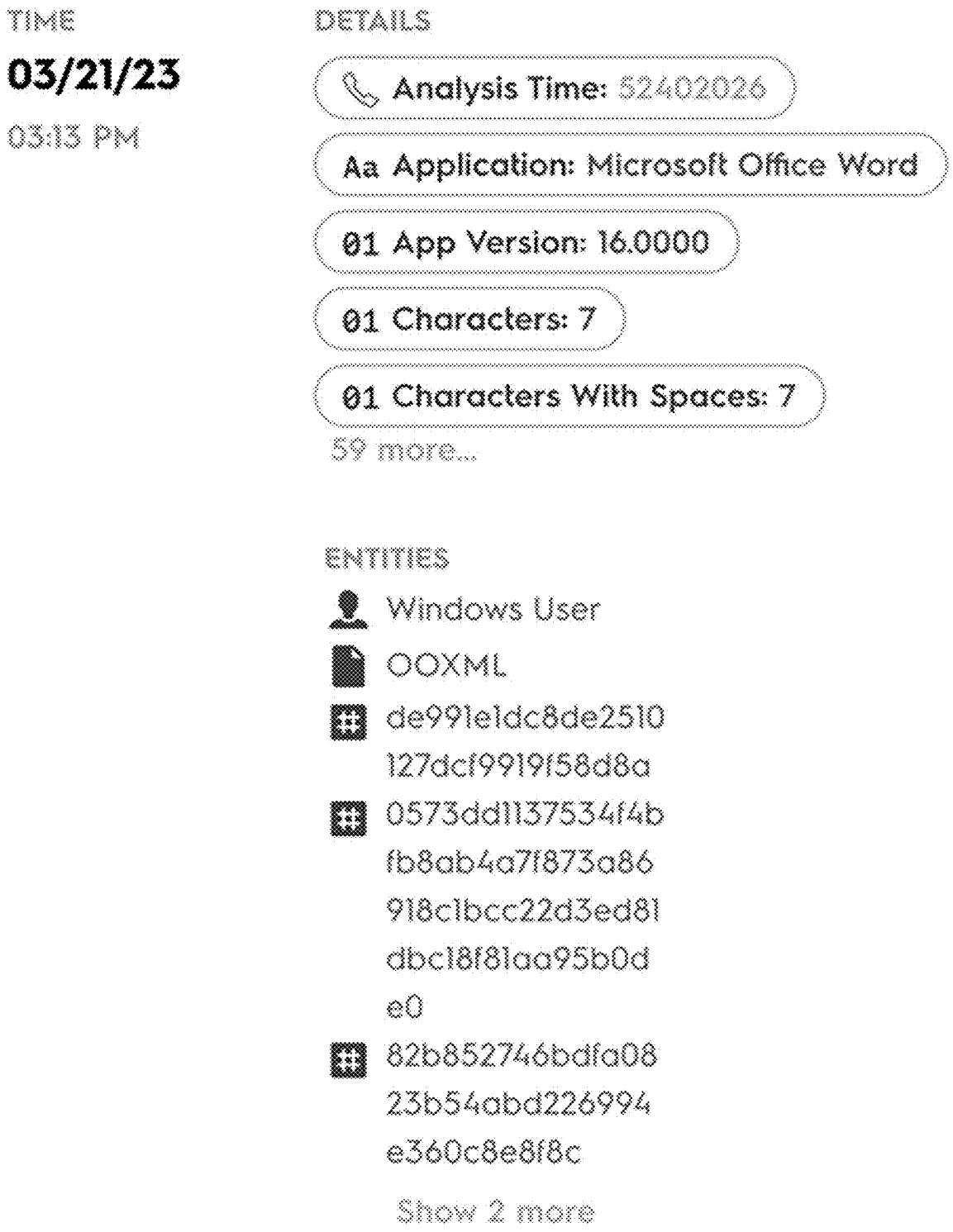

TIME
03/21/23
03:13 PM
DETAILS
Analysis Time: 52402026
Aa Application: Microsoft Office Word
01 App Version: 16.0000
01 Characters: 7
01 Characters With Spaces: 7
59 more...
ENTITIES
Windows User
OOXML
de991e1dc8de2510
127dcf9919f58d8a
0573dd1137534f4b
fb8ab4a7f873a86
918c1bcc22d3ed81
dbc18f81aa95b0d
e0
82b852746bdfa08
23b54abd226994
e360c8e8f8c
Show 2 more

FIG. 19

Expansion	Aliases	Description	Examples

`:all`	*	Expands to everything at the current path level. If no events have been specified in a path, the `:all` command will expand to all events. Otherwise, it will expand to all attributes of the selected events and any attributes specified so far.	`*.time`
			`*.*`
			`authentication.actor.*`
			`authentication.*.ip`

	**	Like `:all`, but always expands all paths to non-recursive ends.	** `#network.**`

`:category`	#	Expands to all events in a given category.	`:category(network)`
			`#network`
			`#network.*.ip`

FIG. 20

Modifier	Aliases	Description	Examples

`:deep`	`//`	Causes the `:all` expansion to expand to include *all* attributes until either the maximum depth or a recursive relationship is reached.	`:deep`
			`:deep.#network.*`
			`//#network.*`
			`#network//*`

`:shallow`	`/`	The opposite of `:deep`, the `:shallow` modifier restricts expansions to just one level of the path.	
`:recursive`	`+`	Causes expansions to follow recursive relationships. A depth limit is required.	
`:no_recursive`		The opposite of `:recursive`	
`:depth`		Sets a maximum depth limit. This is especially useful when traversing recursive relationships.	`:depth(4)`

FIG. 21

Filter	Aliases	Description	Examples
`:type`	`@`	Removes all paths that do not end in an attribute of `type`. The type may be a primitive type or an object type, and the trailing `_t` may be omitted for primitive types.	`*.*.*.:type(ip_t)` `@ip_t`

`:child_of`	`~`	Removes all paths that do not end in an attribute of a type that is a descendant of `type`. This was added to match all types of `endpoint`, but there may be better ways to achieve that goal.	`*.*.:child_of(endpoint)`
			`~endpoint`
			`#network.~endpoint`

`:observable`	`%`	Removes all paths that do not end in a given observable. The observable can be specified by its `caption` (in quotes) or its `type_id`. There are also several aliases like `ip` that match the Query Splunk App.	`//*.*.:observable(2)`
			`#network//*.:observable('IP')`
			`%ip`

`:primitive`		Removes all paths that do not end in an attribute of a primitive data type.	
`:object`		Removes all paths that do not end in an attribute of an object data type.	
`:scalar`		Removes all paths that are arrays.	
`:array`		Removes all paths that are scalars.	
`:root`		Truncates all paths to the root event.	
`:distinct`		Removes all duplicate paths.	
`:min_depth`		Removes all paths that are not at least n levels deep.	`@ip_t.:min_depth(3)`
`:max_depth`		Removes all paths that are over n levels deep.	`@ip_t.:max_depth(7)`
`:parent`		Replaces all paths to primitive type attributes with their parent object.	

FIG. 22

Set Operator	Name	Description
|	Union (or)	A | B selects all paths from both A and B.
&	Intersection (and)	A & B selects all paths from A that are also in B.
-	Difference (minus)	A - B selects all paths from A that are not in B.

FIG. 23

Operator	Description
=	Field equals <value>
!=	Field does not equal <value>
~, contains	String field contains <value>
^=, startswith	String field starts with <value>
$=, endswith	String field ends with <value>
in	Enum field is one of <values>, where values is a comma-separated list.
<, >, <=, >=	Numeric field is less than, greater than, less than or equal to, or greater than or equal to <value>
empty	Field is empty (ex: null or None)

FIG. 24

PARALLEL AND DISTRIBUTED QUERY ENGINE FOLLOW-UP QUERY LANGUAGE

TECHNICAL FIELD OF THE INVENTION

[0001] The present invention relates in general to information technology data management, and, more specifically, to a parallel and distributed query engine follow-up query language.

COPYRIGHT AND TRADEMARK NOTICE

[0002] A portion of the disclosure of this patent application may contain material that is subject to copyright protection. The owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyrights whatsoever.

[0003] Certain marks referenced herein may be common law or registered trademarks of third parties affiliated or unaffiliated with the applicant or the assignee. Use of these marks is by way of example and should not be construed as descriptive or to limit the scope of this invention to material associated only with such marks.

BACKGROUND OF THE INVENTION

[0004] Enterprise information security architecture (EISA) is the practice of applying a comprehensive and rigorous method for describing a current and/or future structure and behavior for an organization's security processes, information security systems, personnel, and organizational sub-units, so that they align with the organization's core goals and strategic direction. Although often associated strictly with information security technology, it relates more broadly to the security practice of business optimization in that it addresses business security architecture, performance management, and security process architecture as well. The primary purpose of creating an enterprise information security architecture is to ensure that business strategy and information technology (IT) security are aligned. As such, enterprise information security architecture allows traceability from the business strategy down to the underlying technology through data tracking and logging.

[0005] To monitor such architectures, enterprises employ IT security personnel to analyze such data and logs. Such security personnel are able, via the data and logs, to review and consider various parameters related to the behavior of devices, applications, and employees within the enterprise and the handling of data by such entities. Processing through such data and logs, though, may be a cumbersome and time-intensive task, especially where the enterprise comprises a large number of devices, applications, and employees, a small number of IT security personnel, or insufficiently trained IT security personnel.

[0006] Multiple security information and event management (SIEM) software platforms already exist to assist with such a security analysis, such as Splunk, ArcSight, and QRadar, which may aggregate relevant data from multiple data and log sources, identify events of interest such as deviations from normal behavior, and generate alerts to take appropriate action. Such platforms help to reduce the time burden of analyzing security data and logs and improve the efficacy of such analyses by security personnel, though, because such systems require structured search and other structured command-line inputs, the user must still be trained sufficiently to effectively direct and utilize the system. Users lacking in sufficient training are often unable to maximize the effectiveness of their analysis or maximize the potential of the software itself.

[0007] In addition, due to the ever-expanding employee work locations, adoption of more SAAS-based tools, and the increasing use of decentralized cloud-based data and storage, it is no longer practical to aggregate such information into a single location or platform, especially prior to performing a data analysis. No solution exists to provide a single application programming interface that allows a user to access and analyze multiple enterprise data storage locations remotely and simultaneously while presenting and reporting information from the multiple sources in a single, uniform display. Such a solution would allow a user to analyze and cross-reference data stored in multiple locations and using multiple query languages in real time without requiring the actual data files to be displaced or combined.

[0008] Thus, there is a need in the art for a parallel and distributed query engine follow-up query language that streamlines and augments the data analysis process by aggregating decentralized enterprise information. The system may further implement interactive artificial intelligence assistant, natural language processing, and workflow-based operations for improved user access and functionality. It is to these ends that the present invention has been developed.

BRIEF SUMMARY OF THE INVENTION

[0009] To minimize the limitations in the prior art, and to minimize other limitations that will be apparent upon reading and understanding the present specification, the present invention describes a parallel and distributed query engine follow-up query language.

[0010] It is an objective of the present invention to provide a distributed query engine that may be implemented on a computing device and exposed on the internet as Software as a Service (SAAS).

[0011] It is another objective of the present invention to provide a distributed query engine that may comprise a proprietary software.

[0012] It is another objective of the present invention to provide a distributed query engine that may comprise a central cloud hosted configuration database.

[0013] It is another objective of the present invention to provide a distributed query engine that may comprise a cloud-based application programming interface.

[0014] It is another objective of the present invention to provide a distributed query engine that may comprise a cloud-based proprietary software.

[0015] It is another objective of the present invention to provide a distributed query engine that may interact with existing enterprise data storage and information context platforms. Such platforms may themselves be on-premise or cloud hosted services themselves.

[0016] It is another objective of the present invention to provide a distributed query engine that may interact with existing security information and event management software platforms.

[0017] It is another objective of the present invention to provide a distributed query engine that may interact with existing internet search engine platforms.

[0018] It is another objective of the present invention to provide a distributed query engine that may obfuscate any personally identifiable information (PII) when the user is interacting with the invention's cloud-based API service.

[0019] It is another objective of the present invention to provide a distributed query engine that may comprise machine learning technology loaded and learning from the user's input.

[0020] It is another objective of the present invention to provide a distributed query engine that may comprise a plurality of workflows.

[0021] It is another objective of the present invention to provide a distributed query engine that may comprise natural language processing for normalizing and interacting with the data present in the third-party platforms.

[0022] It is another objective of the present invention to provide a distributed query engine that can lookup and correlate data from multiple data sources.

[0023] It is another objective of the present invention to provide a distributed query engine that may comprise visual-interactivity.

[0024] These and other advantages and features of the present invention are described herein with specificity so as to make the present invention understandable to one of ordinary skill in the art, both with respect to how to practice the present invention and how to make the present invention.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

[0025] Elements in the figures have not necessarily been drawn to scale in order to enhance their clarity and improve understanding of these various elements and embodiments of the invention. Furthermore, elements that are known to be common and well understood to those in the industry are not depicted in order to provide a clear view of the various embodiments of the invention.

[0026] FIG. **1** schematically presents a parallel and distributed query engine follow-up query language, as contemplated by the present disclosure;

[0027] FIG. **2** schematically presents a parallel and distributed query engine follow-up query language, as contemplated by the present disclosure;

[0028] FIG. **3** schematically presents a parallel and distributed query engine follow-up query language, as contemplated by the present disclosure;

[0029] FIG. **4** schematically presents a parallel and distributed query engine follow-up query language, as contemplated by the present disclosure;

[0030] FIG. **5** schematically presents a computing system configured to carry out and actualize methods and tasks described herein, as contemplated by the present disclosure;

[0031] FIG. **6** schematically presents a parallel and distributed query engine follow-up query language, as contemplated by the present disclosure;

[0032] FIG. **7** illustrates an exemplary user login screen of a parallel and distributed query engine follow-up query language, as contemplated by the present disclosure;

[0033] FIG. **8** illustrates an exemplary platform instances interface of a parallel and distributed query engine follow-up query language, as contemplated by the present disclosure;

[0034] FIG. **9** illustrates an interactive interface of a parallel and distributed query engine follow-up query language, as contemplated by the present disclosure;

[0035] FIG. **10** illustrates an interactive interface of a parallel and distributed query engine follow-up query language, as contemplated by the present disclosure;

[0036] FIG. **11** illustrates an exemplary visualized output of a parallel and distributed query engine follow-up query language, as contemplated by the present disclosure;

[0037] FIG. **12** illustrates a user and system training interface of a parallel and distributed query engine follow-up query language, as contemplated by the present disclosure;

[0038] FIG. **13** illustrates a federated dashboard interface of a parallel and distributed query engine follow-up query language, as contemplated by the present disclosure;

[0039] FIG. **14** illustrates a plan structure of a parallel and distributed query engine follow-up query language, as contemplated by the present disclosure;

[0040] FIG. **15** illustrates a plan structure of a parallel and distributed query engine follow-up query language, as contemplated by the present disclosure;

[0041] FIG. **16** illustrates an add platform connection of a parallel and distributed query engine follow-up query language, as contemplated by the present disclosure;

[0042] FIG. **17** illustrates a plurality of platform connections of a parallel and distributed query engine follow-up query language, as contemplated by the present disclosure;

[0043] FIG. **18** illustrates a federated search input and platforms selection of a parallel and distributed query engine follow-up query language, as contemplated by the present disclosure;

[0044] FIG. **19** illustrates a plurality of federated search results of a parallel and distributed query engine follow-up query language, as contemplated by the present disclosure;

[0045] FIG. **20** illustrates a plurality of path expansions of a parallel and distributed query engine follow-up query language, as contemplated by the present disclosure;

[0046] FIG. **21** illustrates a plurality of expansion modifiers of a parallel and distributed query engine follow-up query language, as contemplated by the present disclosure;

[0047] FIG. **22** illustrates a plurality of filters of a parallel and distributed query engine follow-up query language, as contemplated by the present disclosure;

[0048] FIG. **23** illustrates a plurality of set operators of a parallel and distributed query engine follow-up query language, as contemplated by the present disclosure; and

[0049] FIG. **24** illustrates a plurality of filter operators of a parallel and distributed query engine follow-up query language, as contemplated by the present disclosure.

DETAILED DESCRIPTION OF THE INVENTION

[0050] Certain terminology is used in the following description for reference only and is not limiting. Unless specifically set forth herein, the terms "a," "an," and "the" are not limited to one element, but instead should be read as meaning "at least one." The terminology includes the words noted above, derivatives thereof, and words of similar import.

[0051] Applicant is the owner of registered patents U.S. Pat. Nos. 10,846,342 and 11,397,832, which disclose a similar software technology implemented in a browser-based system. The present technology, by contrast, is implemented in a cloud-based system, which provides significant advantages as discussed in detail below. The use of a

cloud-based system further allows for the implementation of additional services that would not be available in a browser-based system.

[0052] The present invention relates in general to information technology data management, and, more specifically, to a parallel and distributed query engine follow-up query language. As contemplated by the present disclosure, the system may provide a single application programming interface that allows a user to access and analyze multiple enterprise data storage locations remotely and simultaneously while presenting and reporting information from the multiple sources in a single, uniform display. Such a solution may allow a user to analyze and cross-reference data stored in multiple locations and using multiple query languages in real time without requiring the actual data files to be displaced or combined. The system may further implement interactive artificial intelligence assistant, natural language processing, and workflow-based operations for improved user access and functionality.

[0053] The illustrations of FIGS. **1-4** schematically present a parallel and distributed query engine for federated searching. As contemplated by the present disclosure, the parallel and distributed query engine for federated searching may comprise a cloud-based user interface **100** accessible by a user via the internet. The proprietary software of the system may be served as a SAAS service and may comprise a cloud-based investigations response intelligence service (IRIS) application program interface (API) **102**.

[0054] One part of the present system is intended to interface seamlessly with already-existing information technology (IT) platforms. For this reason, the present system creates and presents a parallel and distributed query engine for federated searching comprising data components served from a large number of individual third-party platforms such as, for example, Security Information and Event Management (SIEM) platforms, Log Management Systems (LMS), Endpoint Detection and Response (EDR) platforms, Threat Information Provider (TIP) platforms, Identify and Access Management (IAM) platforms, and Cloud Infrastructure and Security (CIS) platforms. It is contemplated that the parallel and distributed query engine for federated searching may function by, at a minimum, only receiving data remotely and performing its own analyses, or by utilizing analyses performed by SIEM software **108** already installed in an organization's IT network.

[0055] Once accessed, the cloud-based user interface **100** may be interacted with by text-based input **110**, voice-based input **112**, or visual-based input **114**. In an embodiment comprising text-based input **110**, the user may type plain English questions or specific queries and commands into the user interface **100** using any appropriate input source, such as a physical or virtual keyboard or a smartphone or tablet device connected to the system, whether physically or wirelessly. In an embodiment comprising voice-based input **112** the user may interact with the system using a microphone, whether individually or integrated into a smartphone or tablet device, and the system may comprise speech recognition and language interpretation components to understand and interpret the input. In an embodiment comprising visual-based input **114** the user may interact with the system using interactive controls and clickable shortcuts in the web-based graphical user interface (GUI).

[0056] Central to the concept of the present disclosure is the application of a natural language to structured commands convertor **300**, which may be based on applying existing natural language processing concepts to derive and execute commands on the third-party platforms like SIEM **108**. Current natural language processors **300** work by applying lemmatization and tokenization concepts to language inputs to extract the entities and intent of the given instruction. This process involves the analyzing of input terms and the analyzing of used syntax and inflection to determine blocks of entities in the input, and then converting those entities and intents into relevant structured commands **302**. Where current SIEM software **108** requires the user to input structured commands appropriate to the language of the software, the present system receives a natural language input from the user and automatically generates the appropriate command line instruction or sequence of command line instructions **302** for the chosen third-party platforms like SIEM **108**.

[0057] The displaying of results via the cloud-based user interface **100** may be by any appropriate means. In one embodiment the cloud-based user interface **100** may display results as text. As an example, where the user requests information via the remote data source **104**, the cloud-based user interface **100** may display a general summary of the information or the first few lines of text returned by the search along with a hyperlink to the source of the search results. The user's interaction with the hyperlink may then open the source of the search result in a web browser for the user to view directly.

[0058] As another example, where the user requests information via the SIEM software **108**, the cloud-based user interface **100** may display a list of data or log references called for or otherwise matching the user's search parameters. The results of the operation executed on the source platforms like SIEM may have millions of records, but that entire big data is not transferred from that platform to the browser. Instead, the system creates a native SIEM console-like appearance providing pagination, sorting and other navigation support via subsequent commands executed inside of the SIEM letting the end user jump around in the results, giving an appearance that entire data is available.

[0059] The user may also modify the displayed output, as desired, by directing the cloud-based user interface **100** to return the requested results as a visual or graphical display. The innovation running in the browser examines the available fields and value distributions and other parameters to dynamically generate a set of relevant visuals called "data driven visualization". This provides the end user with easy visual analytics to understand and act based upon the data.

[0060] As contemplated by the present disclosure, a user input may be any inquiry, command, or other instruction appropriate for use in the IT security field. As used in the field, though, such user inputs may often follow a logical or standardized and repetitive sequence. To reduce the amount of work or skill required by the user to achieve their sought results, the parallel and distributed query engine for federated searching may further comprise a programmed sequence of inputs, which may be known as a "workflow." A user accessing the cloud-based user interface **100** may load a workflow that the cloud-based IRIS API **102** may then execute.

[0061] The results of each input in the sequence may be displayed, or only the final results of the workflow sequence may be displayed, as desired. The plurality of workflows may be stored on the central cloud service of the system, and

new workflows may be written by users of the system for access by other users of the system. A user may further limit to whom access of workflows may be granted, limiting access, for example, only to other users within.

[0062] By way of example, if a user inputs a natural language command such as "show all VPN logins," the present system receives and translates the natural language input into the SIEM-specific commands for querying VPN login data, and then displays the results of the command inquiries. If the user inputs a natural language command such as "show me a bar graph of login failures by location," the present system receives and translates the natural language input into the SIEM-specific commands for calculating VPN login failures, returning the results of the query, and then visually displaying the results in a bar graph of login failures per remote location. The user may then further specify a time frame by inputting, for example, "show me the results for the past 30 days," and the system will apply the appropriate commands for narrowing the results of the previous inquiry. Thus, the innovation keeps track of the history of relevant questioning from the end-user so that they can be applied in aggregate.

[0063] The system allows the user to capture the interactive session consisting of a sequence of questions and follow-ups into a record known as a "workflow," and which may be automatically or manually recalled by the user for future use by the system. As contemplated by the present disclosure, the system may associate various workflows with commonly-run investigations performed on a given system, and may prioritize such workflow sequences when appropriate.

[0064] In one embodiment the system may receive an open-ended insight question, which, for example, may comprise a user commanding the system to "tell me some interesting events I should investigate today." The system, recognizing the open-ended nature of the inquiry, runs unsupervised machine learning on the subset of events available, clusters them based upon similar attributes and attributes values, and presents in a UI that represents interesting aspects and commonality among the analyzed events. The system then takes feedback from the user based upon their interest and adapts future analysis results based upon that feedback.

[0065] The system may ask clarifying questions and provide choices between close alternatives if there is not a clear match between the user's intent and the system's capability to generate and execute those instructions on the target platforms. The system may recommend the user's past choices it tracks and also additional relevant workflows from its cloud platform. The selected workflows may be run on the user's platforms, so as to provide a user with additional insights into their data that they may not have previously inquired into or considered relevant. In this way the system remains adapted to the individual user's needs, though also provides the user with additional investigations worth performing.

[0066] If the system is unable to understand the user at all, the system provides a choice to the user to "train for my input". The user can then train the system to execute the right command. The user specifies some common English phrases related to how they will ask the question and then specifies the desired command to execute on the desired platform. The user may tag parameters in the question and those parameters may be processed and passed on to the target platform as the command's arguments. The user may then share this training with their team, and everyone can benefit from that new way to interact with the system using natural language.

[0067] In more detail, the system implements domain knowledge with natural language processing to achieve the desired results. Domain knowledge, as contemplated by the present disclosure, may include the user workflows, configurations, and constraints relevant to each third-party platform **108**. The natural language processor **300** of the system converts the entities and intents of the user's input into the command-line instructions **302** of each unique third-party platform **108** by applying such domain knowledge.

[0068] If a user input comprises conditional language, such as "if," "then," "else," "for," "while," or "loop," along with a domain-specific command by commanding the system to, for example, "for each locked employee account, notify their boss," the system may translate and perform the following sequence of steps:

[0069] Step 1: condition, action, and repetition pattern detection

[0070] "<for-each><condition1><action1>"

[0071] Step 2: condition1 construction "account_status=locked"

[0072] Step 3: action1 construction "notify <boss>"

[0073] Step 4: resolve join field from condition1 to be "employee"

[0074] Step 5: resolve join field from action1 to be "boss"

[0075] Step 6: query employee table to find out boss

[0076] Step 7: repeat for every valid condition1

[0077] Once an input has been received, translated, and understood by the system the cloud-based IRIS API **102** may execute or facilitate the input. If, for example, the user input comprises commands related to analyzing the user's security data or logs, the cloud-based IRIS API **102** may interface with the user's SIEM software **108** to retrieve the requested information and present it to the user via the cloud-based user interface **100**. The SIEM software **108** may refer to any one SIEM software **108** known in the art, or may comprise multiple SIEM software **108** installed in the user's IT network, and the system may query data from the single or multiple SIEM software **108** and return the results in a single, uniform display via the cloud-based user interface **100**. In this way, a user may view, visualize, and interact with data from multiple SIEM software **108** seamlessly, and without worrying from which source the data was retrieved.

[0078] If for example, the user input comprises a request for general information the cloud-based IRIS API **102** may interface with a remote data source **104**, such as Google, to retrieve the requested information and present it to the user via the cloud-based user interface **100**. An example of such an interaction would be "Search online for which port SSH runs on", to which the system would provide an answer based upon Google search results and also provide a reference link to the source webpage.

[0079] The illustration of FIG. **5** schematically presents a computing system that may represent an embodiment of the present invention. In some embodiments the method is executed on a computing system such as computing system **500** of FIG. **5**. For example, storage machine **504** may hold instructions executable by logic machine **502** to provide the method to users.

[0080] Display subsystem **506** may display the various elements of the method to participants. For example, display subsystem **506**, storage machine **504**, and logic machine **502** may be integrated such that the method may be executed while being displayed on a display screen. The input subsystem **508** may receive user input from participants to indicate the various choices or user inputs described above.

[0081] The described method may be executed, provided or implemented to a user on one or more computing devices via a computer-program product such as via an application programming interface (API). FIG. **5** schematically shows a non-limiting exemplary embodiment of a computing system **500** that can enact the method described above. Computing system **500** may be any appropriate computing device such as a personal computer, tablet computing device, gaming device or console, mobile computing device, etc. Computing system **500** includes a logic machine **502** and a storage machine **504**. Computing system **500** may include a display subsystem **506**, input subsystem **508**, and communication subsystem **510**.

[0082] Logic machine **502** may execute machine-readable instructions via one or more physical devices. For example, the logic machine **502** may be configured to execute instructions to perform tasks for a computer program. The logic machine may include one or more processors to execute machine-readable instructions.

[0083] Storage machine **504** includes one or more physical devices configured to hold or store instructions executable by the logic machine to implement the method. When such methods and processes are implemented, the state of storage machine **504** may be changed to hold different data. For example, storage machine **504** may include memory devices such as various hard disk drives or CD or DVD devices.

[0084] Display subsystem **506** may visually present data stored on storage machine **504**. For example, display subsystem **506** may visually present data to form a graphical user interface (GUI). Input subsystem **508** may be configured to connect and receive input from devices such as a mouse, keyboard, or gaming controller. Communication subsystem **510** may be configured to enable system **500** to communicate with other computing devices. Communication subsystem **510** may include wired and/or wireless communication devices to facilitate networked communication.

[0085] The illustration of FIG. **6** schematically presents a parallel and distributed query engine for federated searching. The system may comprise an entirely cloud-based user interface **100** accessible by a user through any internet or web browser known in the art. In this way the system does not require the installation of proprietary hardware or software onto a user's system, but otherwise allows the user to access the system.

[0086] The user may interact with the cloud-based user interface **100** by text-based input **110**, voice-based input **112**, or visual-based input **114** in a command-response manner emulating a conversation. The user may type queries and commands into the user interface **100** using any appropriate input source and may receive command results back from the third-party platforms where these commands are executed using the APIs of that specific external platform. The various inputs issued by the user may be for the purpose of executing commands, or taking actions, or making a configuration change, or querying data from the third-party platforms. The user may enter the direct platform command to be executed as is, or may provide an English sentence which would be first interpreted by the system's natural language processing engine. The distributed query engine may then translate into the platform-specific command and then execute it on the appropriate third-party platform. The interaction between the browser and the third-party system for this command execution may be known as a "command-and-control channel."

[0087] For achieving connectivity between the browser and the third-party platform, the user's administrator may store that platform's connection configuration in the cloud service but the more sensitive credentials such as connection auth-tokens are stored in the cloud service's secrets' store. The user admin is prompted for those credentials during setup time and those credentials are saved in the cloud service's secrets' store.

[0088] When the third-party platform connectivity is needed, the connection configuration loaded from the cloud is combined with the credentials stored in the secrets' store, to create the connection described as the "command and control channel" above. Upon obtaining any query results gathered from various remote data storage platforms, those results are shown to the user on the browser while ensuring the privacy of platforms' credentials from the end user. By such a mechanism, this system of securing the parallel and distributed query engine for federated searching even from itself may be known as "privacy by design."

[0089] To begin using the system a user may first log into the cloud-based user interface **100** and provide a user input **116**, which may be a selection of a remote data storage platform and an issuance of commands in search query or natural language form to access or analyze data on that platform. The system may utilize its natural language processor **300** to convert the commands into relevant structured commands **302** recognized by the selected remote data storage platform. The system may then access the remote data platform, execute the analysis or commands instructed, and then return the results of the instructions to the user as a compiled display of results. The user may then input **116** additional commands or swap between platforms and use the results of the previous instructions to conduct further analyses on data stored remotely in a second location.

[0090] By way of example, the system may receive a plurality of command inputs from a user and return search results based on the input and the selected software platform and return data analyses that the user may then continue with on a second software platform. By way of example, the user may perform the following sequence of steps:

[0091] Step 1: Switch platform to splunk

[0092] Step 2: show me data from o365 index

[0093] Step 3: search 1 month ago to now

[0094] Step 4: show me data where recordtype equals 47 or recordtype equals 28

[0095] Step 5: remember ClientIP into <compromised_host>

[0096] Step 6: switch platform to elastic search

[0097] Step 7: start over

[0098] Step 8: show me all data where source_ip equals <compromised_host.clientip>

[0099] Step 9: switch platform to virustotal

[0100] Step 10: show file report of 10676cf66244cfa91567fbc1a937f4cb19438338b35b69d4

[0101] Because the analytical and investigational processes of the parallel and distributed query engine for

federated searching aggregates and standardizes information stored across multiple remote locations in various languages, this operation may be known as a "federated search." The federated search feature of the system may run parallel or sequential searches, as desired, across the various remote software platforms programmed by the system's user. These searches may be run based on historical data contained within the remote systems, or may be run continuously in real-time such that the user is constantly presented with current and updating information. In this way the system may normalize data across multiple remote storage platforms and aggregate and analyze that data without moving it from the remote location. An example of normalization would be that when a federated parallel search brings results with one platform's tabular results having column "app_name" and another platform's tabular results having column "application_name", the system's schema normalization configuration determines that these fields represent the same information and hence they are merged to be presented in the same normalized column called "application".

[0102] The parallel and distributed query engine for federated searching may further comprise a user or team collaboration system, which may allow a single user or a team of users to share information, search data, and analyses results amongst one another. Many enterprises have teams of specialists dedicated to specific tasks, such as security teams, IT helpdesk teams, and management teams, and these various individuals or groups may need to share data analysis information with each other quickly and efficiently. These various types of extracted and shared results may be known collectively as an "Interest List."

[0103] The illustration of FIG. **7** illustrates an exemplary user login screen of a parallel and distributed query engine for federated searching. An admin user of the system may begin by first selecting the data platforms upon which their system is based, which may be any SIEM software or other appropriate third-party platforms. The selection of the data platforms allows the IRIS API to select the appropriate translational protocols for sending instructions to and receiving data from the data platforms. The admin would have configured the network endpoints of their data platforms' servers so that the IRIS API knows where to send and receive the user's instructions and data. The admin would have configured the credentials to access the data platforms within the system, which may grant users access to any data for which they are authorized.

[0104] The illustration of FIG. **8** illustrates an exemplary platform instances interface of a parallel and distributed query engine for federated searching. In one embodiment a user may navigate to a section of the system where they may enter instances of multiple data platforms that the system may search through and source data from simultaneously or in sequence. The system may store the remote area address of the multiple platforms so that a user may quickly switch between these platforms to retrieve or analyze data. A user may, by way of example, retrieve analyses data from a first platform and then switch to a subsequent platform and use the first retrieved data to perform a secondary analysis.

[0105] The illustrations of FIGS. **9-11** illustrate an exemplary web-based user interface of a parallel and distributed query engine for federated searching. Once the user has logged-into the system, the user may begin by entering instructions in the user interface, whether by text, voice, or visual input. The IRIS API may display inquiries and results within the user interface, and may further display entered text or transcribed voice or visual inputs in sequence to resemble a conversational progression. The format of the display output may be directed, as desired, by the system user, and the IRIS API may create visual or graphical output displays based on user parameters. The format of the display output may also be directed by the system, and the IRIS API may create visual or graphical output displays based on the optimal format for displaying such an output, which may be known as "data-driven visualization."

[0106] The illustration of FIG. **12** illustrates an exemplary workflow interface of a parallel and distributed query engine for federated searching. Workflows may be loaded by the user via the web-based user interface, and may be stored on the system's cloud-based IRIS service's central database. Workflows may be programmed by any user of the system, whether manually by the user or automatically by the system recording the user's input sequences, and made available to other users, and may be further correlated with specific IT security purposes or instructional sequences.

[0107] The illustration of FIG. **13** illustrates a federated dashboard interface of a parallel and distributed query engine for federated searching. Because the analytical and investigational processes of the parallel and distributed query engine for federated searching aggregates and standardizes various types of enterprise information stored across multiple third-party platforms into panels of informative visuals created from the merged data from multiple platforms, this feature may be known as a "federated dashboard." The types of data recorded by an enterprise may vary significantly in scope and purpose such as, for example, employee information, device information, application information, and other types of information. The parallel and distributed query engine for federated searching may be able to extract, aggregate, and analyze these various types of information, regardless of where and in what format it is stored, and present the results of such aggregation and analysis in a uniform output. These various types of extracted IT objects may be known collectively as "monitored objects."

[0108] The illustrations of FIG. **14** and FIG. **15** illustrate a plan structure of a parallel and distributed query engine for federated searching. The query engine may receive a user command and a "query planner" may understand and separate the user command into a series of parallel and sequential calls to the various third-party systems **108**. By way of example, the query engine may understand that it is being asked to determine Dependency 6, which may combine results from an inquiry to Dependency 5 and Dependency 4. The information required from Dependency 4 may require data from Dependency 2 and Dependency 3, which themselves require additional data from Dependency 1 and Dependency 0. The query planner may determine the order of the requirements and assemble a sequence in which the data is to be collected.

[0109] By way of an example, Dependency 0 may comprise user login data related to an $assetTag and a $userDn, which both identify the same user. Dependency 1 may contain security information data related to the $assetTag, while Dependency 3 may contain security information data related to the $userDn. Dependency 2 may then collate the data received from Dependency 0 and Dependency 1, and Dependency 4 may then collate the data from Dependency

2 and Dependency 3. The data of Dependency 4 may then be collated with the data of Dependency 5, which may comprise security information related to a particular device or object in the system, into Dependency 6, which may then be presented to a user of the system as a response to their inquiry.

[0110] By way of a second example, data may be called via a query from a third-party database **108** and then may be grafted together via a federated join. These calls and grafts may be sequential and cumulative until resulting in a concatenation of the requested data that may be presented to the user. The query planner may further prune and regroup subtrees and subcalls within the sequence, as needed, to aggregate as much data as possible while also providing the most efficient return of data in response to the query. All of the various calls, grafts, joins, pruning, and regrouping may be performed by a "query executor".

[0111] Such a plan structure is facilitated by the use of the cloud-based user interface **100**. Web browsers are known to have resources limited to the hardware on which they are running, and, as such, have a limited ability to run multiple parallel and distributed queries simultaneously. Because each inquiry may be broken down into multi-part commands, the implementation of a cloud-based structure allows for the nearly-infinite scaling of resources thus providing the system the ability to run more advanced and more complicated plans and executions in response to user inquiries. The implementation of cloud-based architecture also allows for the caching of query results, which may itself allow for the speedier analysis of commonly or regularly queried data points.

[0112] The parallel and distributed query engine for federated searching may further comprise a platform-specific "driver" that runs ephemerally in cloud and executes the query on its designated target platform. The driver knows how to call the target platform's APIs and has custom logic written to interact with that platform. The driver is available to query engine as a callable function running in an ephemeral container. The driver abstracts the query engine from having to know and deal with running platform-specific queries. Each federated search operation may result in multiple drivers' invocations running in parallel, or in a dependency sequence based upon the query execution plan.

[0113] The Open Cybersecurity Schema Framework (OCSF) is an open-source project, delivering an extensible framework for developing schemas, along with a vendor-agnostic core security schema. Vendors and other data producers can adopt and extend the schema for their specific domains. Data engineers can map differing schemas to help security teams simplify data ingestion and normalization, so that data scientists and analysts can work with a common language for threat detection and investigation.

[0114] The illustrations of FIGS. **20-24** illustrate a parallel and distributed query engine follow-up query language, as contemplated by the present disclosure. The parallel and distributed query engine follow-up query language facilitates the integration of the parallel and distributed query engine for federated searching with the OCSF. The parallel and distributed query engine follow-up query language views data through an OCSF lens: all of the data, regardless of source, is represented as OCSF events and provides a view of these events narrowed based on the search criteria and selected connectors.

[0115] Any query language for the follow-up query language needs to work within the constraints of a query data model and a query search contract. Within the query data model is the schema defined by the OCSF. Under the OCSF all search results are deemed to be events, and each event may comprise a plurality of attributes. These attributes may comprise, for example, strings, numbers, Boolean values, or objects, which are compound attributes having their own additional attributes.

[0116] Events and attributes may be described as paths through the schema in dot notation such as, for example, <event> [.<attribute> [.<attribute> [. . .]]]. All events and attributes have an internal "name", using all lowercase letters with underscores, and a human-friendly "caption", using mixed case letters and spaces.

[0117] Enumerations are coded attributes. Their value must be one of a list of values, and they are very useful for normalizing things like status code across data sources. Enumerations have an identifier, usually an integer, and a caption.

[0118] Observables are aliases to multiple attributes across the schema. Observables usually lead to attributes that represent key facts or indicators of compromise. For instance, it is not uncommon for an event to have half a dozen possible internet protocol (IP) addresses associated with it, and remembering all of the paths to an IP in the schema can be cumbersome. One observable can expand to show all of those IP addresses for simplicity.

[0119] The query search contract is the combination of the query data model and filters allowed on that data. The query search contract requires users to select events and, optionally, attributes from the data model to be included in results. Filters can be specified for each desired event type to narrow the result set, and connections can be specified. A time range may be specified to avoid large scanning queries to data sources as all events have a time property.

[0120] Filters describe characteristics of records to be included in the result set. The most basic form of filters, which may be known as predicates, describe a field, operator, and a value for comparison. The allowed operators for a field are determined by its data type, which is defined in the query data model.

[0121] Filters on lists can be thought of as a predicate that must be true for either ANY or ALL elements in the list. The search contract also allows for grouping filters with AND or OR parameters. It also supports negating filters, so that entire groups can be negated. While different from using negated forms of operators, both can be used to the same effect.

[0122] Based on this framework, the parallel and distributed query engine follow-up query language should work within the existing OCSF search capabilities by describing events and attributes to be included in search results, filters for these events, and connectors to be queried. The follow-up query language should also allow for simple specifying of a time range for events and simplify navigating the schema. Finally, the follow-up query language should output these various parameters as a single query request.

[0123] User patterns within the query data model may allow for a number of observables aggregators that may simplify searching through the schema. The query data model defines most observables by type but, in some cases, defines them by specific locations in the schema. Describing attributes by their type, such as an object such as "user" or

a primitive type such as "ip", can provide the same shortcuts as most observables. Describing events by their category may let users describe multiple event types at once, which may enable broad searches and alleviate the need to identify specific event types and to which type a source is mapped.

[0124] While there are many closely related object types in OCSF, that schema requires that a user choose only one in their search. By way of example, an "endpoint" object may further comprise a "network_endpoint", a "device", or a "proxy_endpoint". Allowing for the searching of endpoint objects and all of its descendants will catch all four of these objects in a single search. Also, finding events and attributes by both their "name" and/or their "caption" could maintain consistent terminology.

[0125] To perform a basic query in the parallel and distributed query engine follow-up query language the user may utilize the format:

[0126] QUERY <fields> WITH <filter>

[0127] By way of example, for the user to perform a search of network activity by source IP address they may enter:

[0128] QUERY network_activity.status_id, network_activity.src_endpoint.ip

[0129] WITH network_activity.src_endpoint.ip='172.10.10.153'

[0130] The parallel and distributed query engine follow-up query language may then allow for the implementation of additional parameters, allowing the user to utilize the format:

[0131] QUERY <fields>

[0132] [WITH <filter>]

[0133] [BEFORE <time>]

[0134] [AFTER <time>]

[0135] [FROM <connectors>]

[0136] By way of example, for the user to perform a search of network activity within a specific timeframe they may enter:

[0137] QUERY network_activity.*BEFORE 24 hrs AFTER 48 hrs

[0138] A user may additionally specify connectors by providing the desired connector names as a list of strings:

[0139] QUERY network_activity.*FROM 'My S3 Bucket', 'Crowdstrike Falcon'

[0140] The parallel and distributed query engine follow-up query language may further comprise a plurality of fields, which may comprise paths through the schema. These fields may describe events by category, attributes by type, entity searches by observables, and combine events and attributes into set operations using a more concise syntax.

[0141] To determine, if a source is mapped to "http_activity" or "network_activity" a user may specify:

[0142] QUERY #network.*

[0143] Or, in the alternative, could specify both activity types:

[0144] QUERY (http_activity|network_activity).*

[0145] If the user wanted to perform the entity search with a MAC address they may specify:

[0146] QUERY #network/* WITH % mac='FD:6D:43:3B:F0:E8'

[0147] If the user wanted to perform a search for all devices by an IP address they may specify:

[0148] QUERY **: type (device) WITH @ip='172.10.10.153'

[0149] If a user wanted to request all of the properties of an event except for the start and end time they may specify:

[0150] QUERY #network. (*-(start_time|end_time)) . . .

[0151] Simple filters (predicates) in the parallel and distributed query engine follow-up query language may comprise:

[0152] <field> <operator> <value>

[0153] In the event that a user wished to use a field selector that describes multiple attributes or paths the filter will be expanded to operate on multiple fields across multiple events. This expansion greatly reduces query verbosity, and any field target that does not belong to an event that is included in the field list will be dropped. This allows filters to be written more concisely with wildcards without expanding the result set.

[0154] For instance, the below query uses a wildcard in the filter:

[0155] QUERY (dns_activity|http_activity|ssh_activity).status_id

[0156] WITH *.status_id=SUCCESS

[0157] This query is identical to the below query without the wildcard in the filter:

[0158] QUERY (dns_activity|http_activity|ssh_activity).status_id

[0159] WITH (dns_activity|http_activity|ssh_activity).status_id=SUCCESS

[0160] The parallel and distributed query engine follow-up query language field specifications use a dot (.) to separate parts of the expression, letting simple direct paths read just like the dot notation names commonly found in other interfaces. The follow-up query language field specification commands begin with a colon (:), and strings with spaces must be surrounded by single quotes ('). Command parameters are listed inside of parentheses, and empty parentheses are optional when parameters are not used. Commands and command aliases (like *) do not require a dot in front of them.

[0161] While the invention has been described in connection with what is presently considered to be the most practical and preferred embodiments, it is to be understood that the invention is not to be limited to the disclosed embodiments, but, on the contrary, is intended to cover various modifications and equivalent arrangements included within the spirit and scope of the appended claims.

I claim:

1. A follow-up query language, comprising:

a cloud service;

a proprietary software;

a query planner;

a query executor;

a user interface;

a central configuration database;

wherein said proprietary software is installed on said central configuration database;

wherein said user interface is a web-based interface between a user and said proprietary software;

wherein said proprietary software is run on said cloud service and displayed to said user via said user interface;

wherein said proprietary software is granted access by said user to a remote data storage platform;

wherein said user interface prompts said user for a first input;

wherein said user interface receives said first input from said user;
wherein said query planner converts said first input into a series of parallel and sequential calls;
wherein said series of parallel and sequential calls are in a command language understood by said remote data storage platform;
wherein said series of parallel and sequential calls are issued to said remote data storage platform;
wherein said remote data storage platform returns an initial result to said proprietary software;
wherein said query executor aggregates said initial result into a first output;
wherein said first output is displayed to said user via said user interface.

2. The query language of claim **1**,
wherein said cloud service further comprises a natural language processor;
wherein said user interface prompts said user for a first natural language input;
wherein said user interface receives said first natural language input from said user;
wherein said first natural language input is converted into a series of parallel and sequential calls;
wherein said series of parallel and sequential calls are in a command language understood by said remote data storage platform;
wherein said series of parallel and sequential calls are issued to said remote data storage platform;
wherein said remote data storage platform returns said initial result to said proprietary software;
wherein said query executor aggregates said initial result into said first output; and
wherein said first output is displayed to said user via said user interface.

3. The query language of claim **2**,
wherein said user interface displays said first output and then prompts said user for a subsequent input;
wherein said subsequent input is converted to said series of parallel and sequential calls and issued to said remote data storage platform;
wherein said remote data storage platform returns said initial result that is converted to a subsequent output and displayed to said user;
wherein said prompting for said subsequent input by said user interface is repeated until said user performs a stop action;
wherein said first input and each of said subsequent inputs are recorded in the order received by said proprietary software as a sequence of commands; and
wherein said sequence of commands is stored by said proprietary software as a unique workflow.

4. The query language of claim **3**;
wherein said user interface receives said first input from said user;
wherein said first input is a multiple command input;
wherein said multiple command input is converted into a series of parallel and sequential calls by said proprietary software;
wherein a first of said series of parallel and sequential calls is issued to said remote data storage platform;
wherein a next of said series of parallel and sequential calls is issued to said remote data storage platform until all commands in said series of parallel and sequential calls have been issued;
wherein said remote data storage platform returns said initial result to said proprietary software;
wherein said query executor aggregates said initial result into said first output;
wherein said first output is displayed to said user via said user interface; and
wherein said multiple command input is stored by said proprietary software as a unique workflow.

5. The query language of claim **4**,
wherein said user interface receives said natural language input from said user;
wherein said natural language input is said multiple command input;
wherein said multiple command input is converted into said series of parallel and sequential calls by said natural language processor of said proprietary software;
wherein said first of said series of parallel and sequential calls is issued to said remote data storage platform;
wherein said next of said series of parallel and sequential calls is issued to said remote data storage platform until all commands in said series of parallel and sequential calls have been issued;
wherein said remote data storage platform returns said initial result to said proprietary software;
wherein said query executor aggregates said initial result into said first output;
wherein said first output is displayed to said user via said user interface; and
wherein said multiple command input is stored by said proprietary software as a unique workflow.

6. The query language of claim **5**,
wherein said user instructs said user interface to recall and execute one of said unique workflows;
wherein said sequence of commands of said unique workflow is executed by said proprietary software to generate a progression output; and
wherein said progression output is displayed to said user via said user interface.

7. The query language of claim **6**,
wherein said proprietary software continuously issues said remote platform-specific command to said remote data storage platform;
wherein said remote data storage platform continuously returns an updated result to said proprietary software;
wherein said proprietary software continuously compares said initial result against said updated result to check for an updated output; and
wherein said updated output is displayed to said user via said user interface.

8. The query language of claim **7**,
wherein said first output, said progression output, and said updated output comprise a visualized output;
wherein said visualized output is created dynamically by said proprietary service; and
wherein said visualized output is displayed to said user via said user interface.

9. The query language of claim **8**,
wherein said proprietary software detects an open-ended input;

wherein said proprietary software extrapolates a plurality of queries from said open-ended input;
wherein said proprietary software presents said plurality of queries to said user;
wherein said user provides a feedback to said proprietary software; and
wherein said proprietary software adapts a plurality of future inquiries based on said feedback.

10. The query language of claim **9**,
wherein said federated visualized output is presented on a federated dashboard;
wherein said federated dashboard comprises a plurality of data categories; and
wherein said first output, said progression output, and said updated output are normalized and merged before being displayed to said user via said user interface.

11. The query language of claim **10**, further comprising:
a query data model; and
a query search contract;
wherein said query data model further comprises an open cybersecurity schema framework lens;
wherein said open cybersecurity schema framework lens further comprises a plurality of events;
wherein one each of said plurality of events further comprises a plurality of attributes;
wherein said query search contract further comprises a plurality of filters; and
wherein said plurality of filters of said query search contract dictate which of said plurality of events and said plurality of attributes are displayed to a user.

12. The query language of claim **11**,
wherein a combination of said plurality of events and said plurality of attributes comprise a path.

13. The query language of claim **12**, further comprising:
a plurality of enumerations;
wherein one each of said plurality of enumerations comprise a coded attribute.

14. The query language of claim **13**, further comprising:
a plurality of observables;
wherein one each of said plurality of observables comprise an alias to a plurality of attributes.

* * * * *